Solución Ejercicio Técnico - Arquitecto BP

1. Resumen Ejecutivo

Esta propuesta presenta la arquitectura de una nueva plataforma de banca digital para BP en Ecuador, orientada a ofrecer servicios web y móviles seguros, rápidos y escalables para más de 6 millones de clientes.

La solución se basa en microservicios desplegados en **Azure Kubernetes Service (AKS)**, expuestos mediante **Azure API Management (APIM)** y protegidos bajo **OAuth 2.0/MFA** usando **Azure Entra ID** como proveedor de identidad. El front-end web se desarrolla en **Angular (SPA)** y el móvil en **Flutter**, garantizando una experiencia homogénea y de alto rendimiento en todos los dispositivos.

Se emplean patrones arquitectónicos modernos como **CQRS** para optimizar lecturas y escrituras, **Saga/Outbox** para transacciones críticas, **Event-Driven** con **Azure Service Bus/Event Grid** para integración asíncrona, y **Anti-Corruption Layer (ACL)** para aislar el dominio de la complejidad del **Core Bancario** y el **Sistema Complementario**. La comunicación con estos sistemas se realiza mediante protocolos seguros **(HTTPS/TLS 1.2+, OAuth2.0, mTLS)**.

El flujo de **onboarding** digital incluye captura de documento **(OCR)** y validación biométrica facial con **Facephi**, todo alojado de forma segura en la nube de **Azure**. Las notificaciones multicanal cubren SMS, correo electrónico, push y, como canal adicional no oficial, WhatsApp para interacciones más informales.

La propuesta incorpora medidas de **alta disponibilidad (HA)**, **tolerancia a fallos** y **recuperación ante desastres**, así como **observabilidad** con Application Insights y Log Analytics para un monitoreo integral. El diseño busca un equilibrio entre rapidez de entrega, cumplimiento regulatorio, costos y capacidad de evolución tecnológica, alineándose con las estrategias digitales actuales de BP y las mejores prácticas del sector financiero.

2. Alcance y Objetivos

2.1. Alcance

La solución propuesta cubre el diseño, implementación y despliegue de una plataforma integral de banca digital que incluye:

- Canales de acceso:
    - Banca Web (SPA en Angular)
    - Banca Móvil (Flutter multiplataforma)
- Capa de integración mediante Azure API Management como API Gateway.
- Arquitectura de microservicios en Azure Kubernetes Service (AKS) para funcionalidades clave: movimientos, transferencias, notificaciones, onboarding y auditoría.
- Integración segura con Core Bancario y Sistema Complementario mediante protocolos cifrados (HTTPS/TLS 1.2+, mTLS, OAuth2.0).
- Procesos de onboarding digital con validación biométrica facial (Facephi) y OCR.

- Mecanismos de notificación multicanal: SMS, email, push, y canal adicional no oficial vía WhatsApp.
- Observabilidad y monitoreo con Application Insights, Log Analytics y alertas operativas.
- Cumplimiento regulatorio y prácticas de seguridad alineadas a PCI DSS, ISO 27001 y normativa local.

2.2. Objetivos

2.2.1. Mejorar la experiencia del cliente ofreciendo una banca digital ágil, segura y disponible desde cualquier dispositivo.

2.2.2. Garantizar la seguridad y confianza en las operaciones mediante autenticación robusta (OAuth2.0 + OIDC + MFA) y canales cifrados.

2.2.3. Optimizar el rendimiento aplicando CQRS, caché distribuida y arquitectura event-driven para alta concurrencia y bajas latencias.

2.2.4. Asegurar la resiliencia y continuidad del servicio con alta disponibilidad, recuperación ante desastres y manejo de fallos (DLQ, circuit breaker, auto-healing).

2.2.5. Facilitar la evolución tecnológica manteniendo un diseño modular, desacoplado y compatible con la estrategia cloud-first del banco.

2.2.6. Cumplir la normativa vigente en seguridad de la información, protección de datos y notificación de transacciones.

3. Arquitectura General

La solución se implementa sobre **Microsoft Azure** debido a su alineación con los requisitos regulatorios del sector bancario ecuatoriano, su oferta de servicios certificados para entornos financieros y la integración nativa con herramientas clave como Azure Entra ID, API Management, Key Vault y Azure SQL. Frente a alternativas como AWS o GCP, Azure ofrece ventajas en conectividad híbrida segura (ExpressRoute, VNets y Private Endpoints), capacidades avanzadas de cumplimiento y soporte local/regional, lo que reduce la complejidad de integración con sistemas on-premise y mejora la gobernanza de datos sensibles. Esta elección garantiza coherencia tecnológica, seguridad integral y facilidad de operación bajo un ecosistema único.

3.1. Vista Lógica

- **Canales**: Web Angular (SPA) y móvil Flutter consumen APIs vía APIM.
- **Capa de entrada**: Azure API Management aplica seguridad, versionado, rate limiting y enruta a microservicios.
- **Dominio (AKS)**: microservicios Movimientos, Transferencias, Onboarding, Notificaciones, Auditoría.
- **Integraciones**: Core Bancario y Sistema Complementario detrás de ACL/Facade.
- **Datos/Mensajería**: Azure SQL (operacional), Redis (caché), Data Lake (auditoría), Service Bus (crítico), Event Grid (fan-out).

3.2. Patrones Transversales

- **Microservicios** para desacoplar UI/servicios.

- **CQRS**: lecturas optimizadas en Movimientos (SQL + Redis); escrituras transaccionales en Transferencias.
- **Saga + Outbox** en Transferencias para orquestar pasos con compensación y entrega confiable de eventos.
- **Event-Driven** con Service Bus/Event Grid para desacoplar procesos críticos.
- **ACL/Facade** frente a Core Bancario y Sistema Complementario para aislar contratos.

3.3. Seguridad y Acceso

- **Autenticación/Autorización**: OAuth 2.0/OIDC con Azure Entra ID.
- **APIM** valida JWT y aplica políticas (reintentos, límites, cuotas, caché).
- **Canales cifrados**: REST + HTTPS/TLS ≥1.2; mTLS y OAuth2.0..

3.4. Datos y Mensajería

- **Azure SQL**: órdenes, estados, proyecciones de lectura (CQRS).
- **Azure Cache for Redis**: caché para clientes frecuentes.
- **Azure Service Bus**: colas para procesos críticos (DLQ, reintentos).
- **Azure Event Grid**: eventos informativos y notificaciones.
- **Azure Data Lake**: auditoría y cumplimiento.

3.5. Integración Core y Sistema Complementario

- **Protocolos**: REST/HTTPS (o SOAP si legado) + OAuth2 CC + mTLS.
- **Capa ACL/Facade**: normaliza modelos, controla timeouts/retries/circuit breaker y aplica idempotencia.

3.6. Onboarding Biométrico

- **Flujo**: APIM > MS Onboarding > carga de documento/selfie a Blob > validación con OCR y Facephi.
- **Decisión**: persistencia en SQL, publicación de OnboardingAprobado/Rechazado (Service Bus/Event Grid).
- **Seguridad**: datos cifrados en tránsito y reposo.

3.7. Notificaciones Multicanal

- MS Notificaciones consume eventos y envía SMS/Email/Push; WhatsApp como canal informativo no oficial.
- Azure Communication Services (SMS/Email) y FCM/APNs (push).

3.8. Observabilidad y Operación

- **Application Insights + Log Analytics**: métricas, logs, trazas distribuidas.
- **Alertas** por SLO (latencia, tasa de error, disponibilidad).
- **Auto-healing**: HPA/KEDA para escalar por carga.

3.9. Alta Disponibilidad y DR

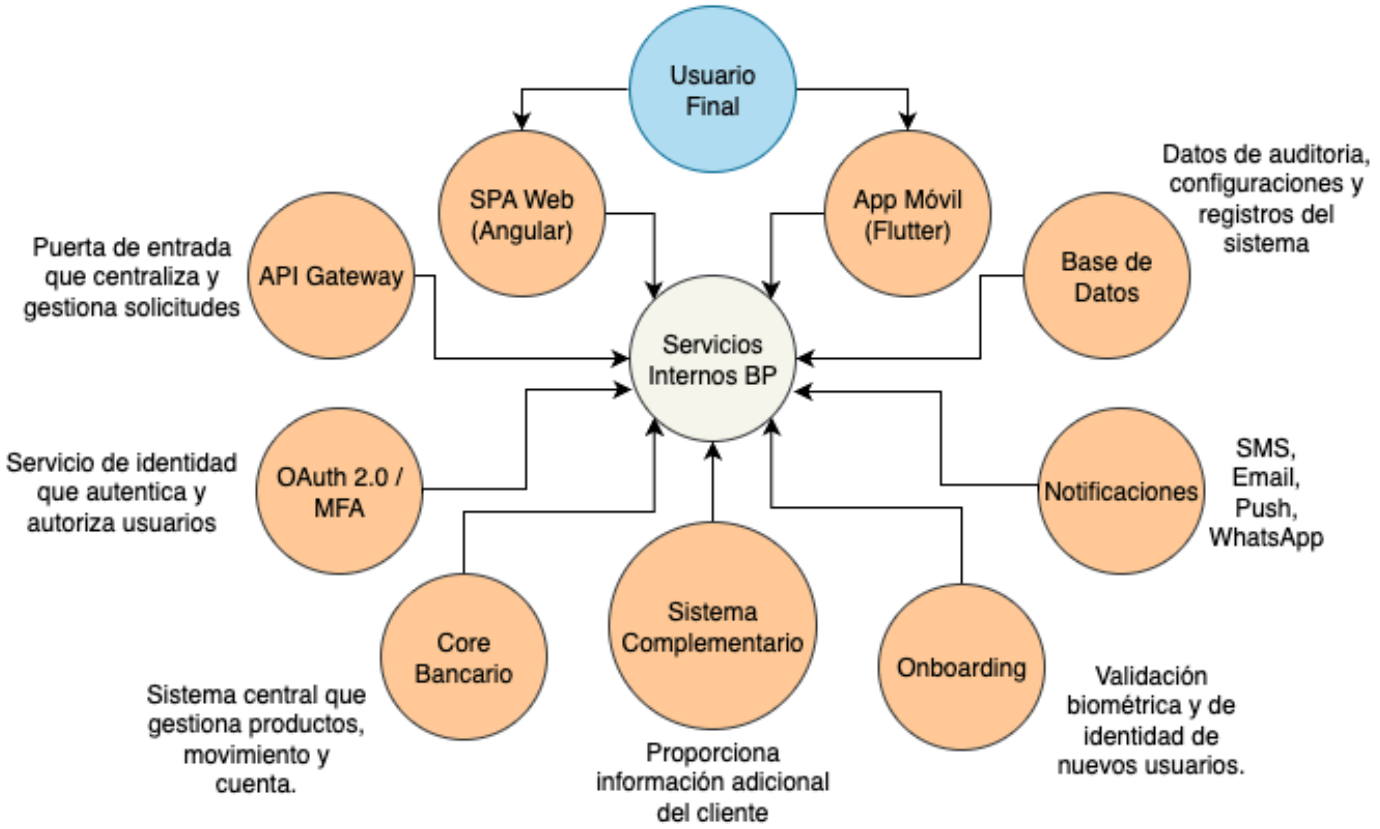
- **HA**: APIM/AKS/SQL/Redis con múltiples réplicas a nivel mundial.
- **DR**: multi-región con Front Door, APIM multi-región, SQL Auto-Failover Groups, Blob GZRS/RA-GZRS, Service Bus.

3.10. Mapeo de Servicios Azure

- **Entrada**: Front Door (WAF), API Management.
- **Cómputo**: AKS.
- **Identidad**: Azure Entra ID.
- **Datos**: Azure SQL, Redis, Blob, Data Lake.
- **Integración**: Service Bus, Event Grid.
- **Observabilidad**: Application Insights, Log Analytics.
- **Seguridad**: Key Vault, Private Endpoints, DDoS Standard.

4. Modelo C4

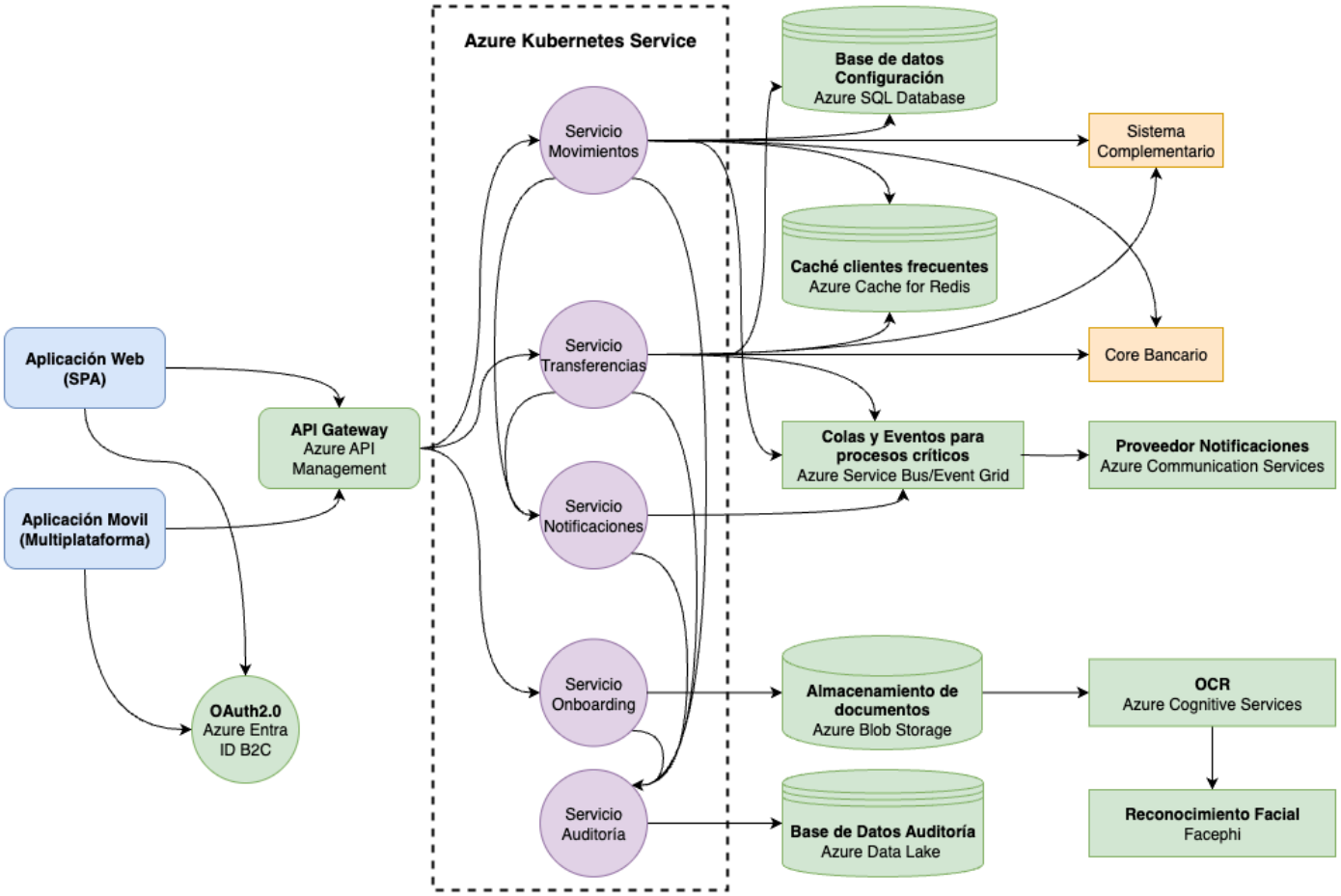
4.1. Diagrama de Contexto



El diagrama de contexto presenta la visión de alto nivel de la plataforma de banca digital propuesta para BP, mostrando los actores clave, los sistemas internos y servicios externos que interactúan con el sistema.

- **Usuario Final**: Clientes del banco que acceden a la plataforma desde la App Web o la App Móvil.
- **API Gateway**: Punto único de entrada que centraliza, enruta y protege las solicitudes hacia los servicios internos.
- **OAuth 2.0 / MFA (Azure Entra ID)**: Servicio de identidad que gestiona la autenticación, autorización y múltiples factores de verificación.
- **Servicios Internos BP**: Conjunto de microservicios responsables de procesar operaciones, consultas y lógica de negocio.
- **Core Bancario**: Sistema central que administra productos, cuentas y movimientos.
- **Sistema Complementario**: Proporciona información adicional del cliente no gestionada en el core principal.
- **Base de Datos**: Almacena datos de auditoría, configuraciones y registros críticos del sistema.
- **Notificaciones**: Servicio que envía alertas transaccionales y mensajes por canales SMS, email, push y WhatsApp.
- **Onboarding**: Proceso de registro digital de nuevos usuarios, con validación biométrica facial y OCR para verificación de identidad.

4.2. Diagrama de Contenedores



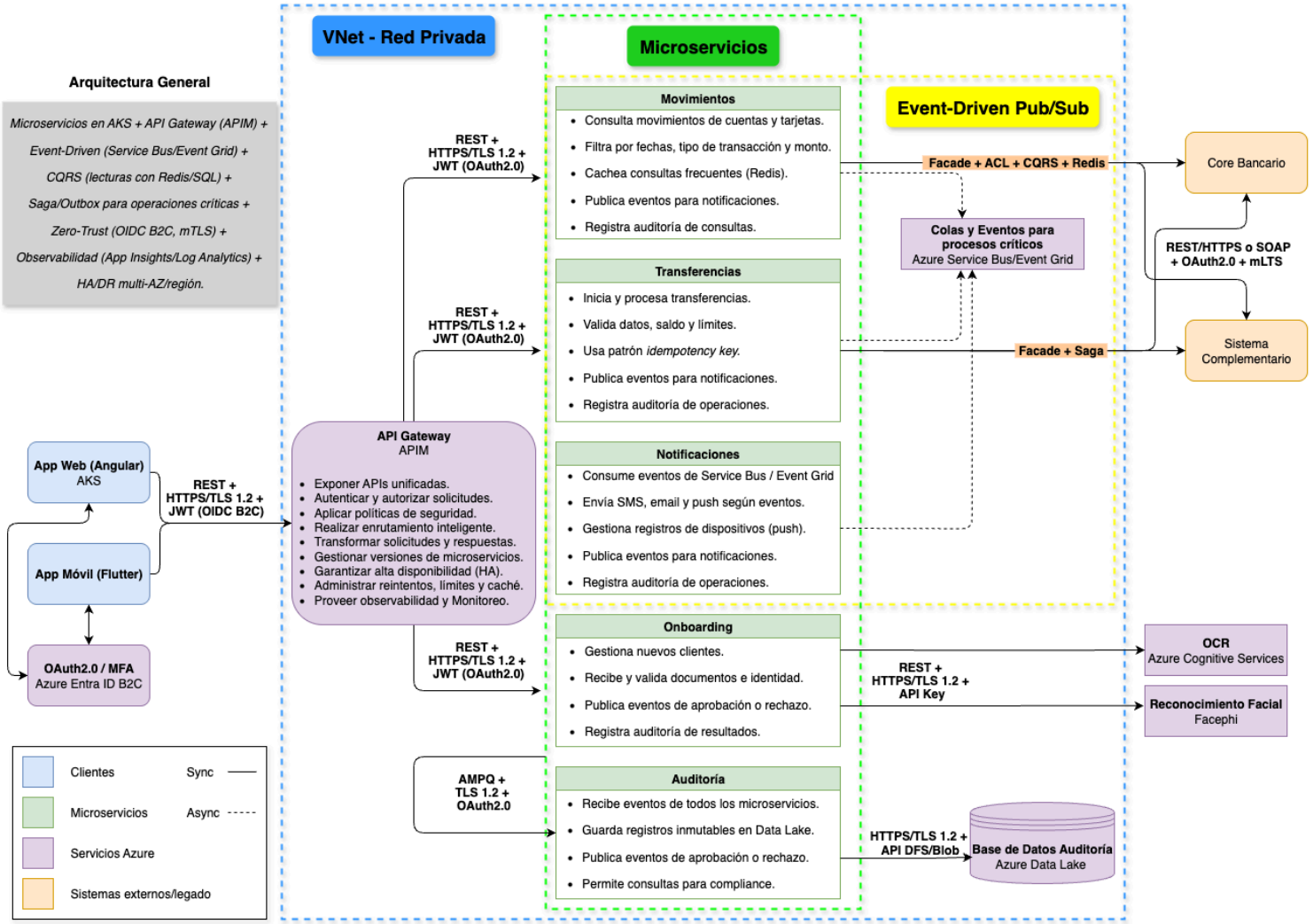
El sistema de banca digital se compone de dos canales principales: aplicación web desarrollada en Angular (SPA) y aplicación móvil multiplataforma con Flutter. Ambos se autentican mediante OAuth 2.0/OIDC con Azure Entra ID B2C (con MFA) y acceden a los servicios internos a través de un API Gateway implementado en Azure API Management, que unifica la exposición de APIs, aplica políticas de seguridad y enruta solicitudes hacia los microservicios alojados en Azure Kubernetes Service (AKS).

Dentro de AKS, los microservicios especializados gestionan funcionalidades clave:

- Movimientos y Transferencias consultan y actualizan datos en Azure SQL Database, optimizan rendimiento con Azure Cache for Redis y publican eventos para otros componentes.
- Notificaciones procesa eventos desde Azure Service Bus/Event Grid para enviarlos a través de Azure Communication Services (SMS, email, push) y canales adicionales como WhatsApp (informal).
- Onboarding integra validación biométrica y documental mediante Azure Cognitive Services (OCR) y Facephi, almacenando la documentación en Azure Blob Storage.
- Auditoría consolida registros inmutables en Azure Data Lake para consultas y cumplimiento normativo.

La integración con Core Bancario y Sistema Complementario se realiza mediante API REST/HTTPS o SOAP, protegidas con OAuth 2.0 y mTLS, garantizando seguridad y cumplimiento regulatorio. Los servicios de mensajería y eventos permiten la comunicación asincrónica y desacoplada, asegurando escalabilidad y resiliencia.

4.3. Diagrama de Componentes



El diagrama representa la arquitectura lógica de la solución de Banca para BP, modelada sobre principios de microservicios en la nube con enfoque Event-Driven y comunicación segura extremo a extremo. Se organiza en capas y zonas claramente delimitadas para facilitar la lectura y la comprensión técnica.

4.3.1. Clientes

- Aplicación Web (SPA – Angular en AKS) y Aplicación Móvil (Flutter) conforman los canales digitales para usuarios finales.
- Ambas aplicaciones se comunican con el backend exclusivamente a través de Azure API Management (APIM), utilizando el protocolo REST sobre HTTPS/TLS 1.2 y portando tokens JWT emitidos bajo OIDC B2C (Azure Entra ID B2C).
- El proceso de autenticación soporta OAuth 2.0 con MFA, integrando verificación multifactor en operaciones críticas.

4.3.2. Api Gateway

- Es la puerta de entrada para todas las solicitudes de clientes externos.
- Expone APIs unificadas, aplica políticas de seguridad, ejecuta enrutamiento inteligente y controla acceso mediante OAuth 2.0.
- Gestiona la transformación de solicitudes y respuestas, control de versiones, cuotas, límites, caché y observabilidad.

- Garantiza alta disponibilidad (HA) y resiliencia, además de integrarse con mecanismos de monitoreo y logging.

4.3.3. Microservicios

Cada servicio implementa funciones específicas con seguridad, escalabilidad y resiliencia:

- **Movimientos**: expone APIs de consulta optimizadas con caché y proyecciones de lectura rápida.
- **Transferencias**: procesa operaciones con patrón Saga, idempotencia y validaciones transaccionales.
- **Notificaciones**: distribuye eventos por múltiples canales (push, email, SMS, WhatsApp no transaccional).
- **Onboarding**: gestiona alta digital con validación biométrica Facephi y OCR.
- **Auditoría**: registra eventos inmutables y consultables para compliance.

Todos aplican autenticación OAuth2.0, observabilidad y comunicación segura (TLS 1.2+).

4.3.4. Event-Driven Pub/Sub

- Implementado con Azure Service Bus para colas y tópicos críticos, y Event Grid para distribución tipo fan-out.
- Los microservicios publican y consumen eventos para desacoplar procesos y mejorar escalabilidad.
- Incluye patrones como CQRS (lecturas optimizadas con Redis), Outbox para entrega garantizada, ACL y Facade para encapsular interacciones con sistemas externos.

4.3.5. Sistemas Externos / Legacy

- Core Bancario y Sistema Complementario: sistemas críticos que gestionan productos, cuentas, movimientos y datos adicionales de clientes.
- La comunicación se realiza a través de canales privados (VPN o ExpressRoute), utilizando REST/HTTPS o SOAP + OAuth 2.0 + mTLS para máxima seguridad.

4.3.6. Servicios Azure

- **Azure Cache for Redis**: optimiza consultas frecuentes, reduciendo carga sobre bases relacionales.
- **Azure SQL Database**: base de datos operacional para configuración y transacciones.
- **Azure Blob Storage**: almacenamiento seguro de documentos durante onboarding, usando SAS tokens para subida directa desde el cliente.
- **Azure Data Lake**: repositorio para auditorías y datos históricos.
- **Azure Cognitive Services (OCR) y Facephi**: validación documental y biométrica en el proceso de onboarding.
- **Azure Communication Services**: envío multicanal de notificaciones.

4.3.7. Seguridad y Observabilidad

- **Zero-Trust**: todas las conexiones están cifradas y autenticadas con certificados o tokens.
- MFA y step-up authentication para transacciones de alto riesgo.
- Observabilidad mediante Azure Application Insights y Log Analytics, recolectando métricas, logs y trazas distribuidas.
- Alta Disponibilidad y Recuperación ante Desastres (HA/DR) multi-AZ/región.

5. Arquitectura Frontend

Se optó por **Angular** (SPA) en lugar de **Next.js** (React SSR) por su arquitectura prescriptiva, por sus herramientas integradas para equipos grandes y menor complejidad operativa en un canal bancario autenticado que no requiere SEO. Esto permite desplegar como SPA optimizada en CDN, reduciendo dependencias de infraestructura Node y facilitando mantenibilidad.

En móvil, aunque en la industria bancaria moderna la mayoría de implementaciones utilizan **React Native** y en segundo lugar **Flutter**, se prefirió **Flutter** por su rendimiento cercano a nativo gracias a su motor gráfico propio, mayor consistencia visual entre iOS y Android, y facilidad para integrar SDKs de seguridad y biometría (como Facephi) sin depender de bridges. Su compilación directa fortalece la seguridad, y el control completo del renderizado minimiza bugs visuales, priorizando la experiencia y robustez en un entorno financiero crítico.

6. Autenticación y Autorización

La arquitectura de autenticación y autorización se basa en **OAuth 2.0** con OpenID Connect, utilizando Azure Entra ID (External ID/B2C) como proveedor de identidad para los clientes finales. Tanto la aplicación web (Angular) como la móvil (Flutter) implementan el flujo **Authorization Code con PKCE** y autenticación multifactor (MFA). Este enfoque permite que las credenciales sensibles nunca se almacenen en el cliente y que la validación se realice de forma segura contra el IdP, emitiendo tokens JWT que posteriormente son validados por Azure API Management antes de enrutar las solicitudes hacia los microservicios internos. Entre el API Gateway y los microservicios, así como en la integración con sistemas como el Core Bancario o el Sistema Complementario, se emplea el flujo Client Credentials junto con mTLS para reforzar la seguridad en la comunicación servidor a servidor.

La elección de **Authorization Code con PKCE** responde a que es el estándar más seguro para aplicaciones SPA y móviles, mitigando ataques de interceptación de código y permitiendo la integración de MFA y autenticación adaptativa. Alternativas como **Implicit Grant** han sido desaconsejadas por la comunidad y los organismos de estándares debido a la exposición directa de tokens, y enfoques como Resource Owner Password Credentials rompen el principio de delegación segura que ofrece OAuth, además de incrementar el riesgo de robo de credenciales. El uso de **OpenID Connect** añade una capa de identidad estandarizada, permitiendo interoperabilidad y facilitando el SSO si fuese necesario.

El sistema soporta varios **métodos de acceso**: usuario y contraseña como credenciales base; MFA mediante OTP por SMS o email, o notificación push; autenticación biométrica del dispositivo (FaceID, TouchID o BiometricPrompt) como factor local para desbloquear la sesión. En el onboarding, se integra validación documental mediante OCR y reconocimiento facial con **Facephi**, asegurando la verificación de identidad antes de la creación de la cuenta digital. Este proceso biométrico inicial es independiente de la biometría utilizada en los inicios de sesión posteriores, que actúa únicamente como mecanismo local de desbloqueo seguro.

Si bien existen alternativas como Auth0, Okta, Keycloak o Amazon Cognito, la adopción de Azure Entra ID responde a su integración nativa con el ecosistema Azure, soporte completo de OAuth2/OIDC, facilidad para aplicar políticas de MFA y step-up authentication, y capacidad de gobernanza centralizada de identidades. Esto reduce la complejidad de integración, mejora

la coherencia de seguridad en todos los canales y garantiza cumplimiento con las regulaciones del sector financiero.

7. Acceso a Datos

7.1. CQRS para Movimientos

Movimientos prioriza lecturas rápidas y masivas. Aplicamos CQRS: las escrituras permanecen en el Core/Sistema Complementario y las lecturas se sirven desde proyecciones optimizadas y caché.

- **Flujo**: Core/Complementario > (eventos/cambios) > Proyección en Azure SQL > Cache en Redis.
- **Consultas**: paginadas, filtros por fecha/tipo/monto. Uso de caché para respuestas rápidas.
- **Consistencia**: eventual; invalidación/refresh de caché cuando llegan eventos de “movimiento registrado”.
- **Rendimiento**: índices por (cuenta, fecha, tipo), particionamiento por período, TTLs diferenciados (saldo corto; movimientos medio).
- **Seguridad**: acceso vía microservicio (nunca directo), TLS ≥1.2, Data Masking y encriptado en reposo (TDE).

7.2. Transacciones de Transferencias

Transferencias es crítico y ACID. El servicio escribe en Azure SQL (estado/orden), coordina pasos con Saga y garantiza no duplicar con Idempotency-Key.

- **Flujo**: POST /transferencias > valida saldo/límites > transacción local > Outbox > comando al Core (REST/SOAP + OAuth2 CC + mTLS).
- **Consistencia**: eventual controlada mediante Saga (débito > crédito > confirmación; con compensación si falla).
- **Idempotencia**: clave por (cliente, monto, cuenta destino, ventana de tiempo).
- **Resiliencia**: Service Bus para mensajes críticos, DLQ para mensajes envenenados, retries con backoff, circuit-breaker hacia Core/Complementario.
- **Seguridad**: step-up MFA en montos/riesgos altos (claim acr/amr), mTLS y OAuth2 client-credentials en server-to-server.

7.3. Persistencia de Auditoría

La auditoría es append-only, inmutable y consultable para compliance. La ingesta es asíncrona para no frenar al usuario.

- **Ingesta**: todos los MS publican evento de auditoría → Service Bus Topic “auditoría” → MS Auditoría escribe en Azure Data Lake (ADLS Gen2) con esquema inmutable.
- **Modelo**: (actor, acción, recurso, timestamp, trace-id, resultado, payload-hash); sin PII en claro.
- **Integridad y retención**: sellado por lote (hash encadenado), contenedores WORM si aplica, políticas de retención (p. ej., 5–7 años).
- **Consultas**: vistas/materializaciones para back-office en SQL/Workbooks; búsquedas por trace-id desde Application Insights/Log Analytics.
- **Seguridad**: acceso por RBAC/ABAC, Private Endpoints, cifrado con CMK en Key Vault.

8. Integración Core Bancario y Sistema Complementario

8.1. ACL/Facade y Protocolos

La comunicación con el Core Bancario y el Sistema Complementario se realiza siempre a través de ACLs (Anti-Corruption Layer) y patrones Facade, que encapsulan la lógica de integración y aíslan los microservicios de la complejidad y las particularidades de los sistemas legados. Esto permite traducir modelos de datos, normalizar respuestas y evitar que cambios en el core impacten directamente en el dominio interno. Los facades también centralizan el manejo de errores, tiempos de espera y reintentos, simplificando la lógica en los servicios de negocio.

En cuanto a protocolos, la arquitectura admite REST/HTTPS o SOAP dependiendo de la capacidad del sistema fuente. En todos los casos, la comunicación se cifra con TLS 1.2 o superior y se autentica con OAuth 2.0 (Client Credentials), añadiendo mTLS para validación mutua entre extremos. El tráfico fluye únicamente por canales privados mediante VPN o ExpressRoute, garantizando seguridad y cumplimiento normativo.

8.2. Estrategias Síncronas y Asíncronas

La estrategia de comunicación combina enfoques síncronos y asíncronos según la naturaleza de la operación. Las operaciones críticas y que requieren confirmación inmediata (por ejemplo, débito y crédito en transferencias) se ejecutan de forma síncrona, asegurando respuesta en línea al usuario y consistencia inmediata en el core. Por otro lado, las operaciones que no afectan la experiencia inmediata o que generan grandes volúmenes de eventos (como actualizaciones de auditoría, replicación de datos o notificaciones) se manejan de forma asíncrona mediante Azure Service Bus o Event Grid, permitiendo desacoplar los procesos y mejorar la escalabilidad.

Este diseño asegura que las integraciones con sistemas legados se mantengan controladas, seguras y resistentes a cambios, a la vez que optimiza el rendimiento global de la plataforma.

9. Notificaciones

El sistema de notificaciones está diseñado para cumplir con la normativa bancaria ecuatoriana, que exige informar al cliente sobre cada transacción relevante, y al mismo tiempo ofrecer una experiencia ágil y multicanal. Un microservicio de Notificaciones centraliza la lógica, consumiendo eventos publicados por otros servicios a través de Azure Service Bus y Event Grid. Este enfoque desacoplado garantiza que las notificaciones no dependan de la ejecución directa de las transacciones, mejorando la resiliencia y evitando demoras para el usuario final.

Los canales principales son SMS, email y notificaciones push (mediante FCM/APNs), asegurando redundancia: si un canal falla, otro puede cumplir la función. Adicionalmente, se integra WhatsApp como canal informativo no transaccional, útil para recordatorios o comunicaciones generales, pero no para confirmaciones de operaciones sensibles.

El microservicio gestiona plantillas y preferencias del usuario, permitiendo personalizar qué eventos generan notificaciones y por qué canal se envían. Para cada envío se almacena un registro en la auditoría, garantizando trazabilidad y cumplimiento normativo. La comunicación con proveedores externos (por ejemplo, gateways SMS o servicios de correo) se realiza mediante API seguras sobre HTTPS/TLS 1.2, autenticadas con claves gestionadas en Azure Key Vault.

Este diseño multicanal, basado en eventos y desacoplado de la lógica transaccional, permite alta disponibilidad, escalabilidad ante picos de demanda y la flexibilidad necesaria para incorporar nuevos canales o proveedores en el futuro sin modificar la lógica principal del negocio.

10. Onboarding Biométrico

El proceso de onboarding permite que nuevos clientes abran su cuenta y obtengan acceso a los canales digitales sin acudir a una agencia física, cumpliendo con las normativas y la prevención de fraude. Este flujo se centraliza en un microservicio de Onboarding, que orquesta la captura, validación y almacenamiento seguro de la información.

Desde la aplicación móvil (Flutter), el usuario inicia el proceso capturando una imagen de su documento de identidad, que es procesada por OCR a través de Azure Cognitive Services para extraer datos de manera automática. De forma paralela, se solicita una selfie para verificación biométrica con Facephi, que incluye detección de prueba de vida (liveness detection) y comparación facial contra la imagen del documento. Ambas validaciones se realizan mediante integraciones API seguras con los servicios especializados, usando REST + HTTPS/TLS 1.2 y autenticación basada en claves protegidas en Azure Key Vault.

Una vez que las verificaciones son aprobadas, el sistema aplica reglas adicionales de negocio (edad mínima, listas de control, scoring) y registra el resultado en la auditoría. Los documentos e imágenes capturados se almacenan cifrados en Azure Blob Storage utilizando claves gestionadas por el banco, con políticas de retención limitadas y acceso controlado para cargas seguras desde el cliente.

El onboarding culmina con la creación del perfil en Azure Entra ID (B2C), habilitando al cliente para iniciar sesión con usuario y contraseña, y ofreciendo opciones de configuración de MFA y biometría en el dispositivo. Gracias a su arquitectura basada en eventos y ejecución paralela de validaciones, este proceso puede completarse en menos de dos minutos en condiciones óptimas, garantizando seguridad, rapidez y experiencia de usuario.

11. Auditoría y Clientes Frecuentes

La auditoría es un componente crítico para el cumplimiento normativo y la trazabilidad de todas las operaciones realizadas por los clientes y el sistema. Su diseño sigue un patrón append-only, donde cada evento relevante (inicios de sesión, transferencias, consultas, cambios de datos) se registra como un mensaje estructurado en un topic de Azure Service Bus. Un microservicio de Auditoría consume estos eventos y los persiste en Azure Data Lake con esquema inmutable, garantizando integridad mediante hashes encadenados y almacenamiento en contenedores. Las consultas para áreas como compliance o seguridad se

realizan sobre vistas preprocesadas, manteniendo separados el entorno de consulta y el de ingestión para preservar el rendimiento.

La persistencia para clientes frecuentes busca optimizar la experiencia de usuarios que consultan repetidamente su saldo o movimientos. Para esto, se implementa un patrón cache-aside con Azure Cache for Redis, que almacena las consultas más comunes y reduce la dependencia del Core Bancario para datos que cambian con menor frecuencia. Cuando se detecta un cambio relevante (por ejemplo, un nuevo movimiento), el evento correspondiente invalida o actualiza la caché para mantener consistencia eventual. En casos donde la disponibilidad de datos recientes es prioritaria, se aplican proyecciones preprocesadas y el patrón CQRS, separando las consultas de las escrituras críticas.

12. Seguridad y Cumplimiento Normativo

La arquitectura está diseñada bajo el principio de seguridad por capas, incorporando controles técnicos y procedimentales que cumplen con las normativas de la Superintendencia de Bancos del Ecuador y estándares internacionales como PCI DSS, ISO 27001 y OWASP ASVS. Cada canal, componente y flujo de datos se protege mediante cifrado, autenticación robusta, autorización granular y monitoreo continuo.

Todas las comunicaciones entre clientes, API Gateway y microservicios utilizan HTTPS/TLS 1.2 o superior, mientras que la comunicación entre servicios internos y con sistemas críticos (Core Bancario y Sistema Complementario) añade mutual TLS (mTLS) y autenticación mediante OAuth 2.0 Client Credentials. El acceso a datos en reposo se protege con cifrado mediante Customer Managed Keys en Azure Key Vault, asegurando control total sobre la gestión de claves y su rotación.

Se implementan políticas de control de acceso basado en roles (RBAC) y atributos (ABAC) para garantizar el principio de mínimo privilegio, tanto a nivel de usuario como de servicio. Esto se refuerza con segmentación de red usando VNets y Private Endpoints, minimizando la exposición pública de componentes críticos como bases de datos, almacenamiento y mensajería.

Para la detección y respuesta ante incidentes, se integran Azure Monitor, Application Insights, Log Analytics y sistemas SIEM corporativos, permitiendo correlacionar eventos y generar alertas en tiempo real. El monitoreo incluye métricas de rendimiento, disponibilidad y seguridad, junto con auditorías periódicas para identificar vulnerabilidades y asegurar la conformidad con los estándares regulatorios.

El cumplimiento normativo incluye:

- Notificación obligatoria de transacciones al cliente por al menos dos canales (SMS y correo electrónico).
- Registros de auditoría inmutables de todas las operaciones, conservados según políticas regulatorias (5–7 años).
- MFA obligatorio en operaciones de alto riesgo, conforme a las directrices de prevención de fraude.
- Protección de datos personales alineada con la Ley Orgánica de Protección de Datos Personales del Ecuador, incluyendo consentimiento informado, minimización de datos y eliminación segura al final de la retención.

Este enfoque asegura que la banca digital no solo cumpla con los requisitos legales y regulatorios locales, sino que también iguale o supere las mejores prácticas internacionales, manteniendo la confianza del cliente y la reputación institucional.

Durante el diseño de esta solución se evaluaron alternativas como Auth0, Okta y Keycloak para la gestión de identidades, así como distintos flujos de OAuth2.0 (Implicit Grant, ROPC) y mecanismos de cifrado y control de acceso. También se consideraron enfoques de red expuesta con filtrado por WAF frente a aislamiento total mediante VNets y Private Endpoints. La elección final priorizó Azure Entra ID con flujo Authorization Code + PKCE y MFA, cifrado con claves gestionadas en Azure Key Vault, segmentación de red privada y control de acceso granular (RBAC/ABAC), garantizando alineación con la regulación local, estándares internacionales y una superficie de ataque mínima, además de tener la infraestructura en Azure lo cual facilita su integración.

13. Alta Disponibilidad, Tolerancia a Fallos y DR

La arquitectura está diseñada para ofrecer alta disponibilidad (HA) en todos los componentes críticos, asegurando la continuidad del servicio incluso ante fallos de infraestructura o picos de demanda. Los microservicios se despliegan en Azure Kubernetes Service (AKS) con Horizontal Pod Autoscaler (HPA) y KEDA para escalar dinámicamente según métricas de uso (CPU, memoria o eventos de cola). Las bases de datos utilizan Auto Failover Groups y replicación geográfica para mantener instancias listas en una región secundaria.

La tolerancia a fallos se implementa mediante patrones como circuit breaker (Resilience4j/Polly) para evitar cascadas de errores, reintentos con backoff exponencial y colas DLQ (Dead Letter Queue) para mensajes que no pudieron procesarse. Los servicios críticos incluyen health probes para supervisar su estado y permitir la autorrecuperación (self-healing) en caso de fallos. Además, el diseño aplica bulkheads para aislar cargas y evitar que un servicio afecte la disponibilidad de otros.

La recuperación ante desastres (DR) sigue una estrategia activo-pasivo entre regiones de Azure, utilizando Azure Front Door para redirigir tráfico en caso de indisponibilidad regional. El almacenamiento en Azure Blob emplea replicación GZRS/RA-GZRS para garantizar la durabilidad de datos, y las políticas de RPO y RTO están definidas para minimizar pérdida de datos y tiempo de inactividad, asegurando que los sistemas puedan restaurarse en minutos u horas, según la criticidad del componente.

Este enfoque integral permite que la plataforma mantenga un servicio seguro, escalable y resiliente, incluso en escenarios de alta concurrencia, fallos de hardware o incidentes a nivel regional.

14. Monitoreo y Observabilidad

La solución implementa un modelo de observabilidad integral con Azure Monitor, Application Insights, Log Analytics y el SIEM, ofreciendo visibilidad de extremo a extremo desde el frontend hasta las integraciones con el Core Bancario. Los microservicios y frontends están instrumentados con métricas, logs y trazas distribuidas mediante OpenTelemetry, permitiendo seguir transacciones completas y detectar problemas rápidamente.

Las alertas se basan en SLOs definidos por el negocio, con notificaciones automáticas y flujos de remediación cuando es posible. Los tableros en Azure Dashboards facilitan el monitoreo en tiempo real, mientras que los logs centralizados y el RUM en frontend ayudan a optimizar rendimiento y cumplir con auditorías regulatorias.

15. Costos y Escalabilidad

La arquitectura propuesta prioriza el time-to-market y la escalabilidad dinámica, optimizando al mismo tiempo el uso de recursos para controlar costos. Los microservicios se despliegan en Azure Kubernetes Service (AKS) con HPA y KEDA, lo que permite escalar automáticamente en función de la demanda y reducir instancias en periodos de baja carga. Servicios como Azure API Management, Azure SQL Database y Azure Cache for Redis utilizan planes de consumo o escalado elástico para ajustar su capacidad sin sobredimensionar la infraestructura.

El enfoque cloud-native elimina inversiones iniciales en hardware y permite pagar únicamente por los recursos consumidos, con visibilidad de gastos mediante Azure Cost Management y prácticas FinOps para optimizar el gasto mensual. La arquitectura de eventos desacoplados reduce el consumo innecesario de recursos, y el uso de caché distribuido y CQRS minimiza las cargas costosas sobre sistemas core.

Este modelo permite que la plataforma atienda picos de demanda de millones de clientes sin degradar la experiencia, al mismo tiempo que mantiene un control estricto sobre el costo operativo, ajustando automáticamente capacidad y gasto a las necesidades reales del negocio.

Se utilizó la herramienta de Azure Pricing Calculator para estimar los gastos mensuales en infraestructura, resultando en un monto aproximado de USD 40,000 al mes. A este total se deben agregar servicios externos no incluidos en la calculadora: la licencia de Facephi para biometría facial (una solución SaaS que suele cobrarse bajo suscripción, estimada en unos USD 5,000/mes para un volumen bancario medio), y servicios de notificación como Twilio para SMS, WhatsApp y correo electrónico. Por ejemplo, enviando 100,000 SMS a USD 0.008/a mensaje, el costo alcanzaría alrededor de USD 800/mes, a lo que se suman tarifas de WhatsApp (USD 0.005/mensaje) o licencias de SLA de correo. Estas cifras son solo estimaciones rápidas; en un entorno real se deben analizar diferentes escenarios, volúmenes, acuerdos de volumen (tiering) y optimizaciones para obtener una proyección más precisa. Estimación Azure.

16. Riesgos y Mitigaciones

El diseño contempla riesgos técnicos, operativos y regulatorios propios de una plataforma bancaria, junto con estrategias claras para mitigarlos:

- Fallas en integraciones con Core Bancario o sistemas complementarios.
    - **Mitigación**: Uso de ACL/Facade para desacoplar, retries con backoff, DLQ para mensajes no procesados y pruebas periódicas de conectividad.
- Sobrecarga o indisponibilidad de servicios críticos.
    - **Mitigación**: Escalado automático (HPA/KEDA), patrones de circuit breaker y bulkheads, y despliegues con failover geográfico.

- Brechas de seguridad o incumplimiento normativo.
    - **Mitigación**: MFA, cifrado en tránsito y reposo (TLS 1.2+, CMK en Key Vault), segmentación de red con VNet/Private Endpoints, y auditoría inmutable conforme a la Ley de Protección de Datos Personales.
- Pérdida de datos ante incidentes.
    - **Mitigación**: Replicación geográfica (GZRS/RA-GZRS, Auto Failover Groups), backups periódicos y pruebas de restauración con RPO/RTO definidos.
- Degradación de experiencia del usuario en picos de demanda.
    - **Mitigación**: Uso de caché distribuido (Azure Redis), CQRS para separar lecturas/escrituras y monitoreo proactivo con alertas basadas en SLOs.

Estas medidas reducen el impacto de incidentes y aseguran la continuidad operativa, incluso en escenarios adversos.

17. Conclusiones

- La arquitectura propuesta ofrece una plataforma bancaria digital segura, escalable y resiliente, alineada con las prácticas actuales de Banco Pichincha y con los estándares internacionales del sector financiero. El uso de microservicios en Azure, integraciones desacopladas con el Core Bancario, y un modelo híbrido de comunicación síncrona y asíncrona garantiza rendimiento óptimo y flexibilidad para evolucionar el sistema sin comprometer la estabilidad.
- La adopción de OAuth 2.0/OIDC con Azure Entra ID, MFA y biometría refuerza la autenticación, mientras que la segmentación de red, cifrado de datos y auditoría inmutable aseguran el cumplimiento normativo. La observabilidad integral, combinada con estrategias de alta disponibilidad, tolerancia a fallos y DR, proporciona una base sólida para la continuidad del negocio.
- Finalmente, el enfoque cloud-native y el uso de IaC optimizan el despliegue, reducen el time-to-market y permiten adaptar la infraestructura a la demanda real, maximizando el valor para el negocio y la experiencia del cliente.

18. Recomendaciones

- Validar la arquitectura en entornos piloto antes del despliegue productivo, incluyendo pruebas de carga, resiliencia y seguridad, para confirmar el cumplimiento de SLOs y requisitos regulatorios.
- Fortalecer acuerdos con proveedores externos (Facephi, SMS, WhatsApp Business API) mediante contratos con SLA claros y pruebas periódicas de integración.
- Optimizar costos continuamente aplicando prácticas FinOps, revisando métricas de uso en Azure Cost Management y ajustando capacidades según la demanda real.
- Mantener actualizadas las políticas de seguridad y configuraciones de acceso en Azure Entra ID, Key Vault y APIM, alineándolas con cambios normativos y nuevas amenazas.
- Fomentar la observabilidad proactiva, integrando métricas de negocio y experiencia del usuario junto con métricas técnicas para anticipar problemas y mejorar la toma de decisiones.
- Planificar ejercicios regulares de DR y respuesta a incidentes, documentando aprendizajes y aplicando mejoras en procesos y configuraciones.