

4 DE OCTUBRE DE 2025

ARQUITECTURA DE INTEGRACIÓN PARA MODERNIZACIÓN BANCARIA

FRANCISCO DANILO JIMÉNEZ PAREDES

Contenido

1. Introducción	2
Objetivo.	2
Alcance.	2
Descripción del escenario.	2
Diagrama C4 Nivel 2 – Sistema Bancario	5
Diagrama C4 Nivel 2 – Core Bancario Legacy.	8
Diagrama C4 Nivel 2 – API Gateway Bancario.....	11
Diagrama C4 Nivel 2 – Prevención de Fraudes.....	13
Diagrama C4 Nivel 3 – fraudDetector (Prevención de Fraudes).....	16
Diagrama C4 Nivel 3 – rulesEngine (Prevención de Fraudes)	16
Diagrama C4 Nivel 3 – alertsService (Prevención de Fraudes)	17
Diagrama C4 Nivel 2 – Plataforma de Pago.....	18
Diagrama C4 Nivel 3 – pagosCore (Plataforma de Pagos).....	21
Diagrama C4 Nivel 3 – compensación (Plataforma de Pagos).....	21
Diagrama C4 Nivel 3 – pagosExternos (Plataforma de Pagos Bancarios)	22
Diagrama C4 Nivel 2 – Canal Digital Bancario	22
Diagrama C4 Nivel 3 – backendCanal (Canal Digital Bancario)	26
Diagrama C4 Nivel 2 – Sistema de Gestión de Identidad y Acceso	27

1. Introducción

Objetivo.

El propósito de este documento es presentar una propuesta arquitectónica integral para la modernización tecnológica de una entidad bancaria, utilizando el modelo C4 como marco de representación. La solución busca facilitar la integración entre sistemas legacy y nuevas plataformas digitales, garantizando escalabilidad, seguridad, cumplimiento normativo. A través de diagramas estructurados y justificaciones técnicas, se pretende ofrecer una visión clara y práctica que sirva como base para la implementación efectiva de la arquitectura propuesta.

Alcance.

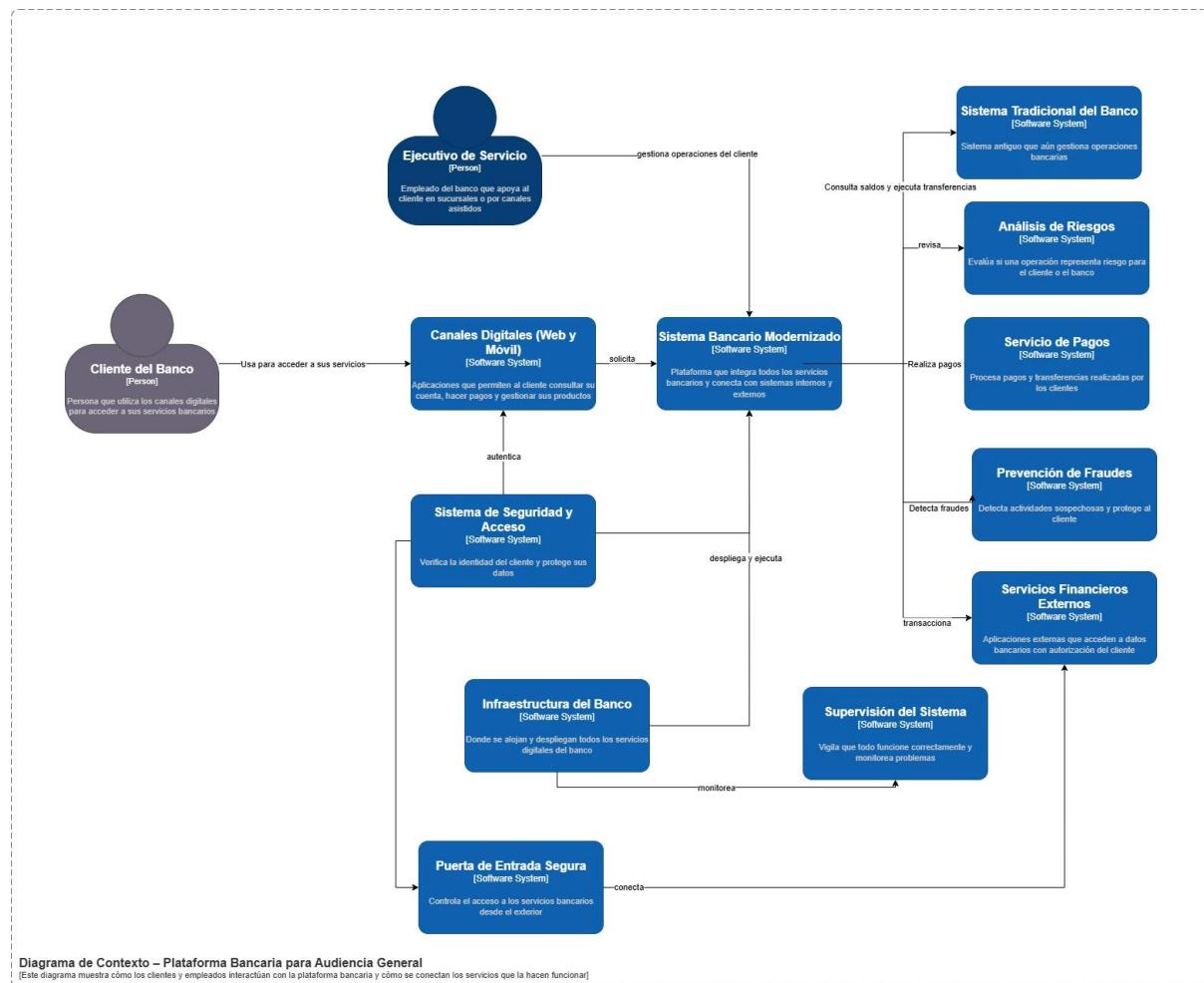
El diseño abarca los siguientes elementos clave:

- Modelado arquitectónico mediante los tres niveles del modelo C4: contexto, contenedores y componentes.
- Integración entre el core bancario tradicional y un nuevo core digital, asegurando interoperabilidad y coexistencia operativa.
- Incorporación de nuevos canales digitales (banca web y móvil), plataforma de pagos, sistemas de gestión de riesgos y prevención de fraudes.
- Diseño de APIs internas y externas bajo estándares modernos de mensajería y seguridad.
- Estrategias de alta disponibilidad, recuperación ante desastres, y protección de datos personales conforme a la normativa vigente.
- Propuesta de gobierno de APIs y microservicios, junto con un plan de migración gradual que minimice el riesgo operativo.

Descripción del escenario.

La institución bancaria se encuentra en un proceso de transformación digital motivado por la necesidad de mejorar sus servicios, adaptarse a nuevas regulaciones y responder a las demandas del mercado. Actualmente opera con un core bancario tradicional que debe integrarse con un nuevo core digital, permitiendo una transición progresiva sin

afectar la operación crítica. Además, se incorporan nuevos canales de atención al cliente (web y móvil), una plataforma moderna de pagos, sistemas de gestión de riesgos y prevención de fraudes, y APIs para habilitar servicios de terceros bajo el marco de Open Finance. La arquitectura debe garantizar interoperabilidad, escalabilidad, seguridad y resiliencia operativa.



Descripción de Componentes del Sistema Bancario Modernizado

Cliente del Banco

Persona que utiliza los canales digitales (web o móvil) para acceder a sus productos financieros, consultar saldos, realizar pagos y gestionar su cuenta.

Ejecutivo de Servicio

Empleado del banco que asiste al cliente en sucursales físicas o canales asistidos, utilizando herramientas internas para consultar y gestionar operaciones bancarias.

Canales Digitales (Web y Móvil)

Aplicaciones que permiten al cliente interactuar con el banco desde su dispositivo. Facilitan operaciones como transferencias, pagos, consultas de saldo y gestión de productos.

Sistema Bancario Modernizado

Plataforma central que integra todos los servicios bancarios. Coordina la comunicación entre los canales digitales, sistemas internos y servicios externos, asegurando una experiencia fluida y segura.

Sistema Tradicional del Banco (Core Bancario Legacy)

Sistema heredado que aún gestiona operaciones críticas como saldos, cuentas y transferencias. Aunque es antiguo, sigue siendo esencial para el funcionamiento del banco.

Servicio de Pagos

Módulo especializado que procesa pagos internos y externos, incluyendo transferencias entre cuentas, pagos de servicios y débitos automáticos.

Análisis de Riesgos

Sistema que evalúa el nivel de riesgo asociado a cada operación financiera. Ayuda a prevenir pérdidas y a cumplir con regulaciones internas y externas.

Prevención de Fraudes

Detecta patrones sospechosos en las operaciones bancarias. Protege al cliente y al banco ante posibles fraudes o actividades no autorizadas.

Servicios Financieros Externos (Open Finance)

Aplicaciones de terceros que acceden a los datos bancarios del cliente con su consentimiento. Permiten ofrecer productos personalizados, consolidar información financiera y facilitar nuevas experiencias digitales.

Sistema de Seguridad y Acceso (IAM)

Verifica la identidad del cliente y gestiona sus credenciales. Emite tokens seguros que permiten acceder a los servicios bancarios sin comprometer la privacidad.

Puerta de Entrada Segura (API Gateway / Manager)

Controla el acceso externo a los servicios del banco. Valida los tokens emitidos por el sistema de seguridad y aplica políticas de protección antes de permitir el ingreso.

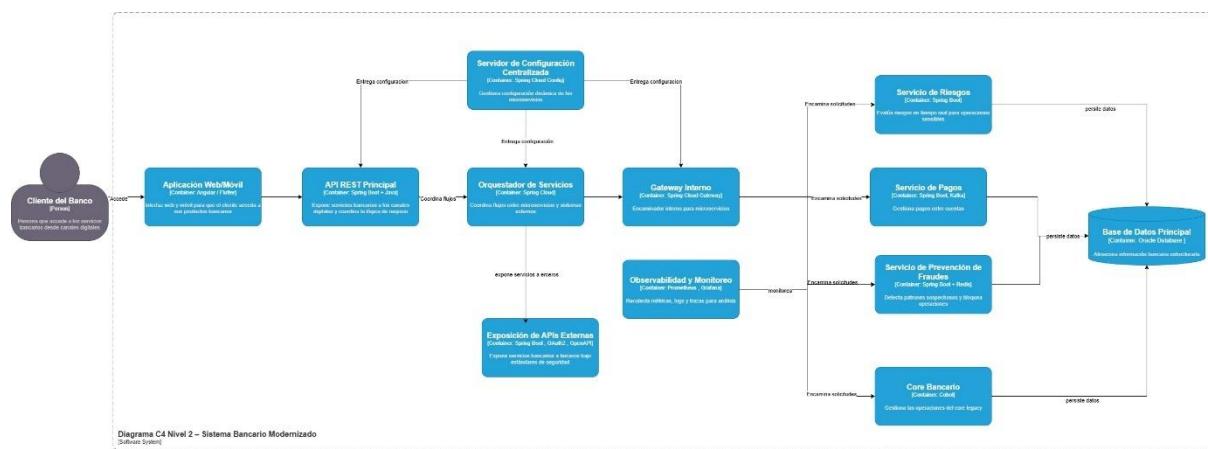
Infraestructura del Banco

Entorno físico y virtual donde se alojan todos los servicios digitales. Permite ejecutar microservicios, distribuir la carga y mantener la disponibilidad de la plataforma.

Supervisión del Sistema

Sistema que monitorea el estado de todos los componentes. Detecta fallos, genera alertas y ayuda a mantener la estabilidad y el rendimiento de la plataforma bancaria.

Diagrama C4 Nivel 2 – Sistema Bancario



Explicación del Flujo – Sistema Bancario Modernizado

Cliente del Banco

- Representa al usuario final que accede a los servicios bancarios desde canales digitales.
- Se conecta directamente con el contenedor Aplicación Web/Móvil.

Aplicación Web/Móvil

- Tecnología: Angular / Flutter
- Es la interfaz visual que el cliente utiliza para interactuar con el banco.
- Se comunica con el API REST Principal para enviar solicitudes como consultas de saldo, pagos, etc.

API REST Principal

- Tecnología: Spring Boot + Java
- Actúa como el punto de entrada lógico para todas las operaciones bancarias.
- Recibe solicitudes desde la web/app y las deriva al Orquestador de Servicios.

Orquestador de Servicios

- Tecnología: Spring Cloud
- Coordina el flujo entre los microservicios internos.
- Deriva operaciones específicas a:
 - Servicio de Pagos
 - Servicio de Riesgos
 - Servicio de Prevención de Fraudes
 - Exposición de APIs Externas

Servicio de Pagos

- Tecnología: Spring Boot + Kafka
- Procesa pagos y transferencias entre cuentas.
- Recibe configuración desde el Servidor de Configuración Centralizada.
- Es monitoreado por el sistema de Observabilidad.

- Recibe tráfico interno a través del Gateway Interno.

Servicio de Riesgos

- Tecnología: Spring Boot.
- Evalúa el riesgo de cada operación en tiempo real.
- Recibe configuración desde Servicio de configuración centralizada, tráfico desde gatewayInterno, y es monitoreado por observabilidad.

Servicio de Prevención de Fraudes

- Tecnología: Spring Boot + Redis
- Detecta patrones sospechosos y bloquea operaciones potencialmente fraudulentas.
- También está conectado a los mismos servicios de soporte que los anteriores.

Exposición de APIs Externas

- Tecnología: Spring Boot + OAuth2 + OpenAPI
- Expone servicios bancarios a terceros (Open Finance) bajo estándares de seguridad.
- Recibe tráfico desde el orquestador.

Servidor de Configuración Centralizada

- Tecnología: Spring Cloud Config
- Proporciona configuración dinámica a los microservicios (pagosService, riesgosService, fraudesService).

Gateway Interno

- Tecnología: Spring Cloud Gateway
- Encaminador interno que distribuye solicitudes entre microservicios.
- Se conecta con los servicios de pagos, riesgos y fraudes.

Observabilidad y Monitoreo.

- Tecnología: Prometheus + Grafana
- Supervisa el estado, rendimiento y trazabilidad de los microservicios clave.

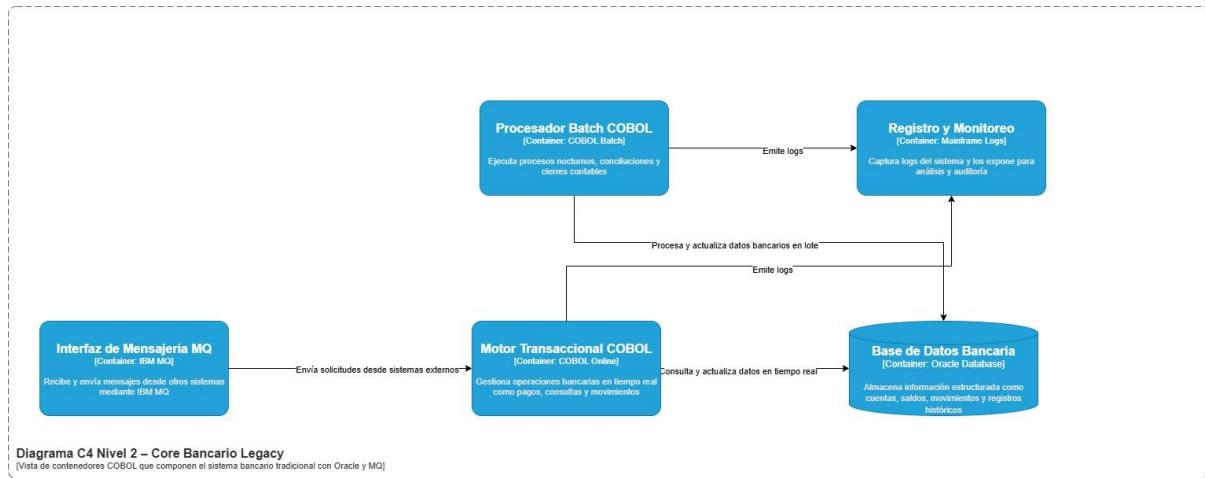
Base de Datos Principal

- Tecnología: Oracle
- Almacena toda la información estructurada del sistema bancario.

Flujo General

1. **Cliente** accede vía webApp o móvil.
2. WebApp/móvil envía solicitudes a apiRest.
3. apiRest coordina con el orquestador.
4. El Orquestador de servicios distribuye tareas a microservicios (pagos, riesgos, fraudes, externalApi).
5. Todos los microservicios reciben configuración desde configServer(servidor de configuración centralizada), tráfico desde gatewayInterno, y son monitoreados por observabilidad.
6. Los microservicios acceden a la Base de Datos Principal para persistencia.

Diagrama C4 Nivel 2 – Core Bancario Legacy.



Explicación del Diagrama – Core Bancario Legacy

Software System: Core Bancario Legacy

Este sistema representa el corazón operativo del banco, desarrollado en COBOL, y encargado de gestionar las operaciones financieras más críticas: saldos, cuentas, pagos, conciliaciones y movimientos. Aunque es un sistema heredado, sigue siendo esencial y está integrado con tecnologías modernas como Oracle y IBM MQ.

Contenedores definidos

1. Procesador Batch COBOL

- **Tecnología:** COBOL Batch
- **Función:** Ejecuta procesos nocturnos como conciliaciones, cierres contables, generación de extractos y actualizaciones masivas.
- **Relaciones:**
 - Se conecta a la **Base de Datos Bancaria** para actualizar datos en lote.
 - Emite logs hacia el contenedor **Registro y Monitoreo (logs)** para trazabilidad y auditoría.

2. Motor Transaccional COBOL.

- **Tecnología:** COBOL Online
- **Función:** Gestiona operaciones en tiempo real como pagos, consultas de saldo, transferencias y movimientos.
- **Relaciones:**
 - Accede a la **Base de Datos Bancaria** para leer y escribir datos en tiempo real.
 - Recibe solicitudes desde la **Interfaz de Mensajería MQ**.

- Emite logs hacia el contenedor **Registro y Monitoreo (logs)**.

3. Interfaz de Mensajería MQ

- **Tecnología:** IBM MQ
- **Función:** Actúa como puente de integración entre el core bancario y otros sistemas (modernos o externos), recibiendo y enviando mensajes estructurados.
- **Relaciones:**
 - Envía solicitudes al **Motor Transaccional COBOL** para su procesamiento.

4. Base de Datos Bancaria

- **Tecnología:** Oracle
- **Función:** Almacena toda la información estructurada del sistema bancario: cuentas, saldos, movimientos, históricos.
- **Relaciones:**
 - Es accedida por el **Procesador Batch** y el **Motor Transaccional** para lectura y escritura.

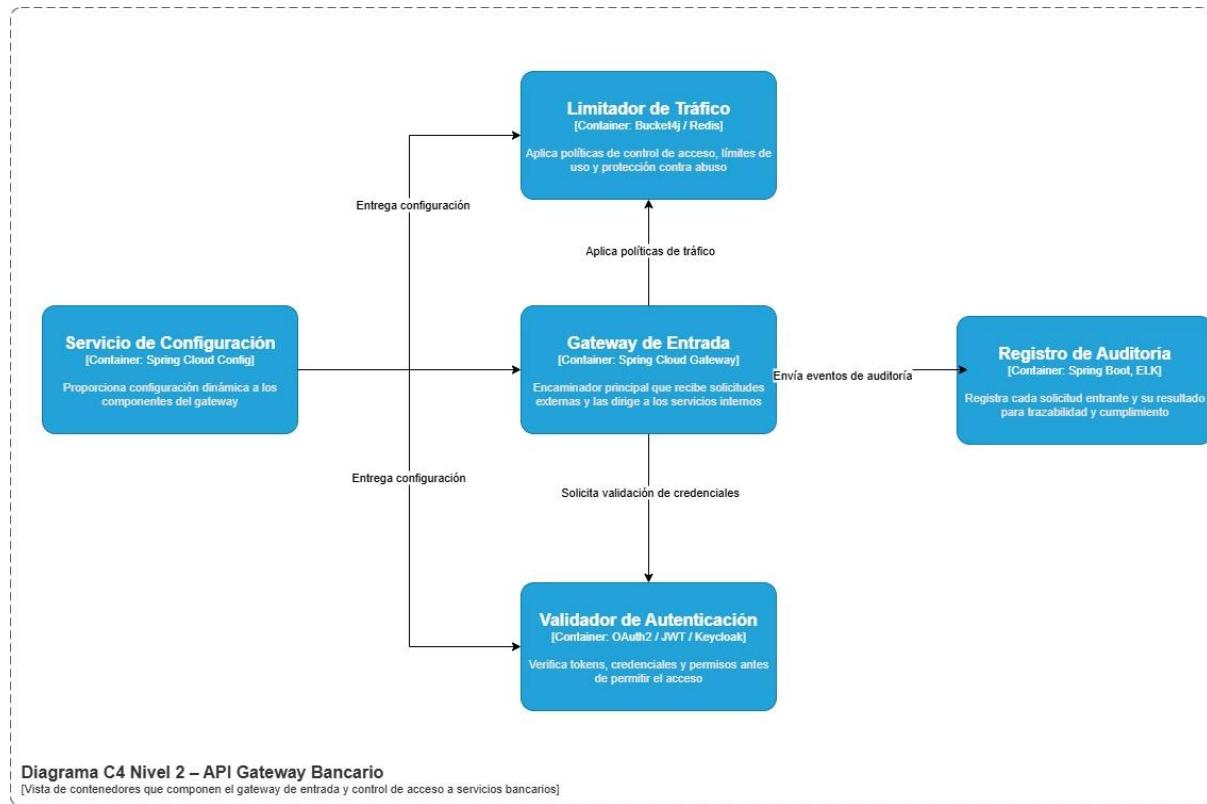
5. Registro y Monitoreo (logs)

- **Tecnología:** Mainframe Logs
- **Función:** Captura los logs generados por los procesos batch y transaccionales, y los expone para análisis, auditoría y monitoreo.

Flujo General del Sistema

1. **Sistemas externos** envían mensajes al **Interfaz MQ**, que los enruta al **Motor Transaccional COBOL**.
2. El **Motor Transaccional** procesa la operación en tiempo real y accede a la **Base de Datos Oracle**.
3. Durante la noche, el **Procesador Batch** ejecuta tareas masivas y también accede a la base de datos.
4. Ambos motores emiten logs hacia el sistema de **Registro y Monitoreo**.
5. Todo el sistema está encapsulado dentro del software system Core Bancario Legacy, que representa el entorno COBOL operativo.

Diagrama C4 Nivel 2 – API Gateway Bancario



Explicación de Componentes – API Gateway Bancario

Gateway de Entrada

- **Tecnología:** Spring Cloud Gateway
- **Función:** Es el punto de entrada principal para todas las solicitudes externas que llegan al ecosistema bancario.
- **Responsabilidades:**
 - Recibe peticiones HTTP desde clientes o terceros.
 - Aplica reglas de enrutamiento para dirigirlas a los servicios internos correspondientes.
 - Actúa como proxy inverso, encapsulando la lógica de acceso.

Validador de Autenticación

- **Tecnología:** OAuth2 / JWT / Keycloak

- **Función:** Verifica que cada solicitud entrante esté autenticada y autorizada antes de permitir el acceso a los servicios.
- **Responsabilidades:**
 - Validar tokens JWT, sesiones OAuth2 o credenciales federadas.
 - Consultar el sistema de identidad (IAM) para confirmar permisos.
 - Rechazar accesos no válidos o caducados.

Limitador de Tráfico

- **Tecnología:** Bucket4j / Redis
- **Función:** Controla el volumen de solicitudes por cliente, IP o token, evitando abusos o sobrecarga del sistema.
- **Responsabilidades:**
 - Aplicar políticas de rate limiting (por segundo, minuto, día).
 - Proteger servicios internos de ataques tipo DoS o uso excesivo.
 - Registrar métricas de consumo por cliente o canal.

Registro de Auditoría

- **Tecnología:** Spring Boot , ELK
- **Función:** Captura y almacena información detallada de cada solicitud procesada por el gateway.
- **Responsabilidades:**
 - Registrar quién accedió, cuándo, a qué recurso y con qué resultado.
 - Facilitar trazabilidad para cumplimiento normativo (auditorías, reguladores).
 - Integrarse con sistemas de monitoreo y análisis (Grafana).

Servidor de Configuración (configServer)

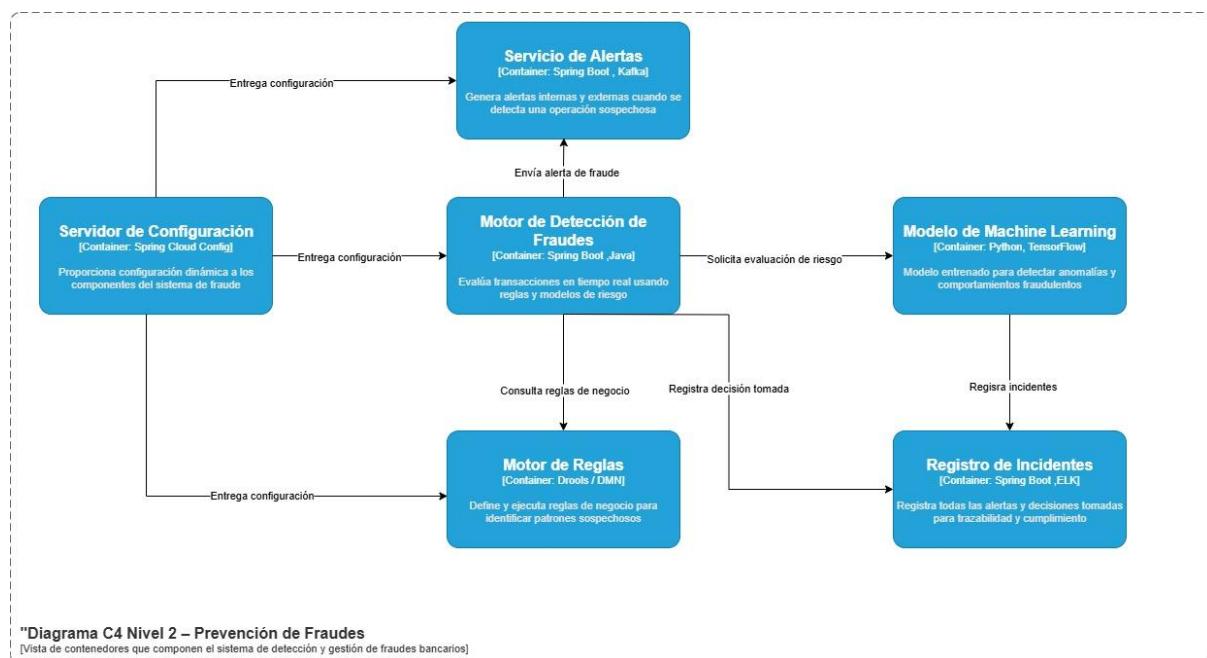
- **Tecnología:** Spring Cloud Config

- **Función:** Proporciona configuración centralizada y dinámica a los componentes del gateway.
- **Responsabilidades:**
 - Entregar propiedades, reglas y parámetros de seguridad.
 - Permitir cambios sin reiniciar los servicios.
 - Versionar configuraciones por entorno (dev, test, prod).

Flujo General

1. Una solicitud externa llega al **Gateway de Entrada**.
2. El gateway consulta al **Validador de Autenticación** para verificar credenciales.
3. Si es válida, pasa por el **Límitador de Tráfico** para aplicar políticas de uso.
4. La solicitud se registra en el **Registro de Auditoría**.
5. Finalmente, se enruta al servicio interno correspondiente.
6. Todos los componentes reciben configuración desde el **Servidor de Configuración**.

Diagrama C4 Nivel 2 – Prevención de Fraudes



Explicación del Diagrama – Sistema de Prevención de Fraudes

Este sistema tiene como objetivo detectar operaciones bancarias sospechosas en tiempo real, aplicar reglas de negocio, evaluar riesgos mediante modelos predictivos y generar alertas para mitigar fraudes antes de que se concreten.

Motor de Detección de Fraudes

- **Tecnología:** Spring Boot, Java
- **Propósito:** Es el núcleo operativo del sistema. Recibe transacciones bancarias y decide si deben ser bloqueadas, marcadas o aprobadas.
- **Responsabilidades:**
 - Orquestar la evaluación de riesgo combinando reglas y modelos predictivos.
 - Enviar alertas cuando se detecta una operación sospechosa.
 - Registrar cada decisión en el sistema de auditoría.

Motor de Reglas

- **Tecnología:** Drools ,DMN
- **Propósito:** Define y ejecuta reglas de negocio que permiten identificar patrones de fraude conocidos.
- **Responsabilidades:**
 - Evaluar condiciones como montos, horarios, ubicaciones, frecuencia, etc.
 - Permitir ajustes dinámicos por parte de analistas de fraude.
 - Registrar cada ejecución de regla en el sistema de auditoría.

Modelo de Machine Learning

- **Tecnología:** Python ,TensorFlow
- **Propósito:** Detecta anomalías y comportamientos atípicos que podrían indicar fraude, incluso si no están cubiertos por reglas explícitas.

Servicio de Alertas

- **Tecnología:** Spring Boot , Kafka
- **Propósito:** Notifica a los sistemas internos o externos cuando se detecta una operación sospechosa.
- **Responsabilidades:**
 - Generar alertas en tiempo real para analistas, sistemas de bloqueo o clientes.
 - Publicar eventos en Kafka para procesamiento asíncrono.
 - Integrarse con canales como email, SMS o dashboards internos.

Registro de Incidentes

- **Tecnología:** Spring Boot , ELK
- **Propósito:** Captura y almacena todas las decisiones, evaluaciones y alertas generadas por el sistema para trazabilidad y cumplimiento normativo.
- **Responsabilidades:**
 - Registrar qué transacción fue evaluada, con qué reglas, modelos y resultado.
 - Facilitar auditorías internas y externas.

Servidor de Configuración

- **Tecnología:** Spring Cloud Config
- **Propósito:** Proporciona configuración centralizada y dinámica a todos los componentes del sistema de fraude.
- **Responsabilidades:**
 - Entregar parámetros, umbrales, rutas de alerta y reglas activas.
 - Permitir ajustes sin reiniciar servicios.
 - Versionar configuraciones por entorno.

Flujo General del Sistema

1. Una transacción bancaria llega al **Motor de Detección de Fraudes**.
2. Este motor consulta el **Motor de Reglas** y el **Modelo de Machine Learning** para evaluar el riesgo.
3. Si se detecta una anomalía o patrón sospechoso, se genera una alerta mediante el **Servicio de Alertas**.
4. Todas las decisiones, reglas aplicadas y resultados del modelo se registran en el **Registro de Incidentes**.
5. La configuración de todos los componentes es gestionada dinámicamente por el **Servidor de Configuración**.

Diagrama C4 Nivel 3 – fraudDetector (Prevención de Fraudes)

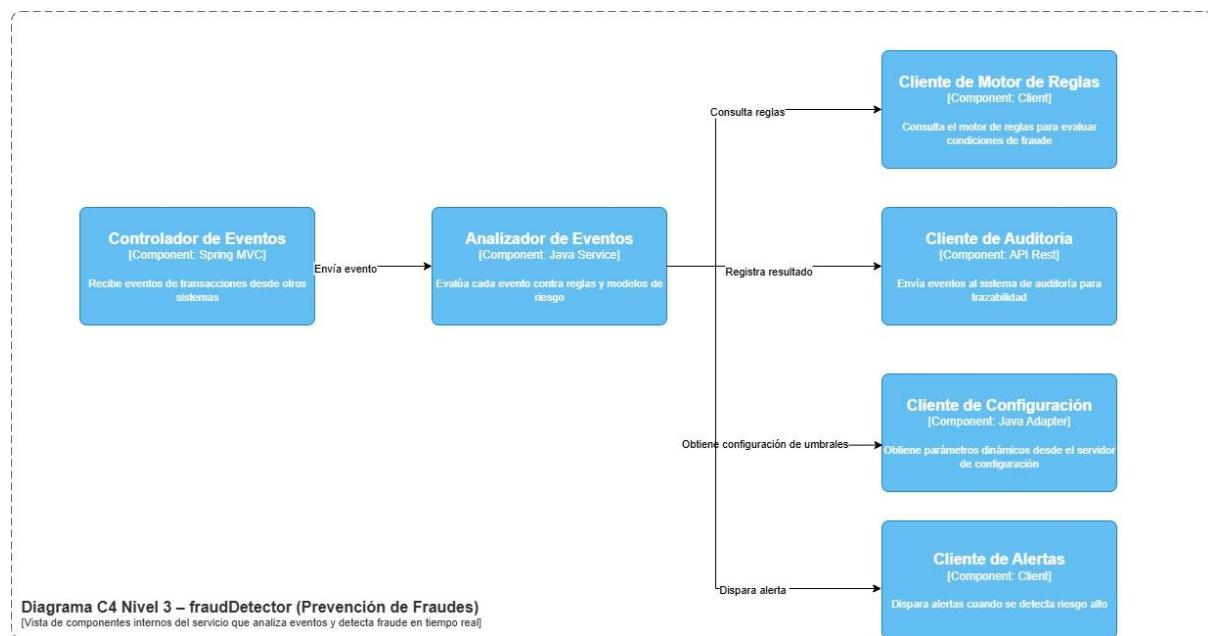


Diagrama C4 Nivel 3 – rulesEngine (Prevención de Fraudes)

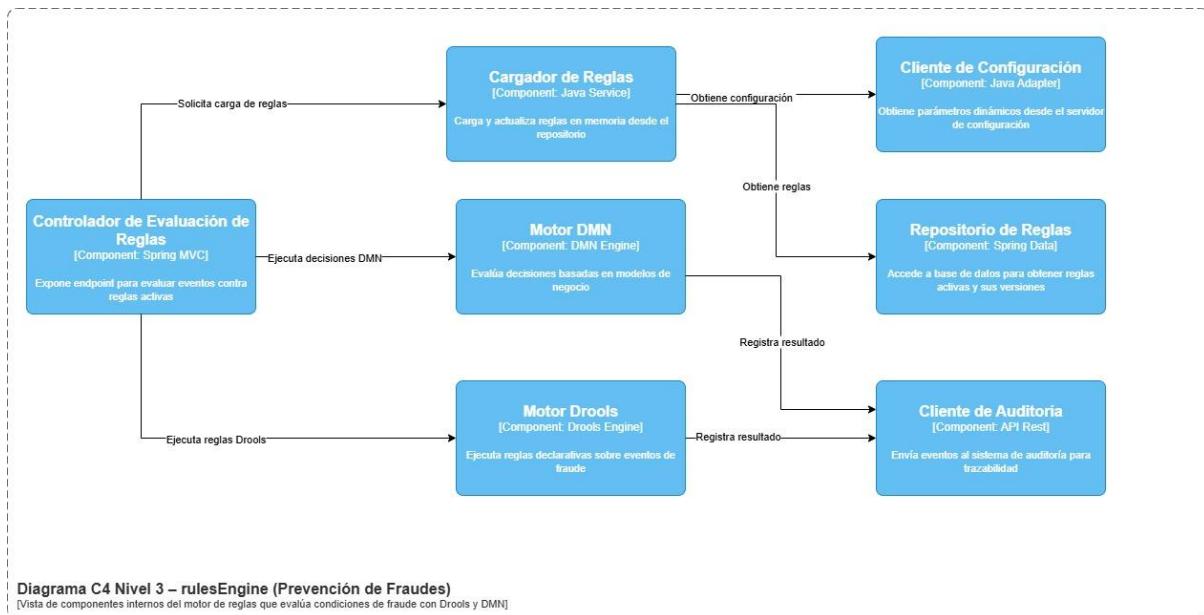


Diagrama C4 Nivel 3 – alertsService (Prevención de Fraude)

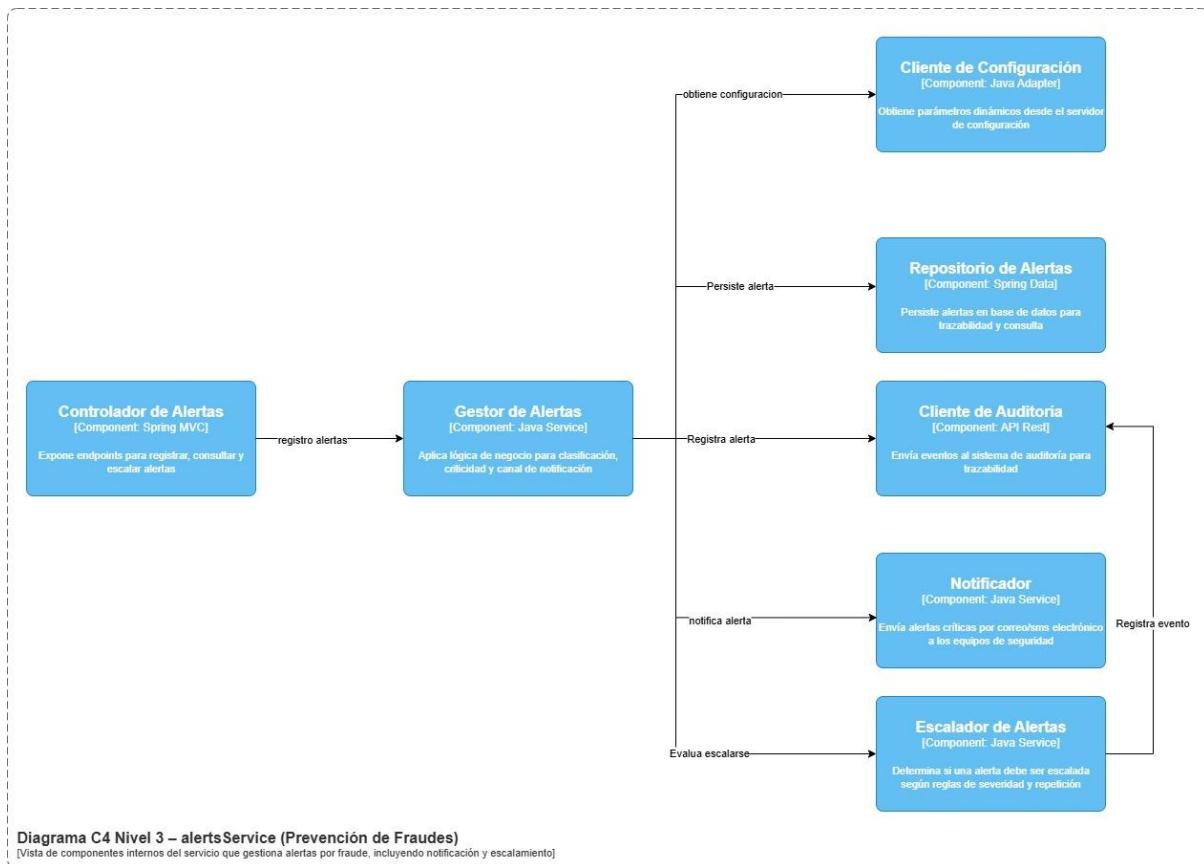
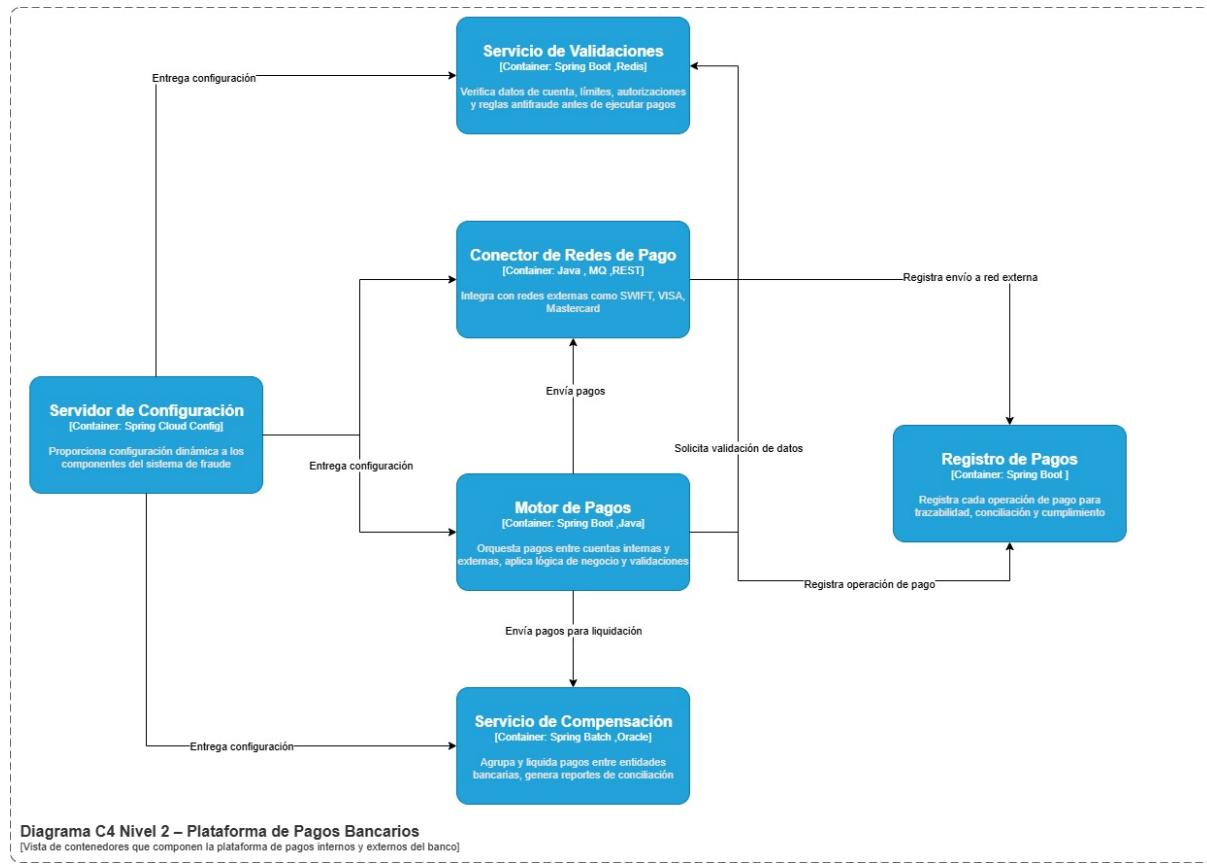


Diagrama C4 Nivel 2 – Plataforma de Pago



Explicación del Diagrama – Plataforma de Pagos Bancarios

Este sistema se encarga de procesar pagos internos y externos, aplicar validaciones, integrarse con redes de compensación y registrar cada operación para trazabilidad.

Motor de Pagos

- **Tecnología:** Spring Boot , Java
- **Propósito:** Es el núcleo de la plataforma. Orquesta el flujo completo de una operación de pago, desde la solicitud hasta la ejecución.
- **Responsabilidades:**
 - Coordinar validaciones, compensación y envío a redes externas.
 - Aplicar lógica de negocio como límites, tipos de cuenta, horarios y autorizaciones.
 - Registrar cada operación en el sistema de auditoría.

Servicio de Compensación.

- **Tecnología:** Spring Batch ,Oracle
- **Propósito:** Agrupa pagos entre entidades bancarias y genera procesos de liquidación y conciliación.
- **Responsabilidades:**
 - Ejecutar procesos batch para compensación interbancaria.
 - Generar reportes de conciliación y liquidación.
 - Registrar resultados en el sistema de auditoría.

Conektor de Redes de Pago

- **Tecnología:** Java + MQ / REST
- **Propósito:** Integra la plataforma con redes externas como SWIFT, VISA, Mastercard, etc.
- **Responsabilidades:**
 - Transformar y enviar mensajes según el protocolo de cada red.
 - Gestionar respuestas y errores de las redes externas.
 - Registrar cada envío en el sistema de auditoría.

Servicio de Validaciones

- **Tecnología:** Spring Boot , Redis

- **Propósito:** Verifica que cada operación cumpla con las reglas de negocio, límites, autorizaciones y controles antifraude.
- **Responsabilidades:**
 - Validar datos de cuenta, montos, horarios, autorizaciones.
 - Consultar reglas antifraude y límites operativos.
 - Responder al pagosCore con resultado de validación.

Registro de Pagos

- **Tecnología:** Spring Boot
- **Propósito:** Captura y almacena cada operación de pago, validación, compensación y envío externo para trazabilidad y cumplimiento.
- **Responsabilidades:**
 - Registrar quién pagó, cuánto, cuándo, cómo y con qué resultado.
 - Facilitar auditorías internas y regulatorias.

Servidor de Configuración

- **Tecnología:** Spring Cloud Config
- **Propósito:** Proporciona configuración centralizada y dinámica a todos los componentes de la plataforma.
- **Responsabilidades:**
 - Entregar parámetros como límites, rutas, claves, reglas y endpoints.
 - Permitir ajustes sin reiniciar servicios.
 - Versionar configuraciones por entorno.

Flujo General del Sistema

1. Una solicitud de pago llega al **Motor de Pagos**.
2. Se valida con el **Servicio de Validaciones**.
3. Si es válida:
 - Se envía a **Compensación** si es pago interno.
 - Se envía a **Redes Externas** si es interbancario o internacional.

4. Cada paso se registra en el **Registro de Pagos** para trazabilidad.
5. Todos los componentes reciben configuración dinámica desde el **Servidor de Configuración**.

Diagrama C4 Nivel 3 – pagosCore (Plataforma de Pagos)

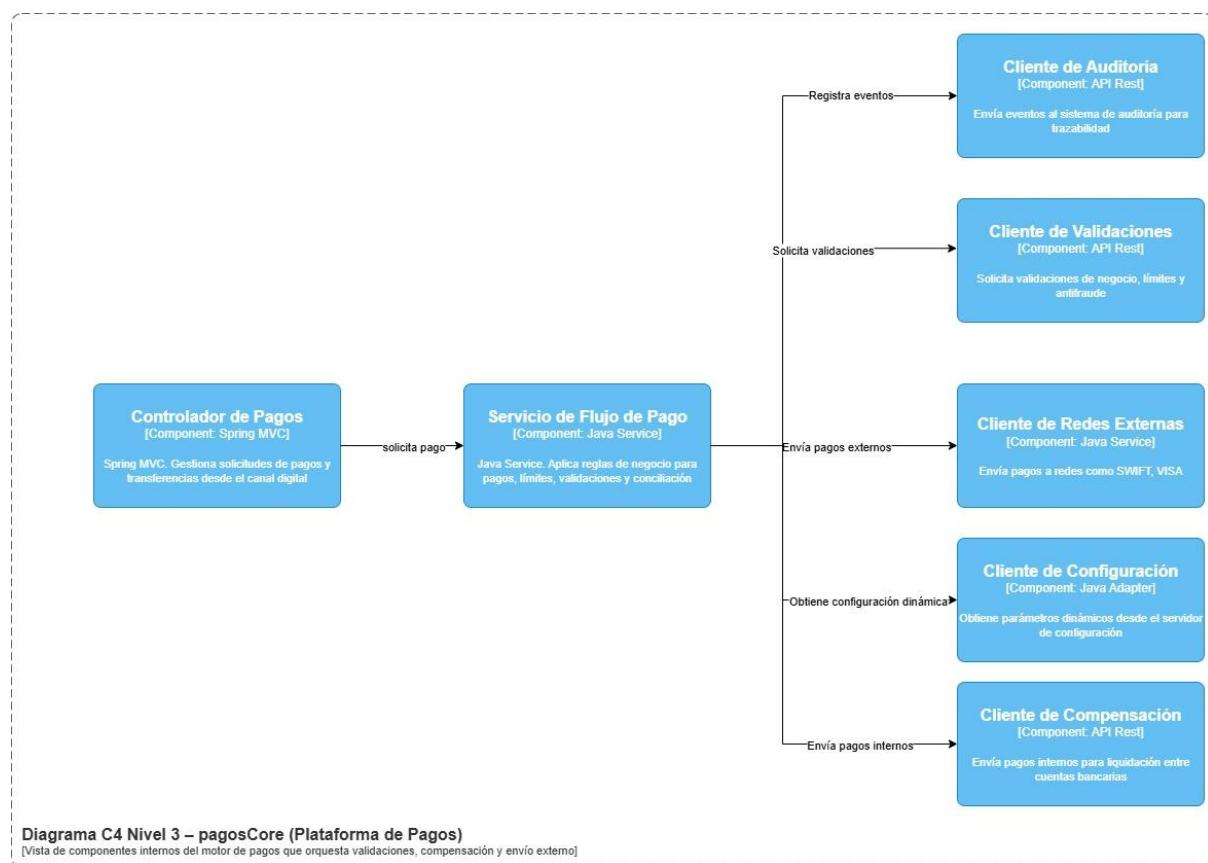


Diagrama C4 Nivel 3 – compensación (Plataforma de Pagos)

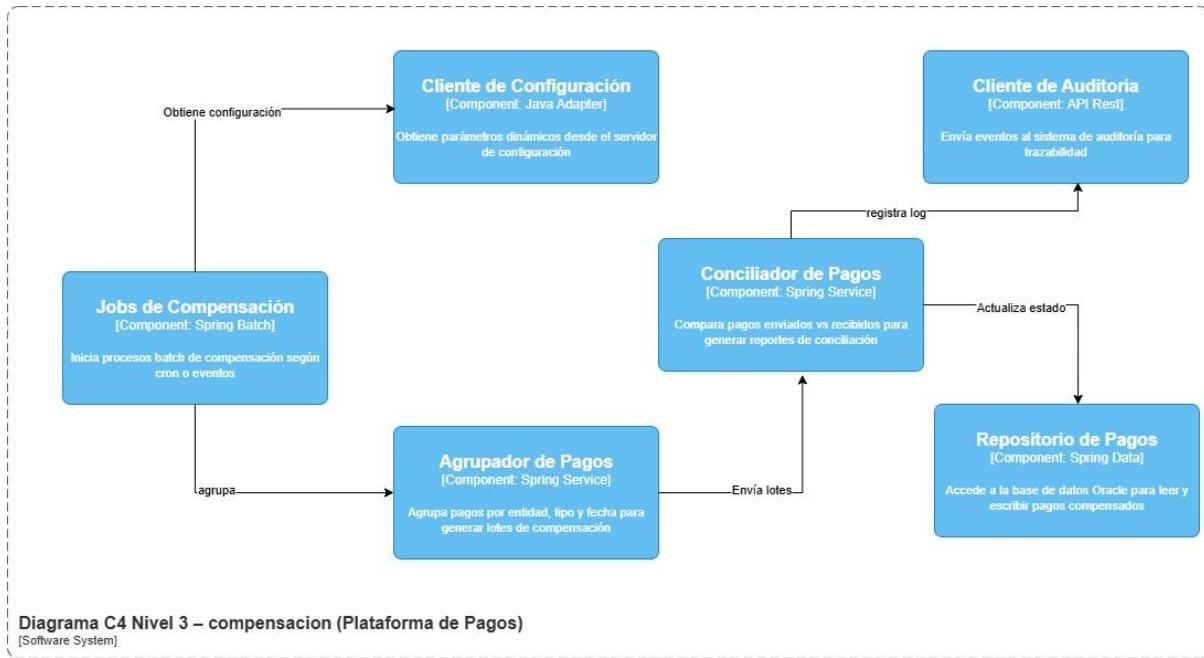


Diagrama C4 Nivel 3 – pagosExternos (Plataforma de Pagos Bancarios)

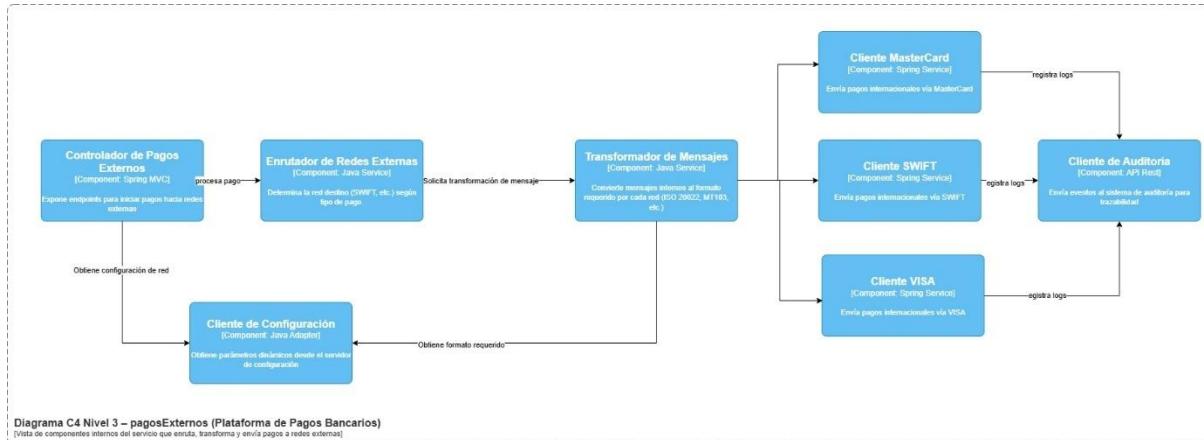
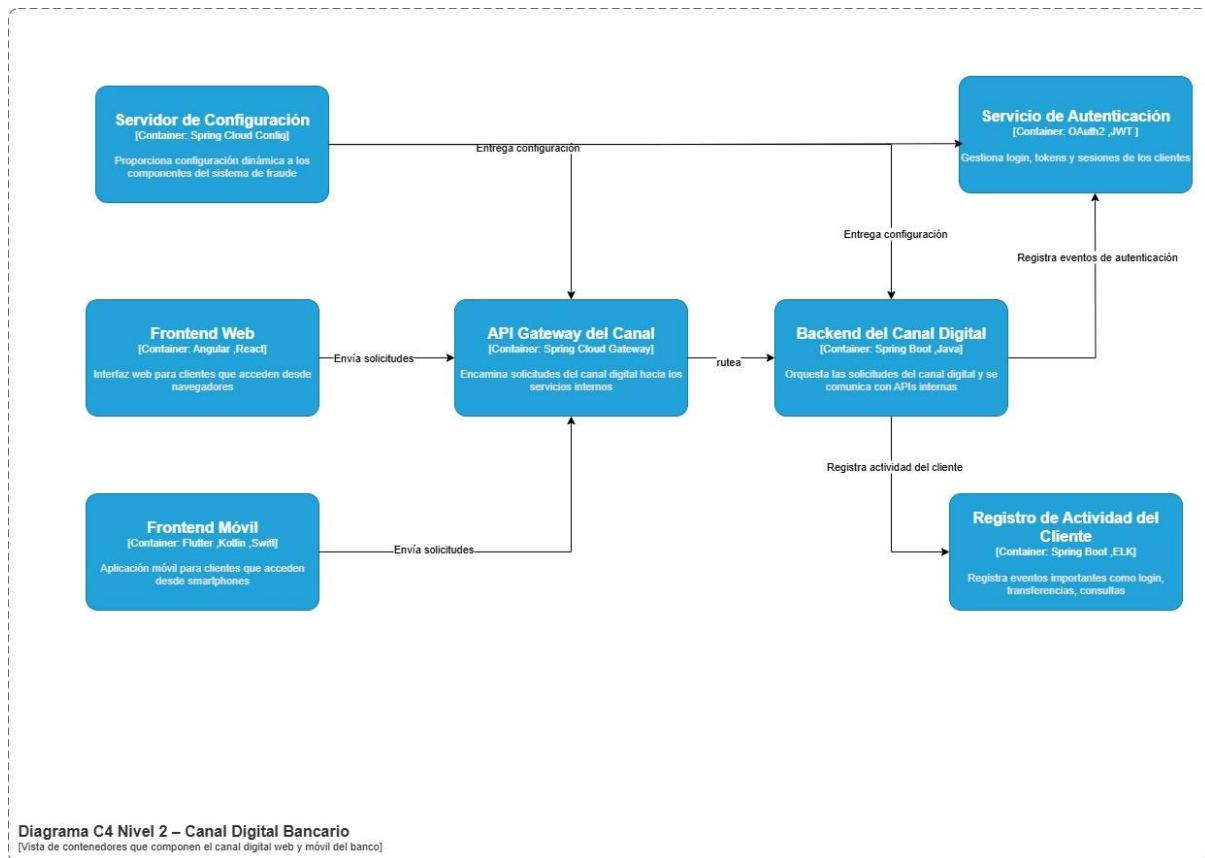


Diagrama C4 Nivel 2 – Canal Digital Bancario



Explicación del Diagrama – Canal Digital Bancario

Este sistema representa el conjunto de componentes que permiten a los clientes del banco interactuar con sus productos y servicios a través de canales digitales: aplicaciones web y móviles.

Frontend Web

- **Tecnología:** Angular ,React
- **Propósito:** Interfaz web accesible desde navegadores, orientada a clientes que operan desde computadoras o tablets.
- **Responsabilidades:**
 - Mostrar información de cuentas, saldos, movimientos y productos.
 - Permitir operaciones como pagos, transferencias, solicitudes.
 - Enviar solicitudes al API Gateway del Canal.

Frontend Móvil

- **Tecnología:** Flutter , Kotlin , Swift
- **Propósito:** Aplicación móvil nativa o híbrida para clientes que acceden desde smartphones.
- **Responsabilidades:**
 - Ofrecer experiencia optimizada para dispositivos móviles.
 - Integrar biometría, notificaciones push y almacenamiento local seguro.
 - Enviar solicitudes al API Gateway del Canal.

API Gateway del Canal (apiGateway)

- **Tecnología:** Spring Cloud Gateway
- **Propósito:** Encaminador que recibe solicitudes desde los frontends y las dirige al backend correspondiente.
- **Responsabilidades:**
 - Aplicar reglas de enrutamiento, autenticación y control de tráfico.
 - Desacoplar los canales digitales de los servicios internos.
 - Recibir configuración desde configServer.

Backend del Canal Digital

- **Tecnología:** Spring Boot , Java
- **Propósito:** Orquesta la lógica de negocio del canal digital, interactuando con APIs internas del banco.
- **Responsabilidades:**
 - Validar sesiones y tokens con el Servicio de Autenticación.
 - Registrar actividad del cliente en la auditoria
 - Transformar y enrutar solicitudes hacia sistemas internos como pagos, cuentas, productos.
 - Recibir configuración desde configServer.

Servicio de Autenticación

- **Tecnología:** OAuth2 , JWT

- **Propósito:** Gestiona el login, la emisión de tokens y la validación de sesiones de los clientes.
- **Responsabilidades:**
 - Validar credenciales y emitir tokens JWT.
 - Verificar sesiones activas y permisos.
 - Recibir configuración desde configServer.

Registro de Actividad del Cliente

- **Tecnología:** Spring Boot ,ELK
- **Propósito:** Captura y almacena eventos importantes generados por los clientes en el canal digital.
- **Responsabilidades:**
 - Registrar accesos, transferencias, pagos, consultas y errores.
 - Facilitar trazabilidad y cumplimiento normativo.
 - Integrarse con Kibana para visualización y análisis.

Servidor de Configuración (configServer)

- **Tecnología:** Spring Cloud Config
- **Propósito:** Proporciona configuración centralizada y dinámica a todos los componentes del canal digital.
- **Responsabilidades:**
 - Entregar parámetros como endpoints, claves, límites y rutas.
 - Permitir ajustes sin reiniciar servicios.
 - Versionar configuraciones por entorno (dev, test, prod).

Flujo General del Sistema

1. El cliente accede desde navegador (frontendWeb) o app móvil (frontendMovil).
2. La solicitud es enviada al API Gateway del Canal, que la enruta al backendCanal.
3. El backendCanal valida la sesión con el servicio de autenticacion y registra la actividad.
4. Si la operación es válida, se enruta hacia los sistemas internos del banco (fuera del alcance de este diagrama).

5. Todos los componentes reciben configuración dinámica desde el Servicio de configuración.

Diagrama C4 Nivel 3 – backendCanal (Canal Digital Bancario)

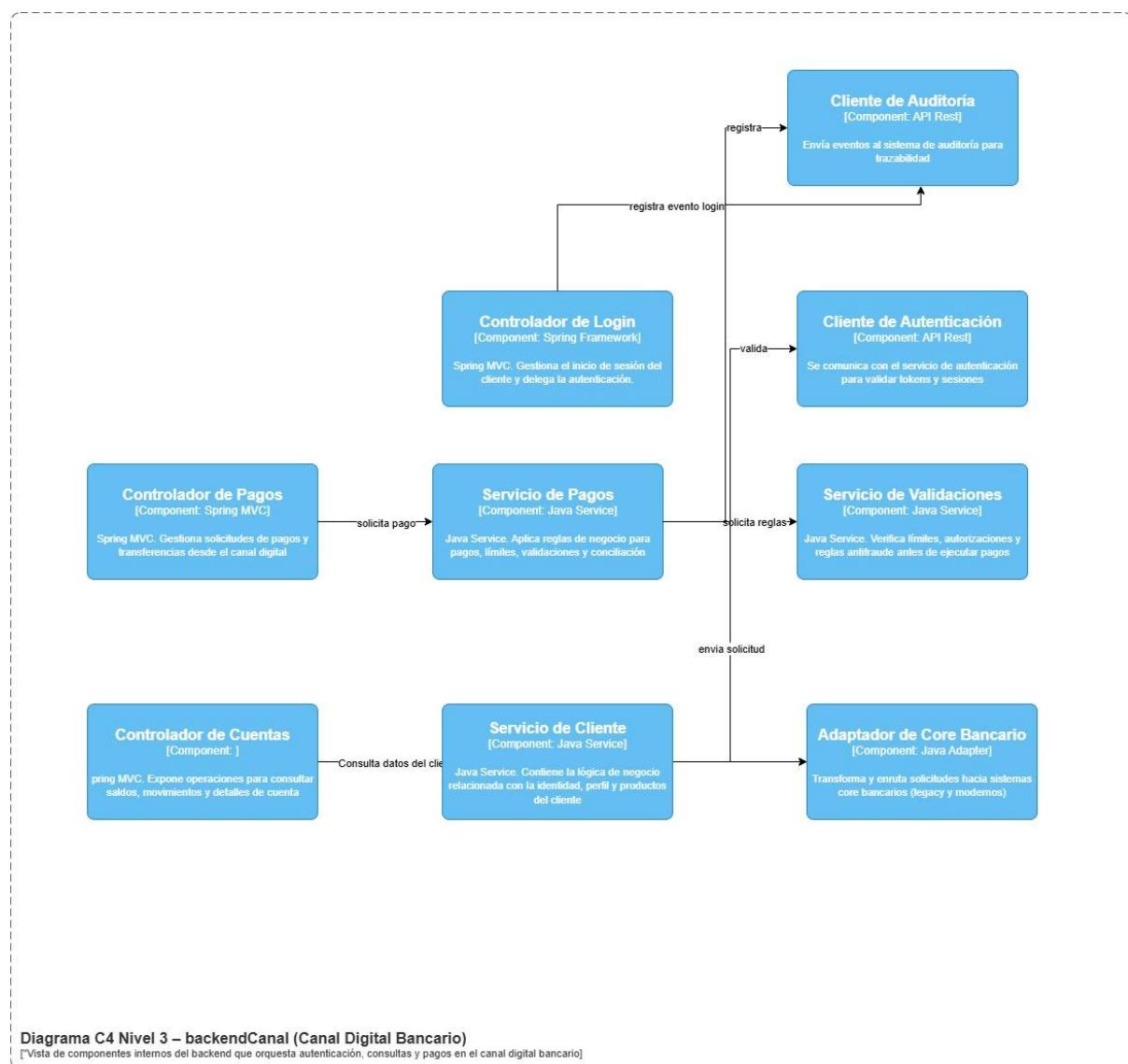
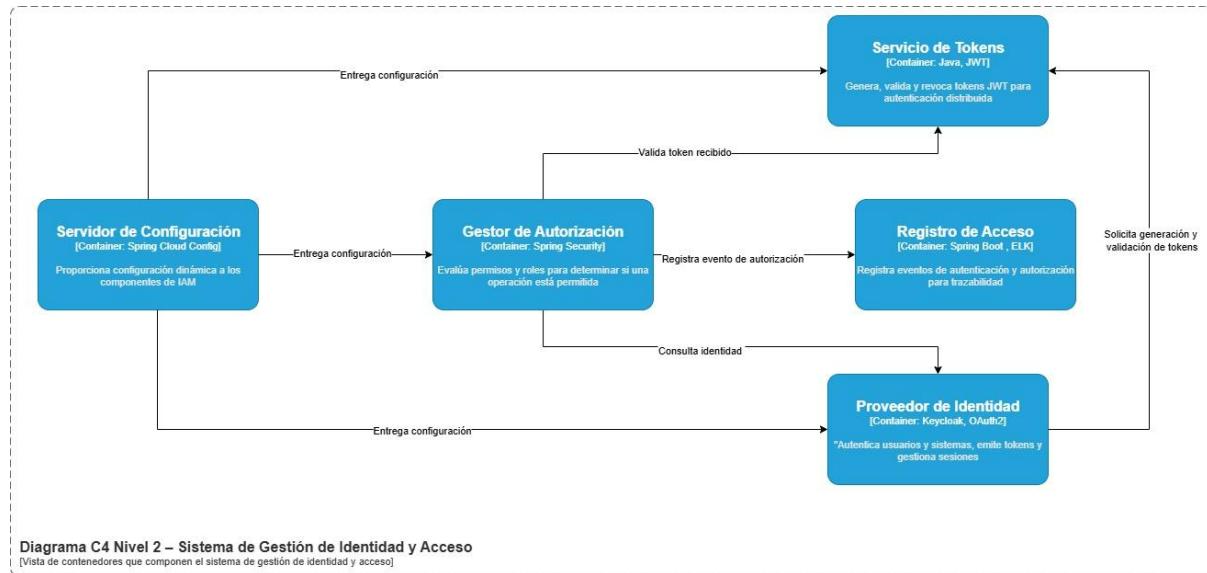


Diagrama C4 Nivel 2 – Sistema de Gestión de Identidad y Acceso



Explicación del Diagrama – Sistema IAM Bancario

Este diagrama representa los contenedores que conforman el sistema de gestión de identidad y acceso (IAM) dentro de tu arquitectura bancaria. Su propósito es garantizar que cada usuario o sistema que interactúa con los servicios bancarios esté correctamente autenticado, autorizado y trazado.

Proveedor de Identidad

- Tecnología:** Keycloak / OAuth2 / OpenID Connect
- Función:** Es el núcleo de autenticación del sistema. Valida credenciales de usuarios y sistemas, gestiona sesiones y emite tokens de acceso.

Gestor de Autorización

- Tecnología:** Spring Security + RBAC
- Función:** Evalúa si un usuario o sistema tiene permiso para realizar una operación específica, según sus roles y políticas de acceso.

Servicio de Tokens (tokenService)

- **Tecnología:** Java + JWT
- **Función:** Genera, valida y revoca tokens JWT que permiten autenticación distribuida entre sistemas.

Registro de Acceso

- **Tecnología:** Spring Boot y ELK
- **Función:** Captura y almacena todos los eventos de autenticación y autorización para trazabilidad, cumplimiento normativo y análisis de seguridad.

Servidor de Configuración

- **Tecnología:** Spring Cloud Config
- **Función:** Proporciona configuración centralizada y dinámica a todos los componentes del sistema IAM.

Flujo General del Sistema

1. Un usuario o sistema solicita acceso a un recurso protegido.
2. El **Proveedor de Identidad** valida sus credenciales y solicita al **Servicio de Tokens** la emisión de un JWT.
3. El **Gestor de Autorización** recibe el token y lo valida con el **Servicio de Tokens**.
4. Si el token es válido, el **Gestor de Autorización** evalúa los permisos del usuario y decide si la operación está permitida.
5. Tanto la autenticación como la autorización se registran en el **Registro de Acceso** para trazabilidad.
6. Todos los componentes reciben configuración dinámica desde el **Servidor de Configuración**, lo que permite ajustes sin reinicios.

Arquitectura de Integración Bancaria

La arquitectura de integración propuesta se basa en una visión modular, escalable y alineada con estándares internacionales como BIAN (Banking Industry Architecture Network). Cada sistema —Canal Digital Bancario, Plataforma de Pagos y Prevención de Fraudes— ha sido modelado con diagramas C4 en sus niveles de contexto, contenedores y componentes, permitiendo una comprensión clara y estructurada de la solución. El uso de BIAN permite mapear cada contenedor a dominios funcionales reconocidos, facilitando la interoperabilidad entre sistemas, la trazabilidad de procesos y la alineación con marcos regulatorios. Esta arquitectura no solo responde a necesidades técnicas, sino que también comunica valor al cliente final, al demostrar gobernanza, claridad y visión de largo plazo.

Los patrones de integración aplicados han sido cuidadosamente seleccionados y justificados en función del rol de cada sistema. En el Canal Digital, se emplea el patrón Backend for Frontend (BFF) para adaptar la lógica según el canal (web o móvil), mientras que el API Gateway centraliza el acceso y aplica políticas de seguridad. En la Plataforma de Pagos, se utiliza orquestación interna en pagosCore para coordinar validaciones, compensación y envío externo, mientras que los servicios como compensacion y pagosExternos aplican patrones batch y enrutamiento por tipo de red. En Prevención de Fraudes, se combinan patrones síncronos y asíncronos para análisis en tiempo real, con desacoplamiento entre detección, reglas y alertas. Las tecnologías seleccionadas (Spring Boot,, Drools, DMN, Kafka) responden a criterios de robustez, trazabilidad, escalabilidad y compatibilidad con entornos bancarios.

La solución aborda de forma integral los requisitos de seguridad y cumplimiento normativo, incluyendo la Ley Orgánica de Protección de Datos Personales vigente en Ecuador. Cada sistema incorpora autenticación federada con OAuth2 y tokens JWT, autorización basada en roles, cifrado en tránsito (TLS) y en reposo, y trazabilidad completa mediante componentes como de auditoria. Además, se aplican políticas de retención, anonimización y control de sesión, garantizando que los datos personales sean tratados conforme a principios de licitud, finalidad y proporcionalidad. La arquitectura también contempla segmentación de datos sensibles, control de acceso por scopes y monitoreo activo de eventos críticos.

La estrategia de alta disponibilidad y recuperación ante desastres está implementada de forma práctica y adaptada al diseño. Los contenedores críticos como backendCanal, pagosCore, fraudDetector y alertsService se despliegan en clústeres redundantes con balanceadores de carga, persistencia distribuida y backups automáticos. Se emplean mecanismos de failover automático, monitoreo activo y alertas en tiempo real, permitiendo continuidad operativa incluso ante fallos parciales o totales. Esta estrategia no se limita a teoría, sino que se refleja en la configuración concreta de cada componente y en la forma en que se gestionan los estados y las dependencias.

La integración multicore se aborda mediante adaptadores específicos en componentes como coreAdapter y pagosService, que permiten coexistencia entre sistemas legacy (COBOL) y modernos (REST, ISO 20022). Esta estrategia se refleja en la arquitectura del Canal Digital y la Plataforma de Pagos, donde los contenedores orquestadores transforman y enrutan solicitudes hacia múltiples cores bancarios sin acoplamiento directo. Esto permite una evolución progresiva del core sin afectar la experiencia del cliente ni la lógica de negocio del canal.

La gestión de identidad y acceso se ha delineado con mecanismos robustos y consistentes en todos los sistemas. Se utiliza autenticación federada con Keycloak, tokens JWT firmados, autorización por scopes y roles, y control de sesión. Los componentes como authService, authClient y configClient garantizan que cada operación esté validada, auditada y controlada. Además, se integran con sistemas IAM corporativos para usuarios internos, y se aplican políticas de revocación, expiración y trazabilidad en cada interacción.

La estrategia de APIs internas y externas se basa en estándares de industria como OpenAPI, ISO 20022 y JSON. Las APIs RESTful se exponen desde contenedores como backendCanal, pagosCore, fraudDetector y alertsService, con documentación estructurada, versionado, validación de payloads y control de tráfico. Se contempla mensajería asíncrona para eventos críticos, y se aplican políticas de seguridad, rate limiting y monitoreo. Esta estrategia permite interoperabilidad, escalabilidad y apertura hacia modelos de Open Banking.

El modelo de gobierno de APIs y microservicios incluye catálogo de APIs, portal de desarrolladores, políticas de ciclo de vida (publicación, revisión, retiro), observabilidad (logs, métricas, trazas) y revisión de contratos. Se propone un comité de arquitectura para gobernanza técnica y funcional, y se aplican prácticas de revisión continua, pruebas automatizadas y control de versiones. Este modelo garantiza que la evolución de los servicios sea ordenada, segura y alineada con los objetivos del negocio.

Finalmente, el plan de migración gradual se diseña para minimizar el riesgo operativo. Se inicia con la exposición de APIs desde el canal digital sin modificar el core, luego se integran adaptadores para coexistencia, se migran funcionalidades específicas hacia microservicios, y se desacopla completamente el canal del core legacy. Cada fase incluye pruebas, monitoreo y rollback controlado, priorizando funcionalidades de alto impacto como login, pagos y alertas. Esta estrategia permite una transición segura, controlada y alineada con la capacidad operativa del banco.