

# ARQUITECTURA DE INTEGRACION

RETO

GEORGE DE LA ROSA

**Tabla de contenido**

*Diagrama de contexto*..... 2

*Diagrama de container*..... 4

*Anexos* ..... 6

    Diagramas Identity..... 6

    Diagramas Notificación..... 8

    Diagramas Auditoria ..... 10

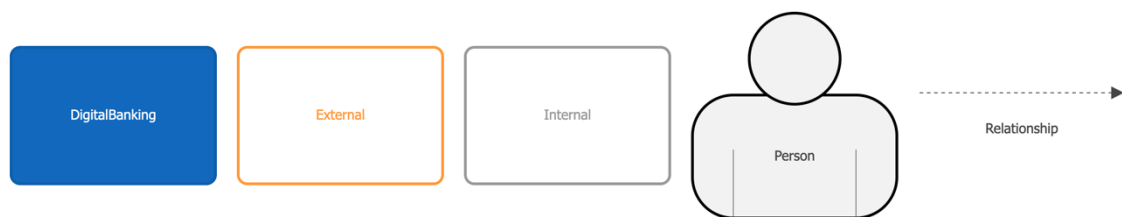
## Diagrama de contexto

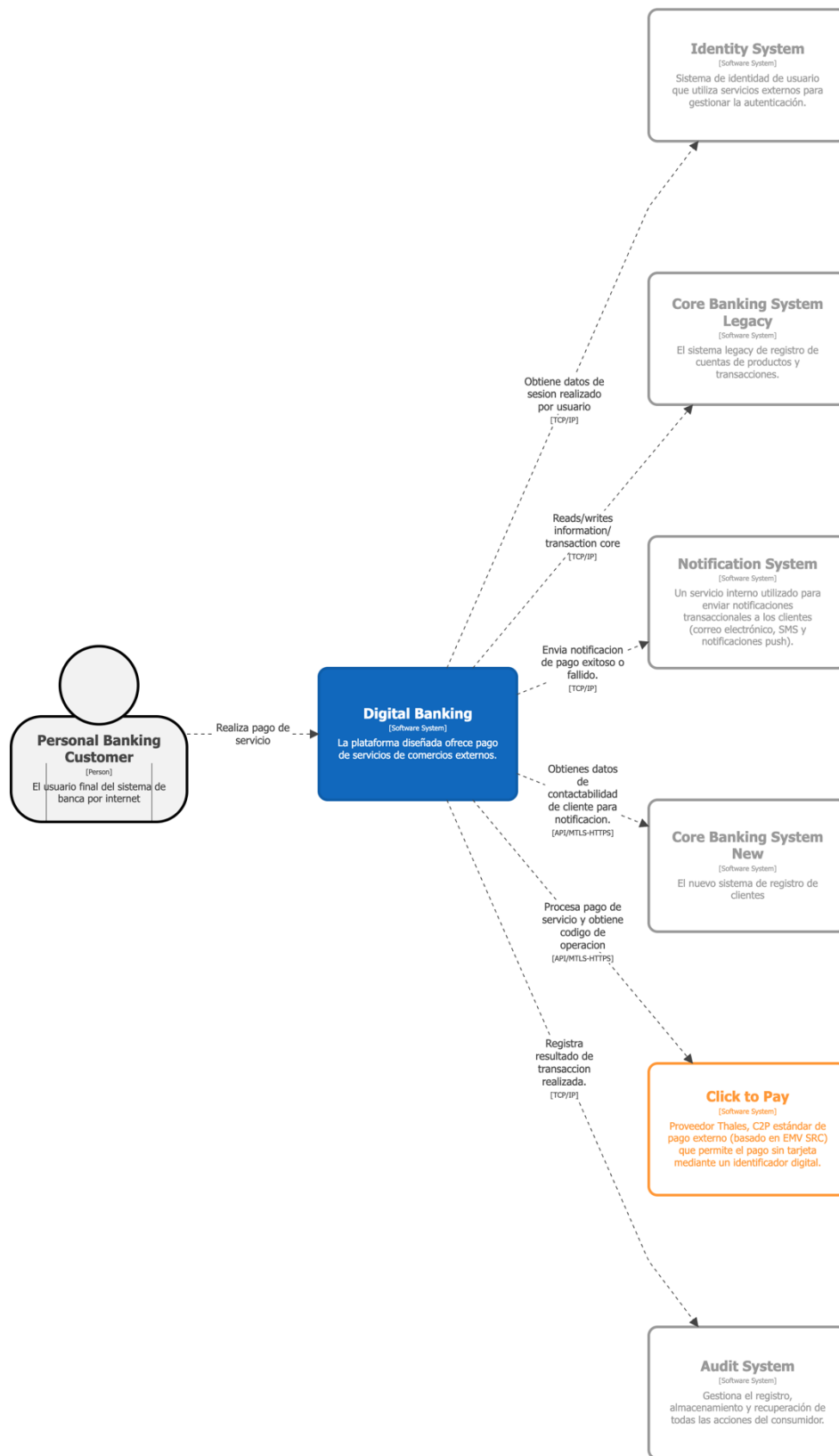
Este diagrama de contexto C4 ilustra el alcance del "Sistema de Pago de Servicios Digitales".

La funcionalidad principal del sistema permite a los usuarios (actores) realizar pagos de servicios a través de sus interfaces principales: la Aplicación Móvil (App mobile) y la Plataforma Web de Banca Digital.

Para operar, este sistema central se comunica con varios sistemas transversales esenciales. Estos sistemas se clasifican en dos categorías:

1. **Sistemas Internos:** Componentes internos de la entidad bancaria, como el Core Bancario y los sistemas de autenticación/perfiles de usuario.
2. **Sistemas Externos:** Entidades fuera del control de la organización, como las Pasarelas de Pago, los sistemas de las Empresas de Servicios (utilitarios) y los sistemas de notificación transaccional.





## 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.

### Optimización y Prevención de Duplicidad con Redis

Para mejorar el performance y la seguridad transaccional, se hace uso del patrón **Cache-Aside**.

- **Tecnología:** Se utiliza **Redis** como almacén de datos rápido.
- **Función:** La capa de caché realiza la validación inicial para determinar si una transacción de pago de servicio es **duplicada**, previniendo la ejecución innecesaria del flujo principal y reduciendo la carga sobre los sistemas Core.

### Patrón Backend For Frontend (BFF)

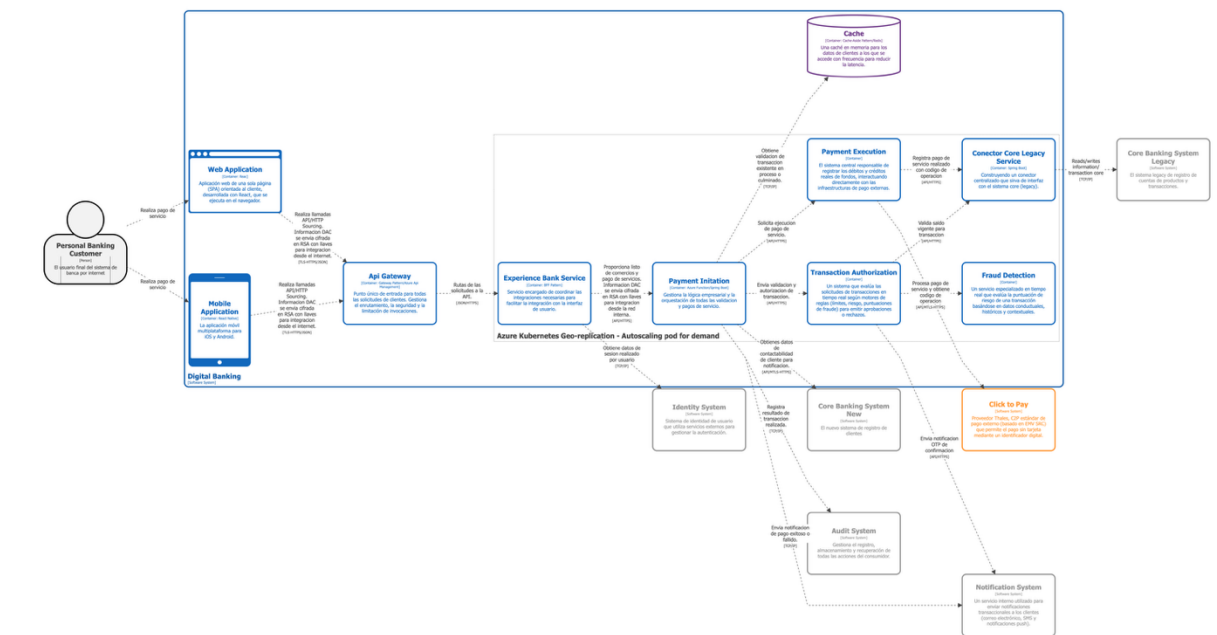
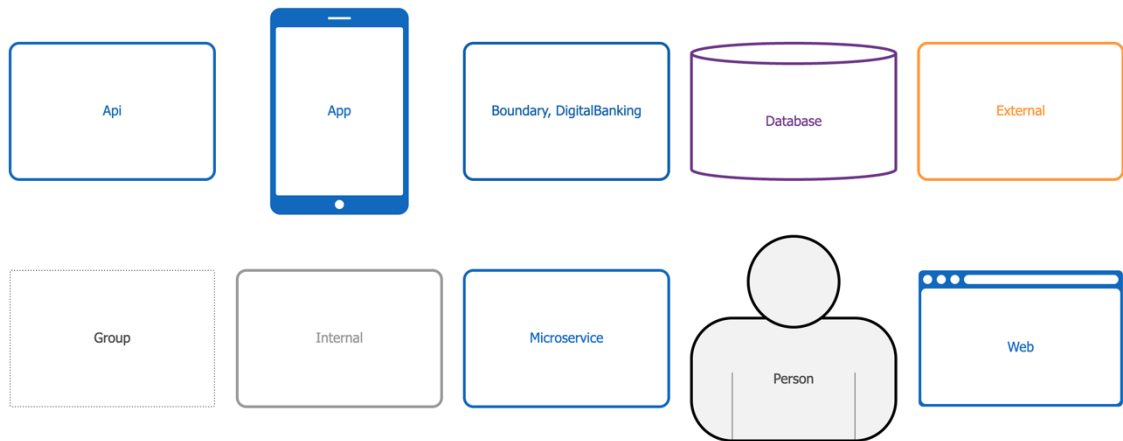
Se aplica el patrón **BFF (Backend For Frontend)**.

- **Objetivo:** 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, evitando que las aplicaciones cliente procesen datos irrelevantes o excesivos.

### Capa Anti-Corrupción (ACL)

Se ha implementado el patrón **ACL (Anti-Corruption Layer)**.

- **Propósito:** Este contenedor actúa como un punto transversal para facilitar la integración con **sistemas legacy**, especialmente aquellos que no utilizan protocolos modernos (como HTTP/JSON).
- **Ventaja de Mantenimiento:** Al encapsular la lógica de traducción de protocolos dentro del ACL, se garantiza que un cambio en la forma de conexión con un sistema *legacy* tenga un impacto mínimo, quedando aislado y gestionado internamente en esta capa.



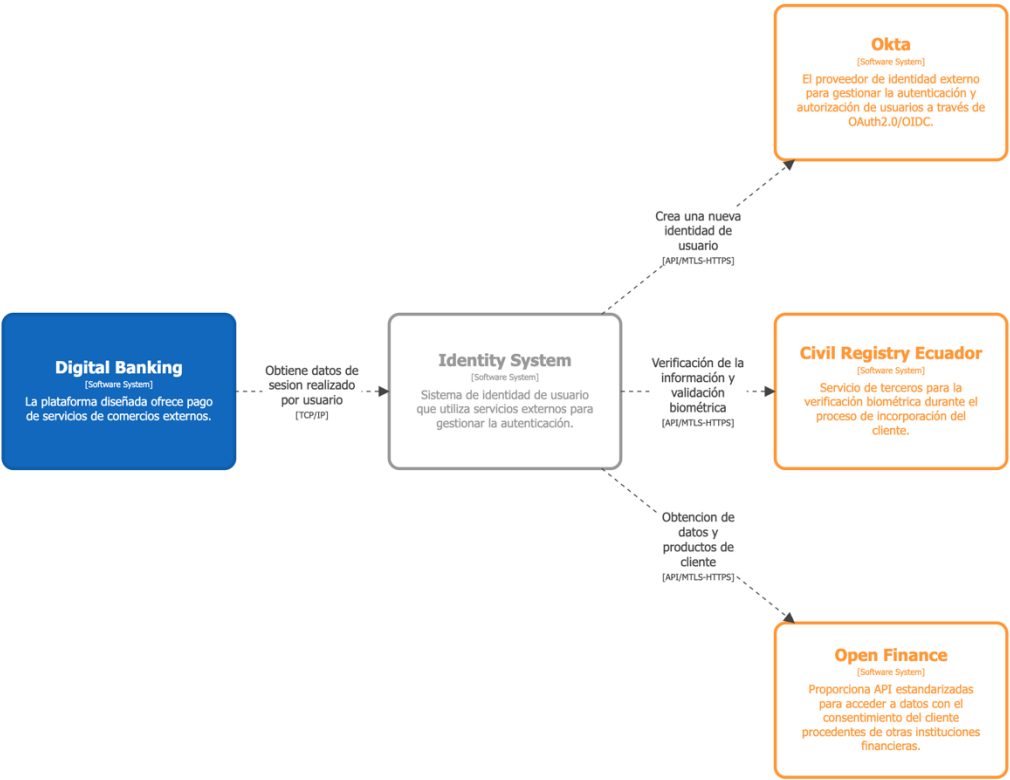
# Anexos

## Diagramas Identity

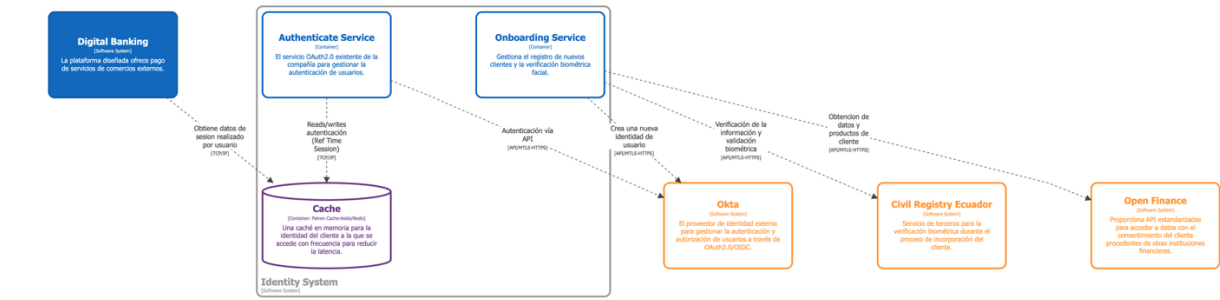
### Autenticación y Validación de Identidad (Onboarding)

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  
domingo, 9 de noviembre de 2025, 20:44 hora estándar de Perú



Container View: Identity System  
domingo, 9 de noviembre de 2025, 20:44 hora estándar de Perú

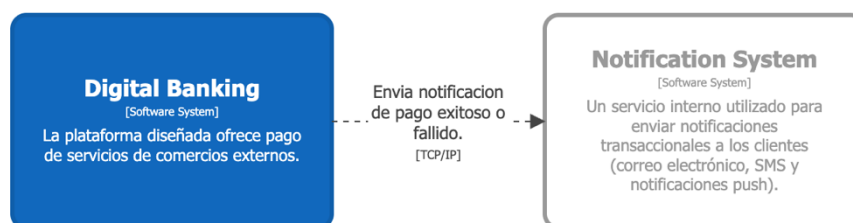


## Diagramas Notificación

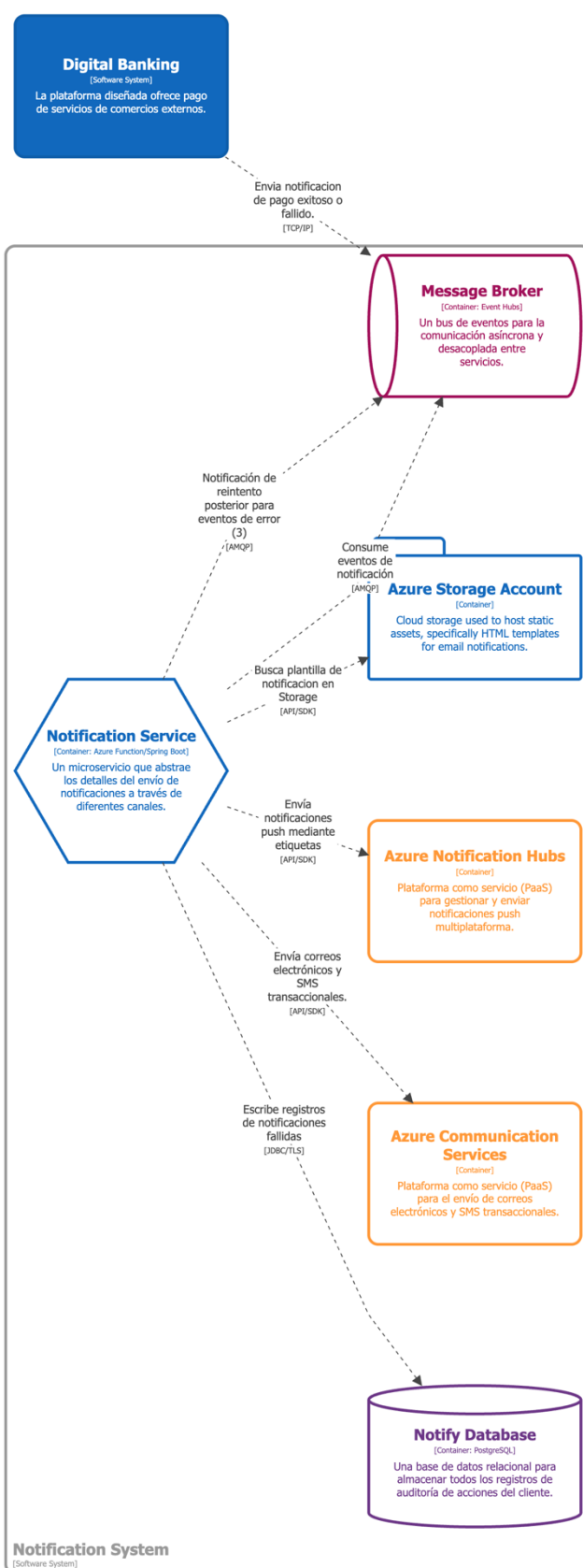
### Sistema de Notificaciones (Arquitectura Asíncrona)

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  
domingo, 9 de noviembre de 2025, 20:48 hora estándar de Perú

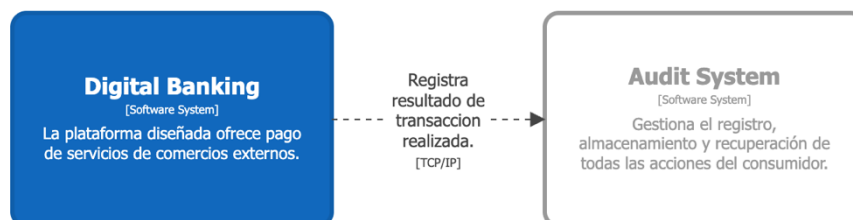


## Diagramas Auditoria

### Sistema de Auditoría Transversal (Asíncrono)

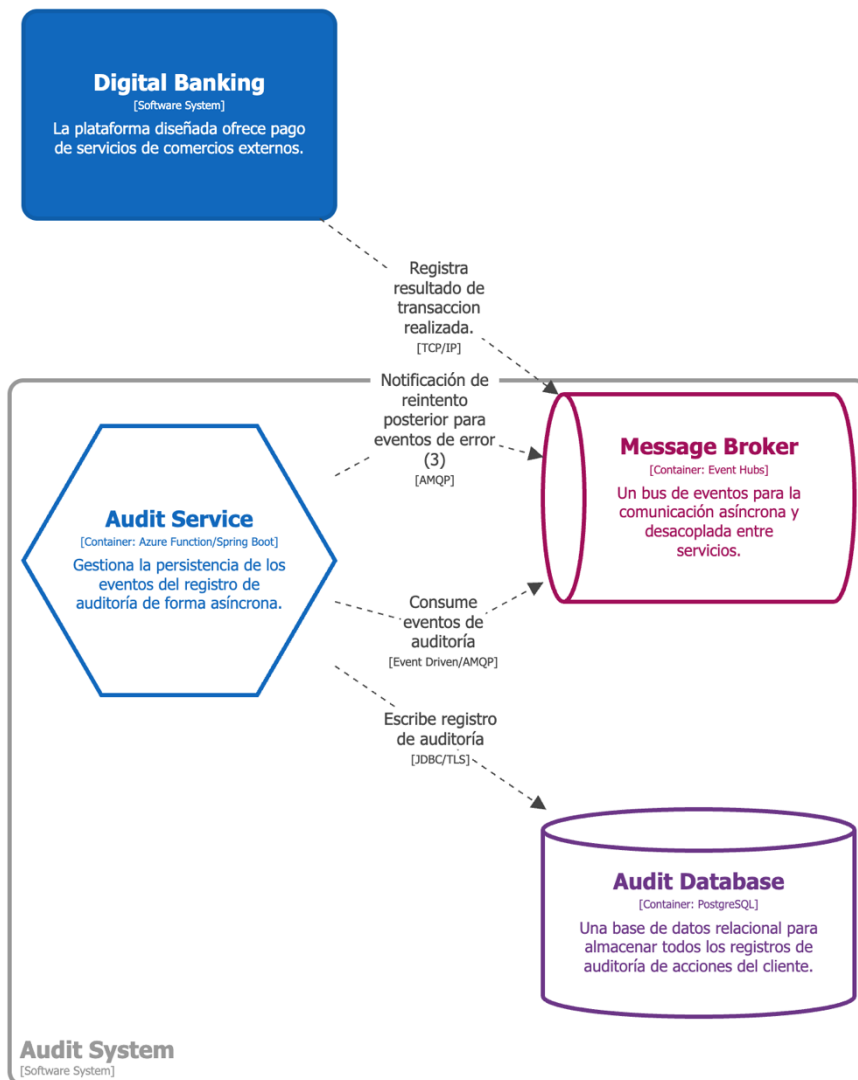
El sistema incorpora una **capa de auditoría transversal** y obligatoria, diseñada para cumplir con las **reglas normativas** de la entidad reguladora financiera. Su propósito es validar las interacciones y registrar de manera inmutable todos los movimientos, transacciones, pagos, solicitudes, y otros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

domingo, 9 de noviembre de 2025, 20:51 hora estándar de Perú



## Container View: Audit System

domingo, 9 de noviembre de 2025, 20:51 hora estándar de Perú

## Link de proyecto

<https://structurizr.com/workspace/108213/diagrams>