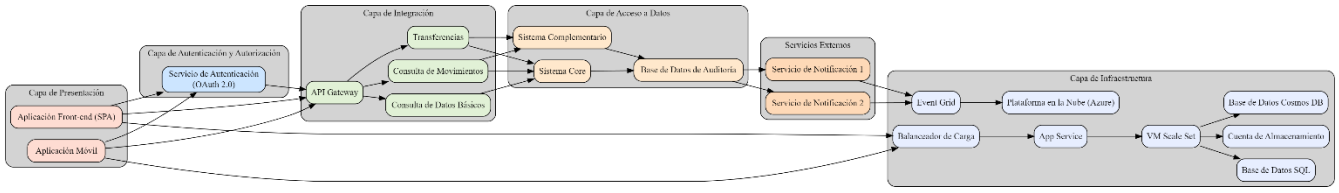
Estimados a continuación se detalla las preguntas:

Lo expuesto en este documento está basado en una arquitectura usada en mi actual Banco y además usado en una empresa Telco más grande del país. Quiero enfatizar que se puede hacer mejores diseños y usar mejores componentes, pero debemos apegarnos a la realidad del Banco local y empresa Telco.

1 - Que la solución satisfaga los requerimientos



La arquitectura propuesta incorpora arquitectura en la nube, como el uso de servicios específicos de Azure, como App Service, VM Scale Set, Base de Datos SQL, Cuenta de Almacenamiento, Base de Datos Cosmos DB y Event Grid. También se incluye un nodo para representar la plataforma en la nube (Azure) como base de toda la infraestructura.

La imagen se agrega en la carpeta para verificar mejor.

2 - Calidad y profundidad de los diagramas (Contexto, Contenedores y Componentes)

Diagrama de Contexto:

Se representa la interacción del sistema con el usuario, los servicios externos y la infraestructura en la nube. Se identifican las principales componentes del sistema, como la Capa de Autenticación y Autorización, la Capa de Integración, la Capa de Acceso a Datos, los Servicios Externos y la Capa de Infraestructura.

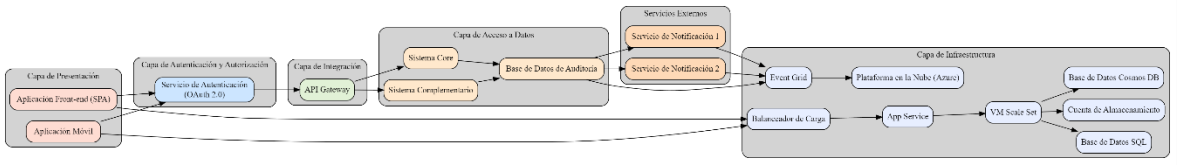


Diagrama de Contenedores:

Se identifican los principales componentes, como el API Gateway, el Sistema Core, el Sistema Complementario, la Base de Datos de Auditoría y los Servicios Externos. Además, se representan los contenedores en los que se ejecutan estos componentes, como el App Service, la VM Scale Set y las bases de datos SQL y Cosmos DB. Este diagrama proporciona una vista más detallada de las tecnologías y los contenedores utilizados.

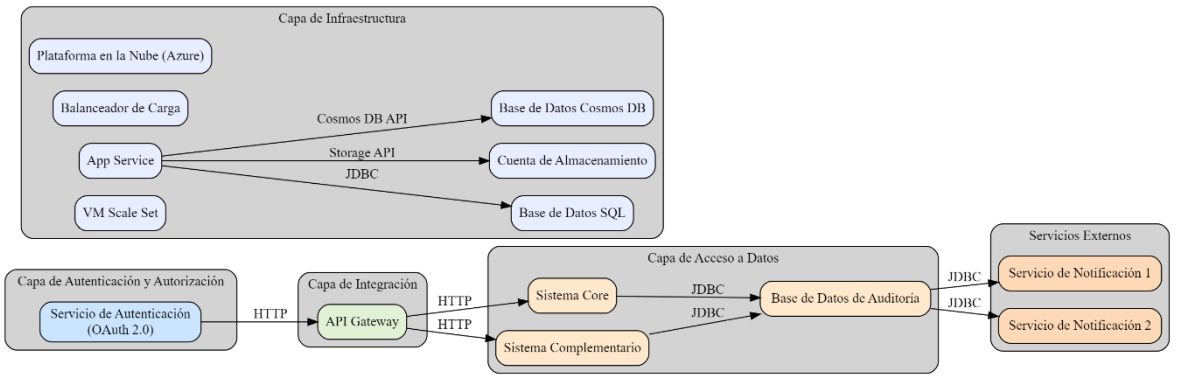
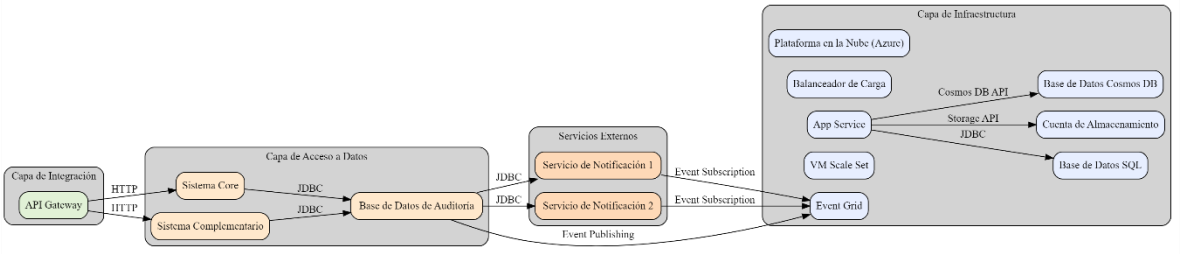


Diagrama de Componentes:

Se representan los componentes clave dentro de cada contenedor, como el Servicio de Autenticación (OAuth 2.0), el API Gateway, el Sistema Core, el Sistema Complementario, la Base de Datos de Auditoría y los Servicios Externos de Notificación. Además, se representan las interacciones entre estos componentes.



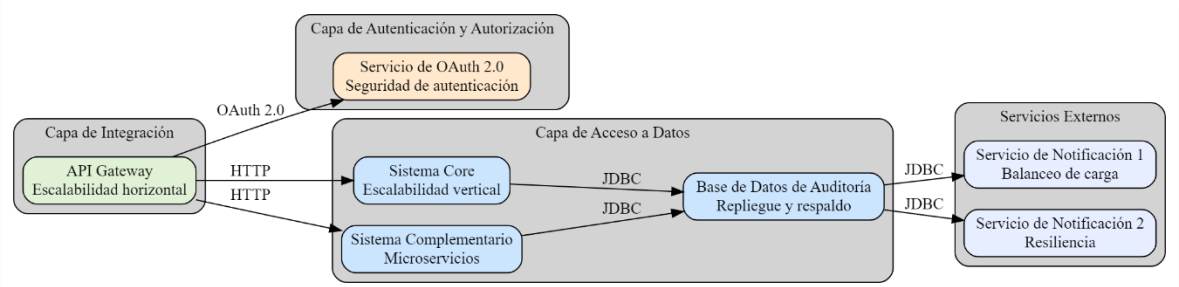
3 - Segmentación de Responsabilidades y Desacoplamiento

Se muestra la segmentación de responsabilidades y el desacoplamiento en la arquitectura propuesta. Se representan las diferentes capas de la arquitectura, como la Capa de Integración, la Capa de Autenticación y Autorización, la Capa de Acceso a Datos y los Servicios Externos. Cada capa tiene componentes específicos que se comunican entre sí mediante protocolos estándar, como HTTP y JDBC.

El API Gateway actúa como punto de entrada para las solicitudes de los clientes y se encarga de enrutarlas a los sistemas Core y Complementario a través de protocolo HTTP. Además, el API Gateway interactúa con el Servicio de OAuth 2.0 para autenticar y autorizar a los usuarios.

Los sistemas Core y Complementario se comunican con la Base de Datos de Auditoría mediante el protocolo JDBC para acceder y registrar información relevante. La Base de Datos de Auditoría también se conecta a los Servicios de Notificación mediante JDBC para enviar notificaciones.

Esta segmentación de responsabilidades y el desacoplamiento entre los componentes permiten una arquitectura modular y flexible, donde cada capa y componente tiene su función específica y se comunica de manera eficiente utilizando protocolos estándar.



• 4- Uso de patrones de arquitectura

En la arquitectura propuesta se pueden identificar varios patrones de arquitectura que ayudan a estructurar y diseñar el sistema de manera eficiente.

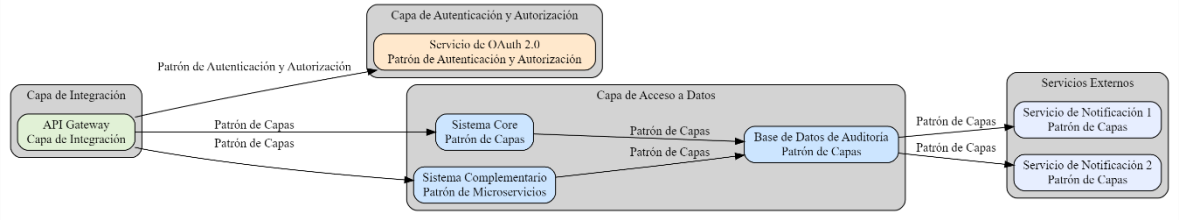
Patrón Modelo-Vista-Controlador (MVC): Este patrón se utiliza para separar la lógica de presentación (Vista) de la lógica de negocio (Modelo) y la lógica de control (Controlador). Puede aplicarse en las aplicaciones front-end y móvil para garantizar una estructura clara y modular.

Patrón de Microservicios: Este patrón se utiliza para dividir la aplicación en servicios pequeños e independientes, cada uno con su propia funcionalidad. Puede aplicarse en la Capa de Acceso a Datos, donde el Sistema Complementario se puede implementar como un conjunto de microservicios que se comunican entre sí.

Patrón de Capas (Layered Architecture): Este patrón se utiliza para dividir la aplicación en capas lógicas, cada una con su propia responsabilidad. Puede aplicarse en la arquitectura propuesta, donde se tienen capas como la Capa de Integración, la Capa de Autenticación y Autorización, y la Capa de Acceso a Datos.

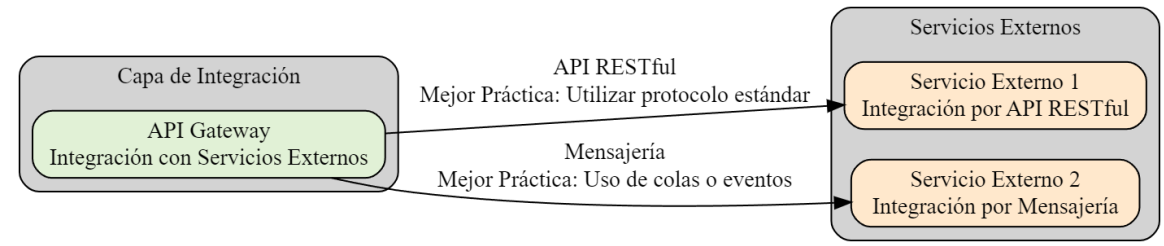
Patrón de Autenticación y Autorización (OAuth 2.0): Este patrón se utiliza para gestionar la autenticación y autorización de los usuarios en aplicaciones distribuidas. Se aplica en la Capa de Autenticación y Autorización, donde se utiliza el estándar OAuth 2.0 y el Servicio de OAuth para autenticar a los usuarios y proporcionarles acceso seguro a los recursos.

Patrón de Mensajería (Message-based Architecture): Este patrón se utiliza para el intercambio de mensajes entre componentes de un sistema distribuido. Se puede aplicar en la comunicación entre el API Gateway y los sistemas Core y Complementario, así como entre la Base de Datos de Auditoría y los Servicios de Notificación.



- 5 - Integración con servicios externos

La Capa de Integración representada por el API Gateway, que se encarga de integrar y comunicarse con los Servicios Externos. Los Servicios Externos, representados por los nodos "external_service1" y "external_service2", interactúan con el API Gateway para proporcionar funcionalidades específicas del sistema.



- 6 - Calidad de arquitectura de aplicación front-end y móvil

La aplicación front-end y móvil en la arquitectura propuesta puede incluir varios aspectos clave. A continuación, se presentan algunos de ellos:

Modularidad: La arquitectura debe estar diseñada de manera modular, dividiendo la aplicación en componentes independientes y reutilizables

Escalabilidad: La arquitectura debe ser capaz de manejar eficientemente el crecimiento y la carga de usuarios. Debe permitir escalar horizontalmente.

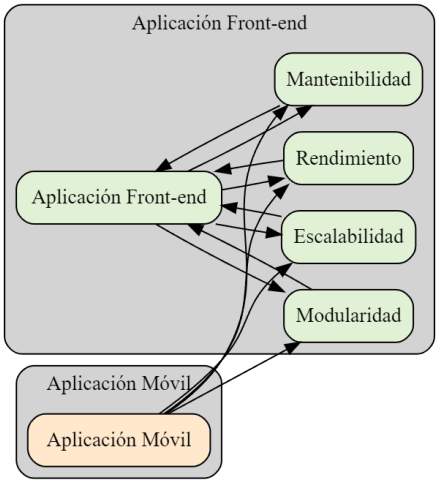
Rendimiento: La arquitectura debe optimizar el rendimiento de la aplicación front-end y móvil. Esto implica minimizar los tiempos de carga, optimizar las consultas a la base de datos y utilizar técnicas como caché y compresión para mejorar la velocidad de respuesta.

Mantenibilidad: La arquitectura debe facilitar el mantenimiento de la aplicación a lo largo del tiempo. Debe estar bien estructurada, con código limpio y documentado

Seguridad: La arquitectura debe garantizar la seguridad de los datos y la protección de la aplicación contra posibles amenazas

Usabilidad: La arquitectura debe tener en cuenta los principios de diseño centrados en el usuario para garantizar una experiencia de usuario intuitiva y agradable

Pruebas: La arquitectura debe permitir la realización de pruebas de forma efectiva, tanto en la aplicación front-end como en la móvil. Debe facilitar la implementación de pruebas unitarias, de integración y de rendimiento, lo que asegura la calidad del software.

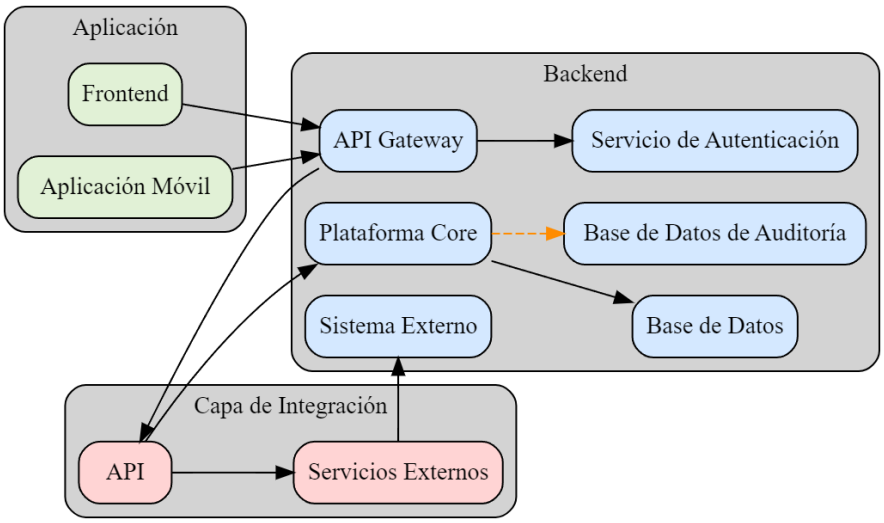


- 7 - Arquitectura de acceso a datos

Se representan los componentes clave de la arquitectura de acceso a datos de la arquitectura propuesta. Los nodos "frontend" y "mobile" representan las aplicaciones front-end y móvil, respectivamente. El nodo "api_gateway" representa el componente de API Gateway que actúa como punto de entrada para las solicitudes de las aplicaciones.

El nodo "autenticacion" representa el servicio de autenticación utilizado para autenticar a los usuarios. El nodo "core" representa la plataforma Core que contiene la información básica del cliente y se comunica con la base de datos representada por el nodo "base_datos". El nodo "sistema_externo" representa el sistema externo del cual se consumen los servicios adicionales.

El nodo "api" representa el componente de API que proporciona los puntos de acceso para las aplicaciones y se comunica con el core y los servicios externos. El nodo "servicios_externos" representa los servicios externos consumidos por la API.



- 8 - Conocimientos de Nube (AWS o Azure)

Actualmente trabajo con plataforma 100% azure y apalanzados con proveedores como Software One, Binaria, entre otros para el soporte.

En mi actuales tareas estoy trabajando en la plataforma de datos lakehouse para todo el Banc usando Azure.

De lo que veo estos serian los componentes base para armar la arquitectura, es claro decir que esto puede ajustarse a la realidad del Banco.

Azure App Service: Se puede utilizar para alojar y escalar las aplicaciones front-end

Azure Functions: Pueden utilizarse para implementar funciones sin servidor que respondan a eventos específicos, como notificaciones o eventos de integración con servicios externos

Azure API Management: Puede utilizarse como una puerta de enlace para administrar, proteger y escalar las API expuestas por la aplicación

Azure Active Directory (Azure AD): Azure AD admite estándares como OAuth 2.0 para la autenticación y proporciona características de seguridad y administración de identidades.

Azure SQL Database: Puede utilizarse como la base de datos para almacenar los datos del sistema, como la información del cliente y los movimientos.

Azure Blob Storage: Puede utilizarse para almacenar archivos estáticos, como imágenes o documentos, que se utilizan en la aplicación.

Azure Queue Storage: Puede utilizarse como una cola de mensajes para la comunicación asíncrona entre componentes del sistema.

9- Manejo de costos

Los costos se calculan dependiendo de varios factores:

- Transaccionalidad
- Interacciones
- Almacenamiento
- Uso
- Trafico
- Replicacion y HA

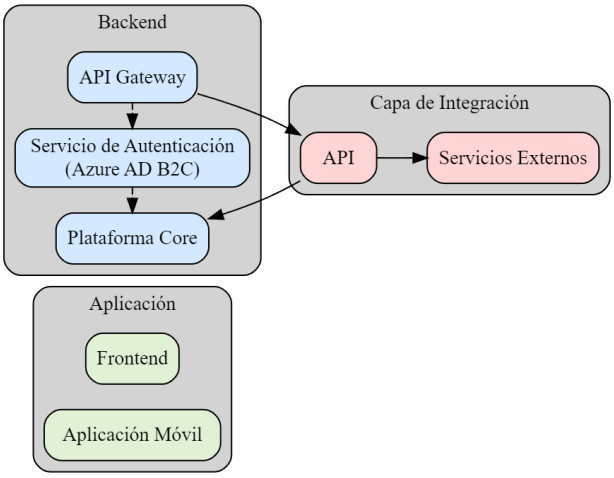
Nosotros usamos un Sizing para calcular previo y luego direccionamos a nuestros partner para que nos coticen los valores estimados y es importante indicar “ESTIMADOS”

10 - Arquitectura de Autenticación

Servicio de Autenticación (Azure AD B2C): es una solución de identidad y acceso en la nube diseñada específicamente para aplicaciones cliente. Azure AD B2C proporciona características avanzadas de autenticación y autorización, como el inicio de sesión social, la autenticación multifactor y la personalización de la experiencia de inicio de sesión.

Conexión directa entre el Servicio de Autenticación y el Core: Se ha agregado una conexión directa entre el Servicio de Autenticación y el Core, lo que permite una comunicación más eficiente y segura para la validación de credenciales y la autorización de acceso a los recursos protegidos.

Estilo de línea: Se ha utilizado un estilo de línea discontinua para representar las conexiones a través de interfaces y protocolos estándar.

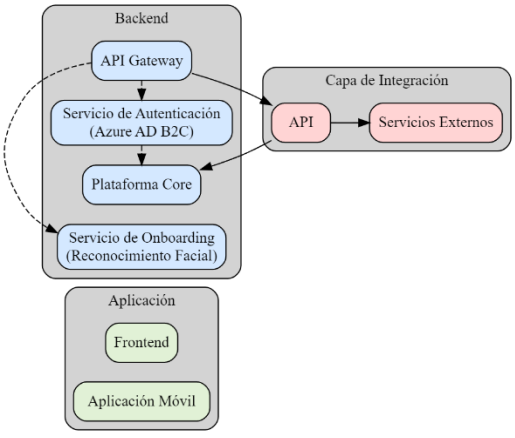


11 Arquitectura de Integración con Onboarding

Servicio de Onboarding (Reconocimiento Facial): Se ha agregado un servicio de Onboarding que utiliza el reconocimiento facial como método de autenticación adicional. Este servicio permite a los nuevos usuarios registrarse en la aplicación mediante el reconocimiento de su rostro y posteriormente acceder a través de métodos de autenticación tradicionales como usuario y contraseña, huella dactilar, etc.

Conexiones directas: Se han establecido conexiones directas entre los componentes clave, como el Servicio de Autenticación, el Servicio de Onboarding, el API Gateway, el Core y los servicios externos.

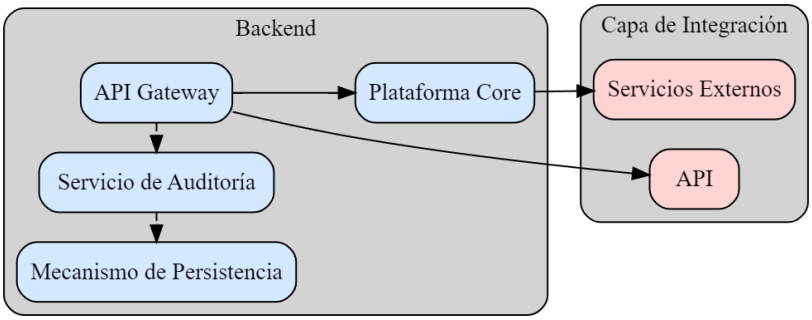
Es clave indicar que esto se tomó del onboarding implementado en la empresa TELCO en la Aplicación Móvil por temas de mejorar la experiencia del Cliente en recargas y pagos de servicios.



- ## 12 Diseño de Solución de Auditoría

Servicio de Auditoría: Se ha incluido un servicio de auditoría que registra todas las acciones realizadas por los clientes. Este servicio es responsable de capturar y almacenar los eventos relevantes para la auditoría, como los movimientos financieros, las transferencias y los pagos realizados.

Tomado del módulo de auditoria del Banco



- ## 13 Conocimientos de regulaciones bancarias y estándares de seguridad

Ley Orgánica de Economía Popular y Solidaria: Esta ley establece el marco legal para las cooperativas y otras entidades de economía popular y solidaria en Ecuador.

Superintendencia de Bancos: La Superintendencia de Bancos del Ecuador (SBS) es el organismo regulador encargado de supervisar y regular el sector bancario en el país

Resolución No. SB-2012-443: Esta resolución emitida por la SBS establece las políticas de seguridad de la información que deben seguir las instituciones financiera

Norma Técnica de Control y Seguridad de la Información: Esta norma establece los controles y requisitos mínimos de seguridad de la información que deben implementar las instituciones financieras en Ecuador

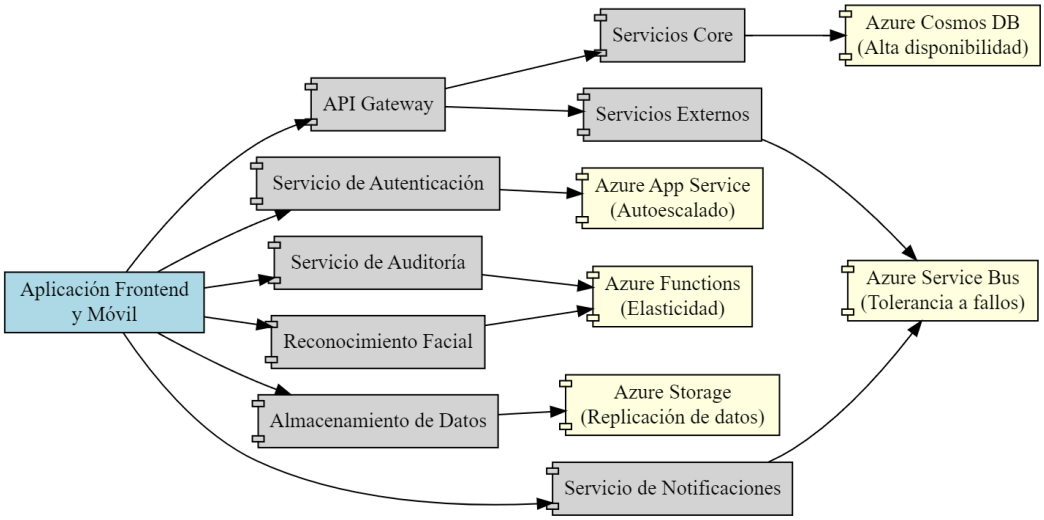
Estándares internacionales: Las instituciones financieras en Ecuador también pueden verse sujetas a estándares internacionales de seguridad, como ISO 27001 (Sistemas de Gestión de Seguridad de la Información) y PCI DSS (Estándar de Seguridad de Datos de la Industria de Tarjetas de Pago), dependiendo de su alcance y operaciones

la Ley de Protección de Datos Personales (Ley Orgánica de Protección de Datos Personales - LOPD) es la legislación que regula la protección y privacidad de los datos personales de los individuos.

- ## 14 Implementación de Alta Disponibilidad y Tolerancia a Fallos

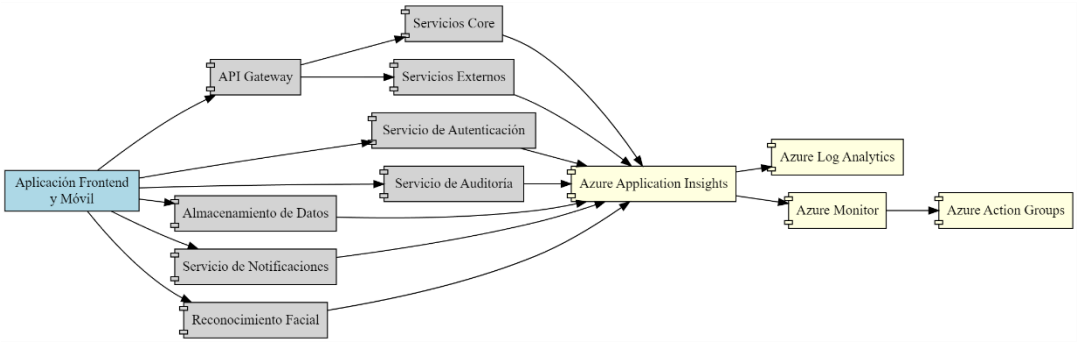
Se diagrama componentes y servicios de Azure que brindan alta disponibilidad y tolerancia a fallos.

Es clave indicar que fui el arquitecto del Banco mas grande del país en la implementación de DCA para Quito y Guayaquil



- ## 16 Implementación de Monitoreo

Los componentes y servicios de monitoreo de Azure para garantizar una implementación robusta. Azure Application Insights se utiliza para recopilar datos de telemetría y seguimiento de la aplicación, y Azure Log Analytics se emplea para almacenar y analizar los registros generados por los servicios. Azure Monitor se utiliza para supervisar el rendimiento y la disponibilidad de los componentes, y Azure Action Groups permite definir acciones automatizadas en caso de eventos o alertas.



Es claro indicar que todas las respuestas fueron tomadas de soluciones ya implementadas en proyectos actuales, pasados y que estamos provisionando en los próximos 6 meses. Estas respuestas son confidenciales y exclusivas para DEVSU.