Arquitectura de Integración

Propuesta de solución de Integración Banking

Ronald Rojas Castillo

Tabla de contenido

1. Descripción del Requerimiento

Requerimiento

A continuación, se describe el requerimiento solicitado:

- Cree un documento PDF con la respuesta al ejercicio.
- El documento debe estar bien organizado y ser fácil de leer.
- Cada uno de los diagramas de C4 es un entregable importante. Asegúrese de hacerlos correctos y detallados. Asegúrese de añadir cualquier texto explicativo que considere necesario.
- Suba el documento como respuesta a este ejercicio. Asegúrese de subirlo dentro del tiempo de resolución.
- Adicional, cree un repositorio público en GitHub y subir el PDF a ese repositorio. Colocar la URL del repositorio en los comentarios de este ejercicio.

Descripción de la solución:

Diseñar una arquitectura de integración para la modernización de sistemas bancarios.

Escenario:

Un banco tradicional busca modernizar su infraestructura tecnológica para mejorar sus servicios digitales y cumplir con nuevas tendencias de la industria y regulaciones. Se requiere integrar:

1. Core Bancario tradicional existente, más un nuevo Core bancario digital.
2. Nuevo sistema de banca web y banca móvil
3. Plataforma de servicios de pago.
4. APIS para servicios de terceros (Open Finance)
5. Sistema de gestión de riesgos.
6. Sistema de prevención de fraudes.

Tareas:

1. Diseñar una arquitectura de integración de alto nivel.
2. Especificar patrones de integración y tecnologías a usar.
3. Abordar requisitos de seguridad, cumplimiento normativo, ley orgánica de protección de datos personales.
4. Proponer estrategia para garantizar alta disponibilidad y recuperación ante desastres.
5. Proponer estrategia de integración Multi Core.
6. Delinear un enfoque para la gestión de identidad y acceso en todos los sistemas.
7. Diseñar una estrategia de API internas y externas, bajo estándares de la industria respecto a la mensajería.
8. Proponer un modelo de gobierno de Apis y Microservicios.
9. Proponer un plan de migración gradual de minimice el riesgo operativo.

Consideraciones:

- Garantice la alta disponibilidad (HA) y tolerancia a fallos (DR), seguridad y monitorio.
- Si lo considera necesario, su diseño de integración puede contener elementos de infraestructura en la nube como Azure o AWS; API Manager. Debe garantizar la baja latencia.
- El diseño debe estar bajo C4 (Diagrama de contexto, contenedores, componentes)
- Contemplar en el diseño el uso de eventos, siguiendo los lineamientos de la industria.

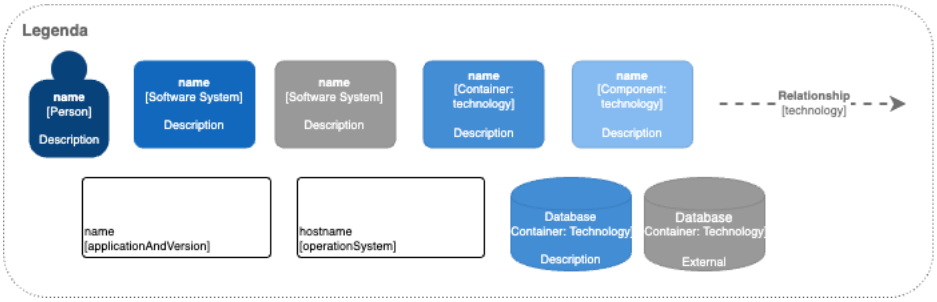
Entregables:

- Diagrama C4
- El aspirante podrá entregar los insumos necesarios para justificar la propuesta.

2. Propuesta de solución

2.1. Diseño de la arquitectura de Integración de alto nivel.

Los siguientes son los componentes y leyendas que se utilizarán en los diagramas del modelo C4:



Con base en los requerimientos solicitados a continuación se describen los diagramas de diseño Modelo C4.

Diagrama C4 – Contexto

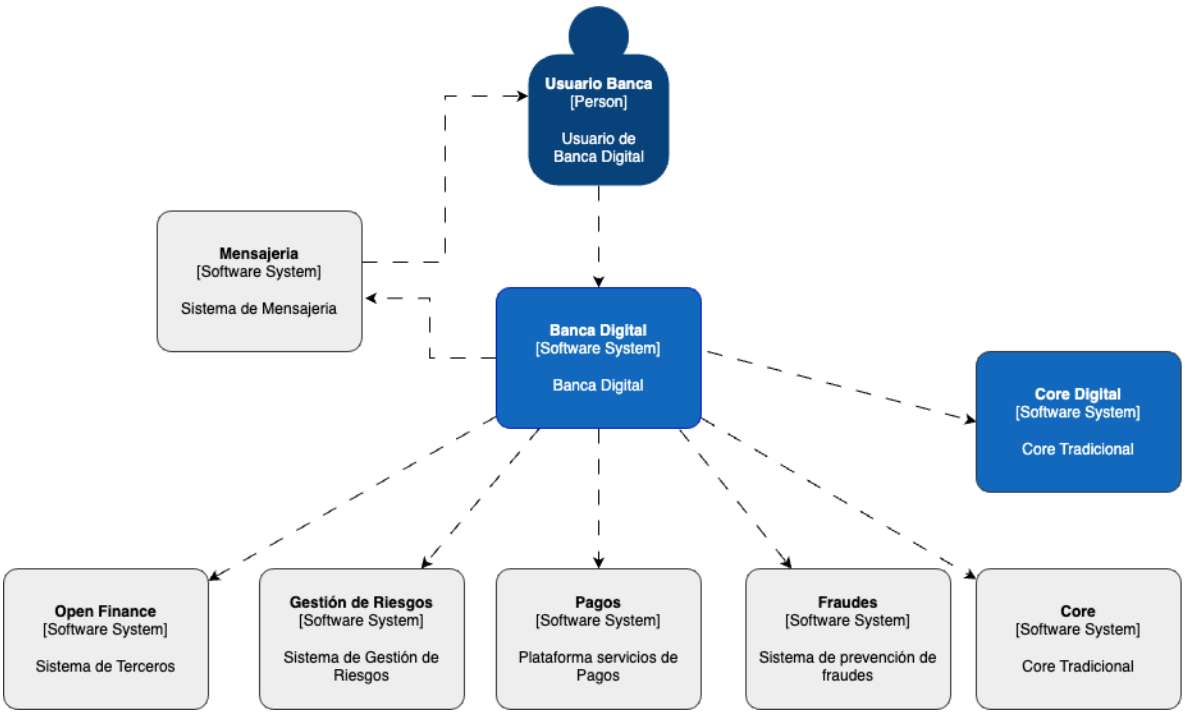
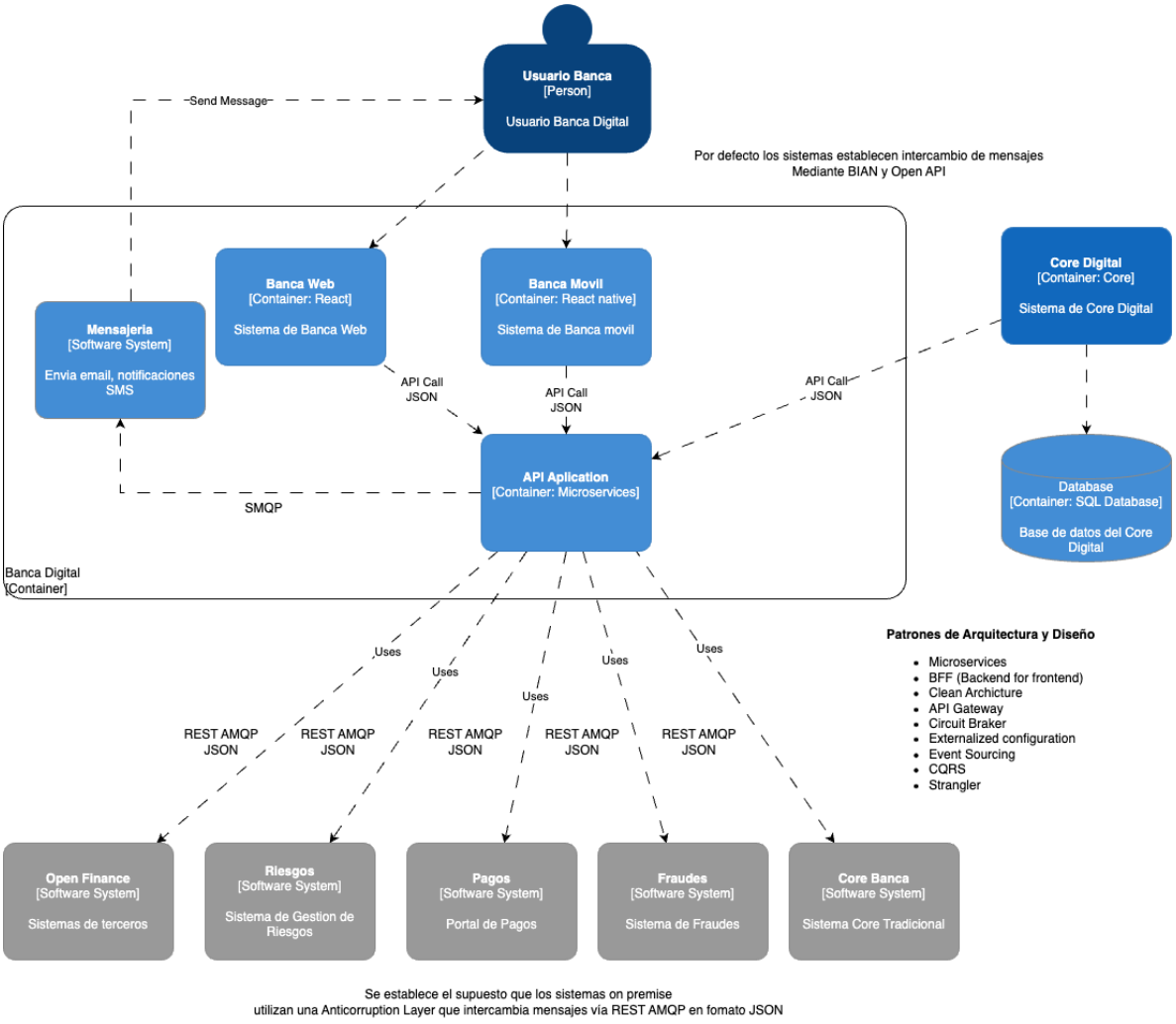
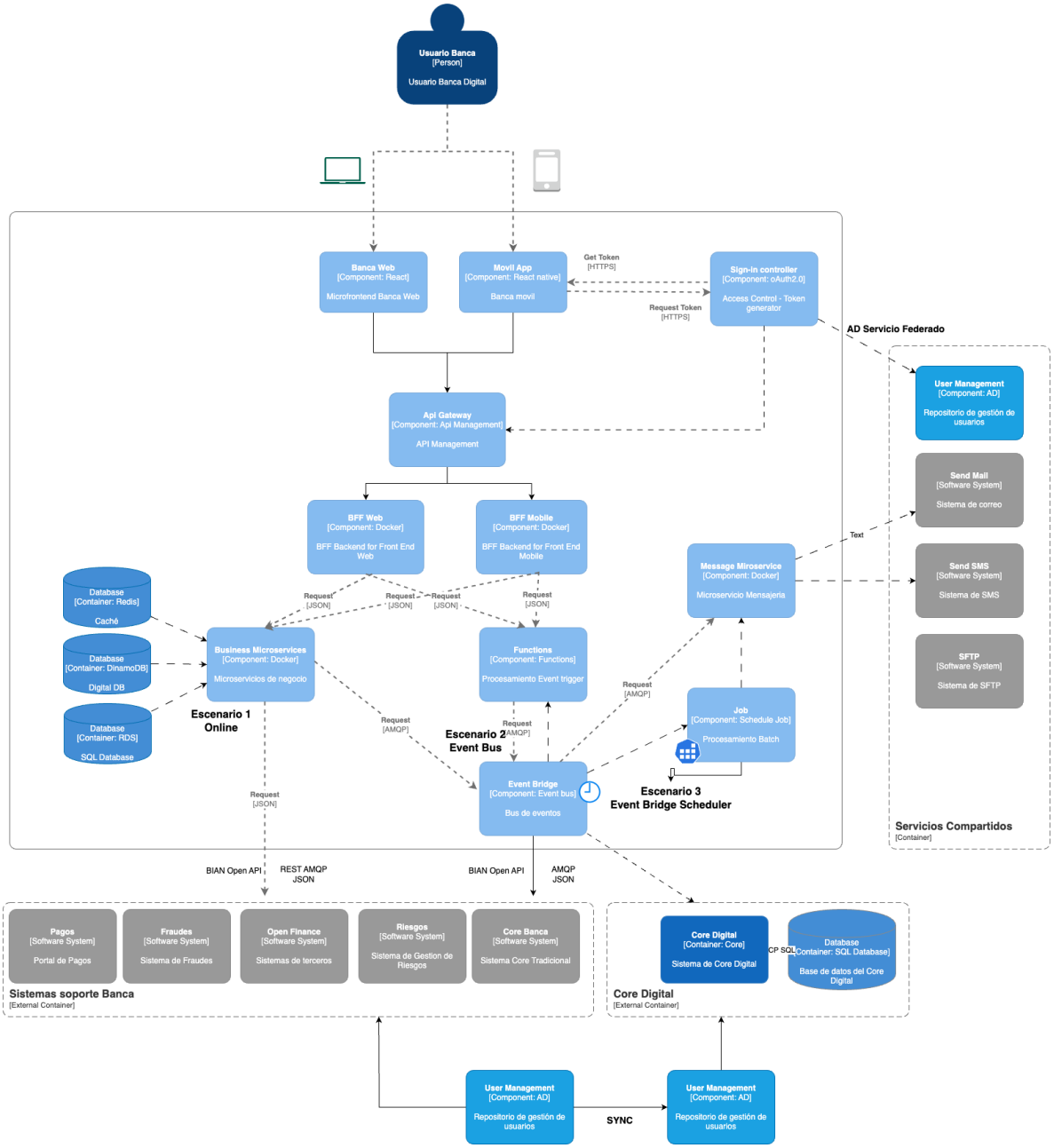


Diagrama C4 – Contenedor



Se establece el supuesto que los sistemas on premise
utilizan una Anticorruption Layer que intercambia mensajes vía REST AMQP en fomato JSON

Diagrama C4 – Componente



2.2. Patrones de integración y tecnologías

Los siguientes son los principales patrones que se implementan en la arquitectura de Integración.

Patron BFF (Backend for frontend)
Se utiliza el patrón BFF como fachada sobre el resto de las APIs de negocio, adaptando los datos que el cliente realmente necesita, simplificando la carga y comportamiento.

Clean Archicture
Patrón de arquitectura que mediante la separación de capas permite un código más limpio, mejor estructurado y fácil de mantener.

API Gateway
Este patrón permite, mediante un componente, centralizar las peticiones a las APIS, agregar mecanismos de retención ante altas inesperadas (limitar API Calls), incorporar mecanismos de caching y seguridad.

Circuit Braker
Patrón que resuelve los problemas a llamadas remotas u otras Apis que no reciben respuesta. Su foco es evitar la propagación de errores y permitir que el circuito de ejecución termine.

Externalized configuration
Este patrón permite que se externalicen que todas las configuraciones de la aplicación, incluyendo credenciales de base de datos y ubicaciones de red.

Event Sourcing
Este Patrón proporciona una nueva secuencia ordenada de eventos, la cual se puede utilizar para reconstruir el estado de la aplicación.

CQRS
Permite separar las operaciones de lectura y actualización de un almacén de datos. La implementación de CQRS en la aplicación puede maximizar el rendimiento, la escalabilidad y la seguridad.

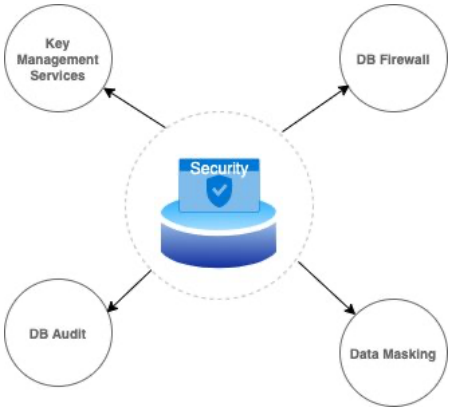
Strangler
El patrón de diseño Strangler permite transformar gradualmente una aplicación monolítica en microservicios, sustituyendo la funcionalidad antigua con un nuevo microservicio.

La arquitectura de integración propuesta está basada en componentes de infraestructura en la nube de AWS.

Las principales tecnologías que conforman la solución son:

Dominio	Componente	Tecnología
Infraestructura	Plataforma Cloud	**Amazon Web Services**
Infraestructura	Contenedor	Docker
Infraestructura	Orquestador Docker	EKS
Infraestructura	Registry	ECR
Infraestructura	Almacenamiento	S3
Infraestructura	Redes	VPC
Infraestructura	Redes	Transit Gateway
Infraestructura	Redes	Customer Gateway
Infraestructura	Redes	VPN Site to Site
Infraestructura	Observabilidad	CloudWatch
Infraestructura	Auditoria EKS	Via API Kubernetes
Infraestructura	Métricas	Prometheus
Software	Base	Nodejs
Software	Front Web	React
Software	Mobile(IOS,Android, Huawei)	React Native
Integración	API Management	API Gateway
Integración	API development Framework	Java Spring boot
Integración	FaaS	Lambda Functions
Integración	Event Bus	Event Bridge
Integración	Job scheduler	Event Bridge Scheduler
Datos	NOSQL Database	DinamoDB
Datos	SQL	RDS Mysql
Datos	Cache	Redis
Seguridad	Almacén de usuarios	LDAP
Seguridad	Control de acceso y protección	Cognito
Seguridad	Key Vault	KMS
Seguridad	Auditoria	RDS audit
Seguridad	Normativa	Normativa Ley 19628

2.3. Requisitos de seguridad, cumplimiento normativo, ley orgánica de protección de datos personales.



Se recomienda implementar una plataforma de seguridad de datos basada en la estrategia de defensa en profundidad con los siguientes componentes:

- AWS Key Management Services, permite el establecimiento de la encriptación de la DB en descanso.
- Se recomienda implementar Key Vault para las BDs con datos sensibles.
- Para datos en tránsito se deben utilizar certificados y TLS.
- Habilitar la configuración de seguridad DB Firewall que protege contra ataques de SQL injection, y otras formas de sentencias maliciosas, Genera alerta de amenazas en tiempo real. Se recomienda definir las políticas de reglas de firewall de SQL.
- Habilitar la configuración de DB Audit, permite el tracking de los eventos de seguridad para el seguimiento de actividades sospechosas.
- Habilitar la configuración de Data Masking, permite el ofuscamiento de los datos sensibles entre ambientes de producción y no producción.
- La data sensible deberá ser previamente identificada, y también las reglas de ofuscamiento alineada a la política de la Ley 19628.

2.4. Estrategia de garantía de alta disponibilidad y recuperación ante desastres, Seguridad y monitoreo.

Se disponibiliza la arquitectura de integración con infraestructura en la nube de AWS en una modalidad multi zona para poder asegurar alta disponibilidad y durabilidad del servicio.

En el ámbito de base de datos se utiliza el servicio administrado AWS RDS en modalidad clúster multi zona.

En la capa de microservicios se utiliza EKS con subredes en multi zona.
El servicio de kubernetes permite habilitar la creación de una nueva instancia ante desastres o perdida de un contenedor en forma automática.

Los componentes de integración de AWS Event Bridge, Lambda Function, S3 son servicios administrados por AWS y proveen alta disponibilidad,

Respecto al monitoreo, se utiliza como herramienta transversal CloudWatch.
Como herramienta de supervisión de métricas del cluster EKS se utiliza Prometheus.

Para RD se propone utilizar AWS Elastic Disaster Recovery (AWS DRS) con el fin de minimizar los tiempos de inactividad y tener fiabilidad y mayor completitud ante un evento de recuperación de datos e infraestructura.

Desde el punto de vista de seguridad se propone utilizar AWS Firewall perimetral para poder proteger ataques y accesos desde internet. En el ámbito de comunicaciones se deben establecer conexiones vía TLS con sus correspondientes certificados.
En el ámbito de seguridad en base de datos y datos se encuentra referido en el punto 2.3.

2.5. Estrategia de integración Multi Core.

Se establece como premisa inicial que el Core Digital será el CORE a establecer como único y definitivo.

La sincronización entre COREs se realiza mediante microservicios de ida y vuelta, vía evento bus. No se recomienda utilizar sincronización mediante base de datos o change data capture, principalmente por violación a integridad referencia y flujos de negocio. La sincronización se debe realizar mediante las APIs que provee el CORE digital.

2.6. Gestión de identidad y acceso en todos los sistemas.

Identidad Entre CORES

Se plantea un proceso de gestión de identidades basado en 2 supuestos iniciales.

1. El sistema actual de identidad es Microsoft AD. Los sistemas de banca utilizan el sistema de gestión de identidades Microsoft AD.
2. Si existiese algún sistema con plataforma de administración propietaria ADHOC (Por ej: en base de datos), debe ser actualizado a AD.

Dado que aún no se ha definido el universo total de usuarios a migrar al nuevo Core, y no se encuentra definido en el scope del requerimiento funcional final, se recomienda realizar una carga inicial de usuarios (basado en Core Digital) y a medida que se habilitan funcionalidades y nuevos perfiles se proceda a sincronizar en forma evolutiva el nuevo AD Digital desde el AD Tradicional.

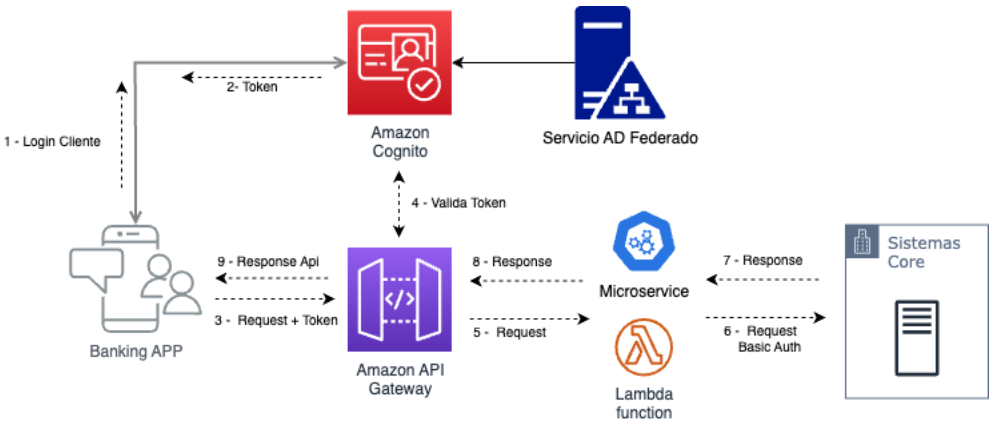
Bajo este supuesto se debe definir una estructura de directorios para el nuevo CORE Digital que sea capaz de poder convivir en forma compatible con el CORE Tradicional.

Una vez realizada esta homologación tecnológica y definición de estructuras, se debe activar el proceso de sincronización entre1 ambos AD (Tradicional y Digital).

Identidad en Banca Digital y Banca Movil

Para el control de acceso y permiso a las plataformas digitales se propone la utilización de AWS Cognito cuyas principales características son:

- Gestión de Grupos de usuarios.
- Autorización de usuarios mediante Servicios de AD Federado.
- Control de Accesos desde las aplicaciones.
- Gestión de Token oAuth2.0.
- Control de acceso a los Endpoint (Lambda, Microservices) de API Gateway.



2.7. Estrategia de API internas y externas.

Para poder diferenciar y aislar las APIs internas de las externas se han definido patrones y consideraciones base.

Por definición de arquitectura se ha contemplado el uso de **AWS API Gateway** como mecanismo único centralizade de consumo y exposición de las APIs.

Se utiliza el patrón **BFF** para definir frontera entre las APIs externas e internas.

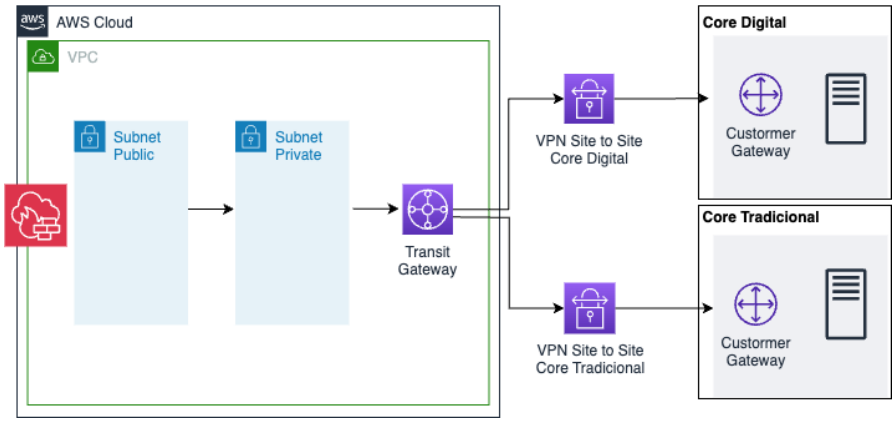
Las APIs internas no están expuestas hacia los canales digitales y su función principal es servir componentes de negocio, consumo de otras APIs internas, APIs Operativas (Logging, monitoreo, mensajería, auditoría). Las APIs internas son las que consumen componentes Backend (COREs y servicios de apoyo).

Se han de definido 3 escenarios de integración en el diseño propuesto:

1) Escenario On line. APIs que requieren una respuesta de negocio online, se consideran **Microservicios** de negocio y **Lambda Function** con http trigger.
2) Escenario basado en eventos. No se requiere una respuesta inmediata, es posible notificar del inicio de la acción posteriormente se gatillan los eventos necesarios mediante **AWS Event Bridge** y un modelo pub/sub.
3) Escenario Batch. Se disponibiliza **AWS Event Bridge Job Scheduler** para los casos que se requiera procesamiento por lotes, agendamiento de trabajos y notificaciones masivas.

Las APIs externas e internas forman parte de una VPC, donde las APIs internas están desplegadas en Subredes privadas. De esta forma se asegura su aislamiento hacia el exterior.

Si se requiere consumir servicios COREs desde otras plataformas como por Ej: Core On premise se debe realizar desde una subred privada mediante VPNs Site to Site como se muestra en la figura.

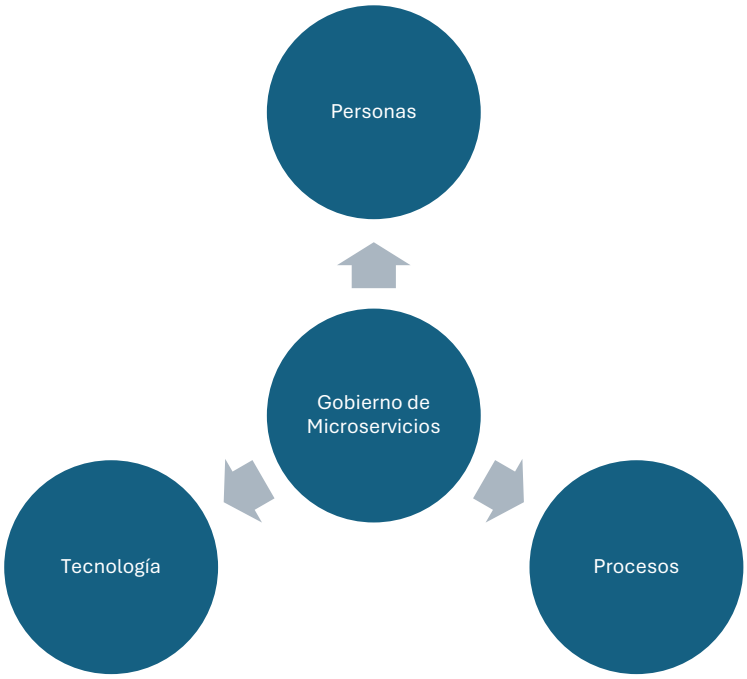


2.8. Modelo de gobierno de Apis y Microservicios.

Como herramienta de gestión del ciclo de vida de una API se propone utilizar API Gateway, API Gateway disponibiliza un catálogo de APIs accesible desde los equipos de desarrollo.

El ciclo de vida de APIs de alto nivel es el siguiente:

El gobierno de microservicios se basa principalmente en 3 pilares:



Desde el punto de vista tecnológico, para un gobierno efectivo, se debe tener presente lo siguiente:

- Herramienta de registro de imágenes.
- Arquitectura de referencia.
- Activos y herramientas de gestión.
- Gestión de la infraestructura de Microservicios.
- Gestión de APIs y su ciclo de vida.
- Captura de métricas.
- Monitorización.

2.9. Plan de migración gradual.

Se propone un plan de migración que considere como primera línea la migración de los datos maestros, para luego gradualmente migrar los módulos funcionales de acuerdo con el siguiente plan macro:

Se propone la ejecución iterativa gradual, dependiendo del Road Map funcional a migrar.