DEVSU

Ejercicio Práctico para Arquitectura de Integración

Miguel Cartagenova

30/06/2025

Introducción:

El presente documento tiene como objetivo diseñar una plataforma de integración de alto rendimiento para la modernización de sistemas bancarios, enfocada en aplicaren el catálogo del estándar BIAN para garantizar una organización común con Open Banking / Open Finance y en un enfoque Azure-First que aprovecha al máximo los servicios PaaS de Microsoft.

A lo largo del documento se irán describiendo, de forma estructurada, los distintos niveles de la arquitectura C4 (Contexto, Contenedores y Componentes), los patrones de integración empleados, que políticas de seguridad y cumplimiento recomendamos, así como las estrategias para alta disponibilidad, resiliencia y recuperación ante desastres, este enfoque incluye también migración multicore, diseño de APIs, gobierno y un plan de despliegue gradual usando mejores prácticas.

Justificación:

El modelo propuesto trata de proporcionar una visión clara y organizada de los diferentes niveles de la solución (Contexto, Contenedores, Componentes), facilitando la comunicación entre equipos de negocio, arquitectura y desarrollo.

El enfoque de mapear cada servicio y microservicio a un "*Service Domain BIAN*", trata de garantizar una taxonomía común y reducir la ambigüedad, promueve la reutilización de componentes y logra acelerar proyectos de integración dentro del sector financiero ecuatoriano, donde la cumplir la reglamentación de interoperabilidad es mandatorio y lograr una eficiencia, velocidad y estandarización es un diferenciador en el negocio.

Optar por un enfoque Azure-First permite maximizar el aprovechamiento de las capacidades PaaS de Microsoft, minimizando la complejidad operativa y los costes de mantenimiento.

Servicios como Azure API Management y Event Grid permiten exponer y orquestar APIs con políticas de seguridad y escalado automático sin desplegar infraestructura propia, mientras que AKS, Azure Functions y Logic Apps ofrecen mecanismos de ejecución

contenerizada y serverless que se ajustan dinámicamente a la demanda. Además, al centralizar la gestión de secretos en Key Vault y aprovechar Managed Identities, RBAC y PIM, se asegura que ningún componente almacene credenciales en texto plano y que todos los accesos se basen en la regla de mínimo privilegio, cumpliendo con los más altos estándares de seguridad y regulaciones locales (LOPDP, PCI DSS, ISO 27001).

Al usar Azure trataremos de garantizar también alta disponibilidad y resiliencia multinodo, con opciones integradas de geo-réplica, balanceo global (Front Door/Traffic Manager) y recuperación ante desastres (Site Recovery, Backups). Su ecosistema de monitorización (Application Insights, Log Analytics) y gobernanza (Azure Policy, Blueprints) permite implementar rápidamente controles de cumplimiento y visibilidad end-to-end, reduciendo el riesgo operacional.

Este conjunto de ventajas justifica la elección de Azure como pilar de la estrategia de modernización, al ofrecer un entorno escalable, seguro y alineado con las mejores prácticas del sector financiero y bancario.

Aunque presentamos un enfoque “Azure-First” para aprovechar sus servicios PaaS y SaaS, procuraremos minimizar el “cloud vendor lock-in” mediante el uso de estándares abiertos (OpenAPI, Kubernetes, Terraform), arquitecturas basadas en contenedores y capas de abstracción (por ejemplo, uso de AKS y Azure Functions tratando de mantener código portable). De este modo, en caso de necesitar migrar o integrar soluciones con otros proveedores, podremos reutilizar gran parte de la definición de infraestructura, APIs y lógica de negocio sin depender exclusivamente de APIs propietarias de Azure.

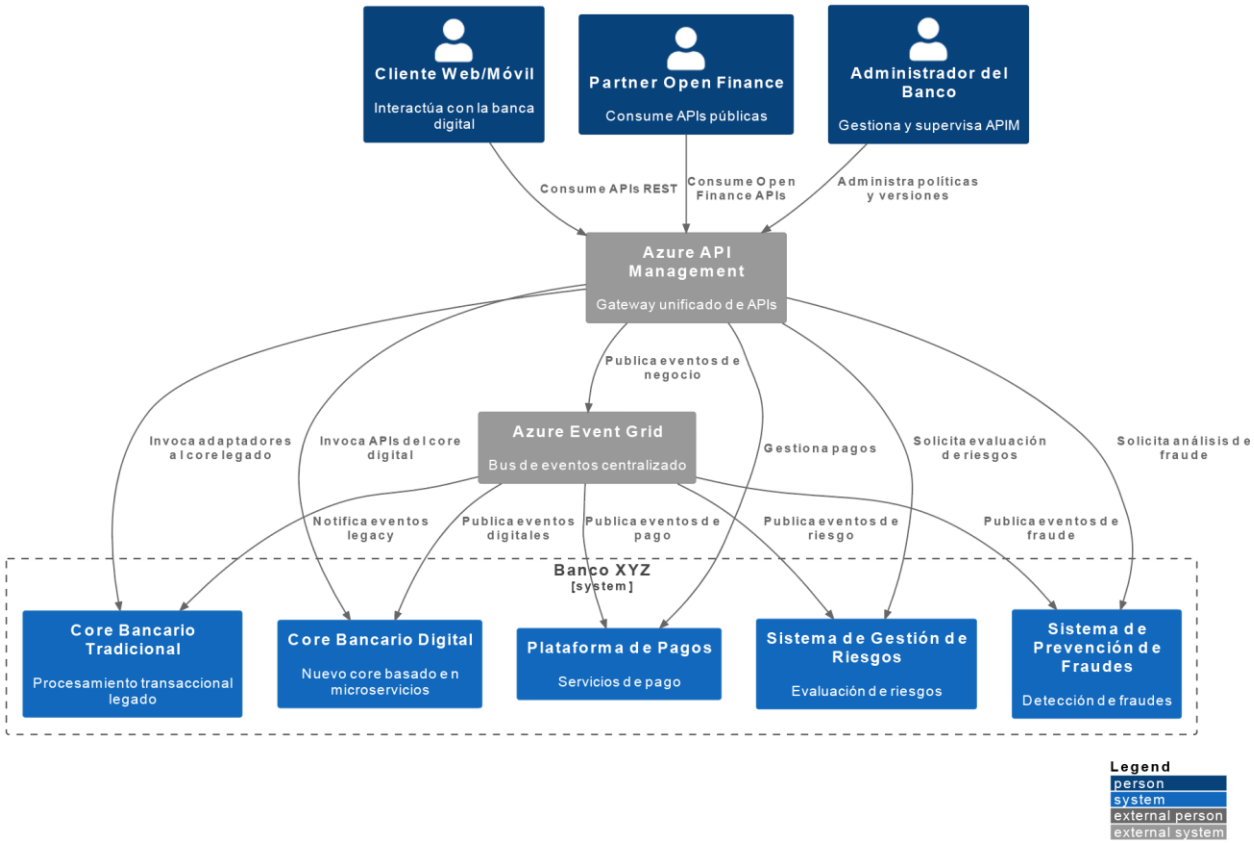
1. Diseño de Arquitectura

C4 – Nivel 1: Diagrama de Contexto

Este diagrama muestra los **actores externos** (Cliente Web/Móvil, Partner Open Finance, Administradores del Banco) y el **sistema de integración central** basado en **Azure API Management** y **Azure Event Grid**, que orquesta y expone las APIs hacia:

- **Core Bancario Tradicional**
- **Core Bancario Digital**
- **Plataforma de Pagos**
- **Sistema de Gestión de Riesgos**
- **Sistema de Prevención de Fraudes**

Cada uno de estos sistemas se mapea a un Service Domain de BIAN (por ejemplo, Payment Initiation, Risk Assessment, Fraud Detection, Customer Fulfillment), garantizando un lenguaje y taxonomía comunes.



(C4.1 Diagrama de Contexto.pdf)

Mapeo a Service Domains BIAN

Payment Initiation	Plataforma de Pagos
Risk Assessment	Sistema de Gestión de Riesgos
Fraud Detection	Sistema de Prevención de Fraudes
Customer Fulfillment	Core Bancario Digital / Legado

Actores Externos

- **Cliente Web/Móvil**: Usuarios finales que interactúan con la banca digital a través de aplicaciones web o móviles. Se autentican mediante OAuth2/OpenID Connect a través de **Azure ENTRA ID B2C**, y consumen APIs expuestas en **API Management**.
- **Partner Open Finance**: Servicios externos (fintech, agregadores de pagos) que, tras un onboarding previo, acceden a APIs públicas / privadas para consultar

información (cuentas, transacciones) y enviar órdenes de pago. La comunicación se asegura con certificados TLS de confianza mutua y scopes esctrictos mediante Entra ID.

- **Administrador del Banco**: Equipo interno de operaciones y gobernanza, que usa el **Developer Portal** de APIM y el **Azure Portal** para:
    - Definir políticas de seguridad y cuotas
    - Versionar y publicar nuevos productos de API
    - Revisar analytics y logs

Punto de Integración Central (Azure-First)

- **Azure API Management** (APIM) actúa como cortafuegos y gateway unificado. Está ubicado en la capa DMZ de la red (subred pública) frente a Internet y peered con la VNet interna donde residen los microservicios.
- **Azure Event Grid** se usa para distribuir eventos de negocio de forma ligera a consumidores internos; se configura dentro de la misma VNet o en un **Event Grid Domain** privado para garantizar tráfico cifrado en tránsito.

Sistemas del Banco

- **Core Bancario Tradicional**
    - Monolito heredado (SOAP/REST), on-premise o IaaS en Azure.
    - Mapeado al *Service Domain* BIAN “Customer Fulfillment” y “Account Management”.
    - Las llamadas pasan por un **Anti-Corruption Layer** (Azure Functions) al exponerlo como API REST moderna.
- **Core Bancario Digital**
    - Conjunto de microservicios .NET Core en AKS, diseñados según el estandar BIAN (“Payment Initiation”, “Transaction Authorization”).
    - Expone APIs de alto rendimiento/alta disponibilidad a través de APIM.
- **Plataforma de Pagos**
    - Servicio en tiempo real para iniciar, confirmar, anular, reversar pagos.
    - Mapeado a “Payment Initiation” y “Payment Execution” de BIAN.
- **Sistema de Gestión de Riesgos**
    - Motor de scoring y análisis estadístico.
    - Publica alertas de riesgo (eventos RiesgoAnañizado) en Event Grid.
    - Corresponde a “Risk Assessment” en BIAN.
- **Sistema de Prevención de Fraudes**
    - Motor analítico real time usando Sentinel (Smartsoft).
    - Emite FraudAlert a Event Grid y Service Bus para flujos de trabajo de investigación.
    - Mapeado a “Fraud Detection” en BIAN.

Conectividad y Seguridad de Red

- **Subred Pública (DMZ)**: APIM, Azure Front Door / WAF.
- **Subred Privada**: AKS, Functions, Logic Apps, Event Grid y Service Bus.

- **Peering y NSGs**: Control de tráfico entre zonas, con reglas estrictas que permiten solo los puertos y protocolos necesarios.
- **Managed Identities**: Todas las identidades de servicio (AKS, Functions, Logic Apps) acceden a Key Vault y bases de datos sin credenciales embebidas.

Observabilidad y Gobierno

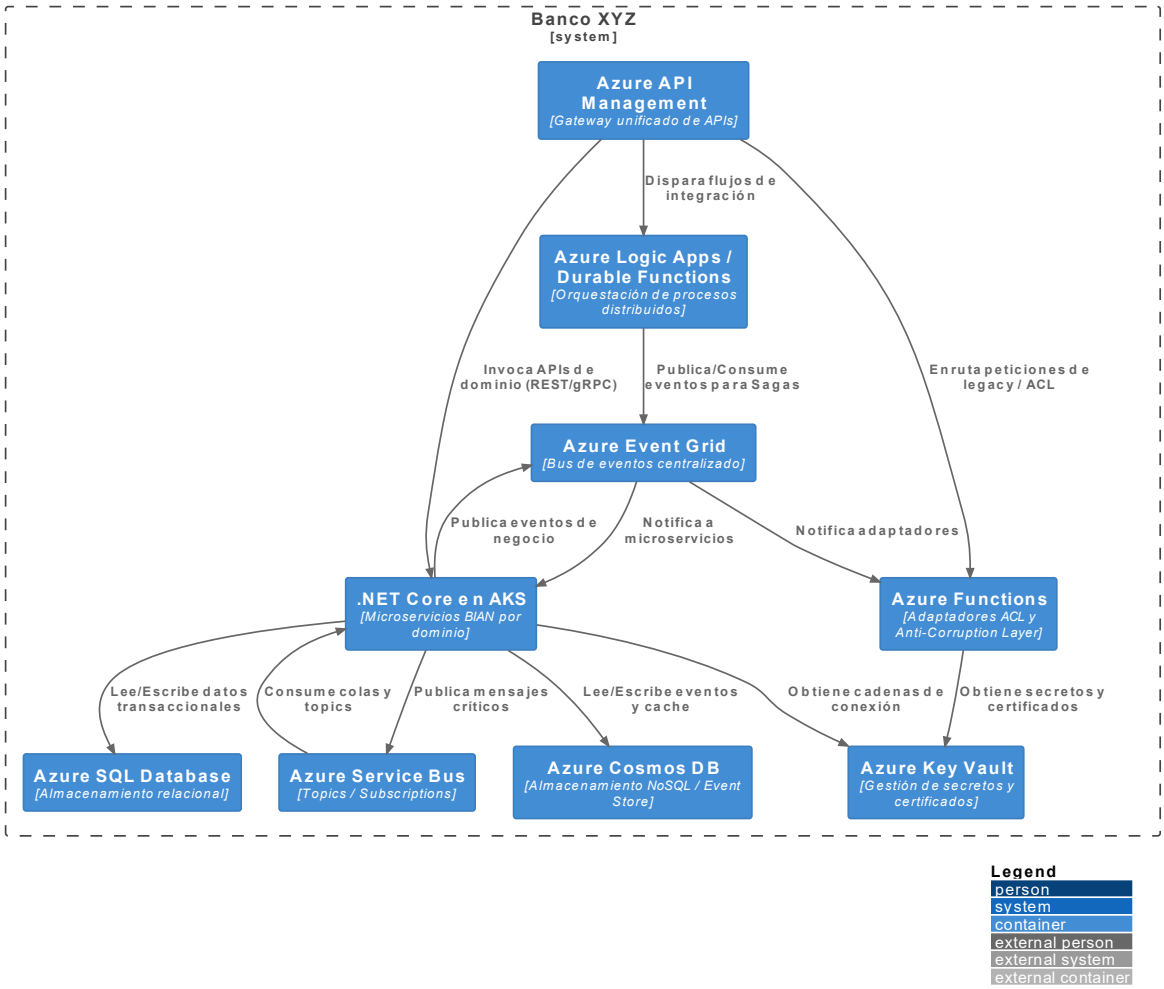
- **Azure Monitor & Application Insights**: Trazas distribuidas con Correlation IDs desde APIM hasta cada microservicio y función.
- **Log Analytics Workspace**: Almacena logs estructurados para auditoría y cumplimiento.
- **API Catalog**: Registro automático de APIs en APIM Registry, enlazado con Azure DevOps para CI/CD.

C4 – Nivel 2: Diagrama de Contenedores y Servicios

Este diagrama muestra los principales contenedores (aplicaciones y servicios) que componen la solución de integración, sus tecnologías basadas en Azure y cómo se comunican entre sí. Aquí vamos a incluir:

- **Azure API Management** como gateway central para todas las llamadas.
- **Azure Event Grid** y **Service Bus** para mensajería y eventos.
- **Azure Logic Apps** y **Durable Functions** para orquestar flujos de negocio distribuidos.
- **Microservicios .NET Core** desplegados en **AKS** que implementan cada *Service Domain* del estándar BIAN.
- **Azure Functions** como capa ligera de adaptación (ACL) al core legado.
- **Azure SQL Database** y **Cosmos DB** para persistencia relacional y NoSQL.
- **Azure Key Vault** para la gestión segura de secretos.

Este nivel de detalle nos ayuda a entender qué piezas ejecutan cada función dentro de la plataforma, cómo escalan y cómo aprovechan los servicios PaaS de Azure para agilizar despliegues, garantizar resiliencia y simplificar la operación.



(C4.2 Diagrama de Contenedores y Servicios.pdf)

Azure API Management

Razón: Actúa como el punto único de entrada y salida para todas las APIs. Permite exponer, versionar y gobernar rutas REST/gRPC sin modificar el código de los microservicios. Sus **políticas** integradas (caching, rate-limiting, transformación de payloads, CORS) y su Developer Portal facilitan la documentación, seguridad y consumo de las APIs por clientes internos y externos.

- **Beneficios Azure**:
    - Políticas de seguridad (throttling, CORS, caching) aplicables sin tocar código.
    - Developer Portal integrado para documentar y versionar tu catálogo de APIs.
    - Escalado automático y SLA alto garantizado por Microsoft.

Azure Event Grid
Es el bus de eventos ligero de Azure para distribuir notificaciones de negocio (p. ej. “Pago autorizado”, “Alerta de fraude”) con latencia mínima. Se integra de forma nativa con Functions, Logic Apps, Storage y otros servicios, permitiendo un modelo event-driven para desacoplar componentes y reaccionar a cambios de estado sin polling.

Azure Service Bus (Topics & Subscriptions)
Ofrece colas y topics con garantías transaccionales, dead-lettering y retry automático, ideal para flujos donde la entrega de mensajes críticos no puede perderse. Se utiliza para orquestar compensaciones de Saga, procesar lotes de transacciones y asegurar la **fiabilidad** de la mensajería en procesos financieros.

Azure Logic Apps / Durable Functions
Orquestan flujos de trabajo distribuidos y transacciones complejas.

- **Razón**: Orquestar procesos distribuidos (workflows) **y transacciones distribuidas** (patrón Saga).
- **Durable Functions**:
    - Código en C#/.NET para lógicas complejas de orquestación con estado.
    - Se integra con Event Grid y Service Bus sin infraestructura adicional.
- **Logic Apps**:
    - Low-code para casos sencillos (transformaciones, aprobaciones humanas, llamadas a APIs externas).

.NET Core en AKS (Azure Kubernetes Service)
Agrupa los microservicios por dominio, cada uno alineado a un Service Domain BIAN. AKS ofrece escalado automático, networking en VNet y gestión simplificada de contenedores. Ideal para cargas constantes, control de versiones y despliegues Canary o Blue Green tratando de lograr una mínima interrupción.

- **Microservicios por Dominio**:
    - Desplegados en **Azure Kubernetes Service** para escalar horizontalmente según carga.

    - Cada microservicio implementa un *Service Domain* BIAN (p. ej. “Payment Initiation”, “Risk Assessment”), garantizando que tu arquitectura siga la taxonomía estándar del sector.

- **Ventajas**:
    - Control total sobre contenedores, versiones y redes (VNet, peering).
    - Integración nativa con Azure Monitor, Container Insights y Politicass de seguridad.

Azure Functions (Adaptadores – ACL / Anti-Corruption Layer)
Implementa capas ligeras de adaptación para el Core Bancario Tradicional. Arrancan en milisegundos y se facturan por ejecución, perfectas para traducir protocolos legacy, aplicar lógica de enriquecimiento o enrutar llamadas sin desplegar contenedores completos.

- **Por qué Functions**:
    - Sirven de **módulo ligero** para traducir o enriquecer llamadas hacia el Core Bancario Tradicional (Strangler Fig Pattern).
    - Se facturan por ejecución y arranca en milisegundos, ideal para cargas intermitentes y picos.
- **Patrón ACL**:
    - Aísla la lógica legacy de la nueva capa de microservicios, reduciendo el riesgo de impactar al core antiguo.

Bases de Datos – Capa de persistencia

- **Azure SQL Database**: almacén relacional gestionado para datos transaccionales (cuentas, movimientos), con geo-réplica, alta disponibilidad y backups automáticos.
- **Azure Cosmos DB**: base NoSQL multimodelo para event store, caching distribuido y consultas de baja latencia en varias regiones.

Azure Key Vault
Centraliza la gestión de secretos, certificados TLS y llaves HSM. Se integra con Managed Identities para que ningún servicio guarde credenciales en texto plano. Ofrece rotación

automática de claves y auditoría detallada de accesos, cumpliendo requisitos de seguridad y cumplimiento normativo.

- **Beneficios**:
    - Integración con Managed Identities: ningún secreto en tu código o configuración.
    - Rotación automática y logging de accesos para auditoría.

Este diagrama y elección de servicios te permite:

- **Desacoplar** sistemas (legacy vs. digital) con patrones claros.
- **Escalar** automáticamente según demanda.
- **Monitorear** y asegurar tu plataforma con mínimos esfuerzos operacionales, apoyándote en las capacidades PaaS de Azure.

Flujo General de Peticiones y Eventos

1. **Entrada de la petición**
    - El **Cliente Web/Móvil** o **Partner Open Finance** envía la solicitud HTTP(s) o gRPC al endpoint público o privado de **Azure API Management** (APIM).
    - APIM aplica de inmediato políticas de:
        - **Autenticación** (OAuth2 con Azure AD o mTLS)
        - **Autorización** (scopes, claims)
        - **Rate-limiting** y **throttling**
        - **Validación de esquema** (OpenAPI)
2. **Enrutamiento inteligente**
   Según la naturaleza de la operación y la configuración de producto en APIM:

1. **Microservicio .NET en AKS**
        - Para operaciones CRUD de dominios BIAN y lógica core digital.

- APIM reescribe headers y propaga el **Correlation ID** para trazabilidad.

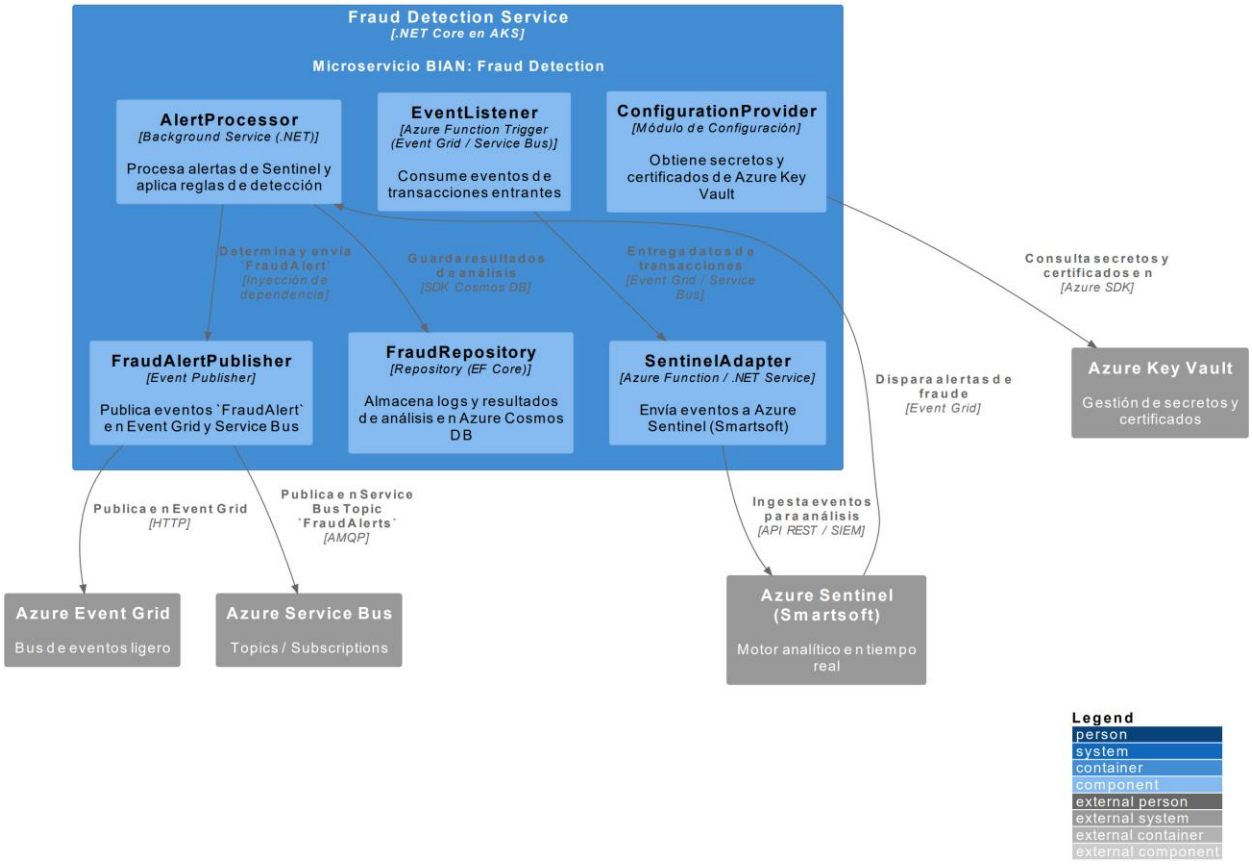
2. **Azure Function (ACL / Anti-Corruption)**
    - Para llamadas al core bancario legado o transformaciones puntuales.
    - Se escala instantáneamente ante picos y factura solo ejecuciones.
3. **Logic App / Durable Function**
    - Para orquestar workflows largos (p.ej. onboarding de cliente, procesos de aprobación).
    - Mantiene estado, retries configurables y checkpoints automáticos.

3. **Publicación de eventos y mensajes**
    - **Microservicios** y **orquestadores** emiten eventos de negocio a:
        - **Azure Event Grid** (eventos ligeros, fan-out inmediato)
        - **Azure Service Bus Topics** (mensajes con garantía de entrega, dead-lettering)
    - Cada mensaje incluye metadatos: tipo de evento (p.ej. PaymentInitiated), timestamp, Correlation ID y payload versionado.
4. **Reacción y persistencia**
    - **Consumidores** (otros microservicios, Functions, Logic Apps) suscritos reaccionan:
        - Validaciones adicionales, enriquecimientos de datos o llamadas a terceros.
        - Escritura de registros transaccionales en **Azure SQL Database**.
        - Almacenamiento de eventos o snapshots en **Azure Cosmos DB** (Change Feed para audit trail).
        - Publicación de eventos secundarios si aplica (p.ej. PaymentConfirmed, FraudAlertRaised).
5. **Seguridad y credenciales**
    - Todas las llamadas de contenedores (AKS, Functions, Logic Apps) usan **Managed Identities**:

        - Se autentican contra **Azure Key Vault** para obtener secretos, cadenas de conexión y certificados HSM.
        - No hay credenciales embebidas en código ni config files.

6. **Observabilidad y resiliencia**
    - **Tracing distribuido** con **Application Insights** (propagación de Correlation ID).
    - **Métricas** en Prometheus/Grafana o Azure Monitor (latencia, tasa de error, throughput).
    - **Logs estructurados** en Log Analytics Workspace con alertas proactivas.
    - **Circuit Breaker** y **Retry Policies** (implementados con Polly en .NET y políticas de Service Bus) para tolerancia a fallos.
7. **Feedback al cliente**
    - APIM devuelve la respuesta al cliente o, en procesos largos, un **callback URL** o **WebHook** con el resultado final.
    - Para flujos asincrónicos, el cliente puede consultar un endpoint de estado (/operations/{id}) hasta obtener el resultado definitivo.

C4 – Nivel 3: Diagrama de Componentes microservicio de “Fraud Detection Service”



(C4.3 Diagrama de Componentes - Deteccion de Fraude.pdf)

Este diagrama de componentes detalla cómo el Sistema de Prevención de Fraude, mapeado al Service Domain “Fraud Detection” de BIAN integra el motor analítico real-time Sentinel, reacciona ante alertas y publica eventos de fraude para alimentar los flujos de investigación, todo ello optimizado en un entorno de alta disponibilidad.

EventListener

- **Tipo**: Azure Function con trigger de **Event Grid** y/o **Service Bus**.
- **Responsabilidad**:
    - Suscribirse a los eventos de transacciones (“TransactionCreated”, “PaymentInitiated”) emitidos por otros microservicios.

    - Deserializar el payload, validar esquema (JSON Schema), descartar o rejugar mensajes corruptos.
    - Insertar metadatos (Correlation ID, timestamp, origen) antes de enviar al siguiente paso.
- **Resiliencia**:
    - Plan de consumo en Consumption Plan para autoescalar según volumen.
    - Política de retry configurable y "dead-letter queue" en Service Bus para evitar pérdida de mensajes.
- **Telemetría**:
    - Emite métricas hacia un Application Insights (tiempo de ejecución, aciertos/fallos de validación), la presentamos usando Graphana o el SIEM existente del banco.

SentinelAdapter

- **Tipo**: Azure Function (o servicio .NET en AKS) que actúa como *bridge* con Sentinel (Smartsoft).
- **Responsabilidad**:
    - Transformar el evento de transacción al formato esperado por Sentinel (Kusto JSON), incluyendo las entidades relevantes (cuenta, monto, canal).
    - Invocar la API REST de Sentinel con autenticación gestionada (Managed Identity + Azure AD) para insertar el registro en el SIEM en tiempo real.
- **Consideraciones de rendimiento**:
    - Agrupamiento por lotes pequeños (batch size tunable) para optimizar llamadas REST.
    - Circuit breaker con Polly para evitar sobrecargar Sentinel en momentos de alta demanda.

AlertProcessor

- **Tipo**: .NET Core *Background Service* corriendo en AKS (o Durable Function con Fan-Out).

- **Responsabilidad**:
    - Escuchar eventos SecurityAlert de Sentinel publicados en Event Grid.
    - Aplicar reglas adicionales (por ejemplo, scoring dinámico, validaciones cross-atlas) basadas en librerías de detección de fraude.
    - Correlacionar alertas (por ejemplo, múltiples transacciones sospechosas en corto periodo) para generar un solo FraudAlert consolidado.
- **Patrón Saga**:
    - Orquestación de sub-procesos de investigación (enviar tareas a Service Bus, actualizar estados en Cosmos DB).
- **Escalado**:
    - Kubernetes Horizontal Pod Autoscaler (HPA) basado en métricas de CPU y encolamiento de eventos pendientes.

FraudAlertPublisher

- **Tipo**: Componente .NET inyectado en el AlertProcessor.
- **Responsabilidad**:
    - Construir el mensaje FraudAlert con estructura versionada (v1, v2...) indicando severidad, origen, datos de cliente.
    - Publicar de forma idempotente en:
        - **Azure Event Grid** (para notificaciones ligeras y suscripciones fan-out).
        - **Azure Service Bus Topic "FraudAlerts"** (para flujos críticos con garantía de entrega).
- **Garantías**:
    - Clave de idempotencia en el header para que, ante reintentos, no se dupliquen las alertas.
    - Monitoreo de latencia en APIM Analytics cuando se usa Event Grid vía APIM.

FraudRepository

- **Tipo**: EF Core Repository apuntando a **Azure Cosmos DB**.
- **Responsabilidad**:
    - Persistir logs de eventos y resultados de análisis (timestamp, patrones detectados, score calculado).
    - Mantener una cola de mensajes activa para alimentar otras subscripciones (p.ej. auditoría, machine learning).
- **Modelo de datos**:
    - Partición por customerId o region para distribuir la carga.
    - Índices secundarios para búsquedas ad-hoc de incidentes y reporting.
- **Configuración**:
    - Throughput autoscalado en modo serverless o manual (RU/s según SLA).
    - Políticas de Time to Live (TTL) para registros antiguos y optimizar costos.

ConfigurationProvider

- **Tipo**: Biblioteca .NET compartida por todos los componentes.
- **Responsabilidad**:
    - Resolver configuraciones sensibles (cadenas de conexión, API keys, certificados) desde **Azure Key Vault** usando **Managed Identity**.
    - Cache local en memoria de secretos con expiración automática para reducir latencia y llamadas a Key Vault.
- **Seguridad**:
    - No se persiste ningún secreto en variables de entorno ni archivos de configuración.
    - Auditoría de accesos a Key Vault visible en Azure Monitor Logs.

2. Patrones de integración

A continuación, se describen los patrones principales empleados en el diseño, indicando dónde se aplican y por qué resultan críticos en nuestro enfoque Azure-First para la modernización bancaria.

2.1 API Gateway

- **¿Dónde?**
  En la capa de entrada de la plataforma, mediante **Azure API Management**.
- **¿Por qué?**
  - Centraliza el gobierno, la seguridad y el versionado de todas las APIs (internas y externas), sin tocar código.
  - Permite aplicar políticas declarativas (caching, rate-limit, validación de esquemas OpenAPI, transformación de payloads).
  - Facilita la exposición controlada de APIs BIAN a partners Open Finance y clientes web/móvil.

2.2 Strangler Fig Pattern

- **¿Dónde?**
  En la migración del Core Bancario Tradicional hacia el Core Digital, usando **Azure Functions** y adaptadores (Anti-Corruption Layer).
- **¿Por qué?**
  - Permite reemplazar funcionalidades del monolito de forma incremental, “estrangulando” las llamadas al sistema legacy sin interrumpir el servicio.
  - Cada nueva función se implementa como microservicio o Function independiente, reduciendo el riesgo de grandes “big-bang” y facilitando pruebas canary.

2.3 Event-Driven / Pub-Sub

- **¿Dónde?**
    - Eventos ligeros (fan-out inmediato) en **Azure Event Grid**.
    - Mensajes críticos (con garantías de entrega, retries y dead-letter) en **Azure Service Bus Topics/Subscriptions**.
- **¿Por qué?**
    - Desacopla emisores y consumidores para mejorar la **escalabilidad** y la **resiliencia**.
    - Separa flujos de negocio (p. ej. notificaciones de pago) de flujos críticos (p. ej. compensaciones de Saga), usando el medio más adecuado para cada caso.

2.4 Saga (Choreography)

- **¿Dónde?**
    - Orquestación de transacciones distribuidas mediante **Durable Functions** y/o **Azure Logic Apps**.
- **¿Por qué?**
    - Coordina múltiples microservicios (p. ej. debit y credit en pagos) con rollback automático en caso de fallo.
    - Mantiene el estado de la saga de forma durable y observable, sin necesidad de una base de datos central de orquestación (evento emitido, respuesta, confirmación).

2.5 Anti-Corruption Layer (ACL)

- **¿Dónde?**
    - **Azure Functions** componentes ligeros que exponen adaptadores hacia el core legacy (SOAP, base de datos monolítica).
- **¿Por qué?**
    - Filtra y transforma llamadas, evitando que la nueva capa de microservicios se “contamine” con modelos de datos antiguos.
    - Aísla los cambios en el sistema legacy, simplificando futuras migraciones o reemplazos.

2.6 Bulkhead, Circuit Breaker y Retry

- **¿Dónde?**
    - En cada microservicio .NET Core en AKS, implementado con **Polly**.
    - Políticas de retry y dead-letter en **Azure Service Bus**.
    - **Azure Application Gateway** con health-probes y WAF.
- **¿Por qué?**
    - Aísla fallos de componentes (bulkhead) y previene cascadas de error (circuit breaker).
    - Garantiza la entrega eventual y ordenada de mensajes, y los reintentos automáticos en rutas críticas.

2.7 Adapter Pattern

- **¿Dónde?**
    - Conectores a sistemas externos (por ejemplo, pasarelas de pago de terceros) implementados como **Azure Functions** o microservicios en AKS.
- **¿Por qué?**
    - Permite integrar rápidamente nuevos proveedores sin impactar la lógica de negocio interna.

- Facilita la estandarización de llamadas (REST, gRPC, mensajería) en torno a contratos OpenAPI o esquemas de eventos.

3. Tecnología propuesta

A continuación, se explica en detalle las elecciones tecnológicas de cada capa de la solución, con base a buenas prácticas para un entorno bancario de misión crítica.

3.1. Azure API Management

- Usar al menos un tier **Premium** que permite habilitar multi-región activa-activa, VNet injection (lado público y privado) y SLA del 99,95 %.
- **DevOps**: Definir la infraestructura par los APIs y productos en ARM/Bicep o Terraform; versionar en GitHub y desplegar con GitHub Actions o Azure DevOps pipelines.
- **Políticas Clave**:
  - **Validate-content**: asegurarse de cumplir esquemas OpenAPI en entrada.
  - **Rate-limit-by-key**: limitar llamadas por suscripción.
  - **Cache**: respuestas de consulta de datos estándar.
  - **JWT-validation**: validar tokens de Azure AD B2C/Enterprise.

3.2. Mensajería y Eventos

Azure Event Grid

- **Modelo**: Dominio privado para separar entornos (dev/test/prod); topic "BusinessEvents" con filtros por tipo de evento.
- **Entrega**: Mejor-esfuerzo, baja latencia (< 200 ms).
- **Integración**: Suscripciones push directas a Functions, Logic Apps o topics de Service Bus.

Azure Service Bus

- **Tier**: **Premium** para baja latencia (< 10 ms), aislamiento de red (VNet), throughput garantizado.

- **Patrones**:
    - Topics/Subscriptions con filtros SQL para enrutar mensajes por severidad o tipo.
    - Sesiones para mantener orden en flujos de compensación de Saga.
    - Dead-letter queues para controlar mensajes no procesables.
- **Escalado**: Escalar mediant el uso de Messaging Units; monitorear métricas de actividad en los requests.

3.3. Orquestación

Azure Logic Apps

- **Uso**: Flujos de trabajo automatizados (robots RPA), aprobaciones manuales y transformaciones de payload con conectores (SQL, Service Bus, HTTP).

Durable Functions

- **Patrones**: Function Chaining, Async HTTP APIs.
- Usar al menos un plan Premium para evitar cold-starts y habilitar VNet, escalado instantáneo y entorno always on.

3.4. Cómputo – AKS y Azure Functions

Azure Kubernetes Service (AKS)

- **Clúster**:
    - Pools de nodos Linux según necesidad.
    - Usar cluster autoscaler y pod autoscaler.

- **Red**:
    - Network Policies para aislamiento.
- **Seguridad**:
    - Enforce Pod Security Policies y Azure Policy para restringir el uso a imágenes aprobadas.
    - Integración con mediante Entra ID para acceder a Key Vault.
- **Ingreso de datos**: Azure Application Gateway con WAF y TLS mutuo end-to-end.

Azure Functions

- Usar siempre un plan Premium para evitar latencia por cold-start.
- **Durabilidad**: Durable Functions para orquestación con checkpoints automáticos.

3.5. Persistencia de datos

Azure SQL Database

- **Tipo**: Business Critical para latencia reducida en I/O, alta disponibilidad GP zonal.
- Habilitar al menos one-way geo-replica en otra región de respaldo.
- Agrupar bases de datos de microservicios para optimizar costos y escalado.

Azure Cosmos DB

- Para event store.
- Usar Multi-región para asergurar lecturas locales y fail-over automático.

3.6. Gestión segura de Certificados, Llaves, Configuración y Secretos

Azure Key Vault

- **Tipo**: **Premium (HSM)** para cifrado FIPS 140-2 y llaves asimétricas.
- **Acceso**: Managed Identities por microsservicio; políticas de acceso mínimas (least privilege).
- **Rotación**: Habilitar rotación automática de certificados y llaves con event grid notification.

Azure App Configuration

- Procurar el uso de Feature Flags centralizados y configuración no sensible; preferir replicación geo-redundante.

3.7. Infraestructura como Código & CI/CD

- **Herramientas**: Terraform (módulos compartidos) para despliegues de recursos Azure y evitar un cloud lock-in.
- **Pipelines**:
    - Lint y Test Terraform previo a ejecución de planes.
    - Deploy a entornos de desarrollo → staging → producción siempre con el uso de gates de aprobación manual.
    - Usar Pipelines integrados a Azure DevOps

3.8. Observabilidad y Gobernanza

Monitorización

- **Azure Monitor**: Recolecta métricas de todos los servicios PaaS.
- **Application Insights**: Trazas distribuidas con Correlation ID y Live Metrics.
- **Log Analytics**: Alerts basados en consultas Kusto (p. ej. error rate > 0.5 %).

Gobernanza

- **Azure Policy**: Denegar creación de recursos fuera de naming conventions o SKU no aprobados.
- **Blueprints**: Plantillas preconfiguradas de RG, permisos RBAC y políticas LOPD/PCI DSS.

4. Seguridad y cumplimiento normativo

Para garantizar la confidencialidad, integridad y disponibilidad de la plataforma, así como el cumplimiento de la **Ley Orgánica de Protección de Datos Personales (LOPDP)** de Ecuador y otros estándares del sector financiero, la solución propuesta incorpora los siguientes controles y procesos:

4.1 Protección de la Red

- **Virtual networks y Subredes**: Separación de capas Públicas (APIM, Front Door) y Privadas (AKS, Bases de datos) con Network Security Groups explicitos creando así un VPC (Virtual private cloud) exclusivo del proyecto.
- **Azure Firewall & WAF**: Filtrado de tráfico; protección de aplicaciones web con mediante la aplicación de reglas de OWASP.
- **Peering y Gateway**: Conexiones seguras hacia componentes legacy on-premise si existieren, mediante VPN Gateway o Azure ExpressRoute con circuitos privados, exigiendo el uso de UDR (UserDefined Routes).

4.2 Cifrado y Gestión de Claves

- **En tránsito**: TLS 1.2+ en todas las comunicaciones (APIM, Event Grid, Service Bus, SQL, Cosmos DB) preferir el uso de certificados de confianza mutua entre dos extremos, asegurar rotación anual de llaves y certificados.

- **En reposo**:
    - **Azure SQL**: Transparent Data Encryption (TDE) y Always Encrypted para columnas sensibles (p.ej. PID, números de cuenta).
    - **Cosmos DB** y **Storage**: SSE con claves gestionadas por el usuario en **Key Vault** (CMK).
- **Azure Key Vault (HSM Premium)**: Gestión de secretos, certificados y llaves de cifrado FIPS 140-2; rotación automática y logging de accesos para auditoría, tomando en cuenta que nos conectaremos a un sistema legacy existente se puede acoplar al uso de un HSM de 3ras personas.

4.3 Protección de Datos Personales (LOPDP)

- **Clasificación y etiquetado**: Uso de **Azure Purview** para descubrir y clasificar datos personales y sensibles.
- **Anonimización, Truncado y Pseudonimización**: Flujos de Azure Data Factory para enmascarar o truncar datos cuando no sean necesarios en producción o para un análisis.
- **Consentimiento y Derechos ARCO**: Diseño de APIs (/data/consent, /data/erase, /data/export) para gestionar solicitudes de acceso, rectificación, cancelación y oposición.
- **Políticas de Retención**: Configuración de TTL en Cosmos DB y lifecycle en Storage para borrar datos según plazos legales vigentes en la normativa ecuatoriana.
- **Auditoría**: Mediante el uso de Azure Sentinel.

4.4 Monitoreo y Auditoría

- **Azure Monitor & Application Insights**: Trazas distribuidas con Correlation ID; métricas de seguridad (intentos fallidos, latencia).
- **Azure Security Center & Defender for cloud**: Detección de amenazas, recomendaciones continuas y hardening de recursos.
- **SIEM**: Ingesta de logs y alertas en Azure Sentinel, con playbooks de respuesta automática (Logic Apps).
- **Registro de Cumplimiento**: Azure Policy y Blueprints aplican reglas LOPDP (GPDR), PCI DSS aplicables en Ecuador; informes periódicos de estado de cumplimiento.

5. Estrategia para cumplimiento de Alta Disponibilidad (HA) y Recuperación ante Desastres (DR)

Para poder definir una estrategia de alta disponibilidad y recuperación contra desastres partiremos de los siguientes objetivos supuestos:

- **RTO (Recovery Time Objective)** ≤ 5 min
- **RPO (Recovery Point Objective)** ≤ 1 min
- **Zonas de Réplica:** Azure East US (Primaria) y Azure West US (Secundaria)

Para un entorno bancario crítico, plantemos combinar alta disponibilidad con un plan de recuperación multi-región.

5.1 Infraestructura Multi-Zona (HA intra-región)

- **Contenedores AKS**
    - Pool de nodos distribuido en **3 Availability Zones** de la región primaria (East US), replicado hacia la región secundaria (West US).
    - Habilitar **Cluster Autoscaler** y **Horizontal Pod Autoscaler** para adaptarse a picos repentinos.
- **Azure SQL Database**
    - Tier Business Critical con replicas síncronas en cada AZ.
    - Read-only replicas para reportes y consultas analíticas (provisionado automáticamente como parte de “Business Critical”.
- **Cosmos DB**
    - Configuración multi-zona por defecto; lecturas y escrituras en cada zona.

- **Azure API Management (APIM) Premium**
    - Instancia activa-activa en ambas regiones.

    - **Azure Front Door** en frente de APIM con health probes para fail-over automático.
- **Azure Functions & Logic Apps**
    - Deploy en **Premium Plan** en ambas regiones para evitar cold-starts y habilitar VNet integration.

5.2 Replicación de Datos y Mensajería

- **Azure SQL Database**
    - **Auto-Failover Groups** configurados:
        - Réplica síncrona primaria → secundaria.
        - Failover manual (o automático con base a un health-probe).
- **Azure Cosmos DB**
    - **Escritura Activa-Activa** para todas las regiones, con fail-over automático al sitio alterno.
    - **Consistency tunable**: Session para equilibrio entre latencia y coherencia.
- **Azure Service Bus Premium**
    - Namespace geo-disaster-recovery alias par el DNS entre las regiones.
    - Failover invocado con runbook de Azure Automation (script PowerShell).
- **Azure Event Grid**
    - **Event Grid Domains** privados configurados con replicación entre regiones.

5.3 Gestión de Claves y Configuración

- **Azure Key Vault (Managed HSM Premium)**
    - **Geo-Recovery** con backup/restore automatizado.
    - **Soft-delete** y **Purge Protection** habilitados.
- **Azure App Configuration**
    - Replicación geo-redundante para Feature Flags y settings no sensibles.

5.4 Balanceo Global y Fail-over

- **Azure Front Door**
    - Punto único de entrada con
        - WAF integrado
        - SSL/TLS end-to-end
        - Health Probes a APIM en ambas regiones
        - Reglas de enrutamiento weighted / priority para DR

5.6 Backups y Orquestación de DR

- **Azure Backup**
    - Políticas diarias/horarias para SQL, VMs y Storage.
    - Retención según SLA bancario (> 1 año).
    - Como protección anti ransomware se puede usar Inmutable Backups con expiración definida, usando Inmutable Vault y Storage Account con Inmutable Storage.
- **Azure Site Recovery**
    - Para VMs on-premise o IaaS críticos (p.ej. legacy SOAP en VM) a partir del Vault.
- **Runbooks de Azure Automation**
    - Scripts que validan estado de réplicas, ejecutan failover de Service Bus, notifican al equipo de operaciones.

5.6 Pruebas y Validación

- **Chaos Studio**
    - Inyección controlada de fallos de red y caída de región primaria.
    - Verificación automática de fail-over y RTO/RPO.

- **Simulacros Periódicos**
    - Simulacros trimestrales documentados en runbooks; informes de lecciones aprendidas.

En conjunto, esta estrategia de alta disponibilidad y recuperación ante desastres trata de asegurar que, ante cualquier fallo, desde la caída de un datacenter hasta desastres regionales, la plataforma bancaria se mantenga operativa dentro de los objetivos de RTO y RPO fijados.

Gracias a despliegues multi-zona y multi-región, backups inmutables con retención extendida, fail-over automatizado mediante runbooks y un riguroso programa de pruebas y simulacros (Chaos Studio), nos enfocamos en garantizar la continuidad del negocio, la integridad de los datos y el cumplimiento de normativas como PCI DSS, minimizando el impacto para usuarios y operaciones.

6. Estrategia de Integración Multicore

La estrategia “multicore” que planteamos a continuación busca que la solución para el nuevo Core Digital pueda coexistir y sincronizarse de forma controlada con el Core Bancario Tradicional, logrando minimizar riesgos operativos y facilitando la migración incremental.

En el diseño planteado, esto se refleja en:

- ## Dual-Write desde la capa de entrada
    - **Dónde**: En **Azure API Management**, mediante una política de set-backend-service que duplica la llamada a dos rutas (Core Legado y Core Digital), o invocando una **Azure Function** que escribe simultáneamente en ambos.
    - **Por qué**: Garantiza que ninguna transacción se pierda durante la transición y permite comparar resultados en tiempo real.
- ## Dual-Read y Canary Releases
    - **Dónde**: APIM enruta un porcentaje configurable de lecturas (weight-based routing) al Core Digital y el resto al Legacy.
    - **Por qué**: Permite validar funcionalidad, rendimiento y consistencia antes de encender el 100 % del tráfico al nuevo core.
- ## Sincronización de datos
    - **Dónde**:
        - **Event-Driven**: Ambos cores publican eventos en Azure Event Grid o Service Bus, activando Azure Functions Trigger based que replican cambios (event sourcing) en el otro core.
        - **Azure Data Factory**: Para cargas iniciales y reconciliaciones periódicas (modo batch), usando Change Feed de Cosmos DB o tablas SQL.
    - **Por qué**: Mantiene la coherencia de datos entre sistemas y soporta trazas de auditoría histórica de transacciones.

- Fallback dinámico
    - **Dónde**:
        - En el microservicio de enrutamiento (Function o Logic App) se implementan políticas de Polly (circuit breaker + retry) que, ante fallo del Core Digital, redirigen automáticamente al Core Legado.
        - APIM puede usar backend health probes para conmutar en caliente.
    - **Por qué**: Asegura continuidad del servicio y evita que una degradación en el core nuevo impacte a los usuarios.
- Monitoreo y métricas comparativas por Core
    - **Dónde**:
        - **Application Insights** captura trazas y métricas con un tag core=legacy|digital.
        - **Dashboards en Azure Monitor** comparan latencia, tasa de error y throughput de cada core.
    - **Por qué**: Facilita la toma de decisiones realistas para avanzar o retroceder fases de migración con datos objetivos.

Con esta combinación de dual-write, dual-read, sincronización continua, fallback automatizado y observabilidad granular, la plataforma garantiza una migración segura y controlada entre ambos Cores, reduciendo al mínimo el riesgo operacional y preparando el camino para desactivar progresivamente el Core Legacy una vez validado el Core Digital, al definir la construcción de APIs por dominio, podemos ir dando de baja cada dominio independientemente y no todo el core legacy.

7. Gestión de Identidad y Acceso a los sistemas

Para garantizar que únicamente usuarios, aplicaciones y servicios autorizados interactúen con nuestros recursos de forma segura, implementamos un modelo centralizado de gestión de identidad y acceso basado en Microsoft Entra ID.

Este enfoque unifica la autenticación multifactor, la autorización granular por API y servicio, y la gestión de credenciales sin secretos embebidos, al tiempo que incorpora auditoría continua y protección adaptada a los exigentes requisitos del sector financiero.

7.1 Autenticación de Usuarios

- **Entra ID B2C** para clientes externos y **Entra ID Enterprise** para empleados.
- Protocolos **OAuth2 / OpenID Connect** con tokens JWT de corta expiración (1 hora).
- **MFA** obligatorio para accesos de alto riesgo y para el personal de operaciones.
- **Conditional Access**: políticas basadas en ubicación, dispositivo y riesgo de sesión.
-

7.2 Autenticación y Autorización de APIs

- **Azure API Management** valida tokens JWT emitidos por Entra ID:
    - Scopes (scp) y roles (roles) en el token para controlar acceso por endpoint.
    - Políticas de transformación y rate-limiting sin tocar código.
- **mTLS** obligatorio para partners externos, con certificados de confianza mutua gestionados en el Key Vault.
-

7.3 Servicio a Servicio (App-to-App)

- Garantizar el aceeso seguro vía **Managed Identities** otorgadas a AKS, Functions y Logic Apps:
    - Permiten obtener tokens OAuth2 sin almacenar credenciales.
    - Usadas para acceder al APIM, Service Bus, Event Grid y Key Vault.

- **Service Principals** para pipelines de CI/CD y herramientas externas, con secretos rotados automáticamente.

7.4 Autorización Fina (RBAC & ABAC)

- **Azure RBAC**: roles integrados y customizados en el nivel de Subscription/Resource Group.
- **Attribute-Based Access Control** en el código: validación de claims, roles en middleware ASP NET Core (usando [Authorize]).

7.5 Control de Acceso a Datos Sensibles

- **Conexiones vía Entra ID** para Azure SQL y Cosmos DB, sin strings en texto.
- **Always Encrypted** en columnas PII (Información Personalmente Identificable) dentro de Azure SQL.
- Acceso a Key Vault restringido a identidades mínimas; logging de cada operación.

7.6 Seguridad de Red y Frontera

- Private Link & Service Endpoints para APIM, SQL, Cosmos y Key Vault dentro de la VNet.
- Network Security Groups con reglas estrictas solo para puertos y protocolos necesarios.
- Azure Firewall y WAF en Azure Front Door / Application Gateway para protección en todas las direcciones.

7.7 Gestión y Monitorización de IAM

- Privileged Identity Management (PIM): elevación just-in-time con aprobación y auditoría.
- Entra ID Identity Protection para detección de riesgos en sign-in.

- Ingesta de logs de autenticación y autorización en Log Analytics para alertas y reportes de cumplimiento.
- Monitoreo y alertas via Azure Sentinel

Con esta estrategia, cada petición, ya sea de un usuario final, un partner o un servicio interno, es autenticada y autorizada de manera segura y consistente, basada en identidades centralizadas y políticas bajo la premisa de mínimo privilegio, garantizando la seguridad y la trazabilidad requeridas por el sector bancario.

8. Mensajería ente Apis internas y externas

Para garantizar interoperabilidad, escalabilidad y gobernanza, definimos una estrategia dual de APIs internas y externas, apoyada en estándares abiertos (OpenAPI, AsyncAPI, ISO 20022) y patrones de mensajería comúnmente usados (REST/gRPC, Pub-Sub, colas).

8.1 APIs Internas

- **Especificaciones y Contratos**
    - Definición de cada API mediante OpenAPI 3.0 (YAML/JSON), versionado semántico (v1, v2…) y publicación automática en APIM interno.
    - Para llamadas de baja latencia o streaming, uso de gRPC.
- **Seguridad y Autorización**
    - Autenticación con tokens JWT emitidos por Entra ID (Managed Identities).
    - Scopes granulares por microservicio y claims específicos para separar entornos (dev/test/prod).
- **Mensajería Síncrona vs Asíncrona**
    - Operaciones CRUD y consultas: REST/gRPC síncrono.
    - Flujos de negocio desacoplados: comandos y eventos asíncronos en Azure Service Bus (AMQP) y Event Grid (HTTP push).

-

- **Governanza y Observabilidad**
    - Documentación y mocking automáticos en el Developer Portal Privado de APIM.
    - Telemetría distribuida (Correlation ID) en Application Insights, con métricas de latencia y carga o saturación de endpoints.

8.2 APIs Externas

- **Gateway de entrada unificado**
    - **Azure API Management** expone solo las operaciones aprobadas a partners / fintechs.
    - Developer Portal Público con “try-it”, suscripciones con clave y cuotas por socio.
- **Protocolos y Estándares**
    - REST/JSON con contratos OpenAPI, complementado con WebHooks para notificaciones push.
    - Para banca abierta, adopción de perfiles de Open Banking sobre JSON o el estándar usado ISO 20022.
- **Seguridad de borde/frontera**
    - Políticas mTLS opcional, OAuth2 Client Credentials y validación de scopes.
    - Rate-limiting, cuotas de consumo y throttling configurables por API.

8.3 Estándares de Mensajería y Esquemas

- **AsyncAPI**: definición de tópicos, canales y payloads en AsyncAPI 2.0, versionados en el repositorio de schemas.
- **Schema Registry**: uso de Azure Schema Registry para gestionar versiones de JSON.
- **Protocolos**:
    - AMQP 1.0 para Service Bus (colas, topics).
    - HTTP push y WebHooks sobre Event Grid.
    - MQTT o Kafka (Event Hubs) para telemetría de IoT/terminales de pago.

- **Buenas Prácticas**:
    - Inclusión de metadatos en headers (correlation-id, message-version, message-type).
    - Diseño idempotente: generación de idempotency-key en cada mensaje, esto busca hacer predecible cada operación critica del negocio.

9. Modelo de Gobierno de APIs y Microservicios

El objetivo de este modelo de gobierno es asegurar calidad, seguridad y alineación con el negocio a lo largo del ciclo de vida de cada API/microservicio.

Roles Clave

- **Product Owner**: define alcance, SLAs y prioridades.
- **Architect**: establece estándares (OpenAPI, BIAN, seguridad).
- **DevOps**: automatiza despliegues y entornos (Terraform/Pipelines).
- **Seguridad/Compliance**: valida políticas (OAuth2, mTLS, LOPDP, PCI DSS).
- **Operaciones**: monitorea salud y gestiona incidentes.

Ciclo de Vida

- **Planificación**: Definir contrato (OpenAPI/AsyncAPI) y SLAs.
- **Diseño y Desarrollo**: generar stubs, tests de contrato en CI.
- **Revisión**: code review + validación de seguridad y datos sensibles.
- **Publicación**: desplegar en APIM, configurar políticas y registrar en catálogo.
- **Operación**: telemetría, alertas y feedback para mejoras.
- **Versionado / Dado de baja (deprecación)**: coexistencia mínima de versiones, notificar retiro.

Políticas Clave

- **Seguridad**: validación de JWT, scopes, mTLS opcional.
- **Calidad**: latencia < 50 ms, error rate < 0.5%.

- **Uso**: cuotas y rate limits por suscripción.
- **Datos**: encriptación, enmascaramiento de PII, cumplimiento LOPDP/GDPR.

Herramientas de Soporte

- **APIM** (Developer Portal privado/público) para onboarding y documentación.
- **API Catalog/Registry** para descubrimiento.
- **CI/CD** (Azure DevOps/GitHub Actions) con lint/tests.
- **Azure Policy & Blueprints** para imponer configuraciones seguras.
- **Azure Monitor & Dashboards, Graphana, Azure Sentinel** para visibilidad de uso, SLAs y manejo de costos.

10. Plan de Migración Gradual que Minimice el Riesgo Operativo

Para mover progresivamente las cargas del Core Bancario Tradicional al Core Digital y al mismo tiempo garantizar continuidad de servicio, proponemos el siguiente roadmap dividido en fases con criterios para evaluar el éxito de cada una y mecanismos de rollback.

Fase	Objetivo	Acciones Clave	Validación y Rollback
0. Descubrimiento & Preparación	Inventario de APIs, flujos y dependencias Mapeo a dominios BIAN	• Catalogar APIs y transacciones críticas • Definir SLAs, RTO/RPO y KPIs de migración	OK si inventario >95 % y SLAs acordados; plan alterno si faltan datos
1. Strangler Fig (Adaptadores)	Introducir capa ACL para llamadas al core legacy	• Desplegar Azure Functions que expongan el monolito como REST • Enrutar tráfico de prueba vía APIM	OK si latencia <100 ms y 0 errores; rollback desactivando la ruta en APIM
2. Dual-Write	Grabar simultáneamente en ambos cores	• Política APIM o Function que duplique escrituras • Añadir log de idempotencia (Redis/Cosmos)	Validar consistencia de datos (>99 %); en caso contrario, deshabilitar dual-write
3. Canary Releases (Dual-Read)	Desviar % de lecturas al Core Digital	• APIM Weighted Routing: 5 % → 25 % → 50 % → 100 % • Monitorizar latencia, error rate por core	Revertir porcentaje si error rate >0.5 % o SLAs incumplidos
4. Sincronización Continua	Mantener ambas BDs alineadas	• Event Sourcing: Change Feed Cosmos + Functions de replicación	Checkpoints diarios de divergencia <0.1 %; fallback a batch completo
5. Deprecación del Legacy	Retirar el monolito y adaptadores	• Cortar rutas antiguas en APIM • Desmontar Functions ACL cuando el 100 % del tráfico esté en Digital	Revertir último cambio en APIM si se detectan errores críticos
6. Optimización y Cierre	Afinar rendimiento y limpieza final	• Eliminar código y tablas obsoletas • Ajustar escalado automático y costos	Completar limpieza cuando no existan dependencias; cerrar proyecto legacy

(10. Plan migración.xlsx)

Puntos Clave para Mitigar Riesgos

- **Feature Flags** en Azure App Configuration para activar/desactivar fases sin redeploy.
- **Circuit Breakers** (Polly) y **Fallbacks Dinámicos** en APIM para conmutar automáticamente al legacy si el core digital falla.
- **Monitoreo Granular**: Dashboards por core en Application Insights y alertas proactivas.
- **Gates de Aprobación**: Solo avanzar de fase cuando se cumplan los KPIs definidos.

Este plan modular, con fases bien definidas, criterios de aceptación claros y mecanismos de rollback automatizados, garantiza una migración controlada, con visibilidad constante y mínima exposición a fallos operativos.

Referencias:

1. **Microsoft Azure Documentation**
   https://learn.microsoft.com/azure/
2. **Polly (.NET Resilience Library)**
   https://github.com/App-vNext/Polly
3. **Martin Fowler – Strangler Fig Pattern**
   https://martinfowler.com/bliki/StranglerFigApplication.html
4. **BIAN – Banking Industry Architecture Network**
   https://bian.org/
5. **OpenAPI Specification**
   https://spec.openapis.org/oas/latest.html
6. **AsyncAPI Specification**
   https://www.asyncapi.com/
7. **ISO 20022 Standard**
   https://www.iso20022.org/
8. **PCI DSS v4.0**
   https://www.pcisecuritystandards.org/document_library
9. **Ley Orgánica de Protección de Datos Personales (Ecuador)**
   https://www.gob.ec/regulaciones/ley-organica-proteccion-datos-personales-0
10. **GDPR (Reglamento Europeo de Protección de Datos)**
    https://gdpr-info.eu/

Estas fuentes respaldan todo el diseño de la solución: servicios Azure, patrones de integración, seguridad, gobernanza y cumplimiento normativo.