Diseño Arquitectónico - Sistema BP SmartBank

Contenido

1. Introducción

Este documento describe la solución arquitectónica para el sistema de banca por internet de la entidad BP. Se sigue el modelo C4 para documentar los niveles de contexto, contenedores y componentes, además de justificaciones teóricas, cumplimiento normativo y uso de servicios cloud (AWS y Azure) para garantizar alta disponibilidad, bajo acoplamiento, seguridad y cumplimiento normativo.

2. Requerimientos del Caso

- El sistema permite a los usuarios:
  Consultar el histórico de movimientos
- Realizar transferencias y pagos
- Acceder a través de SPA y app móvil
- Autenticarse con OAuth 2.0 y reconocimiento facial (onboarding)
- Recibir notificaciones por mínimo 2 canales
- Integrarse con Core Banking y sistema complementario de cliente
- Auditar las acciones del cliente
- Manejar persistencia para clientes frecuentes
- Usar una arquitectura cloud híbrida con alta disponibilidad y monitoreo

3. Normativas y Consideraciones de Seguridad

3.1. Ley Orgánica de Protección de Datos Personales (LOPDP) – Ecuador

Es la ley ecuatoriana que regula el tratamiento de los datos personales de los ciudadanos. Entró en vigor en 2021 y se alinea con principios del Reglamento General de Protección de Datos (GDPR) de la Unión Europea.

Aspectos clave:

- Consentimiento informado: Los usuarios deben aceptar claramente el uso de sus datos.
- Finalidad: Los datos solo deben usarse con el propósito para el que fueron recolectados. Minimización: Solo se pueden recolectar los datos estrictamente necesarios.
- Derechos del titular: Acceso, rectificación, cancelación, oposición (ARCO).
- Responsabilidad proactiva: El banco debe implementar medidas de seguridad para proteger los datos.
- Notificación de incidentes: Ante una violación de seguridad, se debe notificar a la autoridad y al afectado.

Aplicación en BP SmartBank:

- Cifrado de datos en tránsito y reposo.
- Controles de acceso basados en roles (RBAC).
- Trazabilidad de accesos a través del microservicio de auditoría.
- Implementación de mecanismos de consentimiento explícito.

3.2. ISO/IEC 27001 - Sistema de Gestión de Seguridad de La Información (Sgsi)

Norma internacional que define los requisitos para establecer, implementar y mantener un SGSI.

Aspectos clave:

- Gestión de riesgos de seguridad de la información.
- Seguridad lógica y física.
- Controles de acceso y autenticación robusta.
- Auditorías y monitoreo constante.
- Gestión de continuidad de negocio.

Aplicación en BP SmartBank:

- Todos los microservicios están protegidos bajo políticas de un SGSI.
- Mecanismos de autenticación y autorización centralizados (OAuth2).
- Protección del perímetro a través de WAF y API Gateway.
- Registros de actividad a través del microservicio de auditoría.

3.3. NIST Sp 800-53 Controles de Seguridad Y Privacidad

Marco de referencia del Instituto Nacional de Estándares y Tecnología (NIST) de EE.UU. para controles técnicos y organizacionales.

Aspectos clave:

- Familias de controles: autenticación, autorización, monitoreo, cifrado, backup, entre otros.
- Alineado con ISO/IEC 27001 y COBIT.
- Monitoreo continuo y respuesta ante incidentes.

Aplicación en la solución:

- Registro y trazabilidad de eventos con CloudTrail y servicios de auditoría.
- Encriptación en todas las bases de datos y colas.
- Segregación de funciones y control de acceso por roles.
- Uso de políticas de seguridad aplicadas en los contenedores y servicios.

3.4. Aws well-architected framework

Conjunto de buenas prácticas de arquitectura en la nube ofrecido por AWS, basado en 6 pilares fundamentales.

Pilares:

- Excelencia operativa: monitoreo, automatización, documentación.
- Seguridad: gestión de identidades, cifrado, cumplimiento normativo.
- Fiabilidad: tolerancia a fallos, recuperación ante desastres.
- Eficiencia de rendimiento: escalabilidad, equilibrio de carga.
- Optimización de costos: uso eficiente de recursos y modelos de pago por demanda.
- Sostenibilidad: reducción de huella de carbono y recursos innecesarios.

Aplicación en BP SmartBank:

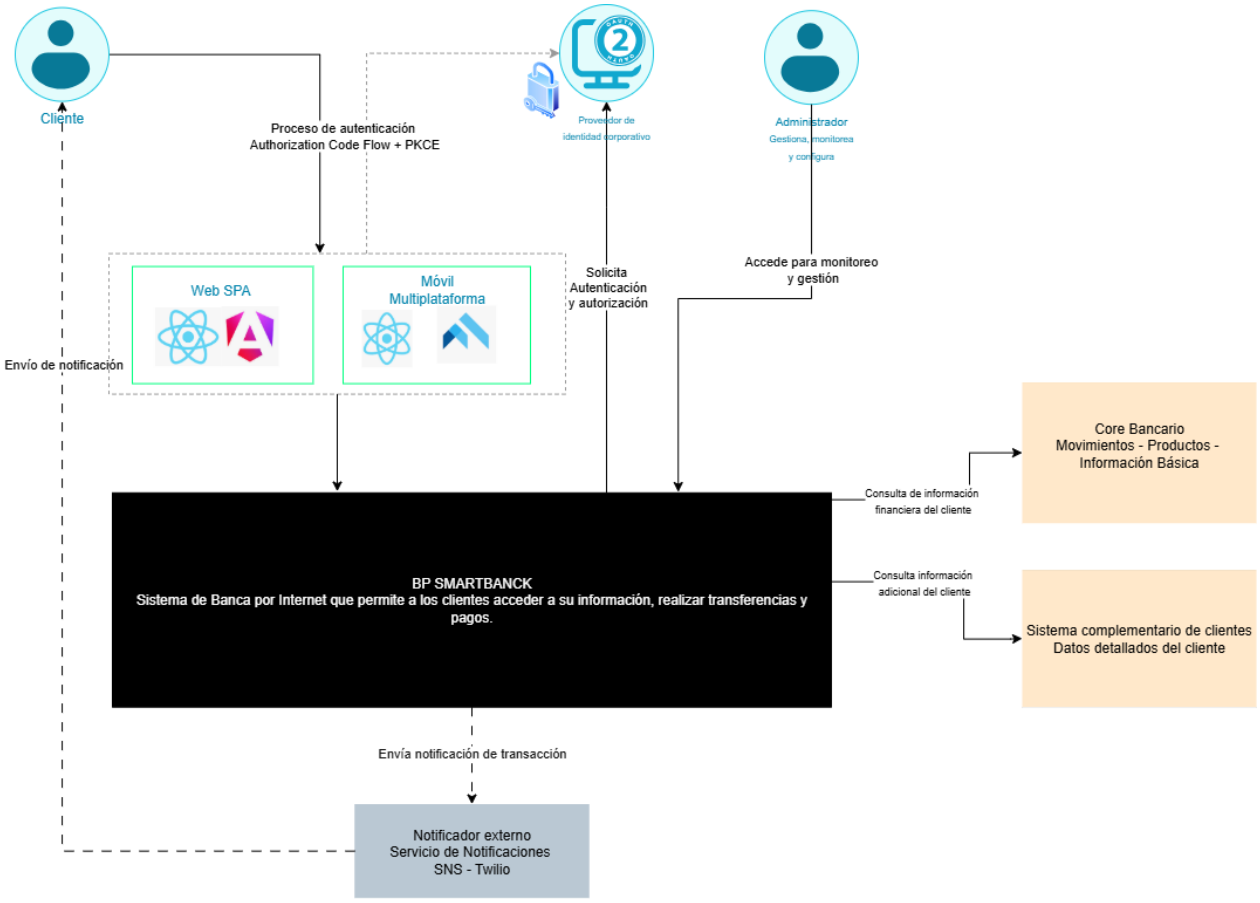
- Alta disponibilidad mediante balanceadores y zonas múltiples.
- Escalabilidad horizontal con ECS Fargate y bases de datos en RDS Multi-AZ.
- Recuperación ante desastres con backups automatizados y políticas de DR.
- Monitoreo con Amazon CloudWatch, X-Ray y alertas con SNS/SQS.
- Uso eficiente de instancias bajo demanda y serverless (Lambda) para ciertas funciones.

4. Justificaciones normativas y técnicas globales

- Normativa de protección de datos (LOPDP, Ecuador)
- Se garantiza la privacidad mediante la autenticación segura (OAuth2), gestión centralizada de identidades y uso de flujos recomendados (PKCE).
- Principio de desacoplamiento y escalabilidad (Clean Architecture + C4 Model)
- Cada sistema externo está desacoplado del sistema principal, facilitando su mantenimiento, reemplazo o ampliación.
- Uso de patrones recomendados en industria
- Autenticación: Authorization Code Flow (PKCE)
- Integración: API Gateway + servicios desacoplados
- Notificación: patrón event-driven (SNS/SQS)
- Compatibilidad multicanal
- Se permiten accesos desde móviles y web usando interfaces modernas, seguras y reactivas, promoviendo UX unificada.

5. Diagrama de Contexto

El diagrama de contexto presenta una vista general para usuarios no técnicos sobre los actores y sistemas que interactúan con el sistema BP SmartBank.

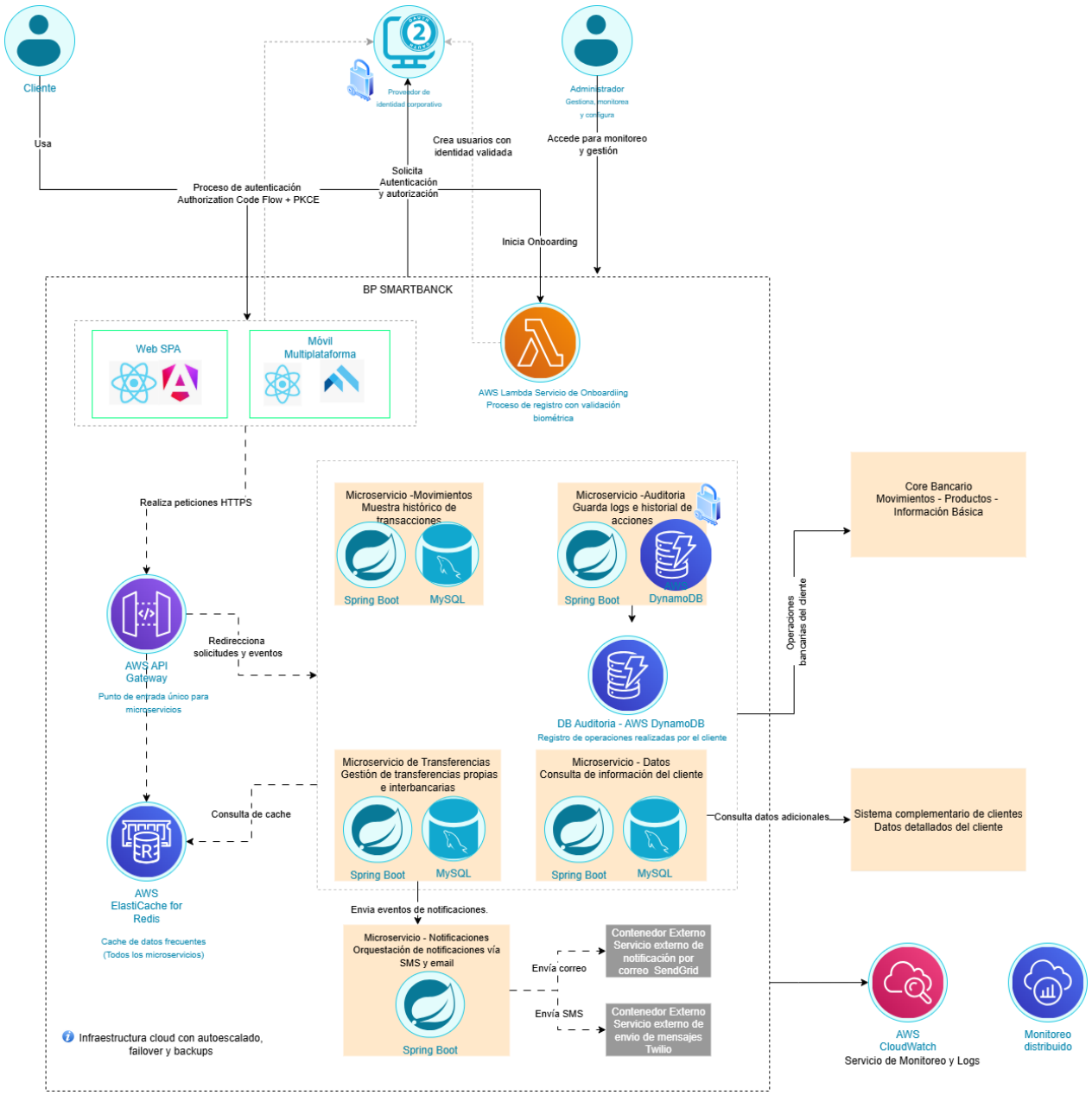


5.1. Justificación

Elemento	Justificación
Cliente	Actor principal del sistema. Puede consultar movimientos, realizar transferencias y recibir notificaciones.
Web SPA / Móvil Multiplataforma	Se eligieron estas dos tecnologías por:
	ReactJS + Angular para SPA (madurez, rendimiento y comunidad).
	React Native / Flutter para app móvil (cross-platform, reducción de tiempos y costos).
Proveedor de identidad (OAuth2)	Autenticación gestionada externamente para mejorar la seguridad y simplificar el flujo. Se implementa el flujo **Authorization Code Flow con PKCE**, recomendado por la RFC 7636 para aplicaciones móviles y SPA.
Administrador	Usuario interno con capacidad para monitorear el sistema, auditar accesos, revisar logs, configurar parámetros o servicios.
BP SmartBank	Sistema núcleo diseñado. Actúa como orquestador de funcionalidades bancarias, integrando fuentes de datos y servicios de terceros.
Core bancario	Sistema legado que contiene los movimientos, productos y saldos. Se accede en tiempo real para obtener información transaccional.
Sistema complementario	Almacena datos detallados del cliente no incluidos en el core (e.g., ocupación, documentos, perfil de riesgo).
Servicio de notificaciones (SNS / Twilio)	Se requieren **mínimo dos servicios de notificación** (según la normativa). Aquí se definen como servicios desacoplados para asegurar entrega confiable y múltiples canales (SMS, email, push).

6. Diagrama de Contenedores

Representa los diferentes contenedores como la SPA, aplicación móvil, API Gateway, microservicios, bases de datos, motores de cache y servicios externos como OAuth y notificaciones.



6.1. Justificación

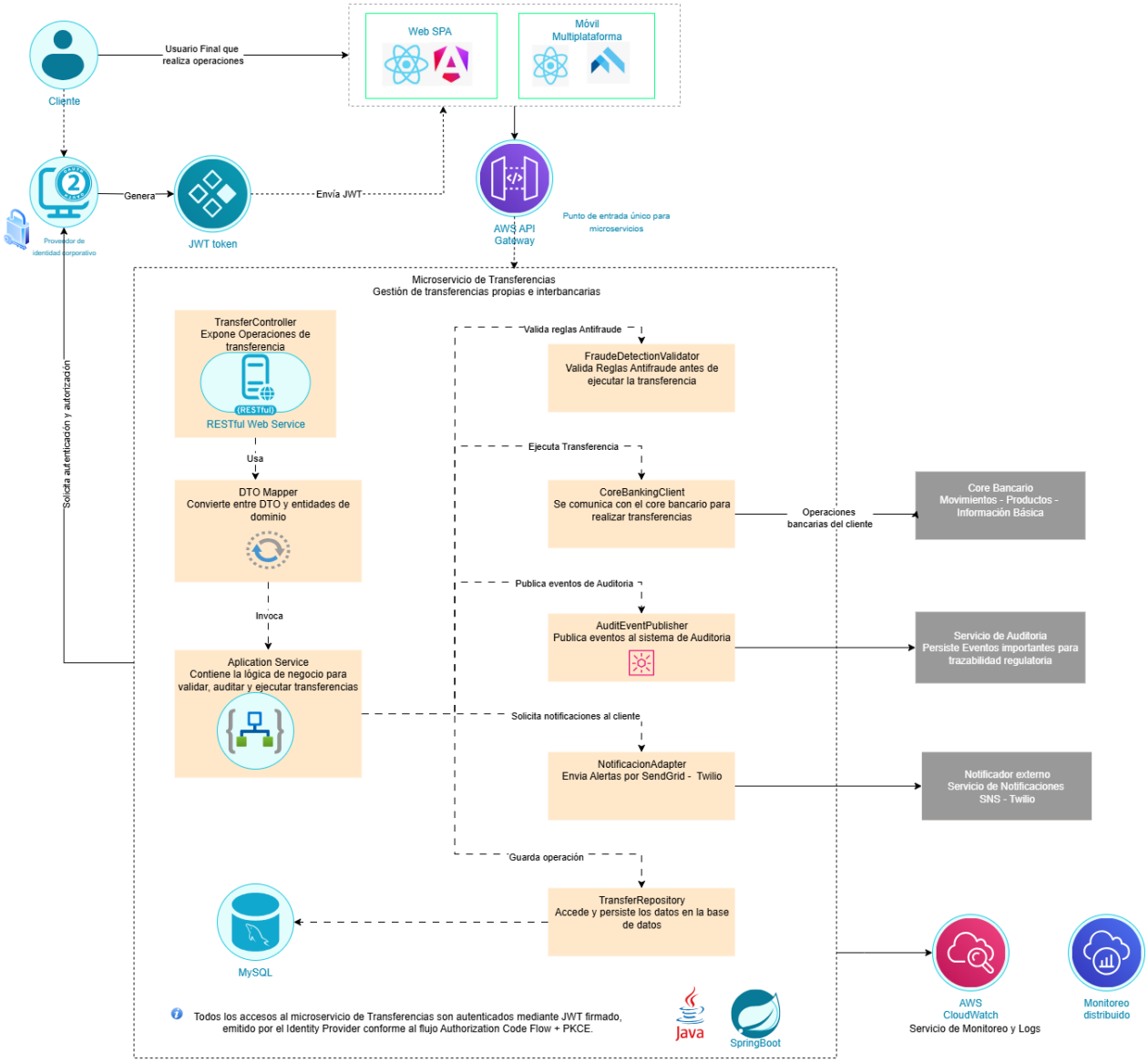
Contenedor	Rol/Función Principal	Tecnología/Framework	Interacciones Clave
Web SPA	Interfaz web para acceso a funcionalidades bancarias	Angular / React	Consume API Gateway, utiliza OAuth2 + PKCE para autenticación, recibe notificaciones
Aplicación Móvil	Interfaz móvil multiplataforma	React Native / Flutter	Consume API Gateway, utiliza OAuth2 + PKCE, inicia Onboarding, recibe notificaciones
API Gateway	Punto único de entrada, enruta solicitudes a microservicios	AWS API Gateway	Redirecciona peticiones HTTPS a microservicios, protege mediante autenticación
Microservicio de Transferencias	Maneja pagos propios e interbancarios	Spring Boot + MySQL	Consume servicios del Core bancario, publica eventos a SNS/SQS
Microservicio de Movimientos	Provee histórico de transacciones	Spring Boot + MySQL	Consulta al Core, usa Redis como cache, responde a clientes
Microservicio de Datos del Cliente	Consulta datos básicos y complementarios del cliente	Spring Boot + MySQL	Consume Core y sistema externo de clientes
Microservicio de Auditoría	Guarda logs y eventos relevantes de la actividad del cliente	Spring Boot + AWS DynamoDB	Persiste logs, integra con CloudWatch, puede enviar alertas
Microservicio de Notificaciones	Orquestador de envíos de notificaciones por SMS y correo	Spring Boot	Recibe eventos, envía mensajes a Twilio y SendGrid
Servicio de Onboarding	Registro de nuevos clientes mediante biometría facial	AWS Lambda	Valida identidad y crea cuenta de acceso
Base de Datos Auditoría	Almacena eventos de auditoría y acciones del usuario	AWS DynamoDB + S3	Persistencia temporal y almacenamiento de largo plazo
Servicio de Cache Compartido	Mejora tiempos de respuesta para consultas frecuentes	AWS ElastiCache (Redis)	Cachea resultados de los microservicios, especialmente movimientos

Notificador Externo (Correo)	Envía notificaciones transaccionales por correo	SendGrid	Usado por el microservicio de notificaciones
Notificador Externo (SMS)	Envía notificaciones por mensajes de texto	Twilio	Usado por el microservicio de notificaciones
Core Bancario	Sistema transaccional principal con información del cliente y sus productos	Sistema externo	Fuente de verdad para transferencias, movimientos, productos
Sistema Complementario de Cliente	Base de datos con datos detallados del cliente	Sistema externo	Consultado para enriquecer la información del cliente
Monitoreo y Logs	Observabilidad, alertas y métricas	AWS CloudWatch + Sistema externo	Recibe logs y métricas de todos los microservicios

7. Diagramas de Componentes

Se diseñan componentes para los siguientes microservicios clave:

7.1. Componente de Transferencias



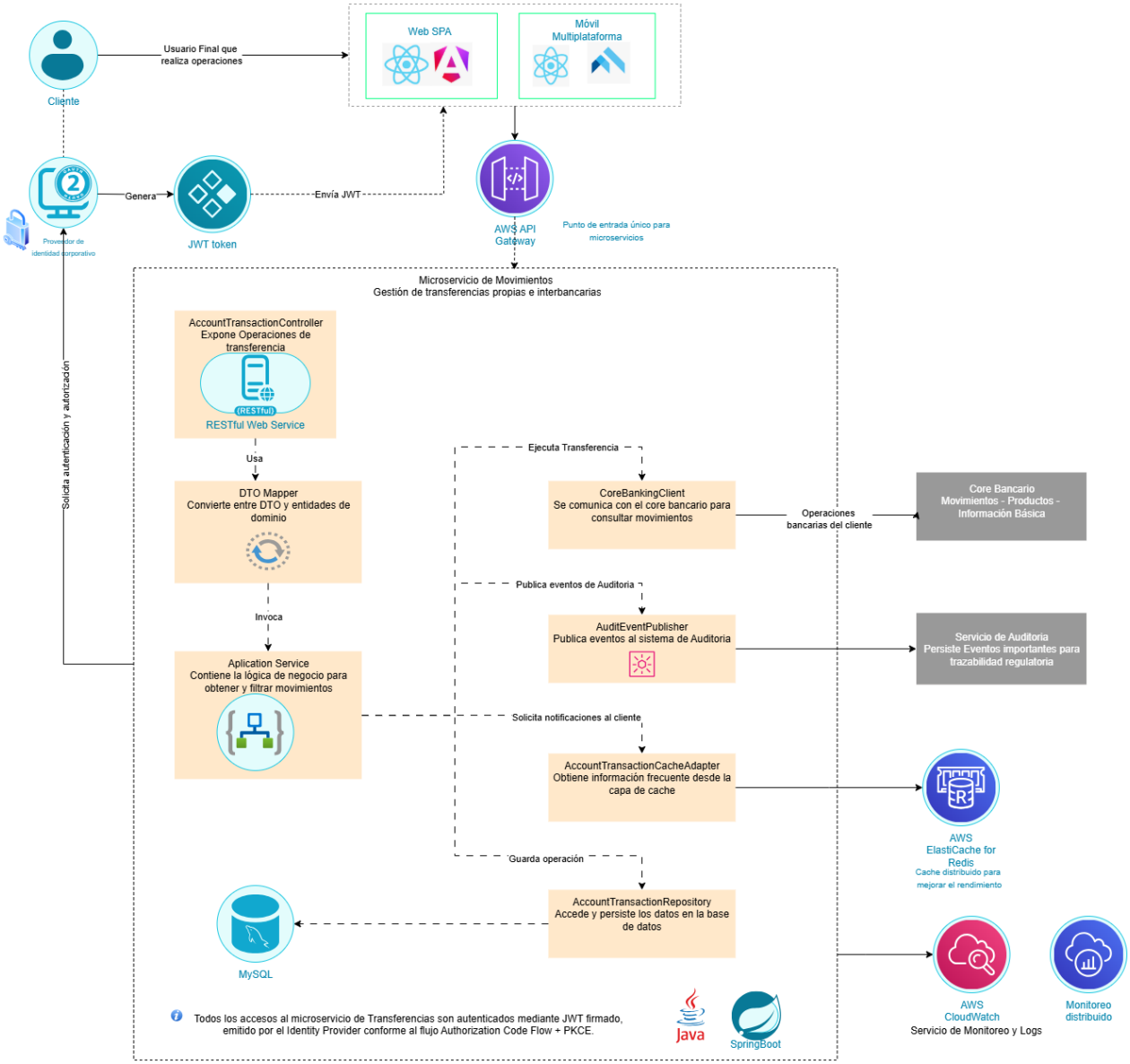
7.1.1. Justificación

Componente	Rol	Justificación Arquitectónica y Técnica
TransferController	Expone las operaciones REST de transferencia	Adaptador de entrada que expone una API REST para consumo externo. Permite desacoplar la lógica del dominio de la interfaz HTTP. Facilita pruebas e integración con canales externos.
DTO Mapper	Convierte entre DTOs y entidades de dominio	Aplica el patrón DTO (Data Transfer Object) y el principio de segregación de responsabilidades (SRP). Aísla el modelo de dominio de formatos de transporte (API).
Application Service	Orquesta la lógica de negocio y coordina acciones	Capa central de dominio de la arquitectura hexagonal. No depende de infraestructura y permite una lógica de negocio cohesionada, mantenible y testeable.
FraudeDetectionValidator	Valida reglas antifraude antes de ejecutar la transferencia	Aplica el principio de seguridad por diseño (Security by Design). Desacopla la lógica antifraude, permitiendo evolución independiente. Mejora la gobernanza regulatoria.
CoreBankingClient	Cliente que consume servicios del Core Bancario para ejecutar transferencias	Actúa como adaptador de salida. Facilita el desacoplamiento mediante interfaces, cumple SRP y permite reemplazar el Core sin afectar la lógica interna.
AuditEventPublisher	Publica eventos de auditoría a un sistema externo	Sigue el patrón Event-Driven. Alineado con LOPDP (Ley Orgánica de Protección de Datos Personales) e ISO/IEC 27001 para trazabilidad. Incrementa el cumplimiento regulatorio y desacopla la auditoría del dominio.

NotificationAdapter	Envia notificaciones usando servicios externos (SendGrid, Twilio)	Permite cumplimiento normativo (informar al cliente) y desacopla el envío mediante integración con múltiples canales. Facilita HA al permitir fallback entre canales.
TransferRepository	Accede y persiste operaciones de transferencia	Aplica el patrón de repositorio para mantener el dominio persistente sin acoplarlo a tecnologías. Se integra con MySQL por consistencia transaccional y soporte de ACID.
MySQL	Base de datos transaccional	Garantiza consistencia fuerte en operaciones financieras críticas. Soporte ACID, replicación y compatibilidad con patrones CQRS.
AWS API Gateway	Punto de entrada único para clientes	Protege los microservicios internos, gestiona seguridad, throttling y redirección. Compatible con JWT y OAuth2. Mejora el rendimiento y la observabilidad.
AWS CloudWatch	Servicio de monitoreo de logs y métricas	Garantiza trazabilidad operativa, detección de fallos y cumplimiento de normas como NIST SP 800-53. Aporta excelencia operativa según el AWS Well-Architected Framework.
Monitoreo distribuido	Observabilidad integral a través de microservicios	Apoya diagnósticos proactivos, análisis de causa raíz y cumplimiento de SLA. Esencial para resiliencia.
Notificador Externo (SMS)	Envía notificaciones por mensajes de texto	Twilio

Core Bancario	Sistema transaccional principal con información del cliente y sus productos	Sistema externo
Sistema Complementario de Cliente	Base de datos con datos detallados del cliente	Sistema externo
Monitoreo y Logs	Observabilidad, alertas y métricas	AWS CloudWatch + Sistema externo

7.2. Componente de Movimientos



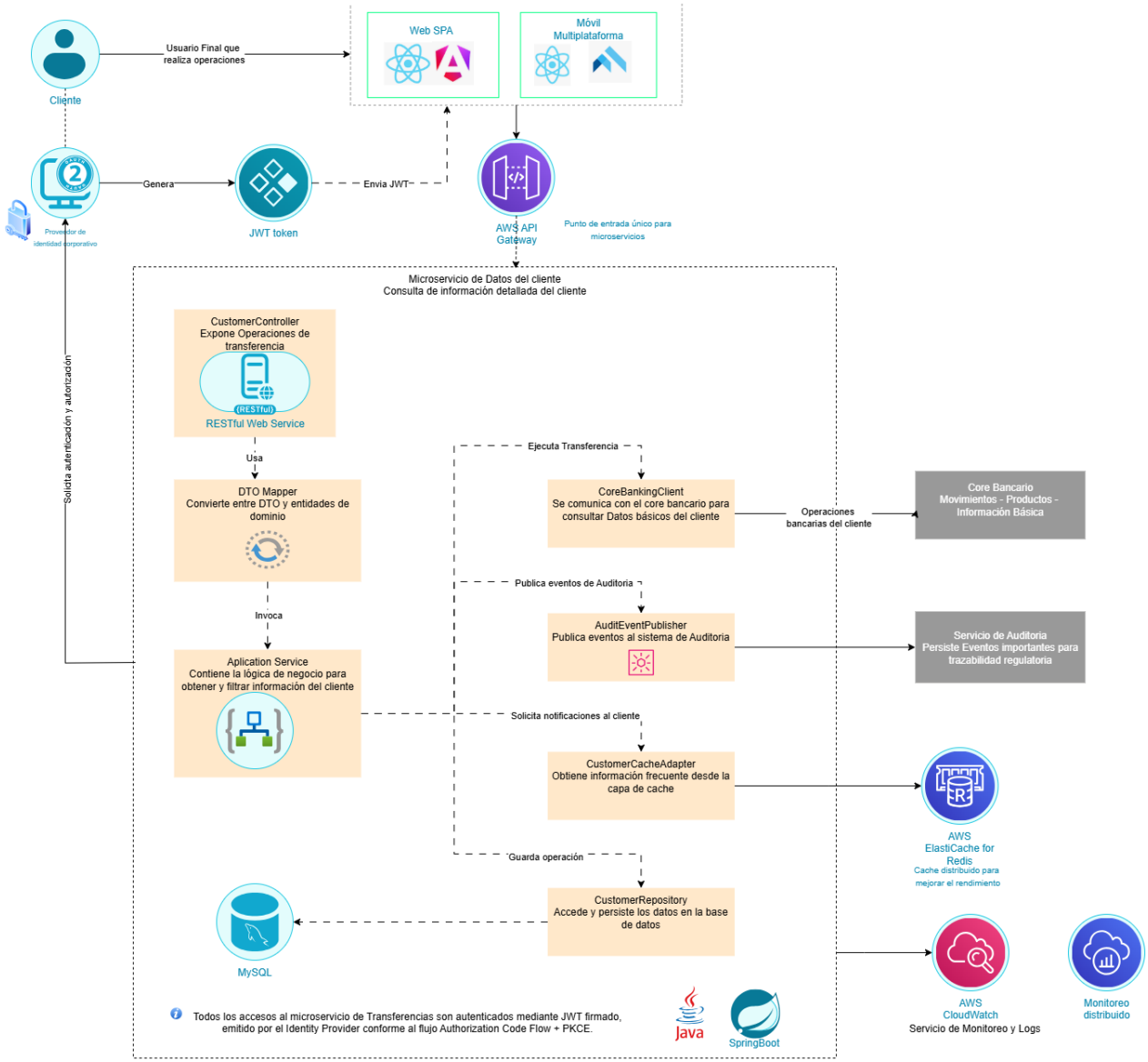
7.2.1. Justificación

Componente	Rol	Justificación Arquitectónica y Técnica
AccountTransactionController	Expone operaciones de consulta de movimientos	Adaptador de entrada que expone una API REST para consumo externo. Permite desacoplar la lógica del dominio de la interfaz HTTP. Facilita pruebas e integración con canales externos.
DTO Mapper	Convierte entre DTO y entidades de dominio	Aplica el patrón DTO (Data Transfer Object) y el principio de segregación de responsabilidades (SRP). Aísla el modelo de dominio de formatos de transporte (API).
Application Service	Orquesta la lógica de negocio para consultar y filtrar movimientos	Capa central de dominio de la arquitectura hexagonal. No depende de infraestructura y permite una lógica de negocio cohesionada, mantenible y testeable.
CoreBankingClient	Se comunica con el Core Bancario para consultar movimientos	Adaptador de salida (driven adapter) que permite abstraer la fuente externa. Favorece el desacoplamiento y facilita pruebas con mocks o stubs.
AuditEventPublisher	Publica eventos de auditoría	Sigue el patrón de arquitectura orientada a eventos (Event-Driven). Se alinea con los requisitos normativos de trazabilidad (ISO/IEC 27001, LOPDP), y desacopla el dominio del subsistema de auditoría.
AccountTransactionCacheAdapter	Obtiene información frecuente desde Redis	Mejora el rendimiento usando caching para transacciones frecuentes. Sigue el principio de separación de preocupaciones (SoC) y desacopla el dominio del mecanismo de almacenamiento en caché.

AccountTransactionRepository	Accede y persiste datos en la base de datos	Aplica el patrón Repositorio para abstraer el acceso a datos. Permite pruebas más sencillas, favorece el aislamiento del dominio y soporta una capa relacional transaccional con MySQL.
MySQL	Base de datos relacional	Uso justificado por requerimientos de consistencia, atomicidad (ACID) y confiabilidad en datos financieros. Complementa al caching en Redis para datos persistentes.
AWS API Gateway	Punto único de entrada para el sistema	Gestiona seguridad, routing, throttling y visibilidad. Compatible con autenticación basada en JWT y flujos OAuth2 (PKCE), protegiendo los microservicios internos.
ElastiCache for Redis	Provee caching distribuido	Mejora tiempos de respuesta. Se integra como cache adapter, lo que permite aplicar el patrón de separación de infraestructura, cumpliendo principios de diseño hexagonal.
AWS CloudWatch + Monitoreo	Recolección de logs, métricas y trazabilidad	Habilita observabilidad completa. Es un requisito clave del AWS Well-Architected Framework (Pilar de Excelencia Operacional). También apoya en cumplimiento normativo (auditoría y recuperación ante fallos).
Monitoreo distribuido	Observabilidad integral a través de microservicios	Apoya diagnósticos proactivos, análisis de causa raíz y cumplimiento de SLA. Esencial para resiliencia.
Notificador Externo (SMS)	Envía notificaciones por mensajes de texto	Twilio

Core Bancario	Sistema transaccional principal con información del cliente y sus productos	Sistema externo
Sistema Complementario de Cliente	Base de datos con datos detallados del cliente	Sistema externo
Monitoreo y Logs	Observabilidad, alertas y métricas	AWS CloudWatch + Sistema externo

7.3. Componente Datos del Cliente

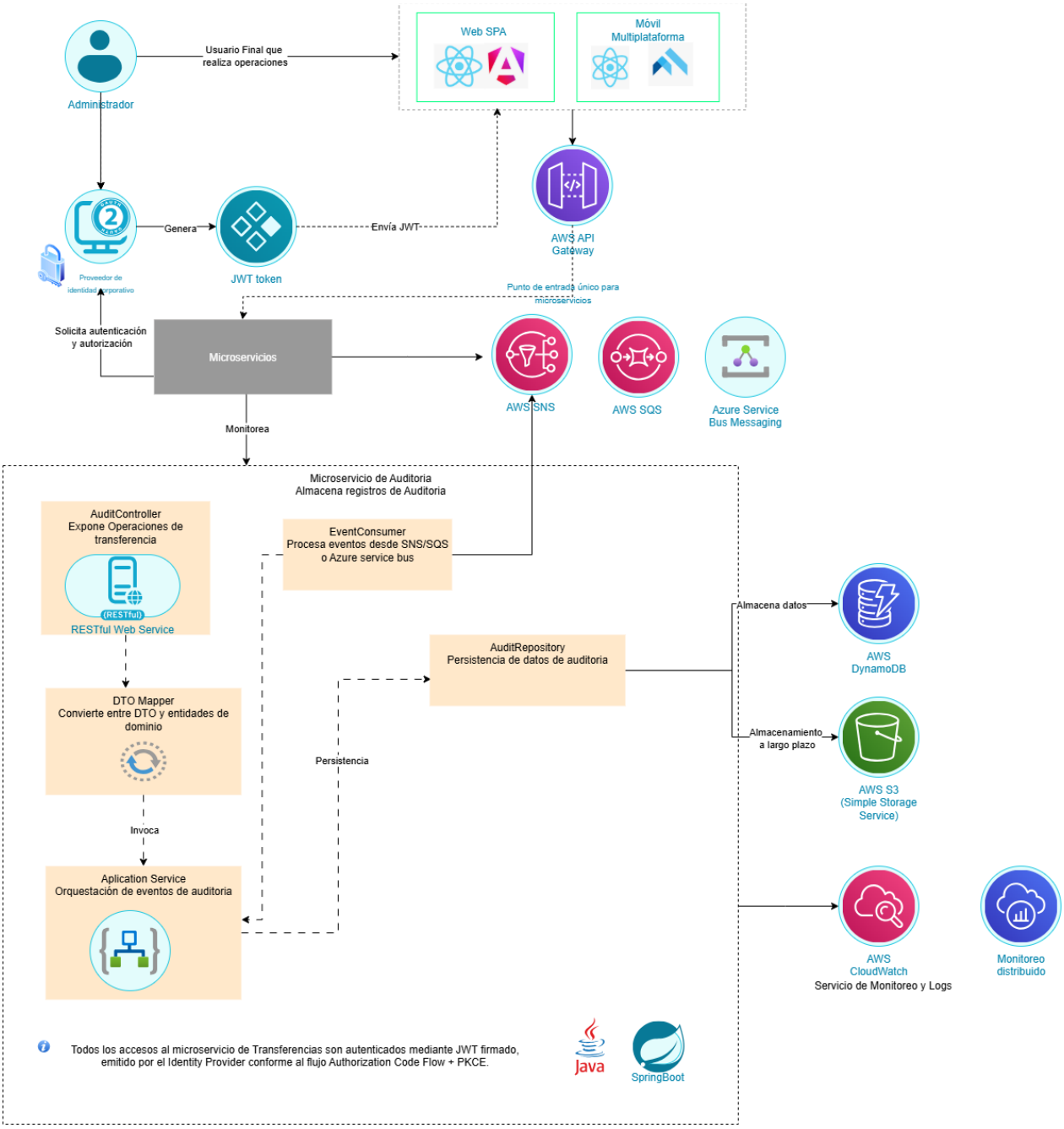


7.3.1. Justificación

Componente	Rol	Justificación Arquitectónica y Técnica
CustomerController	Exposición de operaciones de consulta de información del cliente	Adaptador de entrada que expone una API REST para consumo externo. Permite desacoplar la lógica del dominio de la interfaz HTTP. Facilita pruebas e integración con canales externos.
DTO Mapper	Conversión entre DTOs y modelos de dominio	Aplica el patrón DTO (Data Transfer Object) y el principio de segregación de responsabilidades (SRP). Aísla el modelo de dominio de formatos de transporte (API).
Application Service	Orquesta lógica de negocio para consulta de información del cliente	Núcleo del dominio según la arquitectura hexagonal. Aquí se encapsula la lógica para recuperar y combinar datos desde cache, base de datos y sistemas externos.
CoreBankingClient	Cliente para obtener datos básicos del cliente desde el Core Bancario	Actúa como adaptador de salida. Facilita el desacoplamiento mediante interfaces, cumple SRP y permite reemplazar el Core sin afectar la lógica interna.
AuditEventPublisher	Publicador de eventos al sistema de auditoría	Componente orientado a eventos. Permite cumplir con normativas como LOPDP y ISO/IEC 27001 al garantizar trazabilidad de consultas y accesos.
CustomerCacheAdapter	Obtiene datos frecuentes desde Redis	Mejora la velocidad de respuesta para datos accedidos frecuentemente. Se integra a través de un adaptador, manteniendo el desacoplamiento con la infraestructura.

CustomerRepository	Acceso y persistencia a base de datos MySQL	Aplica el patrón Repositorio. Separa el dominio del acceso a datos, facilitando el uso de MySQL como motor relacional confiable para operaciones críticas.
MySQL	Almacenamiento de datos persistentes	Ideal para mantener consistencia de datos personales. Cumple con requisitos ACID para información regulada.
AWS API Gateway	Punto único de entrada a los servicios	Asegura el acceso centralizado y controlado a través de HTTPS, autenticación JWT y control de tráfico.
AWS ElastiCache (Redis)	Cache distribuido	Reutilizado por todos los microservicios. Reduce la carga al sistema core y mejora la eficiencia general del sistema.
AWS CloudWatch + Logs	Observabilidad, monitoreo y alertas	Asegura diagnóstico en tiempo real y auditoría. Permite cumplir principios de Excelencia Operacional (AWS Well-Architected Framework).

7.4. Componente Auditoría

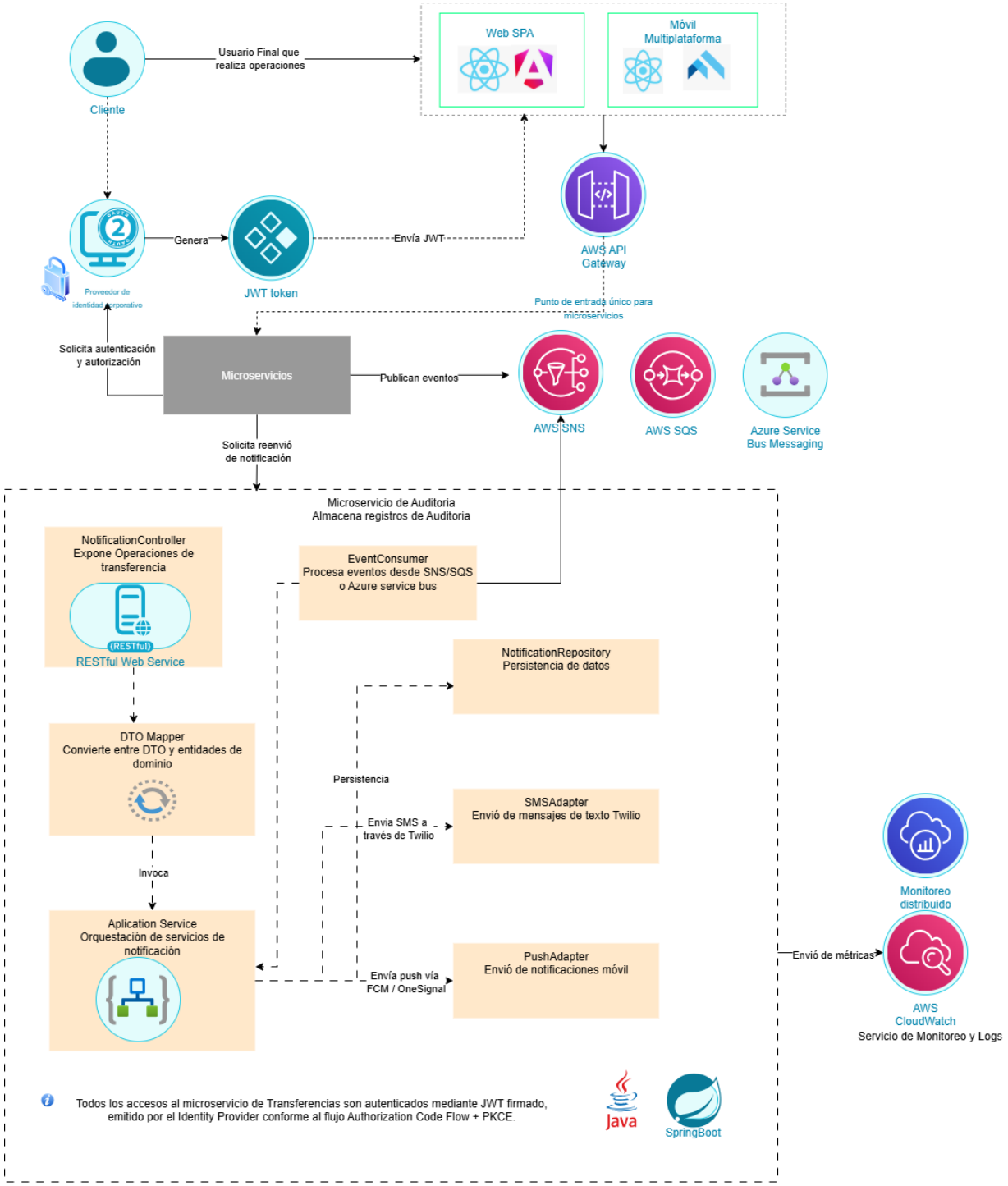


7.4.1. Justificación

Componente	Rol	Justificación Arquitectónica y Técnica
AuditController	Exposición de operaciones de auditoría	Adaptador de entrada que permite consultar eventos auditados. Sigue el principio de inversión de dependencias al separar el acceso HTTP del núcleo del dominio.
DTO Mapper	Conversión entre DTO y entidades de dominio	Permite separar el modelo de transporte del modelo de negocio. Es clave en integración con canales, validación de datos y control de seguridad de entrada.
Application Service	Orquesta eventos y lógica de persistencia de auditoría	Núcleo del dominio que decide cómo manejar los eventos entrantes, aplicar reglas de negocio y persistirlos.
EventConsumer	Procesador de eventos provenientes de SNS/SQS o Azure Service Bus	Adaptador de entrada basado en eventos. Permite desacoplar la publicación desde otros microservicios. Implementa arquitectura event-driven, garantizando trazabilidad y resiliencia.
AuditRepository	Accede a las bases de datos de auditoría	Adaptador de salida para persistencia. Utiliza el patrón Repository para aislar la lógica de almacenamiento (NoSQL + almacenamiento de objetos).
AWS DynamoDB	Almacenamiento estructurado de auditoría en NoSQL	Ideal para lecturas rápidas, escalabilidad horizontal y alta disponibilidad. Garantiza cumplimiento de ISO/IEC 27001 para disponibilidad e integridad.

AWS S3	Almacenamiento de logs a largo plazo	Recomendado para datos históricos, archivos de auditoría pesados o exportables, y cumplimiento de retención legal (LOPDP).
AWS CloudWatch	Observabilidad y monitoreo continuo	Recolecta logs, métricas y alertas sobre los eventos procesados. Alineado con el pilar de Excelencia Operacional del AWS Well-Architected Framework.
SNS / SQS	Orquestación asincrónica de eventos	Desacopla microservicios productores y consumidores. Asegura entrega confiable y evita bloqueos. Mejora la escalabilidad del sistema.

7.5. Componente Notificaciones

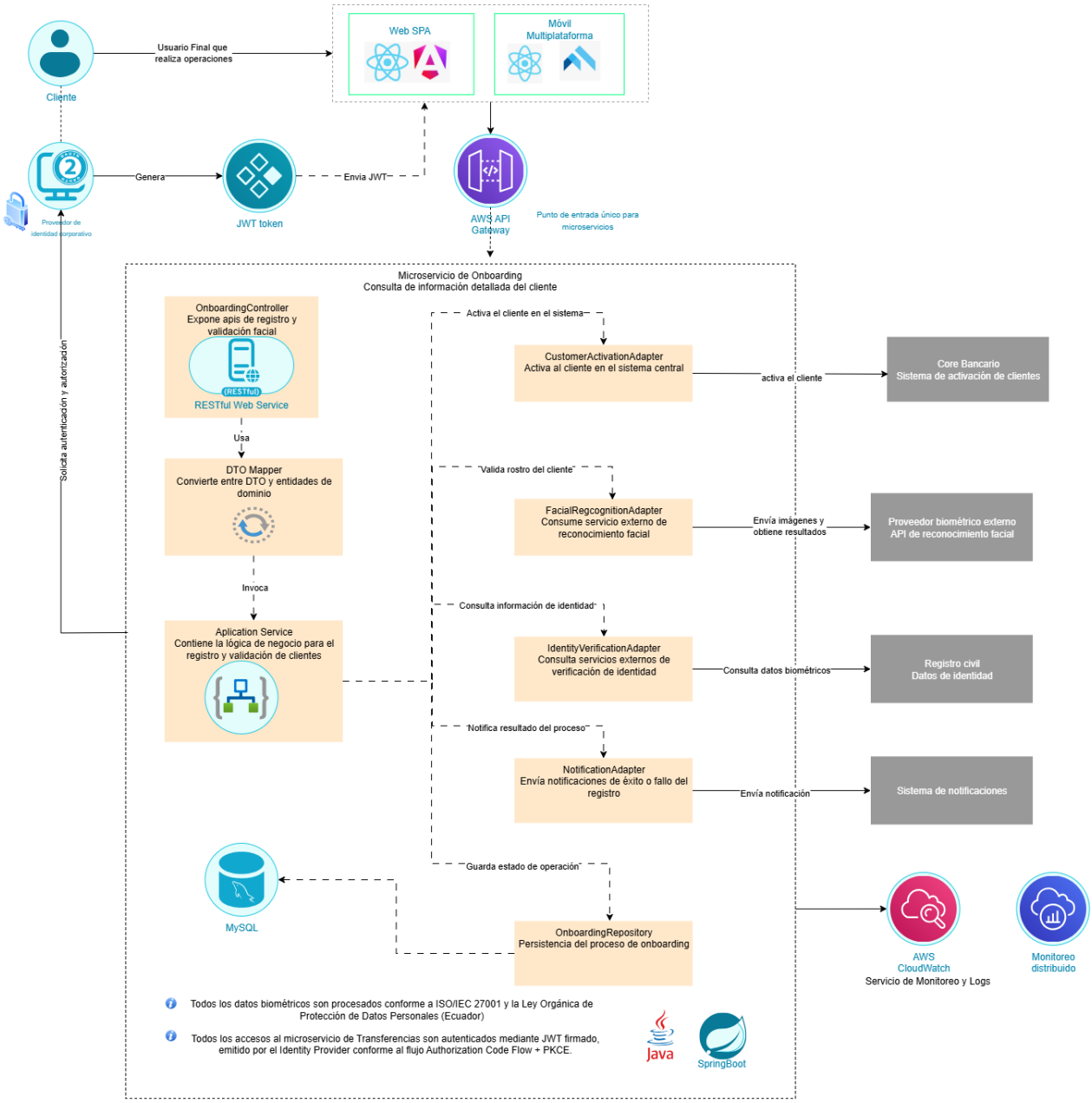


7.5.1. Justificación

Componente	Rol	Justificación Arquitectónica y Técnica
NotificationController	Expone servicios para reenvío y prueba de notificaciones	Adaptador de entrada que permite consultar eventos auditados. Sigue el principio de inversión de dependencias al separar el acceso HTTP del núcleo del dominio.
DTO Mapper	Conversión entre DTO y entidades	Permite separar el modelo de transporte del modelo de negocio. Es clave en integración con canales, validación de datos y control de seguridad de entrada.
Application Service	Orquesta el flujo de envío de notificaciones	Contiene la lógica del dominio de notificaciones, gestionando eventos, validaciones, y enrutamiento hacia los adaptadores correctos.
EventConsumer	Procesa eventos desde SNS/SQS o Azure Service Bus	Adaptador de entrada basado en eventos. Permite recibir mensajes asincrónicos de otros microservicios para generar notificaciones automáticas.
NotificationRepository	Persiste el historial de notificaciones enviadas	Implementa un patrón Repository que abstrae la persistencia, permitiendo auditar o reenviar notificaciones desde base de datos si es necesario.
SMSAdapter	Envía mensajes SMS utilizando Twilio	Adaptador de salida (puerto secundario). Abstrae el servicio externo Twilio. Facilita el cambio a otro proveedor sin afectar la lógica del dominio.

PushAdapter	Envía notificaciones móviles mediante FCM / OneSignal	Otro adaptador de salida. Maneja la lógica de integración con plataformas de notificaciones móviles. Permite escalabilidad hacia distintos dispositivos.
AWS SNS / SQS	Transporte de eventos desde otros microservicios	Proporciona resiliencia y desacoplamiento. Alineado al patrón event-driven. Permite reintentos y gestión de colas en caso de fallos de entrega.
AWS CloudWatch	Observabilidad y monitoreo	Reporta métricas, fallos y trazas del envío de notificaciones. Requisito esencial para sistemas de misión crítica según el pilar de "Operational Excellence" del AWS Well-Architected Framework.

7.6. Componente Onboarding



7.6.1. Justificación

Componente	Rol en la Arquitectura	Justificación Técnica
OnboardingController	Expone operaciones HTTP para iniciar y consultar el estado del proceso de registro.	Adaptador de entrada que permite consultar eventos auditados. Sigue el principio de inversión de dependencias al separar el acceso HTTP del núcleo del dominio.
DTO Mapper	Convierte objetos de entrada en entidades de dominio.	Permite separar el modelo de transporte del modelo de negocio. Es clave en integración con canales, validación de datos y control de seguridad de entrada.
Application Service	Orquesta la lógica de negocio del proceso de onboarding: verificación, validación biométrica y persistencia.	Encapsula la lógica de orquestación conforme a la arquitectura hexagonal, manteniendo desacoplada la lógica de infraestructura.
CustomerActivationAdapter	Activa el cliente en el Core bancario una vez finalizado el proceso.	Actúa como puerto de salida (Outbound Adapter) permitiendo la comunicación con sistemas legados sin acoplar la lógica del dominio.
FacialRecognitionAdapter	Consume el API biométrico externo para validar rostro.	Se ajusta al principio de inversión de dependencias; permite cambiar el proveedor biométrico sin impactar el dominio.
IdentityVerificationAdapter	Consulta datos de identidad con el registro civil u otro sistema gubernamental.	Garantiza validación robusta de identidad, cumpliendo con la **LOPDP (Ecuador)** y prácticas de seguridad recomendadas por ISO/IEC 27001.

NotificationAdapter	Envía alertas de éxito o fallo al cliente.	Desacopla el canal de salida (correo, push, SMS) usando un adaptador que orquesta desde el dominio. Facilita multicanalidad y reintentos.
OnboardingRepository	Persiste el estado del proceso de onboarding.	Implementado sobre MySQL con persistencia transaccional. Sigue el patrón **Repository**, manteniendo la lógica del dominio limpia y desacoplada de la infraestructura.
AWS CloudWatch + Logs	Monitoreo y trazabilidad del proceso de onboarding.	Permite observabilidad, trazabilidad de errores, y auditoría en cumplimiento de la **LOPDP** e ISO/IEC 27001.
MySQL	Almacena el estado del proceso de onboarding y metadatos.	Motor relacional que garantiza ACID y mantiene la trazabilidad completa del proceso.

8. Justificaciones de Diseño

- Patrón hexagonal: Favorece el desacoplamiento, testabilidad y mantenibilidad.
  Spring Boot: Permite construir servicios desacoplados, fácilmente testeables y desplegables en contenedores o FaaS. Compatible con eventos y adaptadores.
- Redis: Cache distribuido de baja latencia.
  AWS Fargate: Alta disponibilidad y escalabilidad.
  SNS + Lambda + EventBridge: Escenarios de eventos desacoplados y resilientes.
  Twilio/SendGrid: Canales de notificación confiables y multicanal.

9. . Aplicaciones Front-end

- Web SPA: Angular o React (modularidad, soporte de comunidad, rendimiento)
- Mobile: Flutter o React Native (desarrollo multiplataforma, comunidad, madurez)

10. . Arquitectura de Autenticación y Onboarding

- OAuth 2.0 con Authorization Code Flow (recomendado)
- Reconocimiento facial (integración desde app móvil como pre-requisito de acceso)
- MFA con OTP o biometría

11. . Persistencia para Clientes Frecuentes

- Uso de Redis y patrón Cache Aside
- Acceso rápido a información sensible o de uso recurrente

12. Solución de Auditoría

- Microservicio especializado con patrón Event-Driven
- Base de datos inmutable para trazabilidad y cumplimiento normativo

13. . HA, DR, Monitoreo y Auto-Healing

- AWS CloudWatch + Azure Monitor
- Auto-scaling con ECS/Fargate + Azure App Service

- Estrategias de DR multizona y backup automatizado

14. . Conclusión

La solución propuesta responde a los requerimientos funcionales, normativos y de calidad del ejercicio práctico. Cumple con buenas prácticas de diseño desacoplado, seguridad, integración y operación en la nube, permitiendo la escalabilidad futura del sistema.