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:

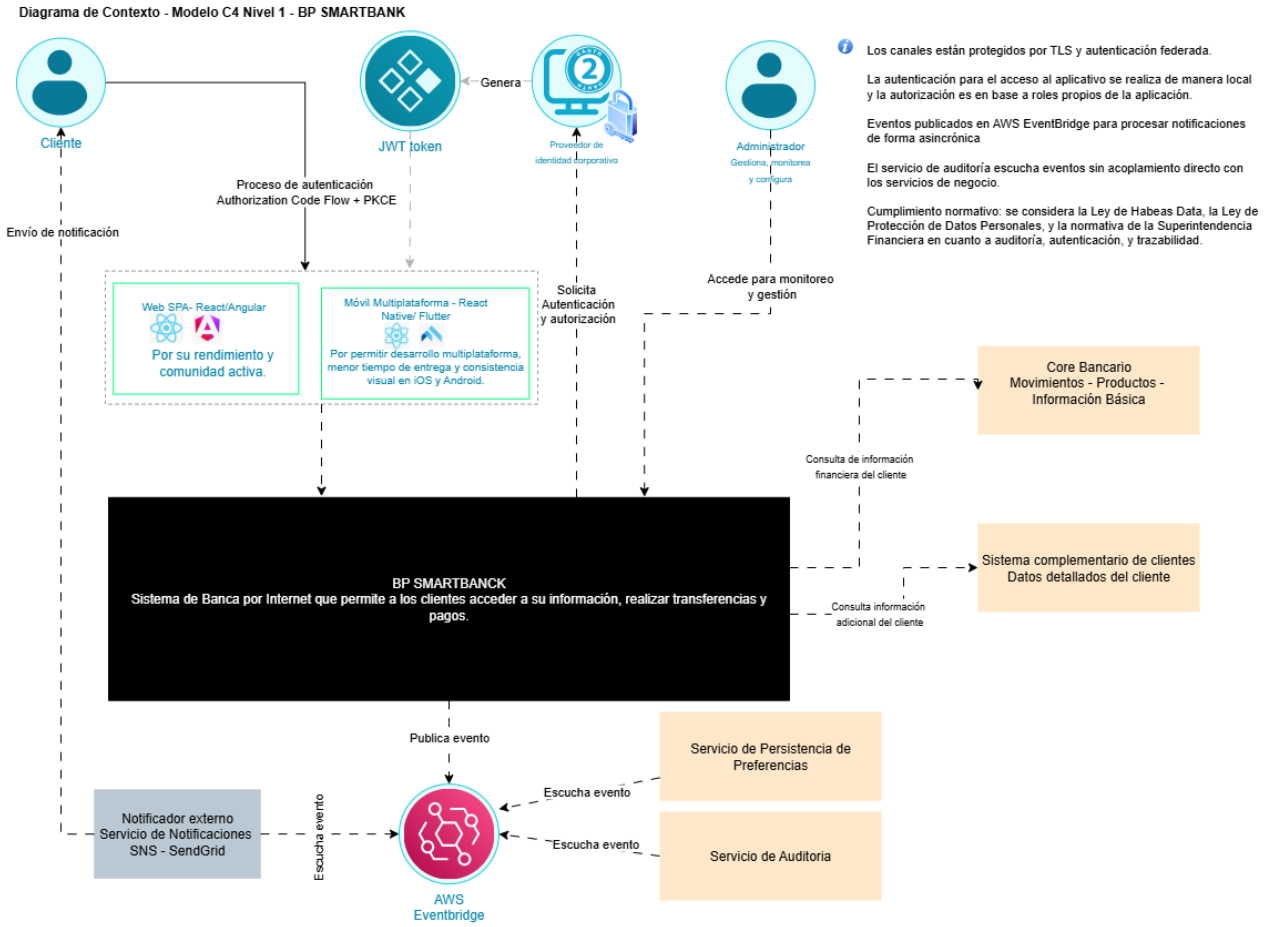
- Alta disponibilidad mediante balanceadores y zonas múltiples.
- Escalabilidad horizontal con ECS Fargate y bases de datos en RDS Multi-AZ.
- Recuperación ante desastres con backups automatizados y políticas de DR.
- Monitoreo con Amazon CloudWatch, X-Ray y alertas con SNS/SQS.
- Uso eficiente de instancias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



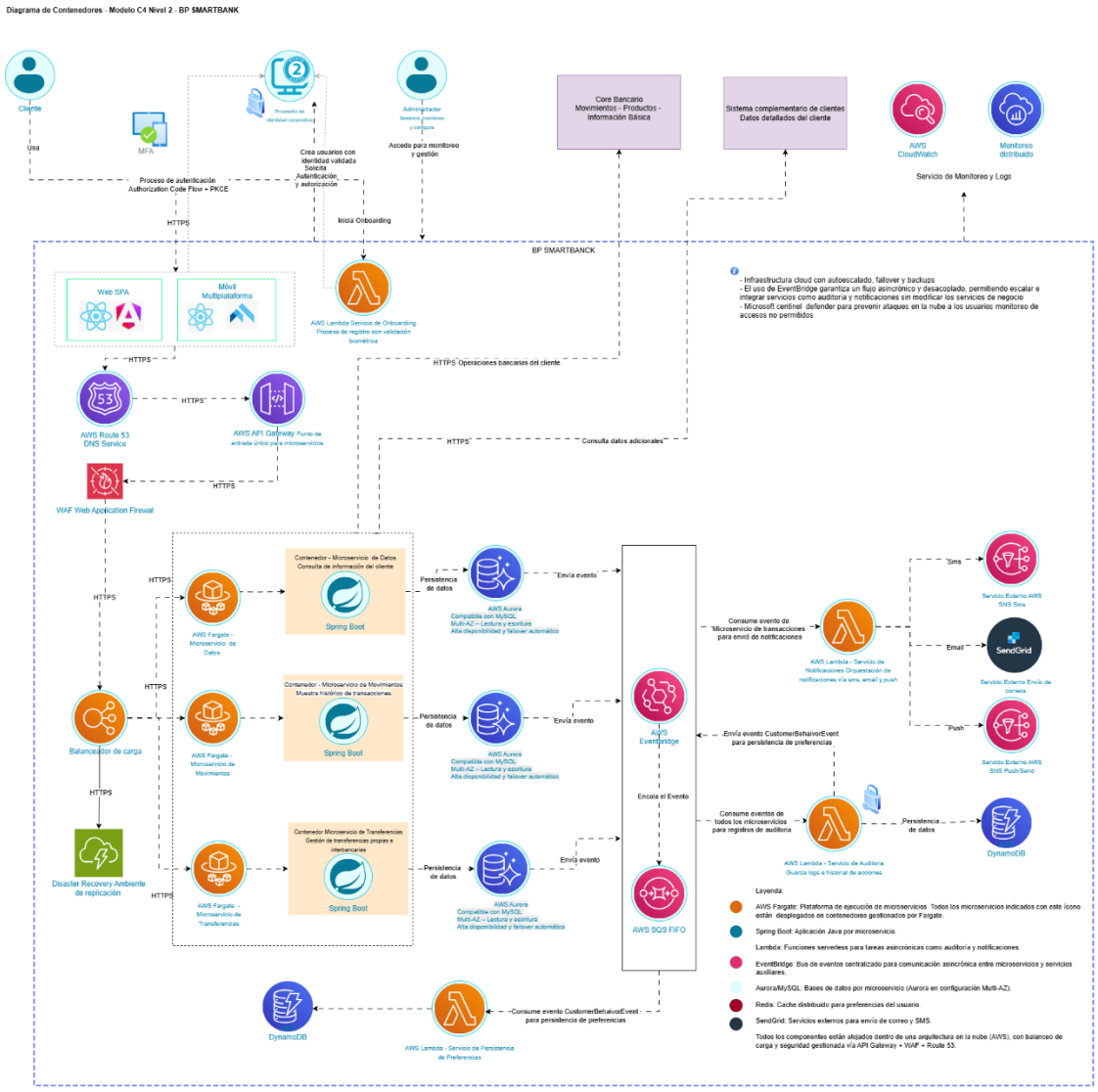
5.1. Elementos del diagrama de contexto y justificaciòn

Elemento	Justificación
Cliente	Actor principal del sistema. Puede consultar información, realizar transferencias y recibir notificaciones personalizadas.
Web SPA / Móvil Multiplataforma	Se eligieron estas tecnologías por su madurez y eficiencia: ReactJS / Angular para SPA: rendimiento, comunidad activa. React Native / Flutter para móvil: desarrollo multiplataforma, menor costo y entrega más rápida.
Proveedor de Identidad (OAuth2)	Gestiona la autenticación federada, mejora la seguridad. Se implementa el flujo Authorization Code Flow con PKCE, recomendado por la RFC 7636 para SPA y apps móviles.
Administrador	Usuario interno con acceso a funcionalidades de monitoreo, gestión de eventos, configuración de parámetros, y revisión de logs.
BP SmartBank	Sistema núcleo del canal digital. Orquesta operaciones de banca por internet (consultas, transferencias, pagos). Publica eventos a AWS EventBridge de forma desacoplada.
Core Bancario	Sistema transaccional que centraliza productos, saldos y movimientos del cliente. Se consulta en tiempo real para obtener datos financieros.
Sistema complementario de clientes	Provee datos detallados del cliente no disponibles en el core (e.g., perfil de riesgo, ocupación, documentos). Se consulta para enriquecer la experiencia del usuario.
Servicio de Notificaciones (SNS / SendGrid)	Microservicios desacoplados que envían mensajes vía email (SendGrid) o SMS (SNS). Se integran por medio de eventos, asegurando confiabilidad y cumplimiento normativo.

AWS EventBridge	Canal de eventos asincrónicos que desacopla a los productores (BP SmartBank) de los consumidores (Auditoría, Preferencias, Notificaciones). Mejora la escalabilidad y flexibilidad del sistema.
Servicio de Auditoría	Microservicio que consume eventos para registrar acciones del cliente. Soporta trazabilidad, control normativo (Habeas Data, Circular 052), y puede disparar eventos derivados.
Servicio de Persistencia de Preferencias	Consume eventos relevantes y persiste patrones o preferencias de usuario (operaciones frecuentes, horarios, canales). Facilita personalización sin acoplarse directamente al cliente.

6. Diagrama de Contenedores

El diagrama de contenedores (Modelo C2) representa los componentes internos del sistema BP SMARTBANK. Este sistema desacopla los distintos servicios siguiendo principios de microservicios y arquitectura hexagonal. Se aprovechan herramientas serverless para manejar eventos, y se garantiza escalabilidad mediante el uso de colas FIFO y funciones en la nube. La seguridad se maneja con OAuth2.0 y JWT, y se integra monitoreo, auditoría y recuperación ante fallos. La arquitectura está preparada para crecimiento, resiliencia y cumplimiento normativo.



6.1. Contenedores del Sistema y Detalle Tecnológico

Contenedor	Rol / Función Principal	Tecnología / Framework	Interacciones Clave / Justificación Técnica
Web SPA	Interfaz web para acceso de clientes	Angular / React	Consume API Gateway, autenticación OAuth2 + PKCE, recibe notificaciones. Se eligió SPA por su rendimiento y comunidad madura.
Aplicación Móvil	Interfaz móvil para clientes	React Native / Flutter	Consume API Gateway, autenticación OAuth2 + PKCE, inicia onboarding, recibe notificaciones. Se eligió multiplataforma para reducir tiempos y costos.
API Gateway	Punto único de entrada al sistema	AWS API Gateway	Enruta peticiones HTTPS a microservicios, autenticación OAuth2, validación de tráfico, integración con WAF.
Servicio de Onboarding	Validación de identidad y registro de nuevos clientes	AWS Lambda	Lógica de onboarding asincrónico, autenticación con biometría facial, crea credenciales.
Contenedor Movimientos	Expone historial de transacciones bancarias	Spring Boot + AWS Fargate + RDS	Consulta al Core bancario y Redis, publica eventos a EventBridge, expone APIs REST.

Contenedor Transferencias	Gestiona pagos propios e interbancarios	Spring Boot + AWS Fargate + RDS	Consulta al Core, publica eventos de notificación y auditoría.
Contenedor Datos del Cliente	Expone información básica y complementaria del cliente	Spring Boot + AWS Fargate + RDS	Consulta al Core y sistema complementario, expone APIs para autenticación y visualización.
Contenedor Auditoría	Registra y persiste eventos del sistema	AWS Lambda + DynamoDB + S3	Consume eventos desde EventBridge, persiste logs estructurados y documentos de auditoría.
Contenedor Notificaciones	Orquestador para envío de notificaciones al cliente	AWS Lambda	Consume eventos desde EventBridge, verifica preferencias, invoca SendGrid (email) o SNS/Twilio (SMS).
Servicio de Preferencias	Persiste y consulta las preferencias de notificación del cliente	DynamoDB	Consultado por el servicio de notificaciones para decidir canal y frecuencia.
Servicio de Cache	Optimiza rendimiento de consultas frecuentes (ej. movimientos)	Redis (AWS ElastiCache)	Consultado por el contenedor de movimientos para evitar consultas repetitivas al Core.

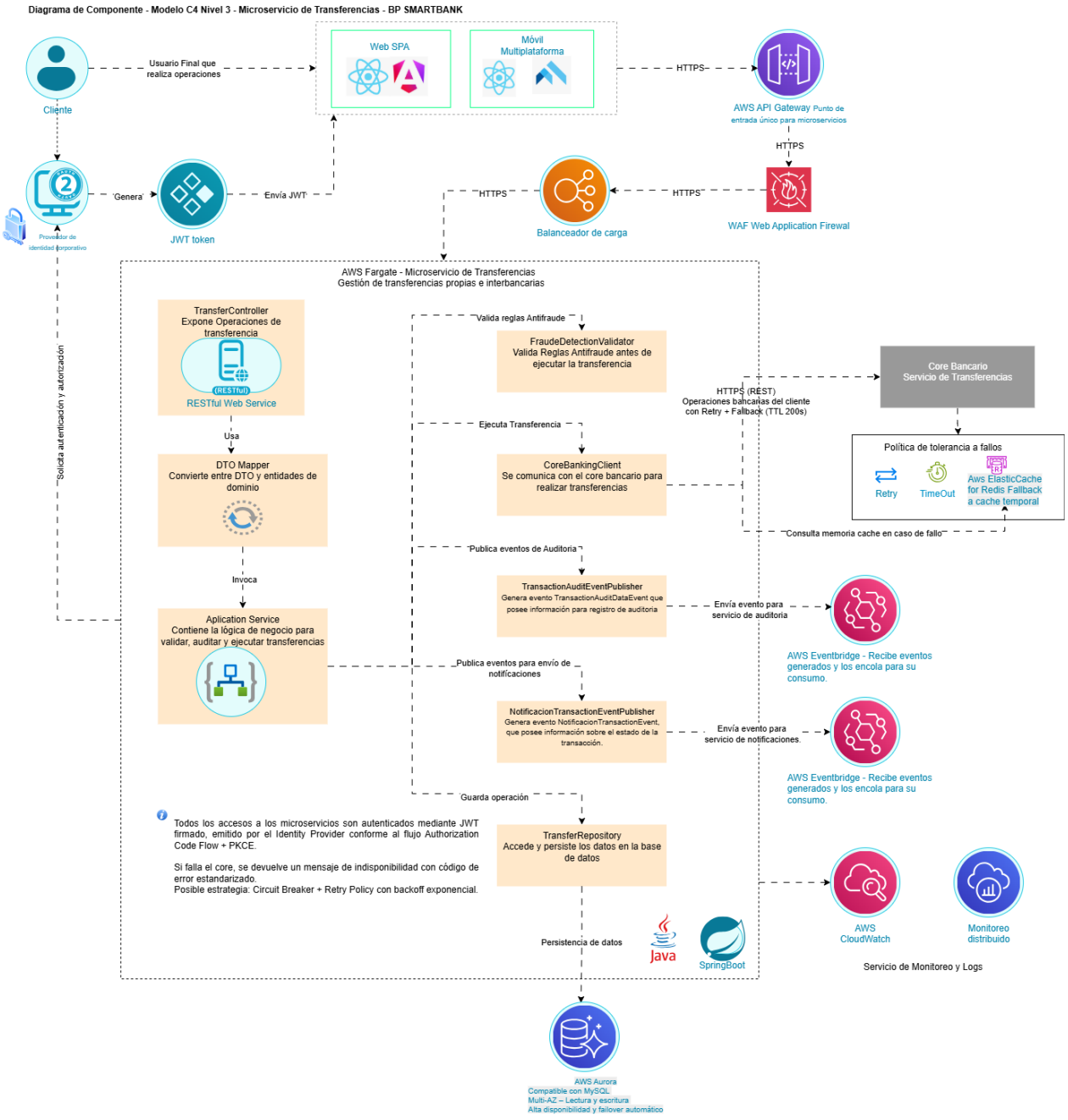
Balanceador de carga	Distribuye tráfico entrante entre los microservicios	AWS ALB (Application Load Balancer)	Balanceo por path y ruta a contenedores protegidos detrás de WAF.
WAF (Firewall)	Protege de amenazas externas (OWASP Top 10, bots, ataques de denegación de servicio)	AWS WAF	Filtra tráfico antes de entrar al ALB / API Gateway.
Notificador Externo (Correo)	Envío de emails al cliente	SendGrid	Usado por contenedor de notificaciones, vía API.
Notificador Externo (SMS)	Envío de mensajes SMS	AWS SNS / Twilio	Invocado por Lambda de notificaciones, como canal alterno de comunicación.
EventBus de Integración	Comunicación asincrónica entre contenedores	AWS EventBridge	Permite desacoplar servicios mediante eventos como TransferenciaRealizada, MovimientoRegistrado, etc.
Cola de Procesamiento	Cola de procesamiento ordenado y tolerante a fallos	AWS SQS FIFO	Procesamiento secuencial de eventos de auditoría y notificaciones.

Repositorio de Archivos de Auditoría	Almacena eventos históricos y trazabilidad	AWS S3	Persistencia a largo plazo para cumplimiento regulatorio.
Core Bancario	Sistema centralizado con información de productos y movimientos	Sistema externo	Consultado por servicios de movimientos, transferencias y autenticación.
Sistema Complementario de Cliente	Información adicional del cliente (ej. perfil, documentos, ocupación)	Sistema externo	Complementa la información que no está en el Core bancario.
Monitoreo y Logs	Observabilidad del sistema	AWS CloudWatch + Sistema externo	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).
DNS Estático	Nombre de dominio para acceder al sistema	AWS Route 53	Punto de entrada para canal web/móvil del cliente.
Disaster Recovery (DR)	Respaldo activo de la infraestructura en otra región	Replicación entre zonas de AWS	Alta disponibilidad y recuperación ante fallos catastróficos.

7. Diagramas de Componentes

Se diseñan componentes para los siguientes microservicios clave:

7.1. Componente de Transferencias



7.1.1. Detalle de componentes internos – Microservicio de Transacciones

Componente	Rol / Función Principal	Tecnología / Framework	Interacciones Clave / Justificación Técnica
TransferController	Exponer endpoints REST para operaciones de transferencia	Spring Boot - RESTful Web Service	Recibe solicitudes del cliente autenticado vía API Gateway. Valida token JWT.
FraudDetectionValidator	Validar reglas antifraude antes de ejecutar una transferencia	Lógica interna del microservicio	Separa la validación de negocio del resto de la lógica. Mejora trazabilidad y auditoría.
DTO Mapper	Convertir datos entre DTOs y entidades de dominio	MapStruct / Manual Mapping	Patrón DTO para desacoplar la capa de presentación de la lógica interna.
ApplicationService	Orquestador interno: aplicar reglas de negocio, ejecutar, auditar y notificar	Spring Boot Service Layer	Contiene lógica de negocio central. Llama validadores, servicios de auditoría y persistencia.
CoreBankingClient	Ejecutar la transferencia llamando al Core Bancario	Spring REST Template / WebClient	Integra con el core mediante API REST. Uso de patrón Retry + Fallback en caso de error.

TransactionAuditEventPublisher	Publicar eventos de auditoría al sistema externo	EventBridge Publisher / AWS SDK	Envía eventos estructurados al servicio de auditoría para trazabilidad y cumplimiento normativo.
NotificationTransactionEventPublisher	Publicar eventos de notificación de la transferencia	EventBridge Publisher / AWS SDK	Desacopla el envío de mensajes del microservicio, mejora escalabilidad y resiliencia.
TransferRepository	Persistir y consultar transferencias	Spring Data JPA + MySQL	Maneja operaciones sobre la base de datos transaccional. Incluye soporte a rollback.

7.1.2. Infraestructura y consideraciones técnicas

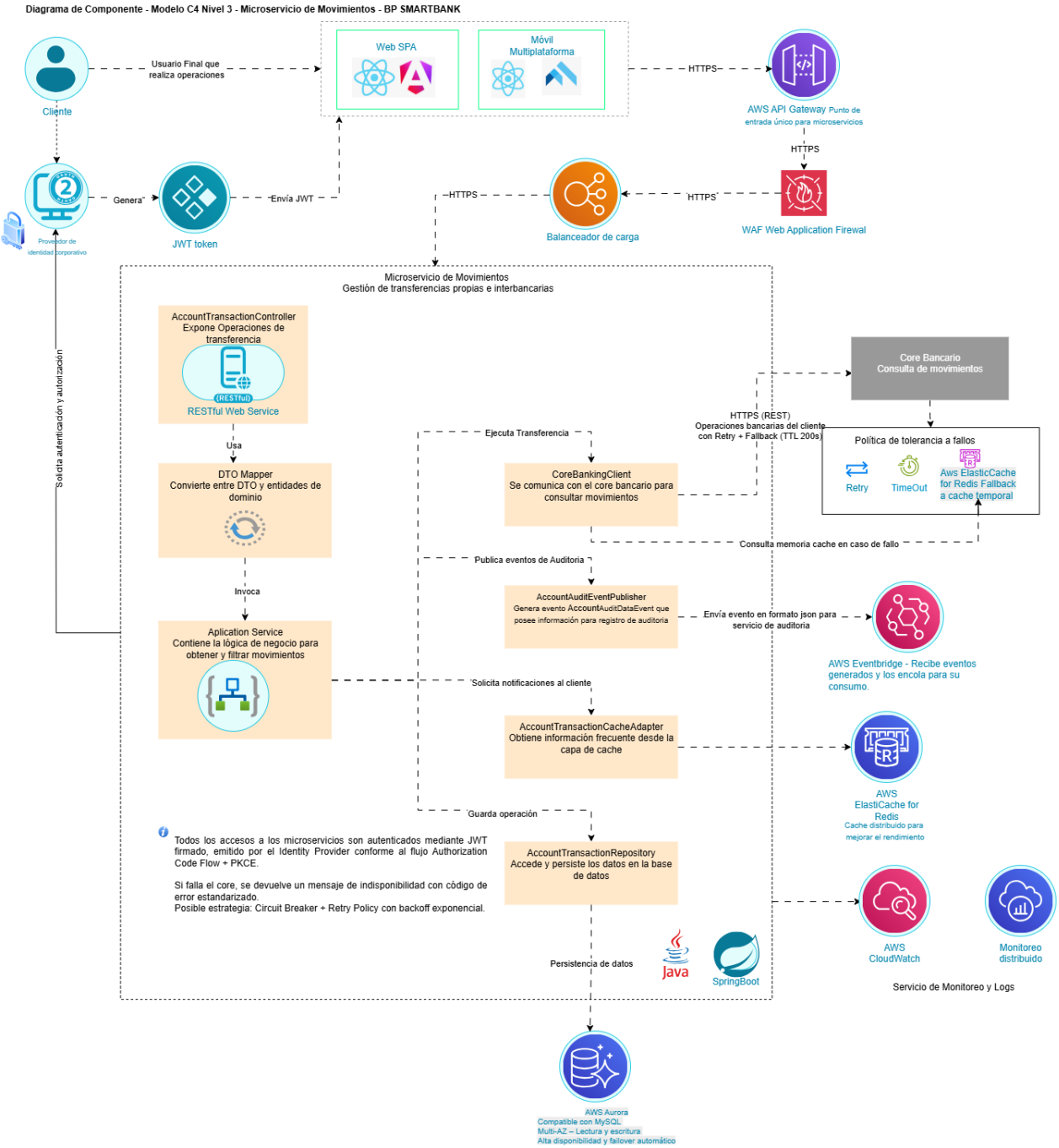
Elemento	Justificación Técnica
JWT Token	Todos los accesos se validan con JWT (OAuth2 + PKCE) para clientes SPA y móviles.
API Gateway + Load Balancer	Control de acceso, escalabilidad y seguridad a través de un WAF.
WAF (Web Application Firewall)	Prevención ante amenazas como inyección de código, bots o DDOS.
ElastiCache (Redis)	Usado como fallback para consultas en caso de falla del core bancario (TTL corto). Mejora disponibilidad.
EventBridge	Facilita arquitectura event-driven. Permite escalar auditoría y notificaciones de forma desacoplada.
CloudWatch / Monitoreo	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).

Política de Tolerancia a Fallos	Uso de Circuit Breaker, Retry Policy y Timeouts para mantener confiabilidad.

7.1.3. Características Clave:

- Expone operaciones críticas para el negocio como envío de dinero entre cuentas.
- Aplica validaciones y lógica de negocio en tiempo real, asegurando la integridad de la operación.
- Publica eventos de auditoría y preferencias, permitiendo trazabilidad y personalización.
- Tolerancia a fallos mediante persistencia eventual y reintentos controlados.
- Arquitectura desacoplada permite escalar horizontalmente según demanda.

7.2. Componente de Movimientos



7.2.1. Detalle de componentes internos – Microservicio de Movimientos

Componente	Rol	Justificación Arquitectónica y Técnica
AccountTransactionController	Expone operaciones de consulta de movimientos	Recibe solicitudes del cliente autenticado vía API Gateway. Valida token JWT.
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	Permite trazabilidad, alertas y dashboards operativos.

7.2.2. Infraestructura y consideraciones técnicas

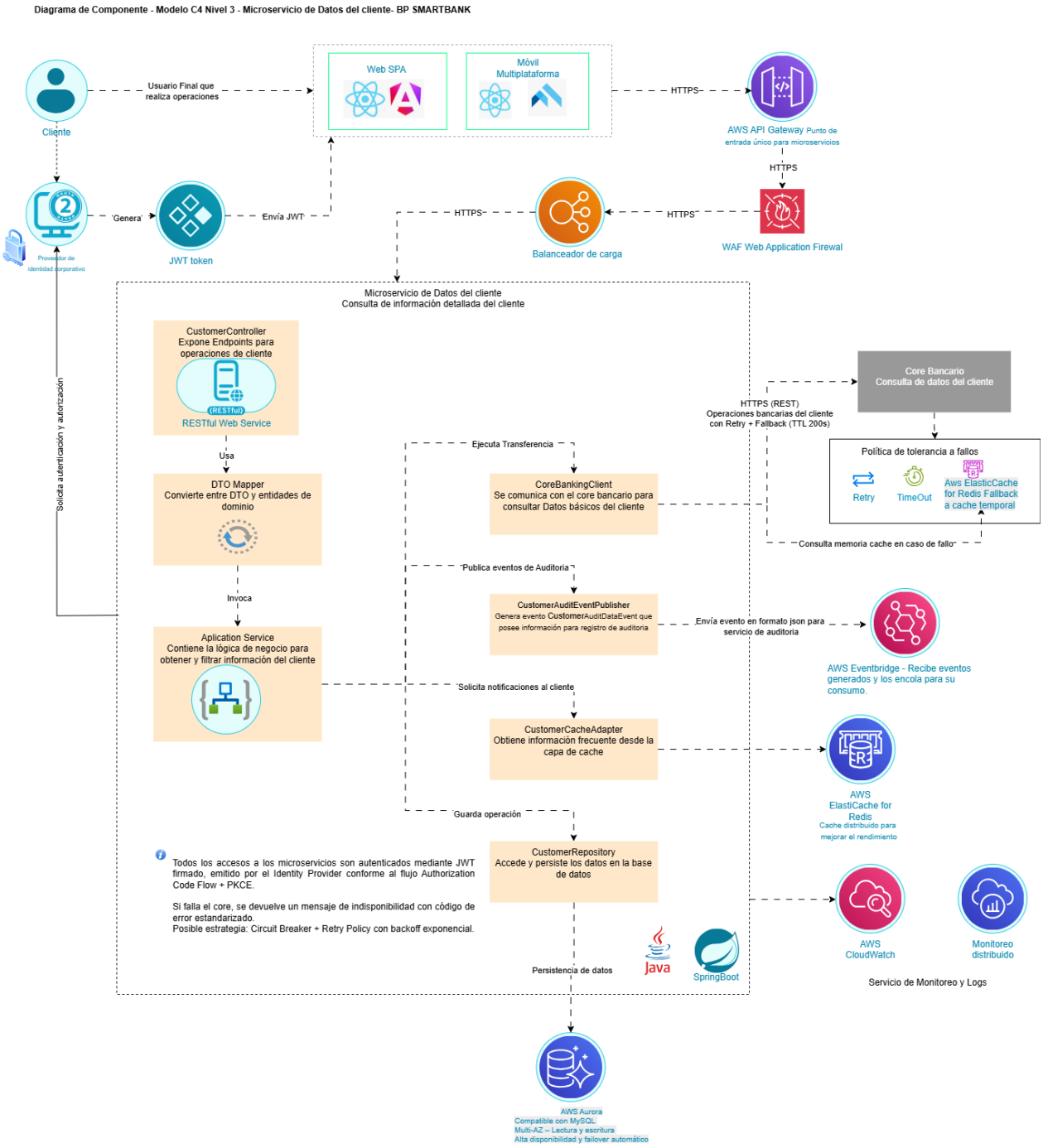
Elemento	Justificación Técnica
JWT Token	Todos los accesos autenticados mediante OAuth2 + PKCE. Asegura integridad y autenticación federada.
API Gateway + Load Balancer	Proveen balanceo de carga, protección por WAF y control de tráfico.
WAF (Web Application Firewall)	Previene ataques como XSS, inyecciones o escaneo de puertos.
ElastiCache (Redis)	Cache de bajo TTL para mejorar tiempo de respuesta en consultas frecuentes y resiliencia ante fallas del Core.
EventBridge	Desacopla la auditoría, permitiendo que otros servicios (como monitoreo) consuman eventos sin impacto en el microservicio.
CloudWatch / Logs Distribuidos	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).

Circuit Breaker + Retry Policy	Patrón de resiliencia en llamadas al Core bancario. Incluye backoff exponencial para no saturar el servicio externo.

7.2.3. Características clave:

- Optimización del rendimiento mediante caché: Utiliza Redis para almacenar resultados de consultas frecuentes, reduciendo la carga en el core bancario y mejorando los tiempos de respuesta para el usuario.
- Consultas asíncronas con consistencia eventual: Soporta sincronización periódica para mantener coherencia con el core, lo que permite mantener disponibilidad incluso en momentos de alta carga o intermitencia en el core.
- Desacoplamiento lógico: La lógica de consulta se encuentra aislada del canal cliente, lo que permite su reutilización en otros canales (e.g. ATM, asistentes virtuales).
- Diseño basado en CQRS: Separación entre comandos (eventos desde auditoría) y consultas (peticiones del cliente), lo que mejora la escalabilidad y mantenibilidad del servicio.
- Preparado para auditoría y trazabilidad: Cada consulta puede ser registrada para análisis posterior, facilitando la trazabilidad y el cumplimiento normativo.

7.3. Componente Datos del Cliente



7.3.1. Detalle de componentes internos – Microservicio de

Componente	Rol / Función Principal	Tecnología / Framework	Interacciones Clave / Justificación Técnica
CustomerController	Expone operaciones REST para obtener información del cliente	Spring Boot - RESTful Web Service	Recibe solicitudes del cliente autenticado vía API Gateway. Valida token JWT.
DTO Mapper	Convierte datos entre DTOs y entidades de dominio	MapStruct / Manual Mapping	Aísla la lógica de presentación de la lógica de negocio. Facilita validación y transformación.
ApplicationService	Contiene la lógica de negocio para obtener y enriquecer datos del cliente	Spring Boot Service Layer	Separa la lógica de acceso a datos del controlador. Puede aplicar reglas o filtros.
CoreBankingClient	Consulta el sistema Core para obtener datos básicos del cliente	REST Template / WebClient	Se comunica con el sistema Core. Usa patrón retry con TTL y tolerancia a fallos.

CustomerCacheAdapter	Consulta caché para obtener datos previamente consultados	AWS ElastiCache (Redis)	Mejora rendimiento, reduce latencia. Usado como fallback en caso de error del Core.
CustomerAuditEventPublisher	Publica eventos de auditoría relacionados a consultas de datos	AWS EventBridge Publisher	Facilita monitoreo y cumplimiento normativo (log de acceso a datos personales).
CustomerRepository	Accede y persiste datos del cliente en la base local	Spring Data JPA + MySQL	Puede usarse para sincronización de datos, perfilamiento o auditoría adicional.

7.3.2. Infraestructura y consideraciones técnicas

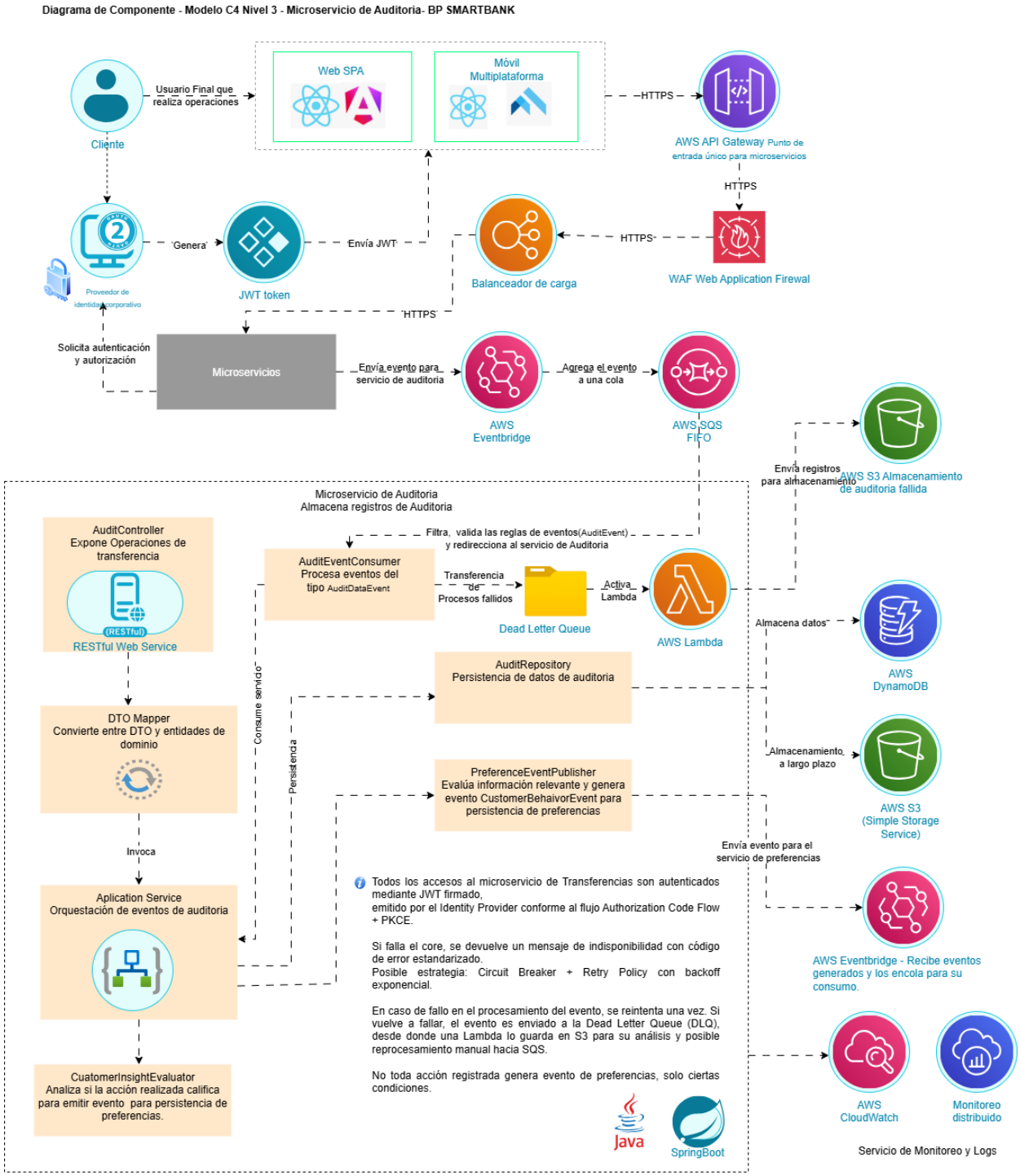
Elemento	Justificación Técnica
JWT Token	Se usa OAuth2 + PKCE para autenticar accesos desde SPA y apps móviles. Flujo Authorization Code Flow.
API Gateway + Load Balancer	Controlan el tráfico entrante, permiten autenticación y protecciones de red.
WAF (Web Application Firewall)	Previene amenazas web comunes (inyecciones, XSS, escaneo).
Redis Cache (ElastiCache)	TTL configurado para reducir llamadas redundantes al Core y dar continuidad ante errores.
EventBridge	Desacopla auditoría para que se consuma de forma asincrónica.

Circuit Breaker + Retry Policy	Protege contra caídas del Core bancario, mejora disponibilidad percibida.
CloudWatch / Logs Distribuidos	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).
Java + Spring Boot	Tecnología estándar en microservicios de misión crítica. Escalable, robusta y con comunidad madura.

7.3.3. Características Clave:

- Centraliza y expone información detallada del cliente autenticado.
- Implementa una arquitectura resiliente ante fallos del core bancario mediante políticas de retry y fallback en caché.
- Publica eventos de auditoría para trazabilidad de acceso a información confidencial.
- Orquesta el consumo de datos en diferentes canales (API REST, caché, core).
- Base para otros servicios como transferencias y preferencias, promoviendo reutilización.

7.4. Componente Auditoría



7.4.1. Detalle de componentes internos – Microservicio de Auditoria

Componente	Rol / Función Principal	Tecnología / Framework	Interacciones Clave / Justificación Técnica
AuditController	Expone endpoints REST para consultar auditorías	Spring Boot - RESTful Web Service	Recibe solicitudes del cliente autenticado vía API Gateway. Valida token JWT.
DTO Mapper	Transforma objetos entre DTO y entidades de dominio	MapStruct / Manual Mapping	Aísla estructuras internas de exposición externa. Facilita consistencia y validaciones.
ApplicationService	Orquesta la lógica de persistencia de eventos	Spring Boot Service Layer	Gestiona la validación, delega persistencia y publica eventos derivados.
CustomerInsightEvaluator	Analiza eventos para determinar si deben emitirse preferencias	Regla basada en condiciones	Evalúa eventos auditados para decidir si deben persistirse como preferencias.
AuditEventConsumer	Procesa eventos de auditoría recibidos desde EventBridge	Spring Boot Consumer (EventBridge)	Extrae eventos AuditDataEvent, los valida y persiste en base o S3. Maneja errores.

AuditRepository	Persiste auditoría en base de datos operativa	Spring Data JPA + AWS DynamoDB	Registro de eventos en almacenamiento estructurado y de consulta rápida.
PreferenceEventPublisher	Publica eventos derivados para preferencias (si aplica)	AWS EventBridge Publisher	Genera eventos CustomerBehaviorEvent solo bajo ciertas reglas. Alta cohesión.

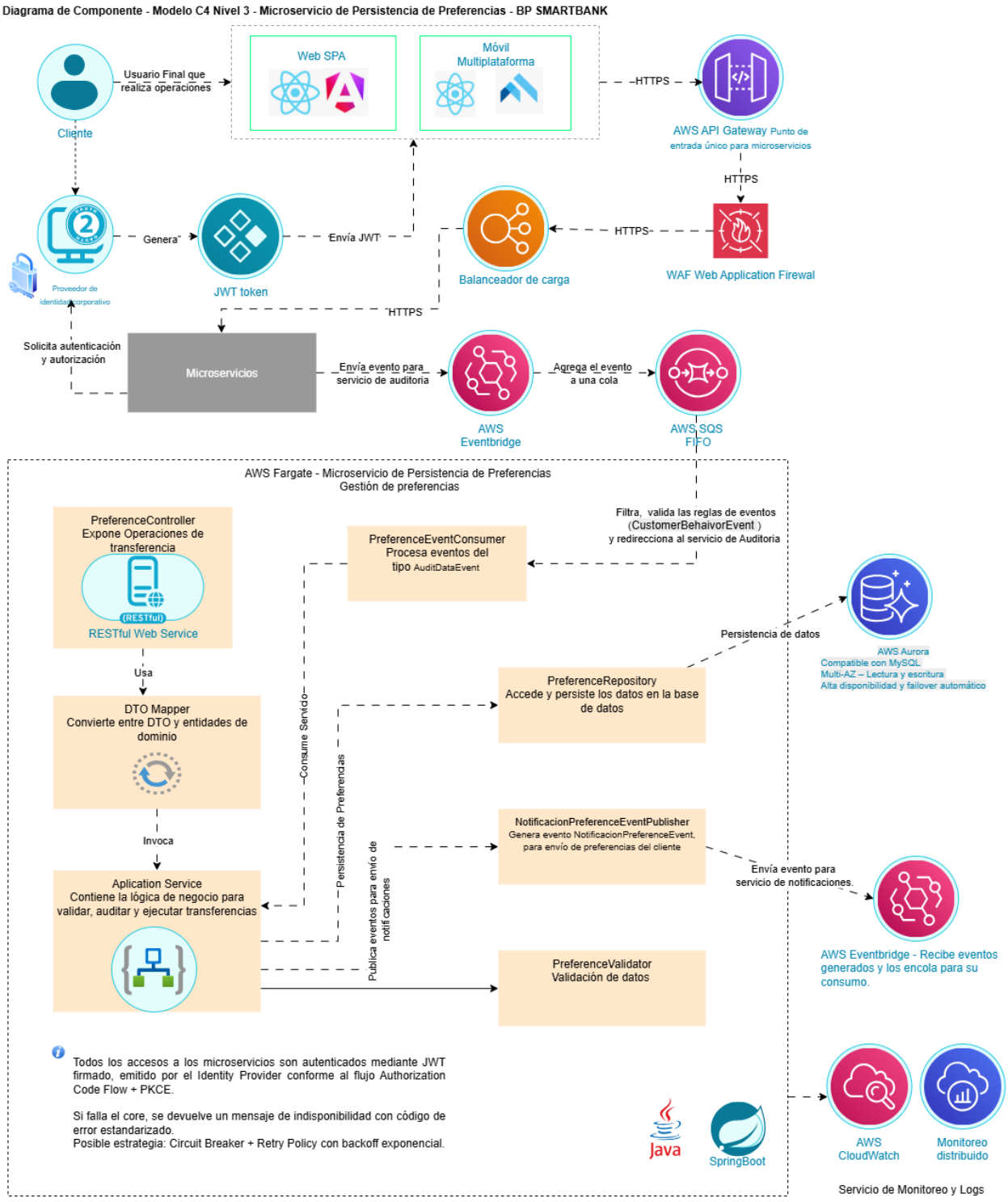
7.4.2. Infraestructura y consideraciones técnicas

Elemento	Justificación Técnica
AWS EventBridge	Recibe eventos desde otros microservicios. Actúa como bus de eventos.
AWS SQS FIFO	Ordena eventos recibidos y garantiza entrega secuencial (relevante en auditoría).
Dead Letter Queue (DLQ)	Permite manejar errores sin pérdida de eventos. Clave en sistemas regulatorios.
AWS Lambda	Maneja auditoría fallida y la envía a almacenamiento en frío (S3).
AWS DynamoDB	Base principal para almacenamiento rápido y escalable de eventos auditados.
AWS S3	Almacenamiento a largo plazo de auditoría fallida o masiva, para consulta posterior.
CloudWatch + Monitoreo distribuido	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).

7.4.3. Características Clave:

- Registra y almacena eventos de auditoría de múltiples servicios, garantizando cumplimiento normativo.
- Implementa una arquitectura desacoplada basada en eventos con EventBridge y SQS FIFO.
- Asegura trazabilidad total mediante almacenamiento redundante en DynamoDB y S3.
- Detecta eventos relevantes para preferencia y los enruta adecuadamente.
- Incluye mecanismos de Dead Letter Queue para reprocesamiento de errores y análisis forense.

7.5. Componente de Preferencias



7.5.1. Detalle de componentes internos – Microservicio de Preferencias

Componente	Rol / Función Principal	Tecnología / Framework	Interacciones Clave / Justificación Técnica
PreferenceController	Expone operaciones REST para gestionar preferencias	Spring Boot - RESTful Web Service	Recibe solicitudes del cliente autenticado vía API Gateway. Valida token JWT.
DTO Mapper	Transforma entre DTOs y entidades de dominio	MapStruct / Manual Mapping	Separación clara de capas, desacoplamiento.
ApplicationService	Orquesta la lógica de validación, auditoría y persistencia	Spring Boot Service Layer	Controla reglas de negocio y validaciones antes de guardar.
PreferenceValidator	Aplica validaciones a los datos de preferencias	Java / Custom Validation	Asegura integridad y formato correcto de la data.
PreferenceRepository	Persistencia de datos de preferencias del cliente	Spring Data JPA + Aurora	Guarda preferencias en base de datos relacional.

PreferenceEventConsumer	Procesa eventos del tipo AuditDataEvent para registrar preferencias	AWS EventBridge Consumer	Permite persistencia asincrónica vía eventos.
NotificationPreferenceEventPublisher	Publica eventos NotificationPreferenceEvent	AWS EventBridge Publisher	Notifica al servicio de Notificaciones sobre cambios de preferencia.

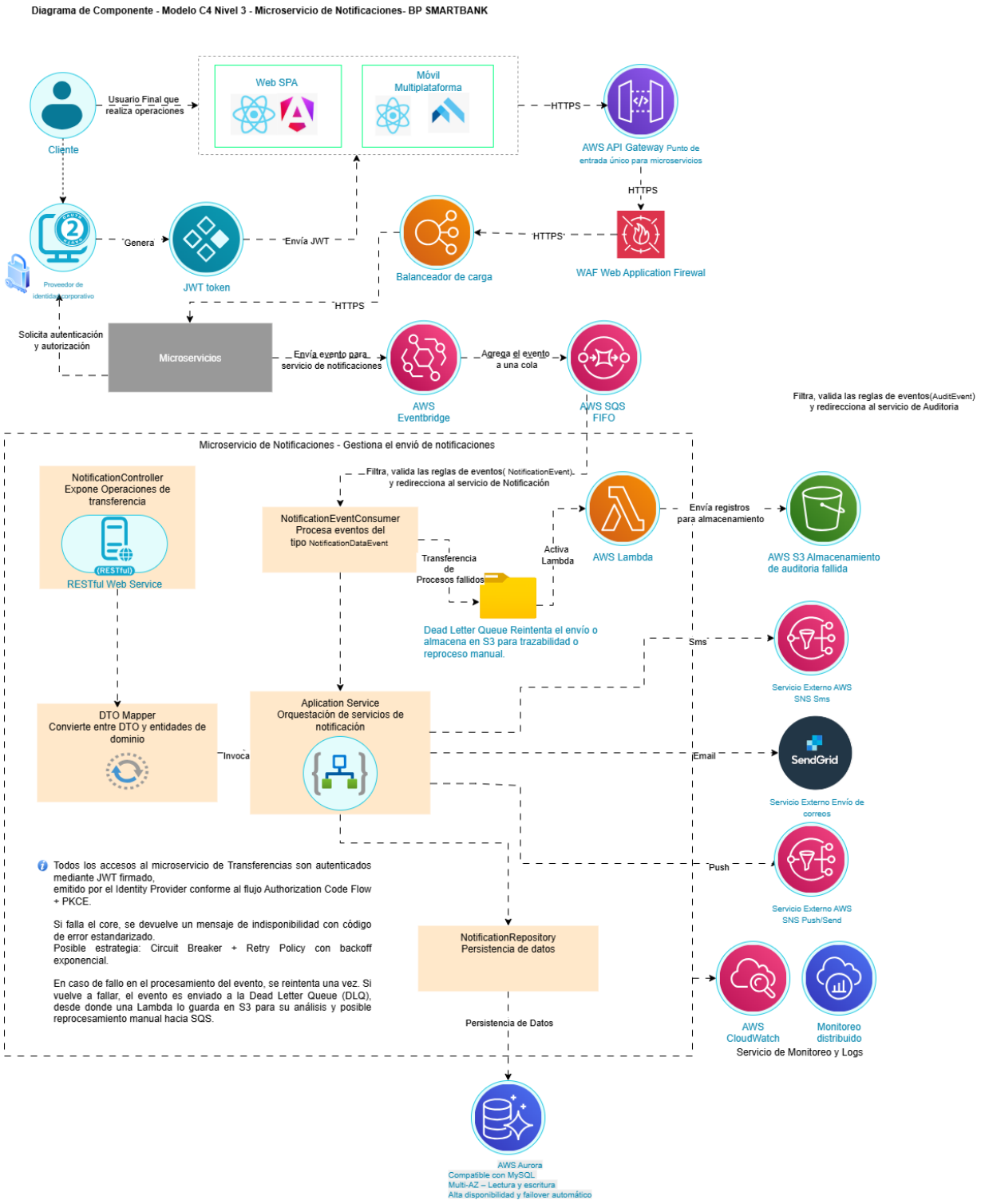
7.5.2. Infraestructura y consideraciones técnicas

Elemento	Justificación Técnica
AWS Aurora (MySQL)	Base de datos relacional altamente disponible y escalable.
AWS EventBridge + AWS SQS FIFO	Recepción y procesamiento asincrónico de eventos con orden garantizado.
AWS CloudWatch + monitoreo distribuido	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).
JWT + OAuth2 PKCE	Acceso seguro desde clientes autenticados.
Circuit Breaker + Retry Policy	Resiliencia ante fallos externos.

7.5.3. Características Clave:

- Gestiona y persiste las preferencias del cliente de forma centralizada.
- Permite personalizar la experiencia del usuario según sus interacciones y patrones de comportamiento.
- Consume eventos generados desde el microservicio de auditoría para identificar comportamientos relevantes.
- Publica eventos a servicios de notificaciones, lo que habilita respuestas automatizadas y proactivas.
- Refuerza la fidelización del cliente al permitirle un mayor control sobre sus comunicaciones y configuraciones.
- Implementa mecanismos de validación de datos y resiliencia ante fallos, asegurando la integridad del sistema.

7.6. Componente Notificaciones



7.6.1. Detalle de componentes internos – Microservicio de Notificaciones

Componente	Rol / Función Principal	Tecnología / Framework	Interacciones Clave / Justificación Técnica
NotificationController	Expone endpoints REST para gestión de notificaciones	Spring Boot - RESTful Web Service	Permite consulta o reenvío de notificaciones desde otros servicios.
DTO Mapper	Convierte entre DTO y entidades de dominio	MapStruct / Manual Mapping	Facilita consistencia entre datos externos e internos. Reduce acoplamiento.
ApplicationService	Orquesta la lógica de notificaciones	Spring Boot Service Layer	Invoca al repositorio, procesa reglas y ejecuta el envío según el canal (SMS, email, push).
NotificationEventConsumer	Consume eventos tipo NotificationDataEvent desde EventBridge	AWS EventBridge + Spring Consumer	Procesa eventos de otros servicios para disparar notificaciones automáticamente.

NotificationRepository	Persiste registros de notificaciones enviadas	Spring Data JPA + Aurora (MySQL)	Permite trazabilidad y consulta histórica de mensajes enviados.

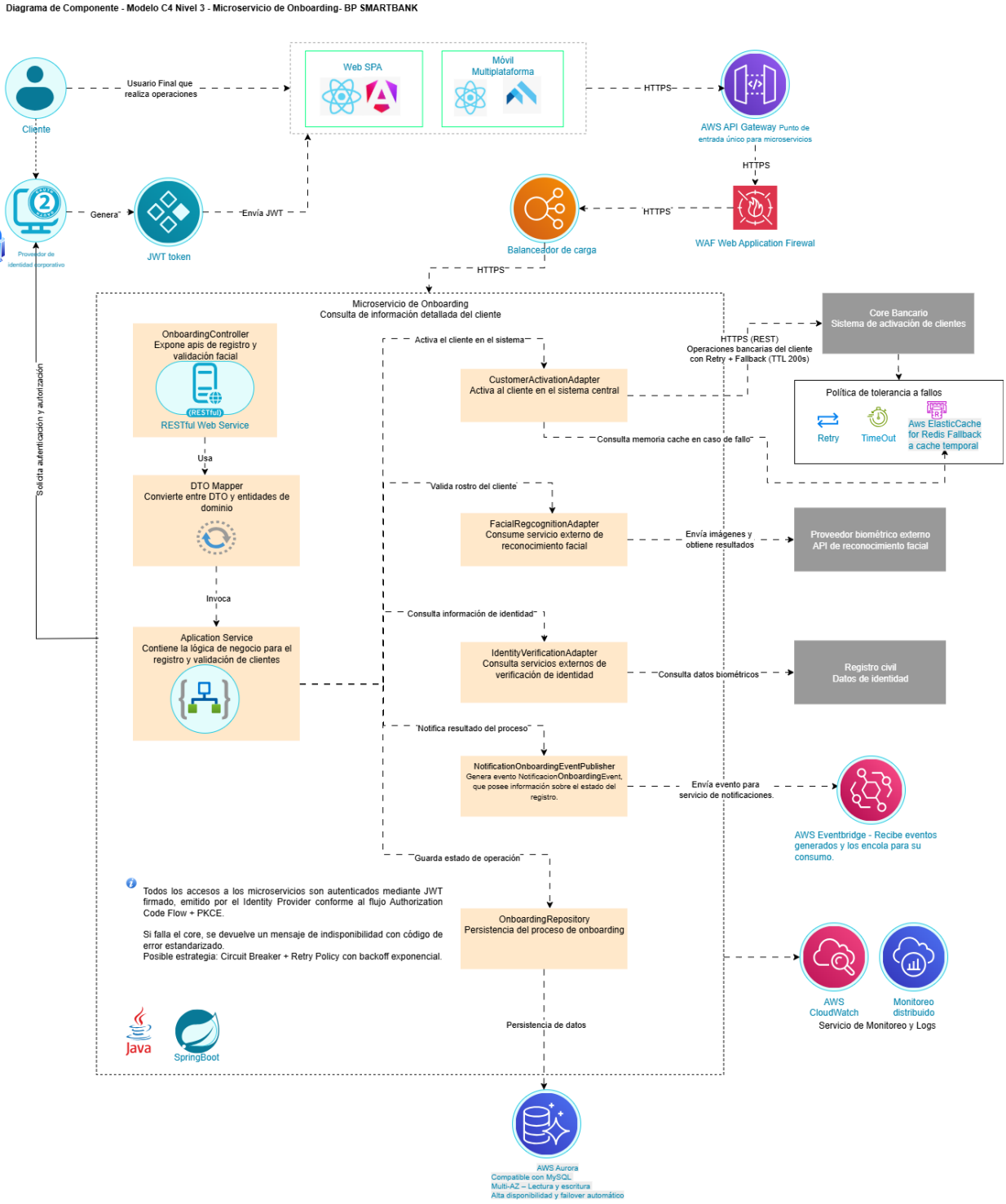
7.6.2. Infraestructura y consideraciones técnicas

Elemento	Justificación Técnica
AWS EventBridge	Recibe eventos generados desde otros microservicios (como transferencias o auditoría).
AWS SQS FIFO	Garantiza orden de entrega y evita duplicidades en procesamiento.
AWS Lambda	Procesa mensajes fallidos desde DLQ o dispara reintentos controlados.
Dead Letter Queue (DLQ)	Asegura trazabilidad de errores, permite auditoría y reprocesos.
AWS SNS (SMS)	Canal para envío de mensajes SMS (informativo, autenticación, alertas).
SendGrid (Email)	Servicio de terceros para envío de correos electrónicos. Alta capacidad y métricas.
AWS SNS Push	Canal para notificaciones push a dispositivos móviles.
CloudWatch + Monitoreo distribuido	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).

7.6.3. Características Clave:

- Gestiona el envío multicanal de notificaciones (SMS, email, push).
- Permite reaccionar a eventos de negocio relevantes de manera proactiva y en tiempo real.
- Desacopla completamente los emisores de eventos de los canales de comunicación.
- Mejora la experiencia del cliente y reduce carga operativa.
- Incluye trazabilidad y reprocesamiento ante errores con DLQ y S3.

7.7. Componente Onboarding



7.7.1. Detalle de componentes internos – Microservicio de Onboarding

Componente	Rol / Función Principal	Tecnología / Framework	Interacciones Clave / Justificación Técnica
OnboardingController	Expone operaciones REST para registro y validación de identidad	Spring Boot - RESTful Web Service	Recibe peticiones desde Web o App para iniciar el proceso de alta del cliente.
DTO Mapper	Convierte entre DTO y entidades de dominio	MapStruct / Manual Mapping	Facilita integración desacoplada entre capa de entrada y lógica interna.
ApplicationService	Orquesta la lógica de onboarding y validaciones	Spring Boot Service Layer	Controla los flujos de validación biométrica, identidad, y activación en core.
CustomerActivationAdapter	Activa cliente en sistema core bancario	REST + Retry Policy	Comunicación con sistema externo; incluye tolerancia a fallos.
FacialRecognitionAdapter	Integra servicio externo de reconocimiento facial	REST API externa	Envía imágenes y obtiene resultado de validación biométrica.

IdentityVerificationAdapter	Valida identidad del cliente ante registro civil	REST API externa	Consulta servicios de verificación con datos biométricos y personales.
NotificationOnboardingEventPublisher	Publica evento con resultado del onboarding	AWS EventBridge Publisher	Dispara evento para el servicio de notificaciones.
OnboardingRepository	Persiste datos del proceso de onboarding	Spring Data JPA + Aurora (MySQL)	Guarda resultados del registro, trazabilidad del flujo y errores.

7.7.2. Infraestructura y consideraciones técnicas

Elemento	Justificación Técnica
AWS EventBridge	Se usa para notificar estado del registro a otros servicios (como Notificaciones).
Core Bancario	Activación final del cliente como entidad válida en el banco.
Registro Civil / Proveedor biométrico externo	Verificación de identidad y rostro en fuentes oficiales externas.
AWS ElastiCache (Redis)	Cache para fallos temporales o respuestas repetidas, mejora rendimiento.
AWS CloudWatch + Monitoreo distribuido	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).

7.7.3. Características Clave:

- Permite la incorporación ágil y segura de nuevos clientes al ecosistema bancario.
- Integra validaciones biométricas y verificación de identidad con servicios externos (registro civil, proveedor biométrico).
- Publica eventos al sistema de notificaciones y auditoría, habilitando flujos asincrónicos.
- Tolerancia a fallos en conexiones con core bancario mediante mecanismos como Retry, Timeout y uso de caché temporal.
- Mejora la experiencia del usuario final reduciendo fricciones en el proceso de alta digital.

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.