[Año]

ARQUITECTURA DE INTEGRACION

RETO

GEORGE DE LA ROSA

Tabla de contenido

Visión General

La arquitectura v4 modela una plataforma de banca digital desacoplada y nativa de la nube, diseñada para cumplir con Requisitos No Funcionales (NFRs) críticos de **alto rendimiento**, **alta disponibilidad (HA/DR)** y **cumplimiento normativo estricto**.

La estrategia central se basa en dos pilares:

1. **Modernización de Legacy:** Uso del patrón **Strangler Fig (ACL)** para migrar progresivamente el Core Legacy sin riesgo.
2. **Optimización de Latencia (CX):** Uso de **Arquitectura Orientada a Eventos (EDA)** para desacoplar operaciones de cumplimiento (Auditoría, Notificaciones) del flujo transaccional principal (*happy path*).

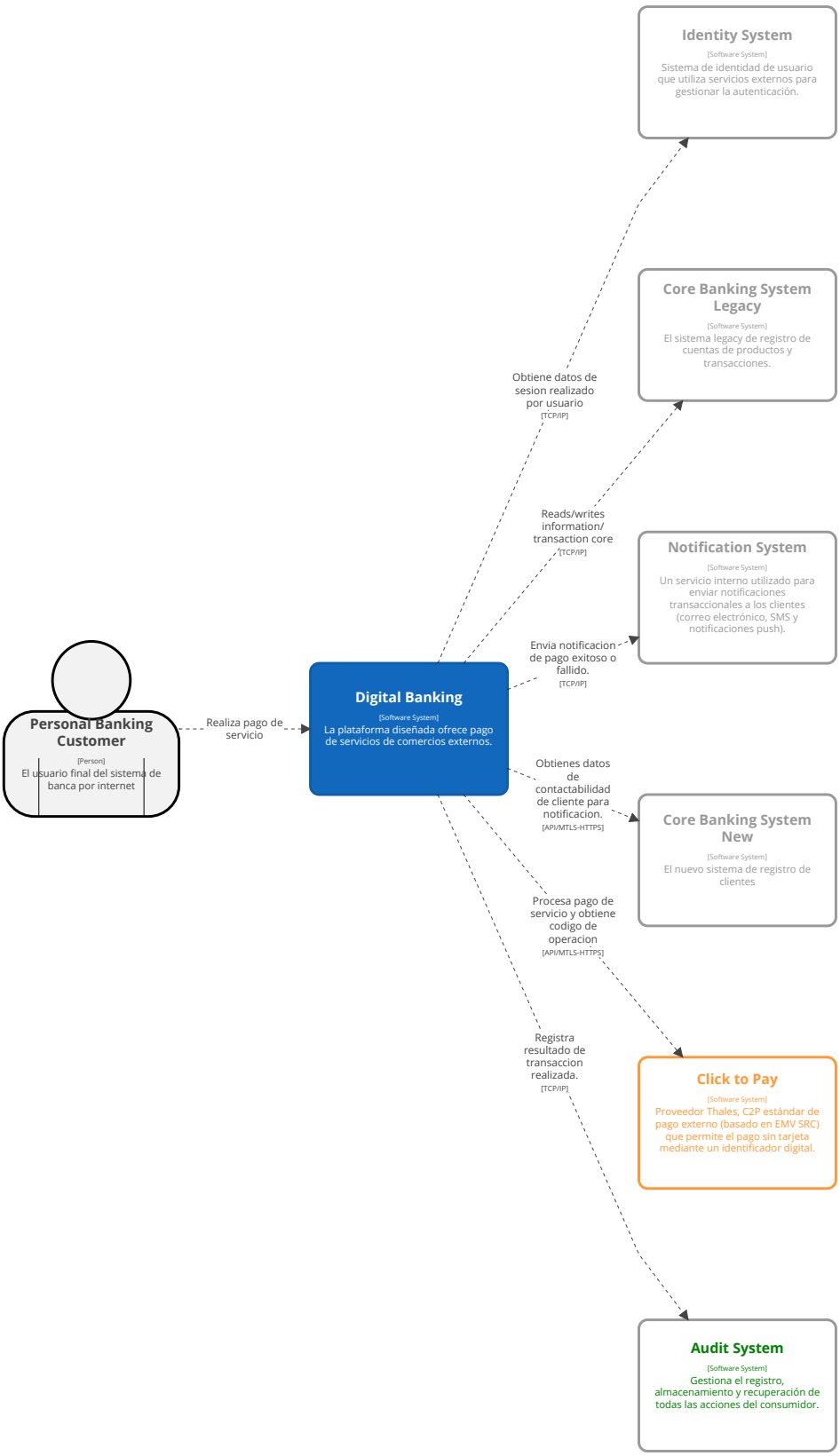
Diagrama de contexto

Este diagrama ilustra el alcance del "Sistema de Pago de Servicios Digitales".

La funcionalidad principal permite a los **Usuarios (Personal Banking Customer)** realizar pagos de servicios a través de la **Aplicación Móvil** y la **Plataforma Web**.

Para operar, el sistema se comunica con varios sistemas transversales:

- **Sistemas Internos:** Componentes de la entidad bancaria (Core Banking System Legacy, Core Banking System New, Identity System, Audit System, Notification System).
- **Sistemas Externos:** Entidades fuera de la organización (Okta, Civil Registry Ecuador, Open Finance, Click top ay).



System Context View: Digital Banking

martes, 11 de noviembre de 2025, 23:25 hora estándar de Perú

Diagrama de container

El sistema utiliza un **proceso sincrónico** para gestionar las transacciones de pago de servicios. Los servicios centrales han sido diseñados siguiendo la arquitectura **BIAN (Banking Industry Architecture Network)**, lo cual es un requisito clave para el sistema financiero y promueve un diseño modular y desacoplado.

Esta arquitectura facilita la integración con **sistemas desacoplados** que complementan el flujo de pagos de servicio, haciendo el proceso más robusto y seguro.

Asimismo, se compone de varios contenedores (microservicios y bases de datos) que implementan los siguientes patrones clave:

1. Patrón Backend For Frontend (BFF)

- **Contenedor:** Experience Bank Service
- **Propósito:** Se aplica el patrón BFF para conseguir una integración más limpia y acorde a las necesidades específicas de cada aplicación frontend (App mobile y Web).
- **Beneficio de Rendimiento:** Este enfoque mejora el performance de las interfaces de usuario (UI), ya que el BFF gestiona y optimiza la información demandada (ej. desencriptación RSA de la capa pública) y reduce el "chattiness" (ruido) de la comunicación.

2. Patrón Cache-Aside (Prevención de Duplicados)

- **Contenedor:** Cache (Redis)
- **Tecnología:** Se utiliza Redis como almacén de datos en memoria.
- **Función:** La capa de caché, consultada por el Payment Initiation Service, realiza la validación inicial para determinar si una transacción de pago de servicio es duplicada (verificación de idempotencia), previniendo la ejecución duplicada del flujo.

3. Estrategia de Migración: Patrón Strangler Fig y ACL

- **Contenedor:** Conector Core Legacy Service
- **Patrones:** Pattern: ACL (Anti-Corruption Layer), Pattern: Strangler Fig
- **Propósito:** Este contenedor es la clave de la estrategia de transformación digital. Actúa como una fachada que "estrangula" al Core Banking System Legacy.
- **Función 1 (ACL):** Oculta la complejidad del *legacy* (protocolos, SOAP, etc.) y expone una API REST limpia a los microservicios modernos.
- **Función 2 (Strangler):** Permite la migración incremental. A medida que la funcionalidad se migra del *legacy* a nuevos servicios (como Core Banking System New), este ACL se actualiza para enrutar las llamadas a los nuevos servicios, sin que sus consumidores se vean afectados.

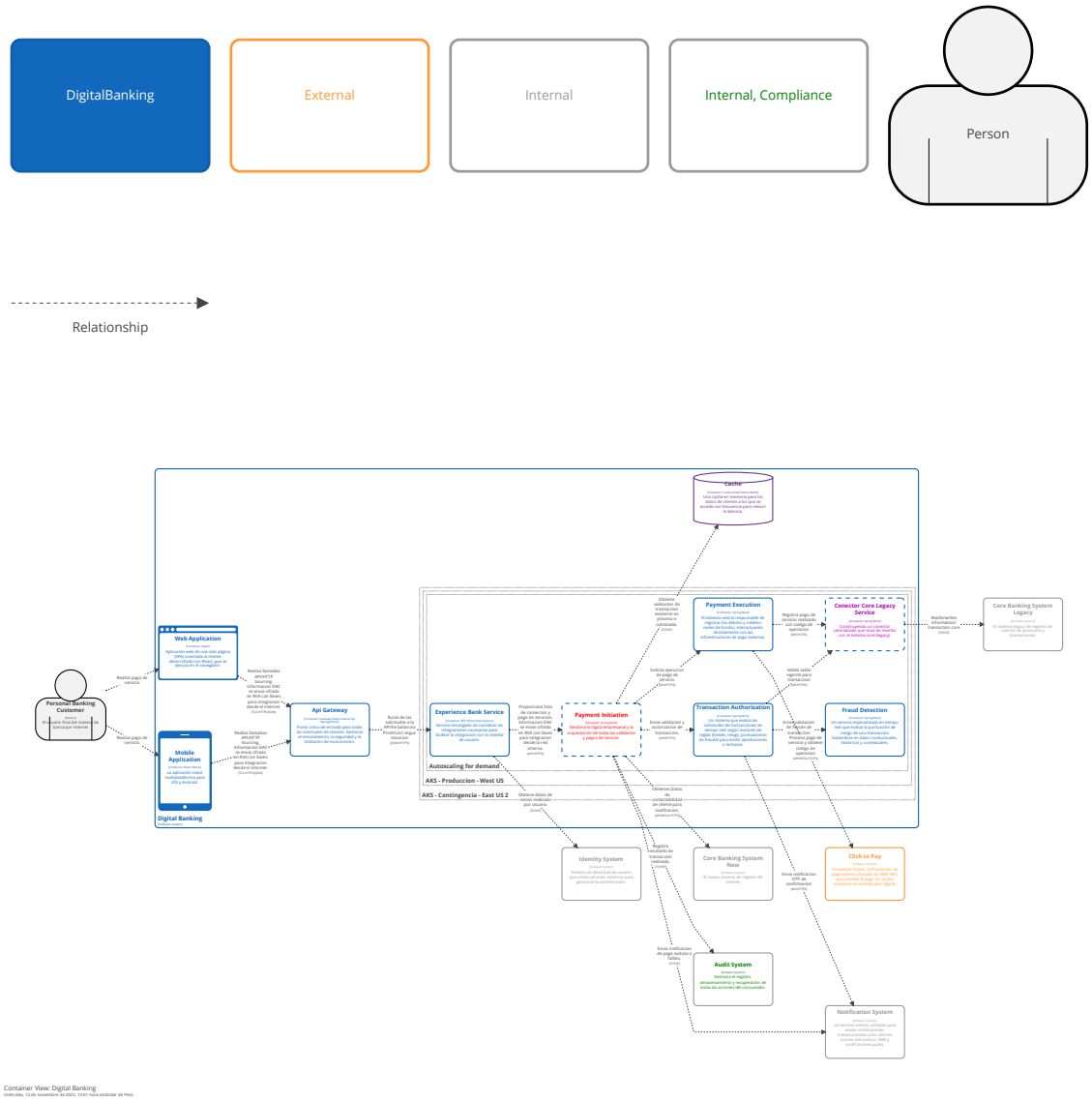
4. Autenticación y Validación de Identidad (Onboarding)

- **Sistema:** Identity System
- **Flujo:** El *onboarding* se ejecuta de manera sincrónica para la validación inmediata del perfil, integrando:
    1. Validación Biometría: Uso del Civil Registry (Ecuador).
    2. Gestión de Identidad (IdP): Almacenamiento en Okta (SSO).
    3. Enriquecimiento de Datos: Integración con Open Finance.
- **Gestión de Sesiones (Cache-Aside):** Al autenticarse con éxito (Authentication Service), la sesión del usuario se almacena en el identitySystem.cache (Redis) para ser reutilizada rápidamente por otros procesos (como el BFF), optimizando la experiencia.

5. Sistemas Asíncronos (EDA) para Cumplimiento y Notificaciones

Para garantizar la mínima latencia en el flujo de pago, las operaciones de auditoría y notificación se manejan de forma asíncrona mediante una **Arquitectura Orientada a Eventos (EDA)**.

- **Sistema de Notificaciones (Notification System)**
    - **Arquitectura:** Usa los patrones **Event Driven** y **Event Sourcing**. El Payment Initiation Service publica un evento (ej. notificar_pago_exitoso) en el notificationBroker (Event Hubs).
    - **Consumidor:** El notificationsService (Azure Function) consume el evento en segundo plano.
    - **Resiliencia:** Las notificaciones fallidas se registran en notifyDatabase para ser reintentadas por un subproceso, garantizando la entrega.
- **Sistema de Auditoría Transversal (Audit System)**
    - **Arquitectura:** Idéntica a Notificaciones, es asíncrona (EDA).
    - **Propósito:** Cumplir con las reglas normativas de la entidad reguladora.
    - **Beneficio:** El registro de auditoría se realiza **en segundo plano** sin interrumpir el pago del cliente.
    - **Resiliencia:** Incluye un proceso de reintento (auditService -> messageBroker) para manejar fallos de conectividad con la auditDatabase.



Container View: Digital Banking

Diagrama de Componentes

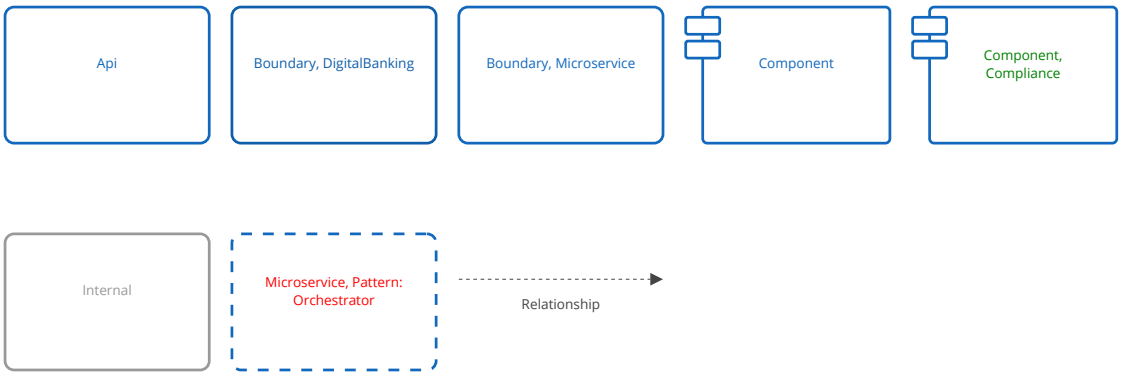
Este nivel detalla la estructura interna de los contenedores (microservicios) más críticos de la arquitectura.

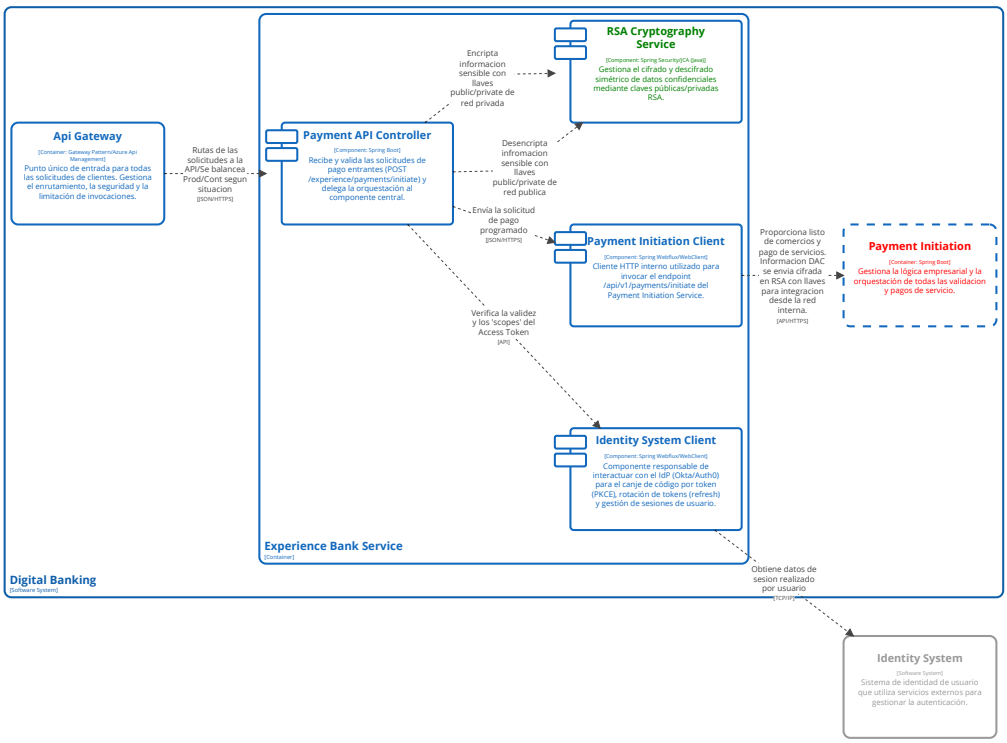
1. Contenedor: Experience Bank Service (El BFF)

Este contenedor actúa como el **Backend For Frontend**. Su responsabilidad principal es servir como una fachada optimizada para las aplicaciones (Web y Móvil), centralizando la lógica de experiencia de usuario.

- **Payment API Controller:** Punto de entrada (entrypoint) del BFF. Recibe las llamadas desde el Api Gateway.

- **Identity System Client:** Componente de seguridad que valida el Access Token del usuario contra el Identity System antes de continuar el flujo.
- **RSA Cryptography Service:** Componente de seguridad que desencripta el payload sensible proveniente de la red pública (internet) usando la llave privada.
- **Payment Initiation Client:** Un cliente HTTP (proxy) que delega la solicitud de pago (ya validada y desencriptada) al siguiente microservicio: el Payment Initiation Service.



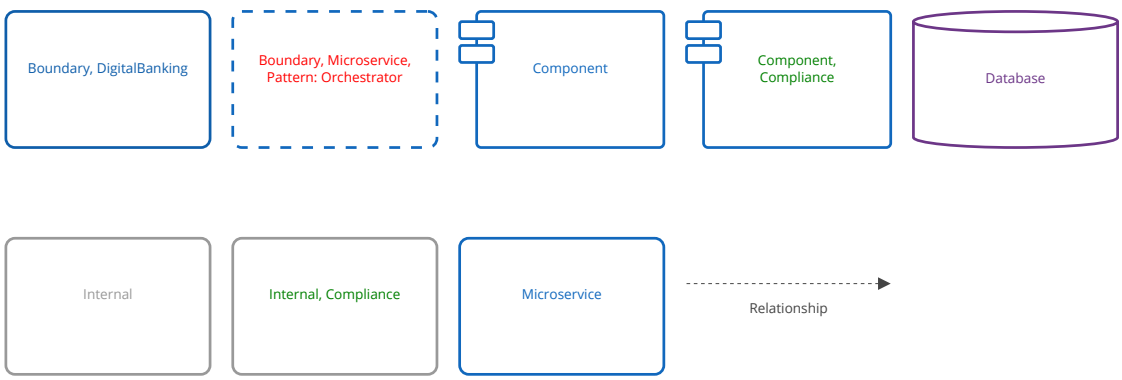


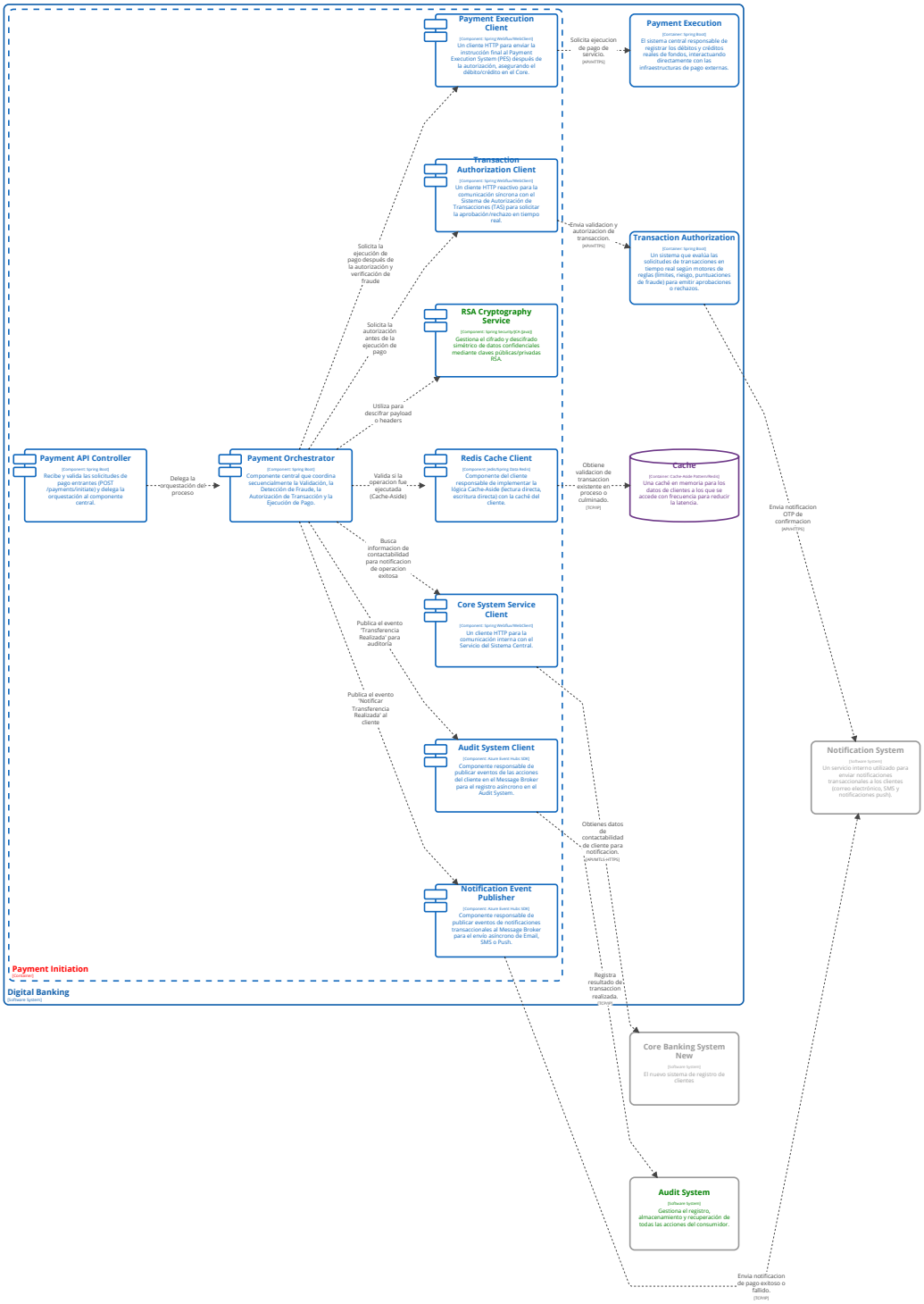
Component View: Digital Banking - Experience Bank Service
miércoles, 12 de noviembre de 2025, 13:01 hora estándar de Perú

2. Contenedor: Payment Initiation (El Orquestador)

Este contenedor es el **cerebro de la transacción** y aplica el **Patrón Orquestador** para coordinar el flujo de pago sincrónico.

- **Payment API Controller:** Punto de entrada del servicio, recibe llamadas del BFF.
- **Payment Orchestrator:** El componente principal que contiene la lógica de negocio. Coordina a todos los demás clientes en la secuencia correcta.
- **Redis Cache Client:** Componente que implementa la lógica de **Cache-Aside**. El orquestador lo usa para consultar el digitalbanking.cache y verificar si la transacción es duplicada (idempotencia).
- **tasClient (Transaction Authorization Client):** Cliente HTTP que llama al Transaction Authorization Service para la validación de reglas de negocio, fraude y saldo.
- **pesClient (Payment Execution Client):** Cliente HTTP que llama al Payment Execution Service para la ejecución final del pago contra el proveedor externo (Click to Pay) y el *legacy*.
- **coreClient:** Cliente HTTP que consulta el Core Banking System New para obtener datos de contacto del cliente (necesarios para la notificación).
- **auditClient:** Componente (SDK de Azure Event Hubs) que publica el evento de auditoría (ej. "Transferencia Realizada") en el auditSystem.messageBroker.
- **notificationPublisher:** Componente (SDK de Azure Event Hubs) que publica el evento de notificación (ej. "Notificar Transferencia Realizada") en el notificationSystem.notificationBroker.





Component View: Digital Banking - Payment Initiation

miércoles, 12 de noviembre de 2025, 13:01 hora estándar de Perú

Flujos Arquitectónicos Clave

1. Flujo Síncrono: Pago de Servicios (Bajo Impacto en Latencia)

El flujo de pago es una coreografía sincrónica diseñada para la velocidad.

1. **Frontend (SPA/Mobile) -> Api Gateway (Azure APIM):** El cliente inicia el pago. La carga útil (payload) viaja encriptada con RSA. El APIM gestiona el balanceo de carga HA/DR entre las regiones de AKS (West/East US).

2. **Api Gateway -> Experience Bank Service (BFF):** El BFF actúa como fachada para la UI. Desencripta el payload (RSA) y valida la sesión del usuario llamando al identitySystem.authenticationService.
3. **BFF -> Payment Initiation (Orquestador):** El BFF delega la lógica de negocio al orquestador.

4. **Payment Initiation (Orquestador):** Este es el cerebro del flujo:
   a) (Cache-Aside): Valida la idempotencia de la transacción contra digitalbanking.cache (Redis).
   b) (Autorización): Llama a transactionAuthorizationService.
   c) (Ejecución): Llama a paymentExecutionService para la ejecución final.

5. **Autorización (TAS):**
   a) Llama a fraudDetectionService.
   b) Llama a notificationSystem (Broker) para enviar un OTP (requerido para continuar).
   c) Llama al coreLegacyService (ACL) para validar el saldo contra el coreLegacySystem.

6. **Ejecución (PES):**
   a) Llama al clickToPayService (Externo) para procesar el pago.
   b) Llama al coreLegacyService (ACL) para registrar la transacción en el legacy.

2. Flujo Asíncrono: Auditoría y Notificaciones (Cero Impacto en Latencia)

Para proteger la latencia del pago, las operaciones de cumplimiento y notificación se manejan fuera de banda (out-of-band).

1. **Payment Initiation (Orquestador):** Al finalizar el flujo síncrono (éxito o fallo), el orquestador publica dos eventos en paralelo (Fire-and-Forget):

a) **Evento de Auditoría:** (Ej. pago_realizado) -> auditSystem.messageBroker.
b) **Evento de Notificación:** (Ej. notificar_pago_exitoso) -> notificationSystem.notificationBroker.

2. **Consumidores (Event Consumers):**

a) **auditService (Azure Function):** Consume el evento de auditoría y lo persiste en auditDatabase. Implementa una estrategia de reintentos (vía DLQ) para garantizar la entrega.
b) **notificationsService (Azure Function):** Consume el evento de notificación, busca la plantilla (Azure Storage) y la envía vía ACS (Email/SMS) o ANH (Push).

Decisiones Clave de Arquitectura (ADRs)

- **ADR-001: Estrategia de Migración (Strangler Fig + ACL)**
    - **Decisión:** Se implementa el coreLegacyService como un **Anti-Corruption Layer (ACL)** y la fachada del **Patrón Estrangulador (Strangler Fig)**.
    - **Justificación:** Permite la coexistencia del *legacy* y los nuevos microservicios. Reduce el riesgo de la migración ("Big Bang") y permite la modernización incremental. Todo el acceso al coreLegacySystem DEBE pasar por este ACL.
- **ADR-002: Desacoplamiento de Carga (EDA)**
    - **Decisión:** Las operaciones de Auditoría y Notificaciones transaccionales son asíncronas, usando Azure Event Hubs.
    - **Justificación:** Garantiza que los NFRs de cumplimiento (auditoría) no impacten la latencia de la experiencia del cliente (pago). Proporciona resiliencia; si el sistema de auditoría está caído, el pago del cliente no falla.
- **ADR-003: Alta Disponibilidad (HA/DR)**
    - **Decisión:** Los contenedores de DigitalBanking se despliegan en clústeres de AKS georreplicados (West US y East US 2).
    - **Justificación:** El Api Gateway gestionará el balanceo de carga activo/pasivo o activo/activo, garantizando la continuidad del negocio ante la falla de una región.

- **ADR-004: Optimización de Frontend (BFF)**
    - **Decisión:** Se usa un Experience Bank Service (Backend for Frontend).
    - **Justificación:** Provee una API optimizada para SPA y Mobile, reduce el *chattiness* y centraliza la lógica de experiencia (como la desencriptación RSA de la capa pública).
- **ADR-005: Seguridad de Datos (Defensa en Profundidad)**
    - **Decisión:** Se aplica encriptación RSA en la capa pública (Cliente -> BFF) y en la capa interna (BFF -> Orquestador). La identidad se delega a Okta (OIDC/OAuth2).
    - **Justificación:** Cumplimiento con PCI/Normativa. Protege datos sensibles (DAC) no solo en el borde, sino también en la red interna.
- **ADR-006: Prevención de Duplicados (Cache-Aside)**
    - **Decisión:** Se utiliza un caché de Redis (digitalbanking.cache) en el paymentInitiationService (Orquestador).
    - **Justificación:** Se implementa el patrón Cache-Aside para verificar la existencia de un ID de transacción antes de procesarlo, garantizando la idempotencia y previniendo pagos duplicados.

Anexos

1. Diagramas Identity

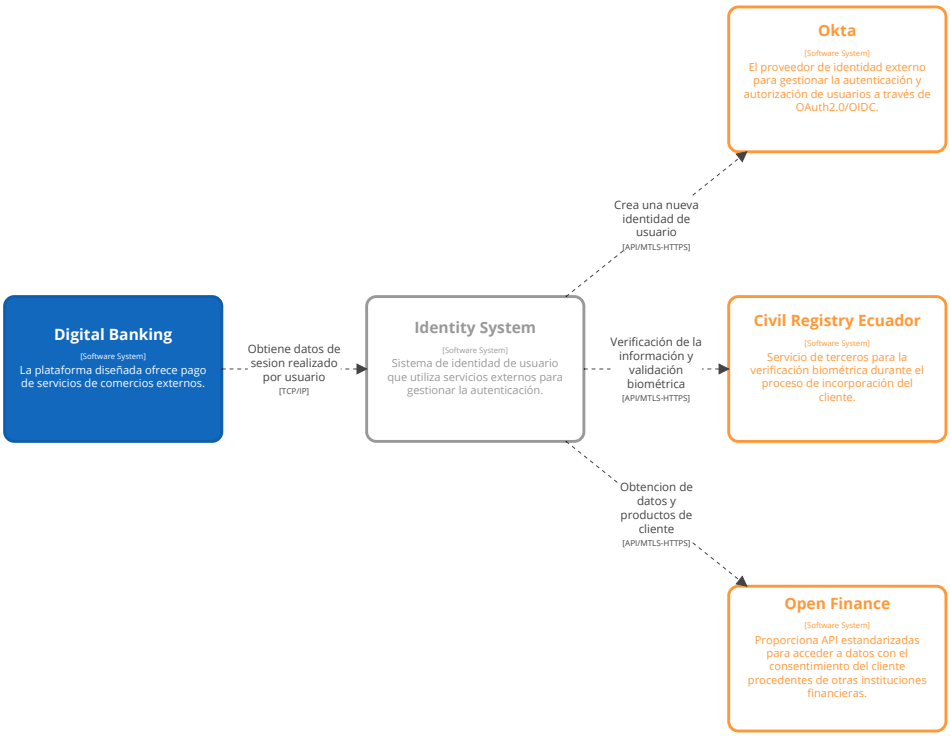
Autenticación y Validación de Identidad (Onboarding)

El sistema incorpora un enfoque transversal para la gestión de la identidad y el *onboarding* de nuevos usuarios, asegurando procesos robustos y verificables:

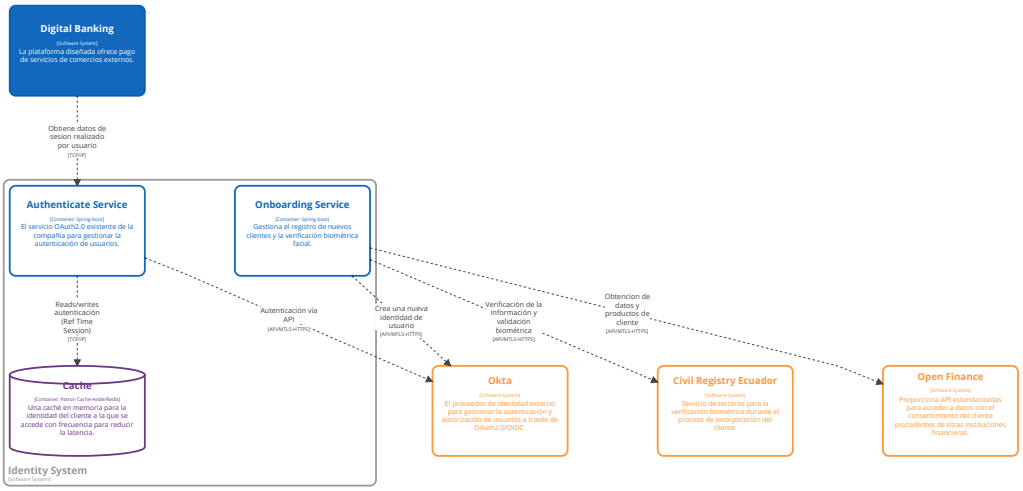
- **Proceso de Onboarding (Sincrónico):** El proceso de *onboarding* del cliente se ejecuta de manera **sincrónica**, un requisito para la validación inmediata y la preparación del perfil. Este flujo integra:
    1. **Validación Biometría:** Utilización del **Registro Civil (Ecuador)** para la verificación biométrica de identidad.
    2. **Almacenamiento y Gestión de Identidad:** La identidad es almacenada y gestionada en **Okta**, sirviendo como proveedor de identidad (IdP) con **SSO (Single Sign-On)**.

3. **Enriquecimiento de Datos:** Integración con plataformas de **Open Finance** para la obtención de información centralizada de otras entidades.

- **Gestión de Sesiones (Patrón Cache-Aside):** Al validarse la autenticación con éxito, la sesión y los datos clave del usuario son almacenados en una capa de caché (patrón **Cache-Aside**). Este mecanismo asegura que la información de la sesión pueda ser **reutilizada rápidamente** desde cualquier otro proceso que el cliente realice posteriormente, optimizando la experiencia y el *performance* general del sistema.



System Context View: Identity System
miércoles, 12 de noviembre de 2025, 13:01 hora estándar de Perú



Container View: Identity System

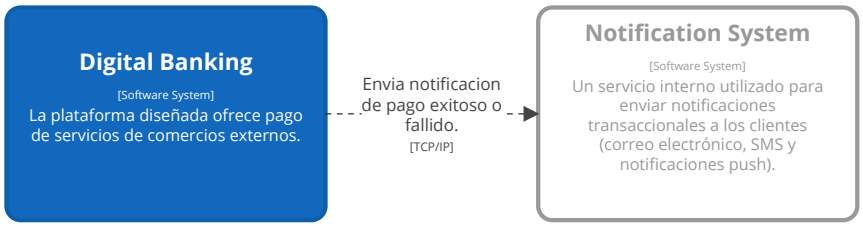
miércoles, 12 de noviembre de 2025, 13:01 hora estándar de Perú

2. Diagramas Notificación

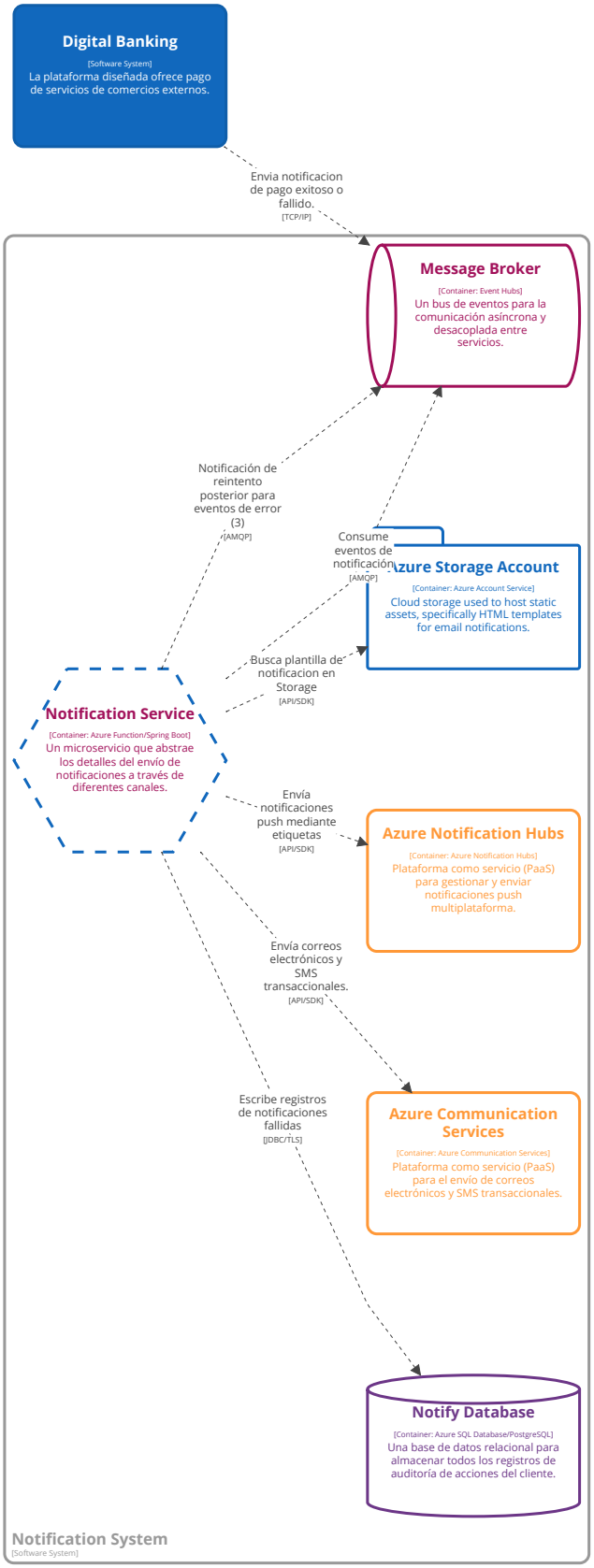
Sistema de Notificaciones (Arquitectura Asíncrona)

El sistema de notificaciones funciona de manera **transversal** para múltiples procesos, garantizando que la operación principal no se vea afectada. Se ha implementado un enfoque altamente desacoplado y resiliente:

- **Arquitectura:** Se utiliza una **Arquitectura Orientada a Eventos (EDA)** basada en los patrones **Event Driven** y **Event Sourcing**. Esto permite que la tarea de notificación se ejecute de forma **asíncrona** en segundo plano, sin interrumpir el flujo del proceso proveniente (e.g., el pago de servicio).
- **Componentes de Plataforma:** Se hace uso de **componentes nativos de Azure** (Servicios de Mensajería y Eventos) para facilitar el desacoplamiento, la escalabilidad y un mínimo impacto ante futuros cambios en la infraestructura.
- **Resiliencia y Reintento de Fallos:** Para manejar fallos de conectividad o disponibilidad externa, se implementa un mecanismo de reintento:
    - Las notificaciones que fallan en su primer intento son registradas en una base de datos de "notificaciones fallidas".
    - Un **subproceso automatizado** se encarga de realizar el reintento del envío de estas notificaciones de forma periódica, garantizando la entrega eventual de la comunicación al cliente.



System Context View: Notification System
miércoles, 12 de noviembre de 2025, 13:01 hora estándar de Perú



Container View: Notification System

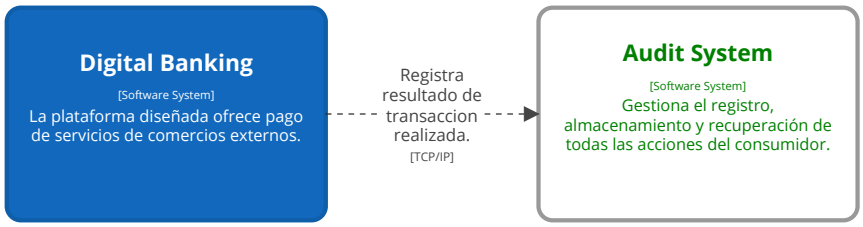
miércoles, 12 de noviembre de 2025, 13:01 hora estándar de Perú

3. Diagramas Auditoria

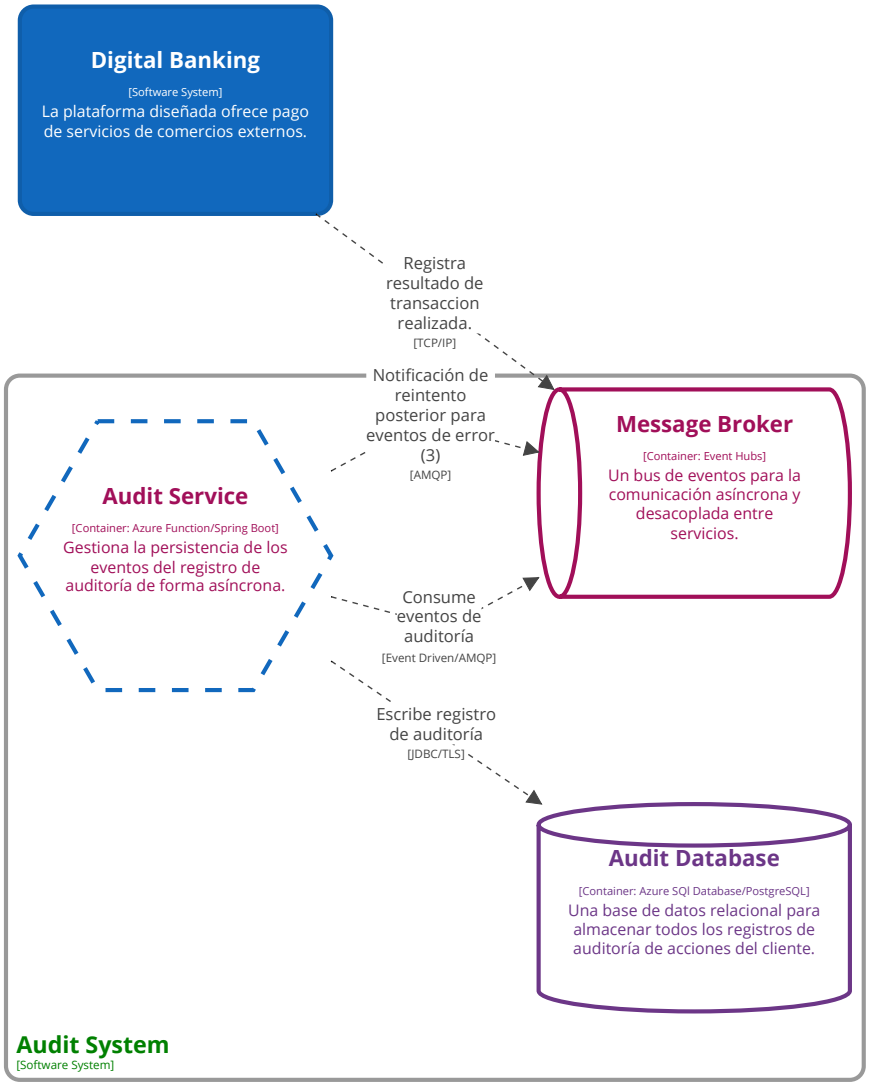
Sistema de Auditoría Transversal (Asíncrono)

El sistema incorpora una **capa de auditoría transversal** y obligatoria, diseñada para cumplir con las **reglas normativas** de la entidad reguladora financiera. Su propósito es validar las interacciones y registrar de manera inmutable todos los movimientos, transacciones, pagos, solicitudes, y otros eventos realizados por el usuario.

- **Arquitectura:** Al igual que el sistema de notificaciones, se hace uso de un **proceso asíncrono** basado en la **Arquitectura Orientada a Eventos (EDA)** y los patrones **Event Driven** y **Event Sourcing**.
- **Beneficio Operacional:** El registro se realiza **en segundo plano** sin interrumpir el proceso principal de negocio (ej. el pago o la solicitud), asegurando la mínima latencia para el cliente.
- **Resiliencia:** Se incluye un **proceso de reintento** para manejar fallos. En caso de inconsistencia o indisponibilidad en la conectividad con la base de datos de auditoría, el registro fallido se almacena temporalmente para ser reprocesado, garantizando la persistencia de la traza de auditoría.



System Context View: Audit System
martes, 11 de noviembre de 2025, 23:25 hora estándar de Perú



Container View: Audit System
martes, 11 de noviembre de 2025, 23:25 hora estándar de Perú

Link de proyecto

https://structurizr.com/share/108289