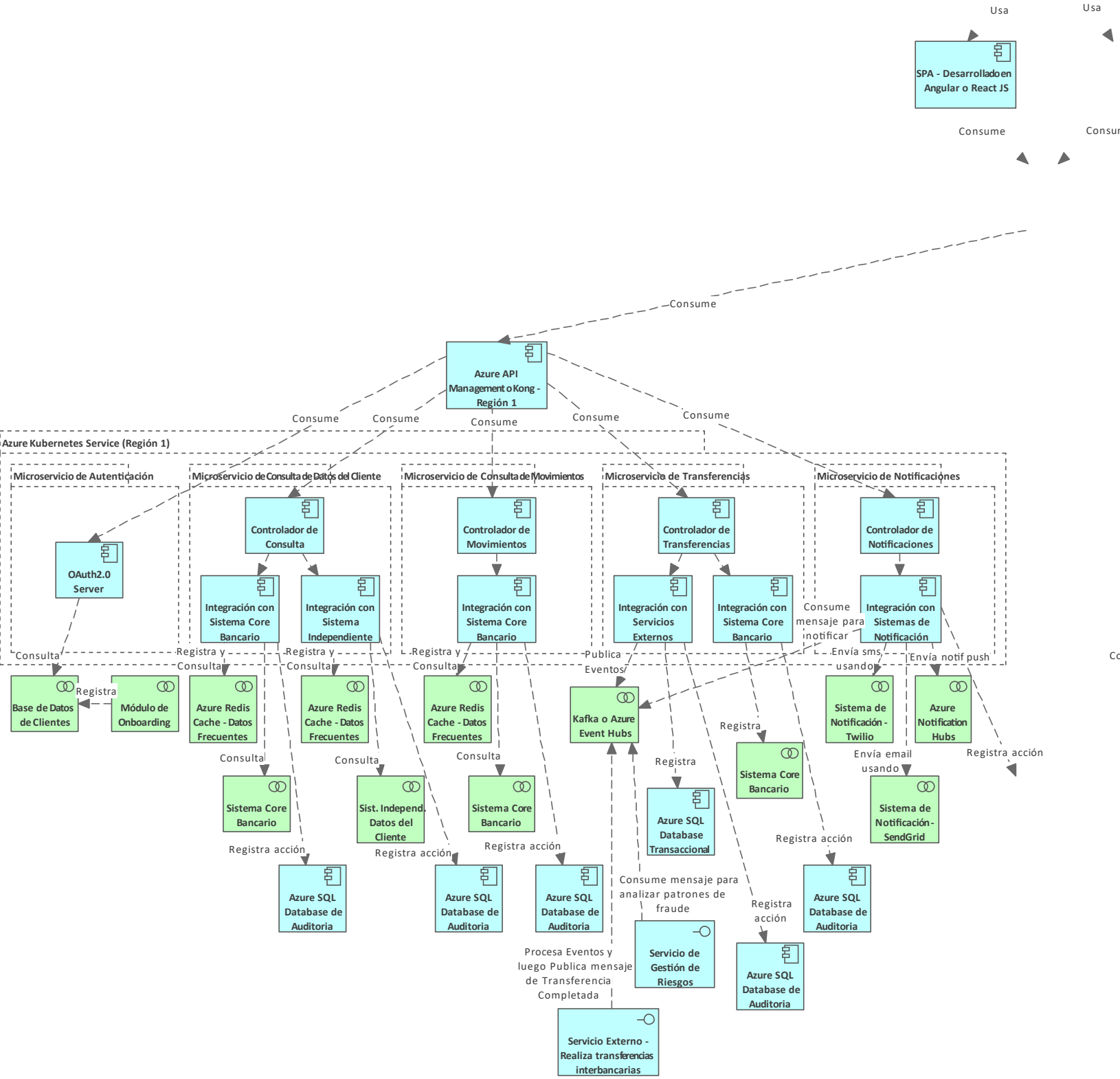
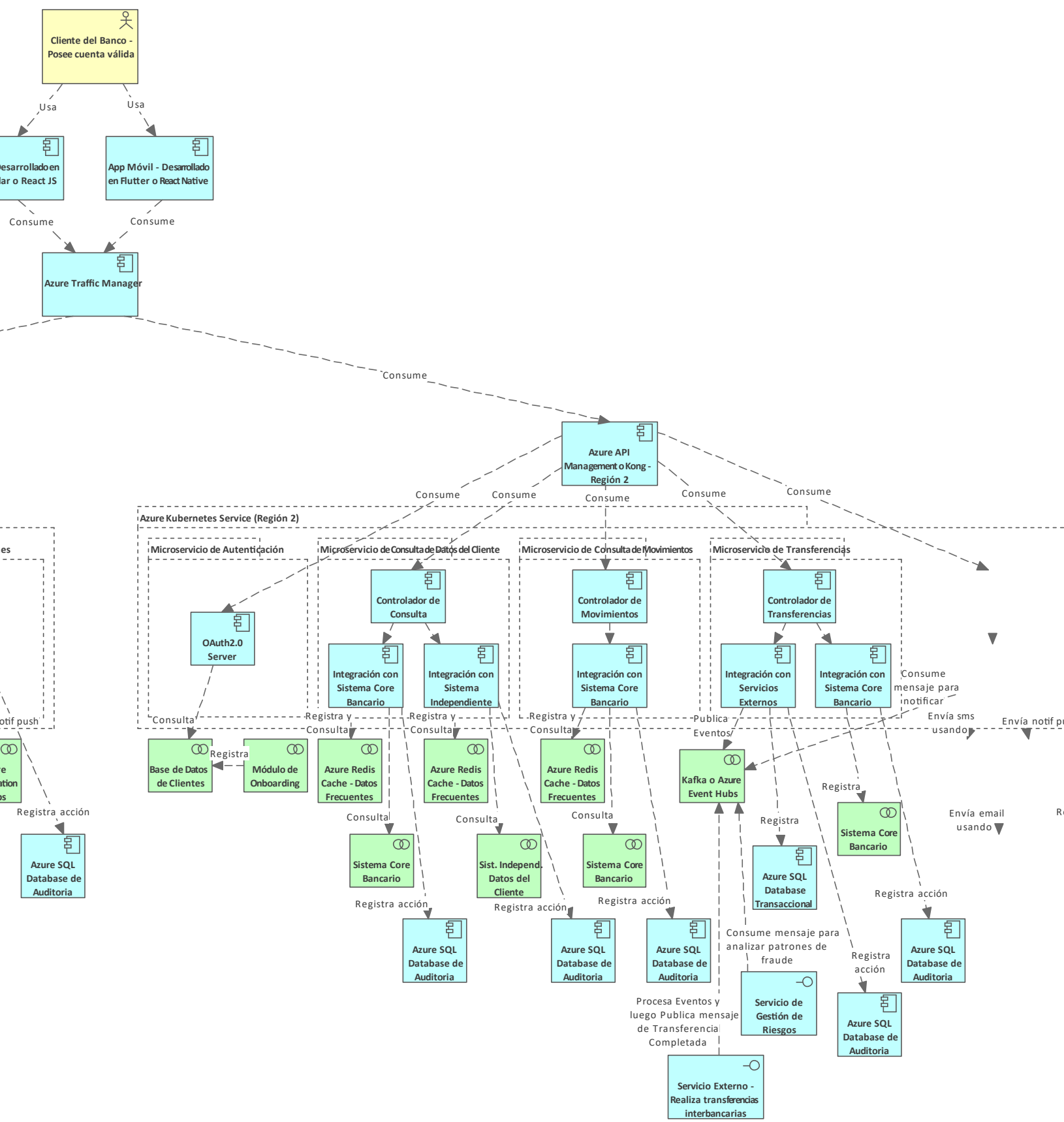

Usa
Usa
SPA - Desarrollado en Angular o React JS
Consume
Consume
Consume
Azure API Management o Kong - Región 1
Consume
Consume
Consume
Consume
Consume
Azure Kubernetes Service (Región 1)
Microservicio de Autenticación
Microservicio de Consulta de Datos del Cliente
Microservicio de Consulta de Movimientos
Microservicio de Transferencias
Microservicio de Notificaciones
OAuth2.0 Server
Controlador de Consulta
Controlador de Movimientos
Controlador de Transferencias
Controlador de Notificaciones
Integración con Sistema Core Bancario
Integración con Sistema Independiente
Integración con Sistema Core Bancario
Integración con Servicios Externos
Integración con Sistema Core Bancario
Integración con Sistemas de Notificación
Consume mensaje para notificar
Consulta
Registra y Consulta
Registra y Consulta
Registra y Consulta
Publica Eventos
Envía sms usando
Envía notif push
Base de Datos de Clientes
Registra
Módulo de Onboarding
Azure Redis Cache - Datos Frecuentes
Azure Redis Cache - Datos Frecuentes
Azure Redis Cache - Datos Frecuentes
Kafka o Azure Event Hubs
Sistema de Notificación - Twilio
Azure Notification Hubs
Registra
Sistema Core Bancario
Envía email usando
Registra acción
Consulta
Consulta
Consulta
Registra
Sistema Core Bancario
Sist. Independ. Datos del Cliente
Sistema Core Bancario
Azure SQL Database Transaccional
Sistema de Notificación - SendGrid
Registra acción
Registra acción
Registra acción
Registra acción
Consume mensaje para analizar patrones de fraude
Registra acción
Azure SQL Database de Auditoria
Azure SQL Database de Auditoria
Azure SQL Database de Auditoria
Servicio de Gestión de Riesgos
Azure SQL Database de Auditoria
Procesa Eventos y luego Publica mensaje de Transferencia Completada
Servicio Externo - Realiza transferencias interbancarias



Propuesta de Plan de Migración Gradual

Fase 1: Preparación de la Infraestructura

1. Configuración de los entornos en Azure (AKS, Redis Cache, SQL Database, API Management).
2. Pruebas de rendimiento y seguridad en la infraestructura.

Fase 2: Integración con los Sistemas Legacy

1. Migrar gradualmente las APIs que interactúan con sistemas legacy al nuevo entorno gestionado por Azure API Management. Realizar pruebas de carga monitoreo.
2. Implementar microservicios paralelos para consult los sistemas legacy y comparar los resultados.
3. Asegurar que las nuevas consultas y operaciones s auditadas en los nuevos microservicios.

Razones de los siguientes Framework para construir la SPA:
Angular.- Es un framework de desarrollo web completo mantenido por Google. Proporciona una estructura sólida y una amplia gama de herramientas y características integradas que facilitan el desarrollo de aplicaciones robustas y mantenibles. Debido a su robustez y soporte a largo plazo por parte de Google, Angular es una opción popular en el desarrollo de aplicaciones empresariales.
React JS.- Es una biblioteca de JavaScript mantenida por Facebook que permite la construcción de interfaces de usuario de manera eficiente y flexible. Se basa en la creación de componentes reutilizables, lo que facilita el desarrollo de interfaces complejas y su mantenimiento. Cuenta con un ecosistema rico y una comunidad activa, lo que proporciona una gran cantidad de herramientas, bibliotecas y recursos adicionales.

Razones de los siguientes Framework para la Aplicación Móvil:
Flutter.- Desarrollado por Google, permite el desarrollo rápido de aplicaciones móviles con una alta productividad gracias a su característica de hot reload, que permite ver los cambios al instante. Ofrece una amplia gama de widgets personalizables que permiten crear interfaces de usuario nativas y de alta calidad para ambas plataformas (iOS y Android). Compila directamente al código nativo, lo que resulta en un alto rendimiento de la aplicación.
React Native.- Mantenido por Facebook, permite el uso de componentes nativos, lo que proporciona una experiencia de usuario similar a la de una aplicación nativa. Permite desarrollar aplicaciones para iOS y Android desde un solo código base, reduciendo el tiempo y los costos de desarrollo. La arquitectura permite integrar fácilmente módulos nativos y bibliotecas externas, proporcionando flexibilidad y extensibilidad en el desarrollo de la aplicación.

Conclusión: La elección de Angular y React JS para la SPA se debe a su robustez, flexibilidad, y la capacidad de manejar aplicaciones complejas y de gran escala. Para la aplicación móvil, Flutter y React Native fueron seleccionados debido a su capacidad de proporcionar una experiencia de usuario nativa, desarrollo rápido y la eficiencia de mantener un solo código base para múltiples plataformas.

Recomendaciones del mejor flujo de autenticación que se debe usar según el estándar:

- **Tanto para la SPA y la App Móvil se recomienda el flujo:** Authorization Code Flow with PKCE que es una extensión del Authorization Code Flow y añade una capa de seguridad adicional para las aplicaciones públicas (como las SPAs) que no pueden mantener secretos de cliente de forma segura; tal como en la App Móvil.
- **Recomendaciones adicionales:** Uso de tokens de actualización (Refresh Tokens), autenticación multifactor (MFA), rotación y revocación de tokens, almacenamiento seguro de tokens de acceso y cifrado de tráfico (HTTPS).

Patrón de Diseño escogido para el trato de la persistencia de información de clientes frecuentes:

- **Cache-Aside (Lazy Loading):** Este patrón carga los datos en la caché solo cuando se solicitan, y los guarda en la caché después de que se recuperan de la base de datos.

Componentes:

* Cache Layer (Redis): Almacena temporalmente los datos para mejorar el rendimiento.
* Database: Almacena de manera persistente todos los datos.
* Application Logic: Maneja la lógica para interactuar con la caché y la base de datos.

Implementación:

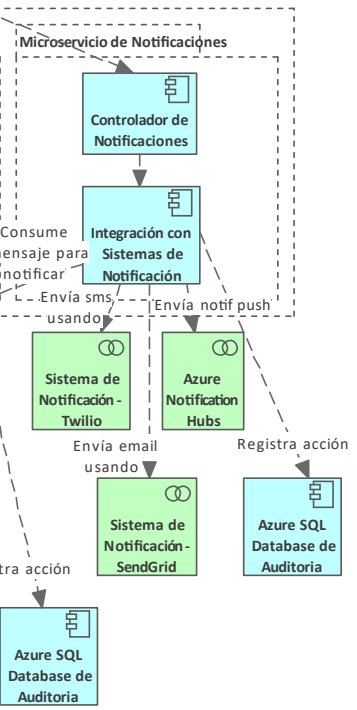
1. La aplicación intenta leer datos desde Redis.
2. Si los datos no están en Redis (miss), se recuperan de la base de datos.
3. Los datos recuperados de la base de datos se guardan en Redis.
4. Las actualizaciones se realizan primero en la base de datos y luego se invalidan o actualizan en Redis.

Beneficios:

* Rendimiento Mejorado: Reduce la carga en la base de datos al servir datos desde la caché.
* Consistencia Eventual: Asegura que los datos en la caché y la base de datos se mantengan consistentes eventualmente.

Normativas importantes para crear aplicaciones para Entidades Financieras:

- **Ley de Protección de Datos Personales:** GDPR (Reglamento General de Protección de Datos). Garantizar el manejo seguro y adecuado de los datos personales de los usuarios, cumpliendo con normativas locales e internacionales.
- **Seguridad de la Información:** ISO/IEC 27001. Un estándar internacional para la gestión de la seguridad de la información.
- **Regulaciones Financieras:** PCI DSS (Payment Card Industry Data Security Standard). Un estándar de



sta de Plan de Migración Gradual
n de la Infraestructura
ón de los entornos en Azure (AKS, Redis
Database, API Management).
rendimiento y seguridad en la
tura.

n con los Sistemas Legacy
dualmente las APIs que interactúan con los
gacy al nuevo entorno gestionado por
Management. Realizar pruebas de carga y

r microservicios paralelos para consultar
s legacy y comparar los resultados.
ue las nuevas consultas y operaciones son
en los nuevos microservicios.

Propuesta de Plan de Migración Gradual

Fase 1: Preparación de la Infraestructura

1. Configuración de los entornos en Azure (AKS, Redis Cache, SQL Database, API Management).
2. Pruebas de rendimiento y seguridad en la infraestructura.

Fase 2: Integración con los Sistemas Legacy

1. Migrar gradualmente las APIs que interactúan con sistemas legacy al nuevo entorno gestionado por Azure API Management. Realizar pruebas de carg monitoreo.
2. Implementar microservicios paralelos para consult los sistemas legacy y comparar los resultados.
3. Asegurar que las nuevas consultas y operaciones s auditadas en los nuevos microservicios.

Fase 3: Migración de Datos

1. Migrar datos históricos a las nuevas bases de dato Azure SQL Database. Establecer una sincronizació entre las bases de datos legacy y las nuevas.
2. Realizar pruebas de integridad de datos y validacio de seguridad.

Fase 4: Activación de los Nuevos Sistemas

1. Activar los nuevos microservicios de manera gradu desactivar las interfaces legacy una vez que se val los resultados.
2. Monitorear el desempeño de los sistemas y realiza ajustes en tiempo real.

Fase 5: Finalización y Desmantelamiento del Sistema Legacy

1. Desmantelar los sistemas legacy una vez que todo los datos hayan sido migrados y validados.
2. Apagar gradualmente los servidores legacy y migr las funcionalidades remanentes.

Propuesta de Gobernanza de APIs

Políticas de Seguridad

- **Autenticación y Autorización:** Todas las APIs debe autenticarse utilizando OAuth 2.0 o JWT para garantizar que solo usuarios o sistemas autorizado tengan acceso.
- **Límites de Tasa (Rate Limiting):** Implementar límit de tasa (rate limiting) en las APIs públicas para evit abusos. Por ejemplo, limitar las solicitudes a 1000 minuto por usuario.

Versionado

- **Versionado de APIs:** Las APIs deben seguir un esquema de versionado claro, por ejemplo, v1, v2 etc. Esto permite que las versiones anteriores siga funcionando mientras se implementan cambios o mejoras en las nuevas versiones.
- **Deprecación de APIs:** Cuando una API es deprecad se debe comunicar claramente a los consumidores mediante avisos con un período de gracia de al me 6 meses antes de la eliminación.

Monitoreo y Auditoría

- **Monitoreo de Uso:** Utilizar herramientas como Azu API Management para monitorear el uso de las AP en tiempo real. Esto incluye supervisar los tiempos respuesta, tasas de éxito, y solicitudes fallidas.
- **Auditoría:** Todas las solicitudes deben ser auditada registradas. Esto incluye quién accedió, cuándo, y datos fueron solicitados. Estas auditorías deben es disponibles para revisiones de cumplimiento.

sta de Plan de Migración Gradual
n de la Infraestructura
ón de los entornos en Azure (AKS, Redis
Database, API Management).
rendimiento y seguridad en la
tura.

n con los Sistemas Legacy
dualmente las APIs que interactúan con los
gacy al nuevo entorno gestionado por
Management. Realizar pruebas de carga y

r microservicios paralelos para consultar
s legacy y comparar los resultados.
ue las nuevas consultas y operaciones son
en los nuevos microservicios.

de Datos
os históricos a las nuevas bases de datos en
Database. Establecer una sincronización
ases de datos legacy y las nuevas.
uebas de integridad de datos y validaciones
d.

n de los Nuevos Sistemas
nuevos microservicios de manera gradual y
as interfaces legacy una vez que se validen
os.
el desempeño de los sistemas y realizar
iempo real.

ón y Desmantelamiento del Sistema

r los sistemas legacy una vez que todos
ayan sido migrados y validados.
dualmente los servidores legacy y migrar
alidades remanentes.

ropuesta de Gobernanza de APIs
dad
ón y Autorización: Todas las APIs deben
e utilizando OAuth 2.0 o JWT para
que solo usuarios o sistemas autorizados
eso.
Tasa (Rate Limiting): Implementar límites
e limiting) en las APIs públicas para evitar
r ejemplo, limitar las solicitudes a 1000 por
usuario.

de APIs: Las APIs deben seguir un
e versionado claro, por ejemplo, v1, v2,
rmite que las versiones anteriores sigan
o mientras se implementan cambios o
las nuevas versiones.
n de APIs: Cuando una API es deprecada,
municar claramente a los consumidores
visos con un período de gracia de al menos
tes de la eliminación.

oría
de Uso: Utilizar herramientas como Azure
ement para monitorear el uso de las APIs
real. Esto incluye supervisar los tiempos de
tasas de éxito, y solicitudes fallidas.
Todas las solicitudes deben ser auditadas y
. Esto incluye quién accedió, cuándo, y qué
n solicitados. Estas auditorías deben estar
para revisiones de cumplimiento.

* Consistencia Eventual: Asegura que los datos en la caché y la base de datos se mantengan consistentes eventualmente.

Normativas importantes para crear aplicaciones para Entidades Financieras:

- **Ley de Protección de Datos Personales:** GDPR (Reglamento General de Protección de Datos). Garantizar el manejo seguro y adecuado de los datos personales de los usuarios, cumpliendo con normativas locales e internacionales.
- **Seguridad de la Información:** ISO/IEC 27001. Un estándar internacional para la gestión de la seguridad de la información.
- **Regulaciones Financieras:** PCI DSS (Payment Card Industry Data Security Standard). Un estándar de seguridad de datos para las organizaciones que manejan información de tarjetas de crédito.
- **Protección contra el Lavado de Dinero:** AML (Anti-Money Laundering). Regulaciones diseñadas para prevenir el lavado de dinero y otras actividades financieras ilegales.

Para asegurar Alta Disponibilidad (HA):

- Se consideró el despliegue en **múltiples regiones de Azure** para garantizar HA y DR.
- Uso de **Azure Traffic Manager** para que distribuya el tráfico entre diferentes regiones para asegurar que el sistema permanezca disponible incluso si una región experimenta problemas.
- Uso del balanceo de carga de **Kong o Azure APIM** para distribuir la carga a nivel aplicativo entre múltiples instancias.

Para asegurar Tolerancia a Fallos:

- Uso de **Azure API Management** para gestionar y proteger las APIs, distribuyendo las solicitudes entre los microservicios backend.
- Implementación de **Azure API Management en múltiples regiones** para asegurar que las APIs sigan siendo accesibles incluso si una región experimenta fallos.
- Se consideró que los microservicios se desplieguen en **Azure Kubernetes Service (AKS)** en ambas regiones para asegurar que el sistema permanezca funcional incluso si una región falla.
- El uso de **Azure Redis Cache** está replicado en ambas regiones para proporcionar tolerancia a fallos y mejorar la disponibilidad de los datos.

Para asegurar Recuperación Ante Desastres (DR):

- Se consideró las bases de datos (**Azure SQL Database**) en ambas regiones para asegurar la disponibilidad y recuperación rápida en caso de desastre.
- También, se consideró que los **servicios externo**s permanezcan accesibles desde ambas regiones.

Con respecto a Seguridad:

- El uso de **Azure API Management** aplica políticas de seguridad, autenticación y autorización para proteger las APIs. Por ejemplo:

1. Uso de **Json Web Token (JWT)** para el manejo de tokens se seguridad de la aplicación.

- Uso de **Azure Redis Cache** para que la información de clientes frecuentes esté almacenada de forma segura.
- Uso de Algoritmo **AES-256 o Azure Key Vault** para gestionar secretos y claves de cifrado y así guardar de forma segura la data sensible en **Azure SQL Database.**

Con respecto a Monitoreo, Excelencia Operativa y Auto-Healing:

- Se debe considerar el uso de **Azure Monitor y Azure Log Analytics** para supervisión de la infraestructura y la aplicación en tiempo real y análisis de logs centralizado.
- Se debe considerar el uso de **Azure Application Insights** para rastreo y monitoreo de rendimiento de la aplicación.
- Uso de **Azure Kubernetes Service (AKS),** ya que orquesta los contenedores y proporciona capacidades de auto-healing. Se puede configurar para que escale automáticamente los contenedores basándose en la carga de trabajo y además reinicia los contenedores fallidos automáticamente para asegurar la continuidad del servicio.