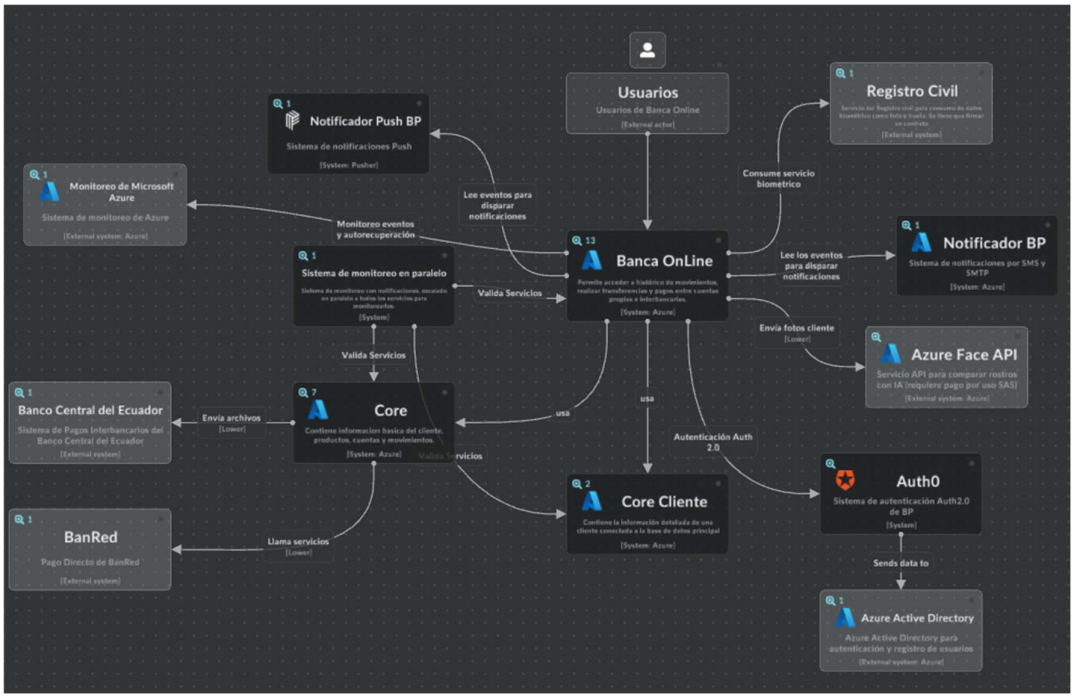
Usted ha sido contratado por una entidad llamada BP como arquitecto de soluciones para diseñar un sistema de banca por internet, en este sistema los usuarios podrán acceder al histórico de sus movimientos, realizar transferencias y pagos entre cuentas propias e interbancarias.

Toda la información referente al cliente se tomará de 2 sistemas, una plataforma Core que contiene información básica de cliente, productos, cuentas, movimientos, y un sistema independiente que complementa la información del cliente cuando los datos se requieren en detalle este sistema core está conectado a una base de datos principal.

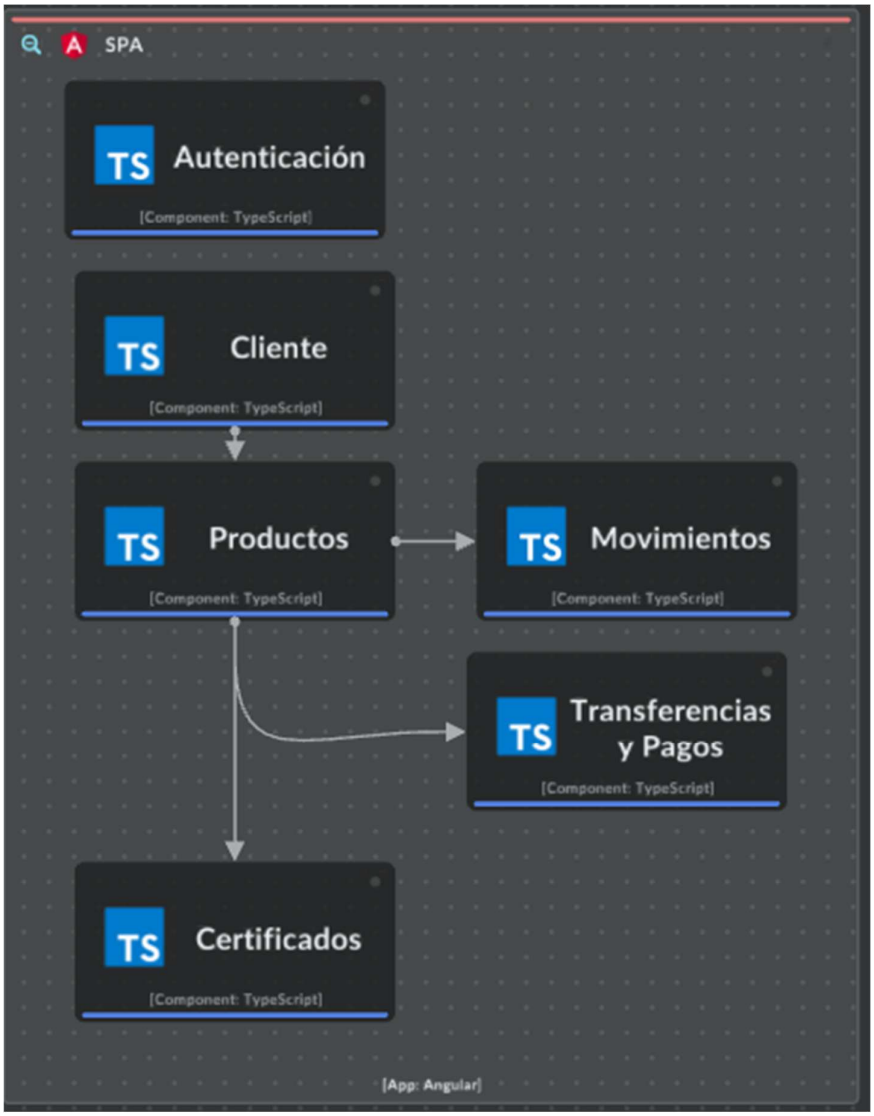


Debido a que la norma exige que los usuarios sean notificados sobre los movimientos realizados, el sistema utilizará sistemas externos o propios de envío de notificaciones, mínimo 2.

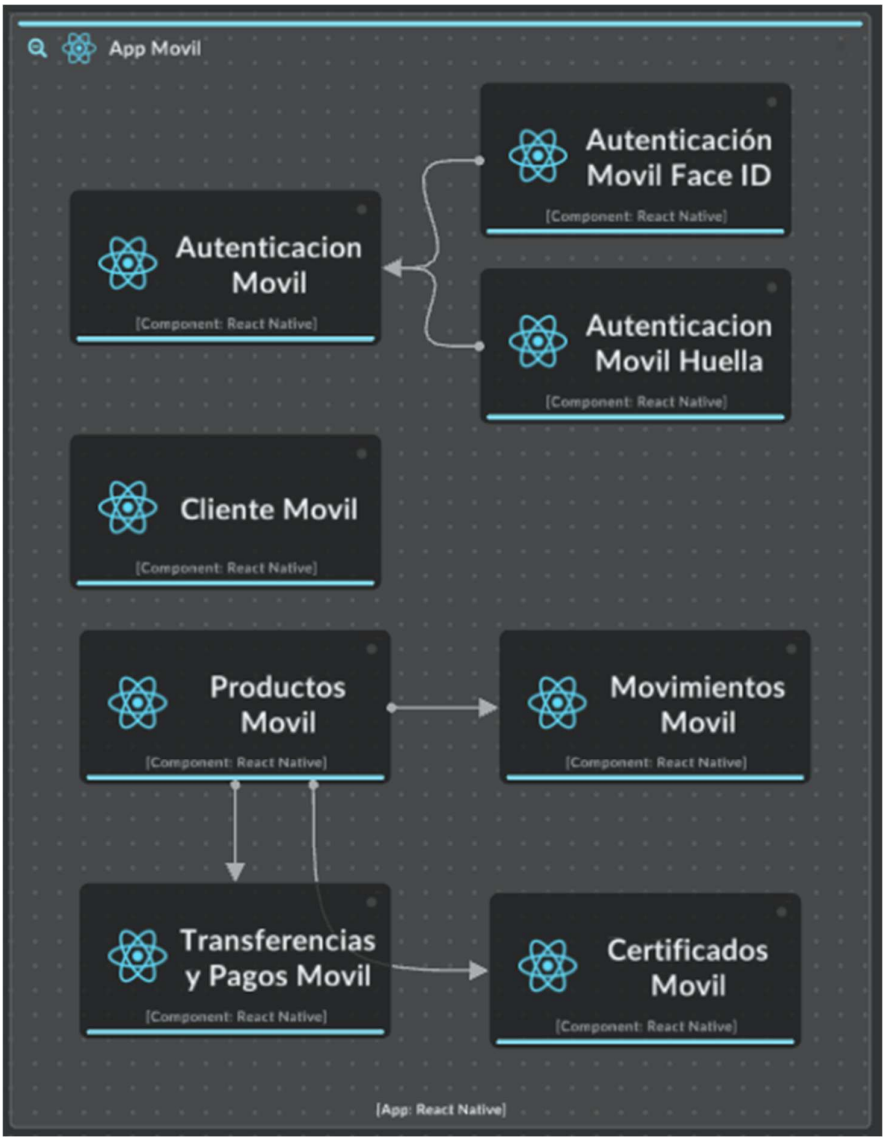
Existirá un notificador basado en Apache Kafka con subscriptores para mensajes SMS y otros SMTP, y otro adicional para notificaciones push para la app móvil.

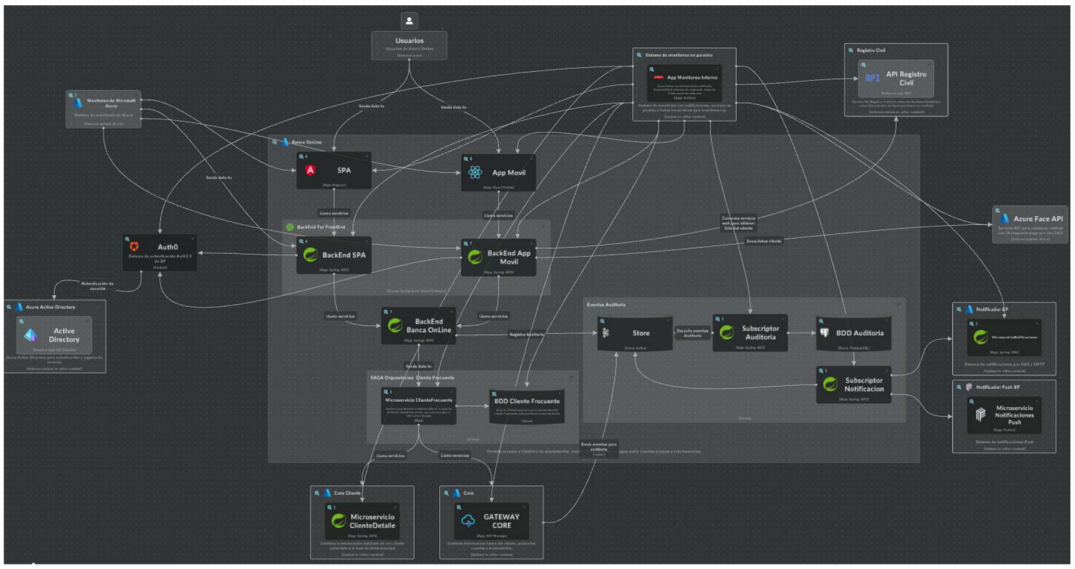
Al realizar transferencias o pagos se publicara un mensaje para que los Sistemas de notificación Push, SMS, SMTP envíen las alertas al cliente.

Este sistema contará con 2 aplicaciones en el Front, una SPA



y una Aplicación móvil desarrollada en un Framework multiplataforma. (Mencione 2 opciones y justifique el porqué de su elección)


App Movil
Autenticación Movil Face ID
[Component: React Native]
Autenticacion Movil
[Component: React Native]
Autenticacion Movil Huella
[Component: React Native]
Cliente Movil
[Component: React Native]
Productos Movil
[Component: React Native]
Movimientos Movil
[Component: React Native]
Transferencias y Pagos Movil
[Component: React Native]
Certificados Movil
[Component: React Native]
[App: React Native]


Usuarios
SPA
App Movil
BackEnd SPA
BackEnd App Movil
Auth0
Azure Face API
Active Directory
BackEnd Banca OnLine
Store
Subscriptor Auditoria
BDD Auditoria
Subscriptor Notificacion
Microservicio ClienteDetalle
GATEWAY CORE

SPA: Angular

APP: React Native

Otras opciones pueden ser VUE para SPA y Xamarin de Microsoft.

Angular + React Native ofrece ventajas sobre **Vue + Xamarin** para una aplicación de banca móvil en términos de seguridad, escalabilidad, rendimiento y soporte.

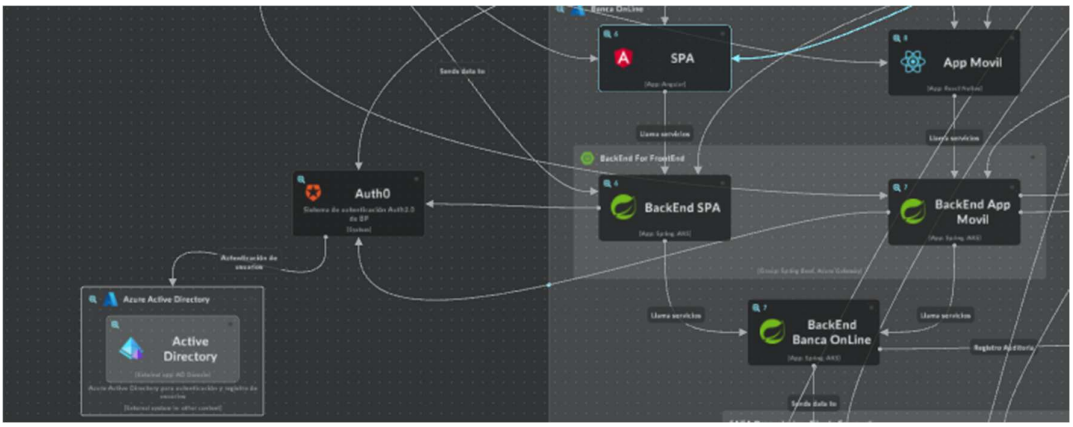
Angular es más robusto y seguro que Vue, con mejor estructura para proyectos grandes y protección contra ataques es desarrollado por Google y tiene soporte continuo y parches de seguridad.

React Native tiene mejor integración con iOS y Android que Xamarin, proporcionando una mejor experiencia de usuario y rendimiento.

Seguridad y mantenimiento: Angular maneja mejor autenticación y protección de datos, y React Native tiene un ecosistema más amplio que facilita la integración y actualizaciones.

Compatibilidad y soporte: React Native recibe mejoras constantes, mientras que Xamarin depende de Microsoft, lo que puede limitar su evolución.

Ambas aplicaciones autenticarán a los usuarios mediante un servicio que usa el estándar Oauth2.0, para el cual no requiere implementar toda la lógica, ya que la compañía cuenta con un producto que puede ser configurado para este fin; sin embargo, debe dar recomendaciones sobre cuál es el mejor flujo de autenticación que se debería usar según el estándar.

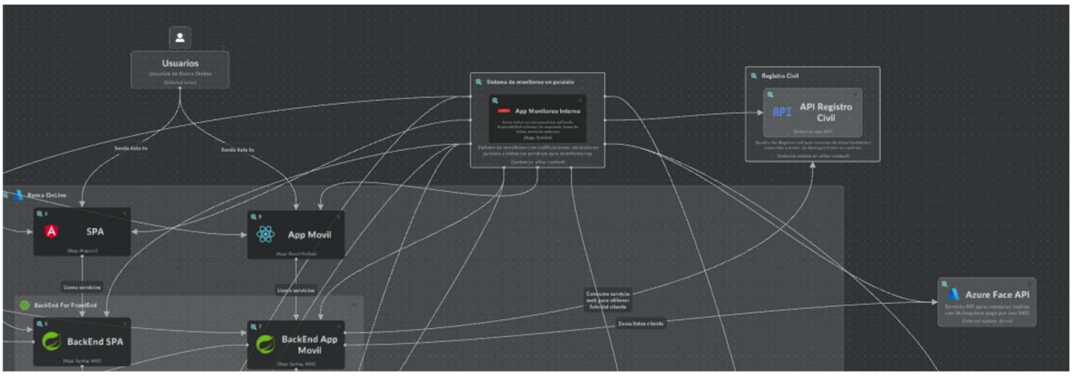


Auth0 manejara la autenticación por MFA, roles y permisos. Adicional posee seguridad contra ataques de bots y fuerza bruta.

El flujo de código de autorización con PKCE (Authorization Code Flow with PKCE) se usa para evitar el ataque de interceptación de código de autorización podemos

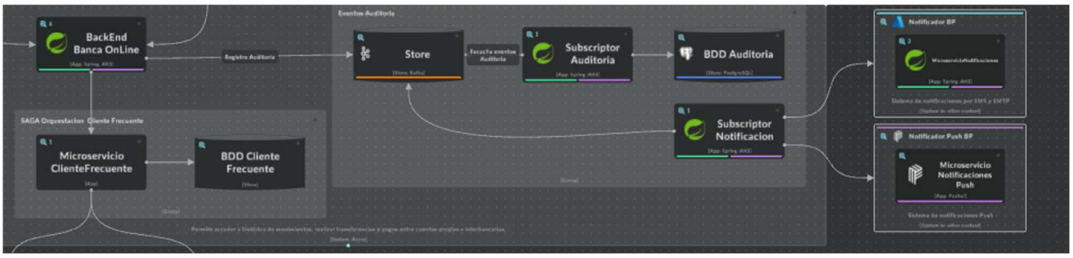
utilizar PKCE que amplía el flujo de código de autorización impidiendo CSRF (opens new window). PKCE, pronunciado "pixy" es un acrónimo de Proof Key for Code Exchange. Este flujo incluye nuevos elementos PKCE (code verifier, code challenge y code challenge method) en varios pasos del flujo encargados de proteger la comunicación entre el cliente y el servidor de autenticación.

Tenga en cuenta que el sistema de Onboarding para nuevos clientes en la aplicación móvil usa reconocimiento facial, por tanto, su arquitectura deberá considerarlo como parte del flujo de autorización y autenticación, a partir del Onboarding el nuevo usuario podrá ingresar al sistema mediante usuario y clave, huella o algún otro método especifique alguno de los anteriores dentro de su arquitectura, también puede recomendar herramientas de industria que realicen estas tareas y robustezca su aplicación.



Se utilizará los servicios del registro civil para obtener la foto del cliente mediante su cedula y código dactilar una vez que haya aceptado y dado el consentimiento del uso de su información para la Ley de Protección de Datos personales, solicitaremos una foto que se comparará con el servicio de Azure Face API para validar que se trate de la misma persona y poder dar de alta al cliente en el Onboarding.

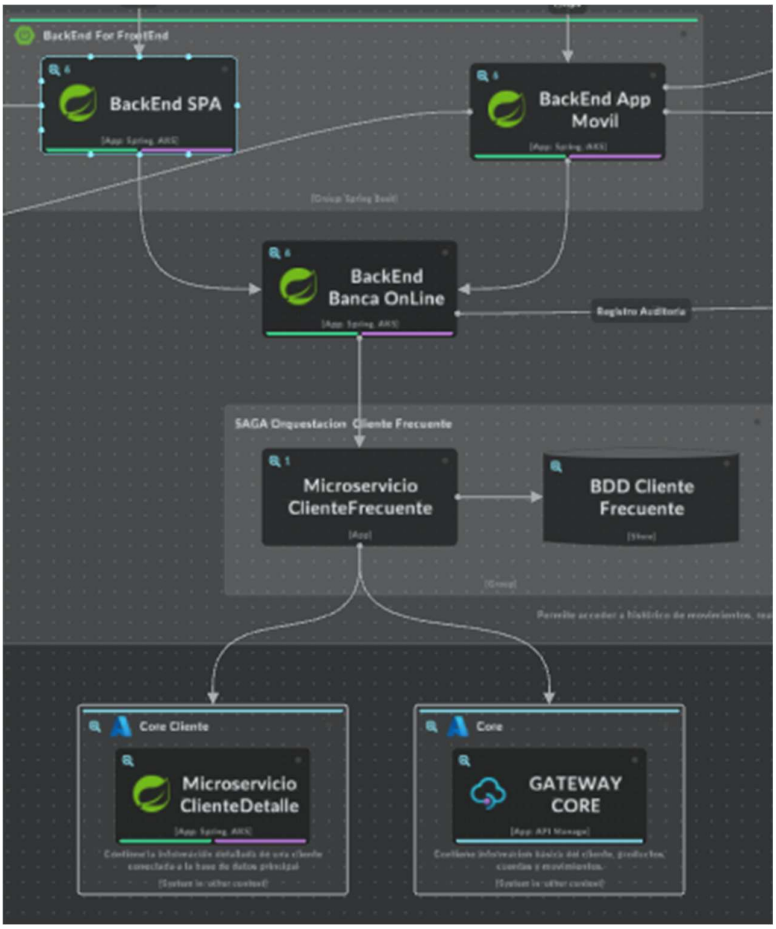
El sistema utiliza usa base de datos de auditoría que registra todas las acciones del cliente



Para este caso vamos a implementar el patrón de publicador/suscriptor mediante Apache Kafka toda acción en la app publicara tópicos para que un subscriptor de

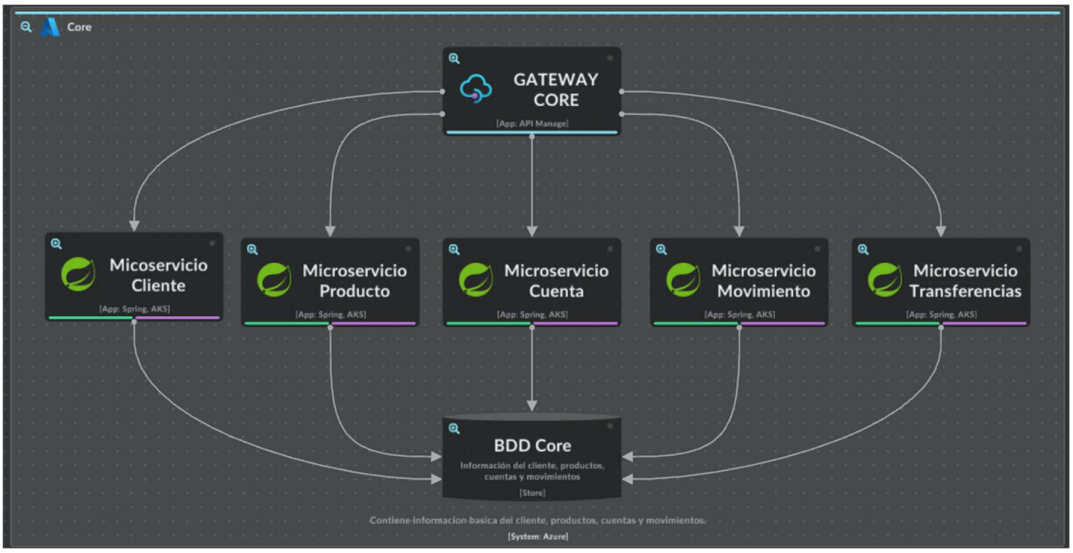
auditoria los esté escuchando y los vaya procesando sin afectar al rendimiento de la aplicación.

y cuenta con un mecanismo de persistencia de información para clientes frecuentes, para este caso proponga una alternativa basada en patrones de diseño que relacione los componentes que deberían interactuar para conseguir el objetivo.



Existirá un servicio que identificara si el cliente es frecuente y persistirá la información del mismo en una BDD usando el patrón SAGA Orquestación

Para obtener los datos del cliente el sistema pasa por una capa de integración compuesta por un api Gateway y consume los servicios necesarios de acuerdo con el tipo de transacción, inicialmente usted cuenta con 3 servicios principales, consulta de datos básicos, consulta de movimientos y transferencias, que realiza llamados a servicios externos dependiendo del tipo, si considera que debería agregar más servicios para mejorar la repuesta de información a sus clientes, es libre de hacerlo.



Para este reto, mencione aquellos elementos normativos que podrían ser importantes a la hora de crear aplicaciones para entidades financieras, Ejemplo ley de datos personales, seguridad etc.

Datos personales.

Intereses, tarifas y comisiones.

ISO 27000

Código Orgánico Monetario y Financiero (COMF):

Define el marco legal para la banca, incluyendo la regulación de la banca electrónica y los sistemas de pago.

Ley Orgánica para Defender los Derechos de los Clientes del Sistema Financiero Nacional:

Protege a los usuarios de la banca, estableciendo normas para la transparencia, la protección de datos y la prevención de abusos.

Resoluciones de la Superintendencia de Bancos:

La Superintendencia de Bancos y Seguros (SB) emite resoluciones que complementan el marco legal y regulan aspectos específicos de la banca electrónica, como la seguridad de los canales electrónicos y la gestión de riesgos.

Resoluciones de la Junta de Política y Regulación Monetaria (JPRM):

La JPRM dicta normativas sobre los medios de pago electrónicos y la actividad de las Fintech.

Aspectos clave de la regulación de la banca electrónica:

Seguridad:

Las instituciones financieras deben implementar medidas de seguridad tecnológica y de control para garantizar la integridad, disponibilidad y confidencialidad de los datos de los usuarios en los canales electrónicos.

Transparencia:

Las entidades financieras deben informar de manera clara y concisa a los clientes sobre los costos, tarifas, riesgos y beneficios de los productos y servicios financieros ofrecidos a través de canales electrónicos.

Protección de datos:

Se debe garantizar la confidencialidad y la protección de los datos personales de los usuarios en los canales electrónicos, cumpliendo con la Ley Orgánica de Protección de Datos Personales.

Derechos de los usuarios:

Los usuarios tienen derecho a reclamar, presentar quejas y denunciar irregularidades ante la Superintendencia de Bancos y Seguros o las instancias competentes.

Garantice en su arquitectura, alta disponibilidad (**HA**), tolerancia a fallos, recuperación ante desastres (**DR**), Seguridad y Monitoreo, Excelencia operativa y auto-healing.

OnPremise

HA

Para servidores físicos o virtuales, es fundamental desplegar un clúster con balanceo de carga para optimizar la distribución del tráfico y garantizar la disponibilidad.

Si se requieren contenedores OnPremise, BP dispone de OpenShift, que ofrece alta disponibilidad mediante la escalabilidad horizontal de PODs. Esta funcionalidad debe configurarse adecuadamente en los archivos de configuración vinculados a los pipelines de implementación.

DR

Banco Pichincha ya cuenta con su DCA implementado, por lo que es necesario desplegar la infraestructura adecuada para la réplica de información. Por ejemplo, los datos y aplicaciones pueden replicarse mediante copias de almacenamiento, mientras que las bases de datos pueden sincronizarse utilizando tecnologías como AlwaysOn o el modelo de Publicador/Distribuidor/Suscriptor.

Seguridad y Monitoreo, Excelencia operativa y auto-healing

Las aplicaciones OnPremise se conectan a una plataforma de monitoreo a través de agentes encargados de detectar eventos, analizar puertos y realizar búsquedas en logs, presentando los datos en dashboards de supervisión y notificaciones.

El estado de los componentes se evalúa mediante reglas predefinidas, mientras que las acciones correctivas pueden ser ejecutadas de forma automática según reglas establecidas o de manera manual a través de tareas específicas.

Cloud

HA

Al implementar esta solución en Azure sobre ASK podemos incluir más seguridad como Microsoft Defender para Kubernetes, incluimos el tema de monitoreo y control y también la excelencia operativa ya que AKS escalara automáticamente de manera horizontal nuestros Microservicios y esto brindara calidad de servicio a nuestros clientes y reducirá el mantenimiento de infraestructura local.

DR

Dentro de Azure se debe configurar planes con redundancia y replica a otras zonas.

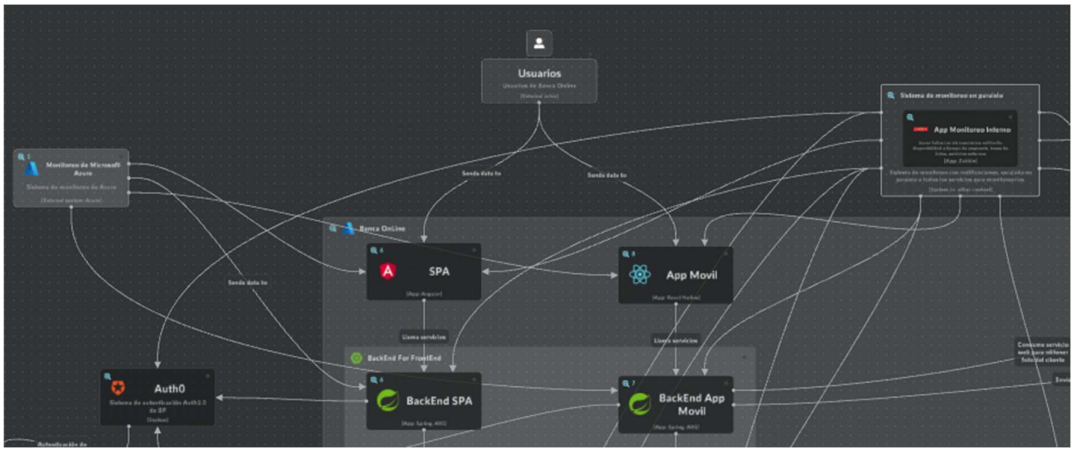
Tolerancia a fallos:

Al implementar nuestra arquitectura desacoplamos y generamos Microservicios con una única responsabilidad, además implementaremos en los componentes críticos como consulta de clientes y productos un patron de reintentos adicional al cliente frecuente para traer la información, y si servicios como el de movimientos no responde para una cuenta, no parara la funcionalidad de todo el sistema por ello.

Seguridad y Monitoreo, Excelencia operativa y auto-healing

Al implementar esta solución en Azure sobre ASK podemos incluir más seguridad como Microsoft Defender para Kubernetes, incluimos el tema de monitoreo y control y también la excelencia operativa ya que AKS escalara automáticamente de manera horizontal nuestros Microservicios y esto brindara calidad de servicio a nuestros clientes y reducirá el mantenimiento de infraestructura local.

Para el Auto-healing vamos a aprovechar la capacidad de Azure dentro del Azure App Service donde configuraremos las reglas y las acciones que Azure auto healing services.



Sí lo considera necesario, su arquitectura puede contener elementos de infraestructura en nube como Azure o AWS, garantice baja latencia en sus servicios, cuenta con presupuesto para esto.

> Se implemento microservicios sobre Azure que se despliegan sobre AKS lo que brinda resiliencia, escalabilidad, supervisión y monitoreo con Azure Monitor y adicional Zabbix para tener un doble monitoreo ya que puede ser que existan problemas de comunicación hacia el exterior de Azure.

En lo posible plantee una arquitectura desacoplada con elementos y reusables y cohesionados para otros componentes que puedan adicionarse en el futuro.

Toda la arquitectura se implemento con Microservicios respetando los principios SOLID, para que cumplan una sola función y se encuentren desacoplados siendo reutilizables.

El modelo debe ser desarrollado bajo **c4**(Modelo de Contexto, Modelo de aplicación o contenedor y Componentes), describa hasta el modelo de componentes, la infraestructura la puede modelar como usted lo considere usando la herramienta de su preferencia.

Link del modelo C4 en IcePanel: https://s.icepanel.io/MJn9sBcJNj3Wp6/AZTo

Link Github: https://github.com/majurado/TestArquitectura

¡Éxitos!