Nombre: Gabriel Salgado

Solución de Arquitectura para el Sistema de Banca por Internet

1. Introducción

- **Descripción del Proyecto**: Se desarrolla un sistema de banca por internet donde los usuarios pueden realizar operaciones como consultar el histórico de movimientos, transferencias y pagos, utilizando información proveniente de dos sistemas: Core Bancario y un sistema complementario.
- **Objetivo**: Diseñar la arquitectura de este sistema utilizando Azure, Keycloak para autenticación (OAuth 2.0), garantizando la alta disponibilidad, seguridad, y la integración con servicios externos y auditoría.

2. Requerimientos del Sistema

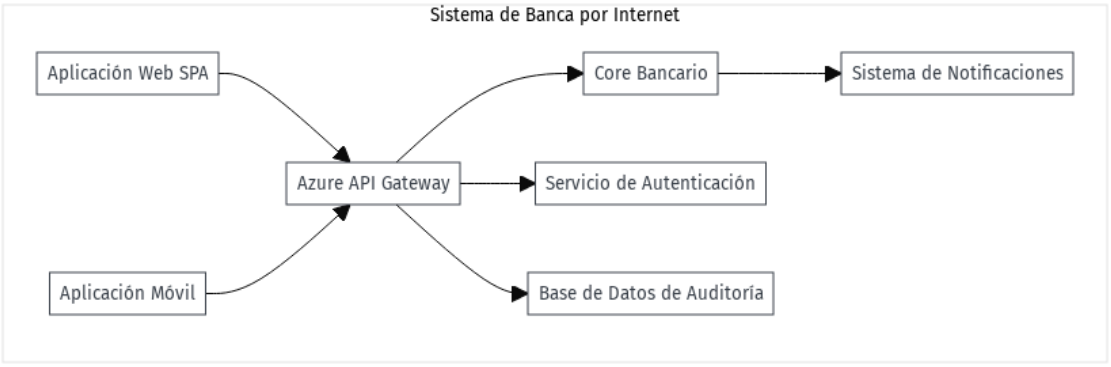
- **Usuarios**: Acceso mediante dos aplicaciones frontend (SPA y móvil) que usan OAuth 2.0 para la autenticación.
- **Backend**: Microservicios para consulta de datos básicos, movimientos y transferencias.
- **Notificaciones**: Uso de al menos 2 sistemas externos de notificación.
- **Auditoría**: Registro de todas las acciones realizadas por los usuarios.
- **Seguridad**: Cumplimiento con regulaciones de protección de datos (GDPR, PCI DSS) y seguridad avanzada (OAuth 2.0).

3. Arquitectura de la Solución

3.1. Diagrama de Contexto

- Descripción: El **Diagrama de Contexto** presenta las interacciones de alto nivel entre los sistemas principales:
    - **Aplicación SPA** y **Aplicación Móvil** interactúan con el **API Gateway** para acceder a microservicios.
    - **Microservicios Backend** que incluyen servicios como consulta de datos básicos, movimientos y transferencias.
    - **Keycloak** para la autenticación mediante OAuth 2.0.
    - **Sistemas Externos de Notificación** (ejemplo: **Twilio** para SMS, **SendGrid** para correos electrónicos).
    - **Base de datos de auditoría** para registrar todas las acciones realizadas por los clientes.

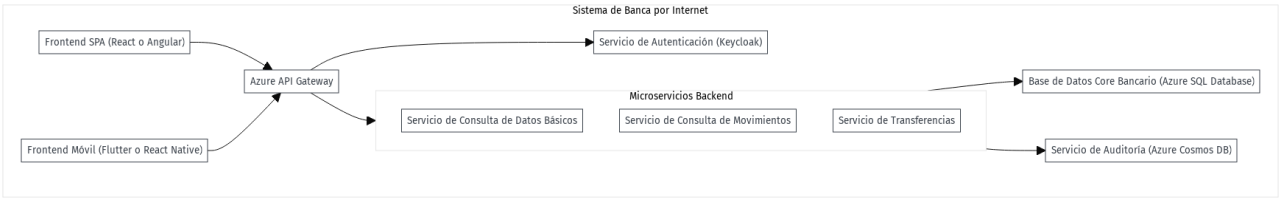
Diagrama:



3.2. Diagrama de Contenedores

- Descripción: El **Diagrama de Contenedores** detalla los componentes técnicos en la solución:
    - **Frontend SPA** y **Frontend Móvil** que son aplicaciones independientes.
    - **Microservicios** que interactúan con una base de datos **Azure SQL Database** y otros servicios externos como el servicio de auditoría.
    - **API Gateway** que maneja las solicitudes entre el frontend y los microservicios.
    - **Keycloak** que se encarga de la autenticación.
    - **Azure Cosmos DB** para la persistencia de la auditoría.

Diagrama:

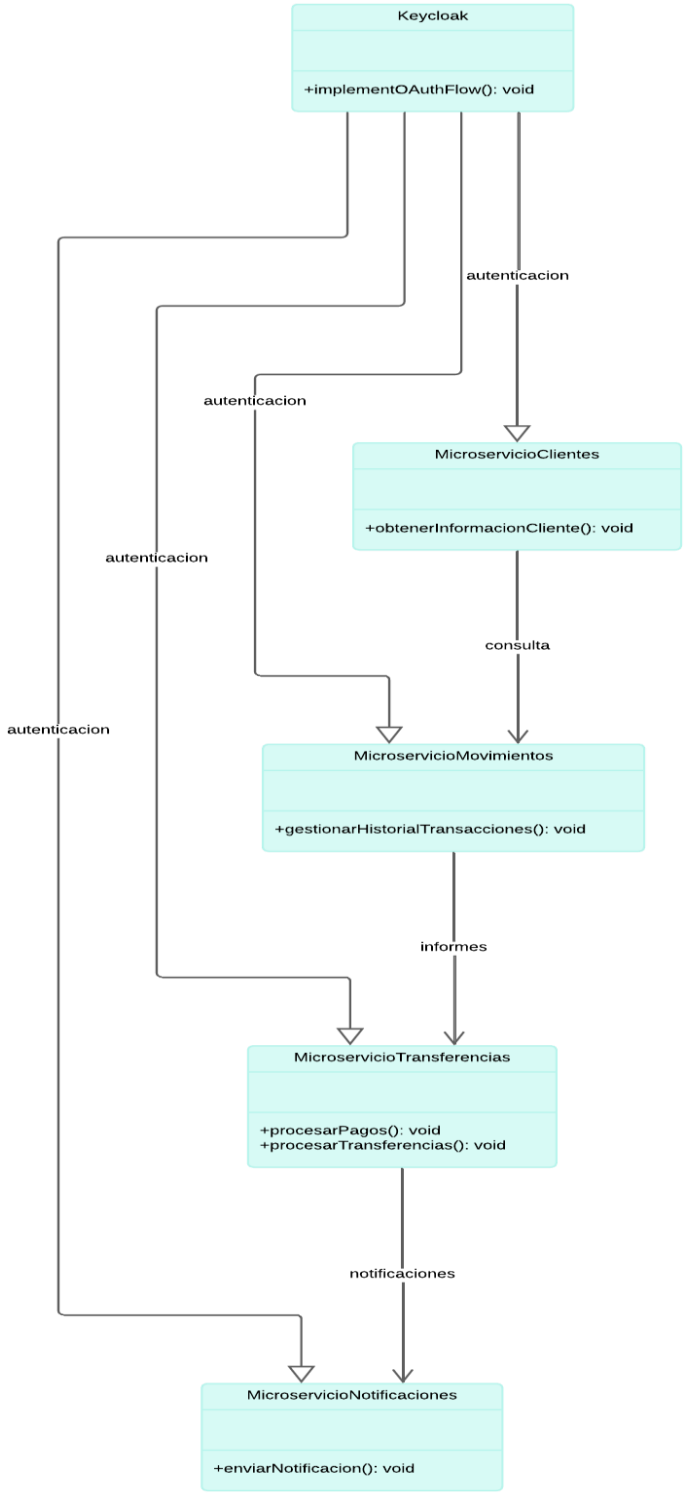


3.3. Diagrama de Componentes

- Descripción: El **Diagrama de Componentes** muestra los componentes detallados de cada microservicio.
    - **Microservicio de Clientes**: Obtiene la información básica del cliente desde el Core Bancario.
    - **Microservicio de Movimientos**: Recupera el historial de movimientos de la cuenta.
    - **Microservicio de Transferencias**: Procesa pagos entre cuentas propias e interbancarias.

- **Microservicio de Notificaciones**: Se comunica con sistemas externos (Twilio, SendGrid) para enviar notificaciones.
- **Keycloak**: Implementa la autenticación y autorización mediante OAuth 2.0.

Diagrama:



4. Flujo de Autenticación (OAuth 2.0 con Keycloak)

4.1. Elección del Flujo de Autenticación

El sistema usará el flujo de **Authorization Code Flow** para las aplicaciones SPA y móvil. Este flujo es adecuado porque:

- **Seguridad**: Minimiza la exposición de los tokens al frontend.
- **Uso de Tokens**: Intercambio del código de autorización por un token de acceso y un token de refresco en el backend.

4.2. Integración con Keycloak

- Keycloak se configura como el **Identity Provider (IdP)** y maneja la autorización de usuarios.
- Los usuarios se autentican mediante su nombre de usuario y contraseña, con soporte para autenticación biométrica (reconocimiento facial) en la aplicación móvil.

5. Consideraciones de Seguridad y Cumplimiento

5.1. Normativas Relevantes

- **GDPR**: El sistema debe garantizar la protección de los datos personales de los clientes con cifrado en reposo y en tránsito.
- **PCI DSS**: Cumplimiento con la normativa para el procesamiento seguro de pagos y transacciones.
- **Autenticación multifactor (MFA)**: Para aumentar la seguridad, especialmente en operaciones críticas como transferencias y pagos.

5.2. Estrategia de Alta Disponibilidad y Tolerancia a Fallos

- **Azure Kubernetes Service (AKS)**: Escalabilidad y distribución de la carga entre microservicios.
- **Azure Cosmos DB**: Almacenamiento distribuido con replicación global para alta disponibilidad de datos de auditoría.
- **Azure Load Balancer**: Para balanceo de carga entre instancias de servicios.

6. Propuesta de Arquitectura para Frontend y Móvil

6.1. SPA (React/Angular)

- Elección de **React** por su flexibilidad y popularidad en aplicaciones modernas.
- Se comunica con los microservicios a través del **API Gateway**.

6.2. Aplicación Móvil (Flutter/React Native)

- Elección de **Flutter** para crear una aplicación nativa multiplataforma, lo que optimiza el rendimiento en dispositivos móviles.
- Implementación del onboarding con reconocimiento facial para la autenticación biométrica.

7. Auditoría y Persistencia de Información

- **Event Sourcing** para capturar todos los eventos de acciones del cliente.
- Los eventos se almacenan en **Azure Cosmos DB** para su persistencia con alta disponibilidad.

8. Implementación y Despliegue en Azure

- **Azure DevOps** para CI/CD, despliegue automático de microservicios y gestión de configuraciones.
- **Azure Monitor** para la supervisión de la infraestructura y servicios backend.

---

9. Conclusión

Este documento presenta una arquitectura detallada para un sistema de banca por internet que se adapta a las mejores prácticas de seguridad, alta disponibilidad y cumplimiento normativo, utilizando **Azure** como plataforma en la nube y **Keycloak** para la autenticación segura. El sistema está diseñado para ser escalable, flexible y fácil de mantener.

---

10. Repositorio de GitHub

- **Enlace al repositorio público en GitHub**: https://github.com/Lgsalgado/solucionBancaInternet.git
- Este repositorio contiene el archivo PDF con la solución y diagramas.