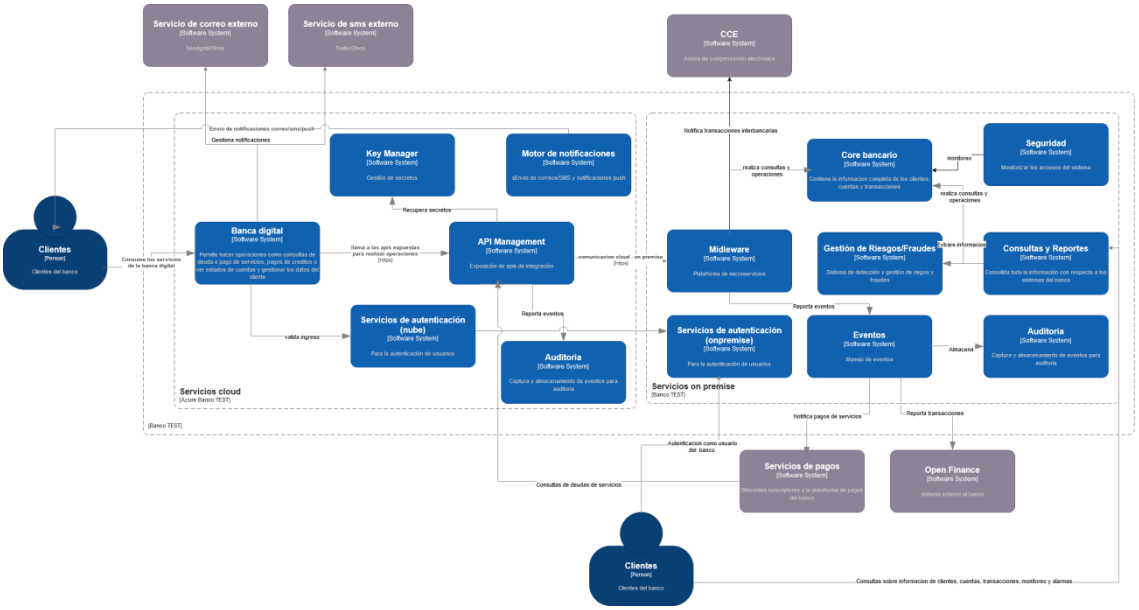
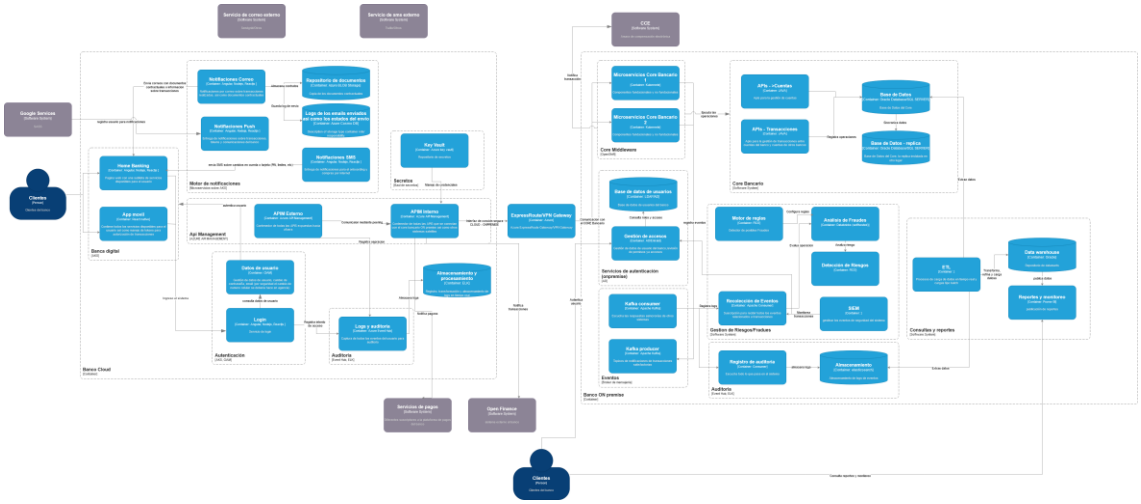
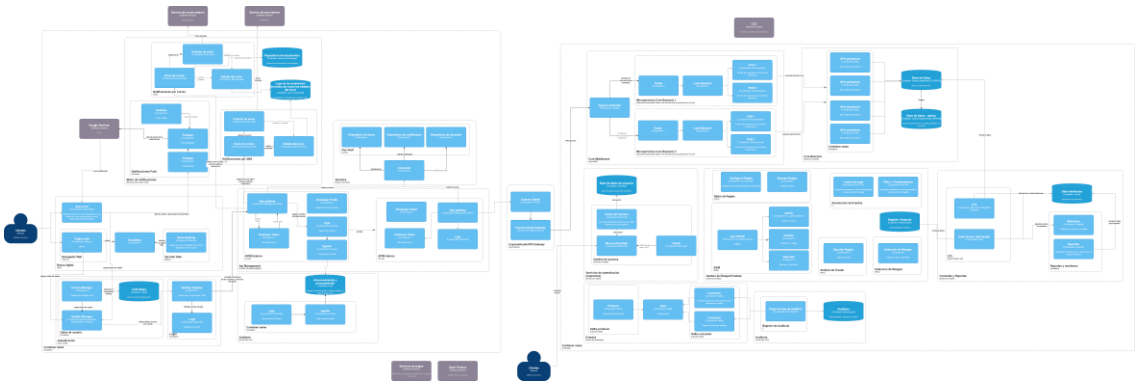
1. Arquitectura de alto nivel

   Se considera el diagrama adjunto, tanto para el diagrama de contexto, contenedores y componentes







2. Patrones de integración

   Se planea utilizar el patrón API Gateway para utilizar microservicios que expondrán todas las capacidades del banco a través de Azure y el patrón de microservicios con Kubernetes y Opensfhift para la orquestación, por último se plantea utilizar el patrón de Message Broker para el control de fraudes utilizando Apache Kafka.

3. Requisitos de seguridad de datos, cumplimiento normativo y ley orgánica de protección
   Dentro de los requisitos de seguridad primeramente tenemos que el manejo de la información sensible del cliente debe ser maneja de manera encriptada, como lineamiento en ningún lugar esta información debe ser almacenada en claro.

4. HA y Plan DRP
   Para mantener la alta disponibilidad se propone levantar los componentes de openshift en cluster configurando el deploy controller para administrar la replicación y actualización de los pods, así como el balance de carga mediante ingress controller para manejar el tráfico que reciba la plataforma, de esta manera que el autoescalamiento sea horizontal y no tener problemas de rendimiento. En cuanto a la base de datos debe ser configurada en cluster (RAC) para manejar multiples instancias, Data Guard configurando la commutacion en modo manual ante errores así como mantener la sincronización entre ambas bases de datos.

   Se propone también manejar redundancia geográfica para los componentes on premise y un despliegue multi región de los componentes de Azure, evaluando los costos que estos implican, para lo cual se debe definir el costo beneficio de esta alternativa, esto dependerá del negocio.

   Sobre el plan de recuperación de desastres se deben definir el alcance de las aplicaciones, la base legal tanto externas (leyes) como internas (normativas del banco) por otro lado se deben definir los tiempos de recuperación (RTO) como el punto objetivo de recuperación (RPO) por ultimo también es necesario definir el tiempo máximo tolerable de interrupción
   Para el plan se deben contar con los recursos tecnológicos, la infraestructura y el soporte de proveedores

   Será importante contar con una infraestructura DevOps en Azure para asegurar una recuperación. Para lograr esto, se deben manejar los siguientes componentes:
   - Implementación de Pipelines de CI/CD
   - Gestión de Infraestructura como Código (IaC)
   - Automatización de Recuperación (Azure automation
   - Se deben realizar pruebas de recuperación controladas de manera periódica.

5. Estrategia integración multicore
   Se propone levantar inicialmente 2 plataformas openshift que contendrían todas las capacidades del core y un api Gateway por delante que distribuya la carga, también sería posible desacoplar ciertas funcionalidades en un nuevo pod logrando cierta independencia

6. Gestión de identidad y accesos al sistema

   Para la gestión de acceso se puede utilizar CIAM para los clientes y para los usuarios internos se puede aplicar el uso de Active Directory y EntraID (Cloud) el cual estaría sincronizado con el AD, de esta manera se podría manejar el acceso a varias aplicaciones con un solo proceso de autenticación, permite utilizar autenticación multifactor y también es posible diferenciar colaboradores y proveedores. Para el manejo de credenciales se puede utilizar OAuth 2.0 para la gestión de permisos y accesos a ciertas aplicaciones o recursos.

7. API externas e internas

   En el diagrama de componentes se explica el funcionamiento del api management interno y externo, pero básicamente la idea es solo exponer las capacidades “publicas” a través del APIM externo ya que este se encontrara expuesto a internet, en cambio el APIM interno tendrá acceso al CORE por lo cual debe ser manejado de manera responsable y en u ambiente donde se garantice la seguridad

8. Gobierno de API y microservicios

   Como lineamiento principal se deben estandarizar las APIS, para esto se puede utilizar el estándar BIAM donde se podrían exponer todas las capacidades del negocio, de acuerdo a eso se pueden organizar las APIS por dominio, posteriormente se deben definir políticas para el versionamiento y despliegue bajo CI/CD, también es importante manejar una políticas de seguridad mediante el uso de token SSL y secretos para la comunicación entre microservicios

9. Plan de migración

   Asumiendo que el banco actualmente se encuentra en bajo de una arquitectura monolítica, lo que sugiere primero es ir migrando las aplicaciones satélites como el envio de correos o notificaciones, posteriormente sería conveniente desacoplar el core bancario y migrarlo a microservicios, esta esta sería la más crítica pues que puede desestabilizar el funcionamiento de todo el banco, es necesario un análisis profundo de las capacidades y destinar entre 6 meses y 1 año para revelar toda la información necesaria.

   Posteriormente definir un cronograma donde se vayan desarrollando todas las apis necesarias, en la etapa de pruebas en OA se deben definir pruebas unitarias, pruebas de estrés para medir el rendimiento y pasar por todas las pruebas de seguridad que estarían en coordinación con el área de riesgos, finalmente se iniciaría con una etapa de Friends and family donde se habilitarían las funcionalidades para un grupo reducido de personas para verificar el correcto funcionamiento de la app, y así paulatinamente se puede ir migrando usuarios de poco en poco (bajo un cronograma de migración) hasta lograr la migración por completo.

   Finalmente se debe migrar la app y el home banking bajo la misma lógica donde primero se realiza la habilitación para un grupo reducido de personas y se va incrementando.

En base a la experiencia y de acuerdo al tamaño del banco esta tarea puede tomar entre 2 y 5 años ya que es un cambio tecnológico muy grande el que se plantea realizar.