Informe Reto Técnico DEVSU

El cliente ha solicitado realizar el siguiente reto técnico, para postular al cargo de Arquitecto de Soluciones

Descripción

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.

Justificación
- *Para el envío de notificaciones al cliente se propone el servicio AWS SNS que permite el envío de notificaciones mediante correo electrónico o mensajes al número de celular del cliente.* - *Adicional para el envío de correos electrónicos a través de Symantec que hace un relay de los mensajes y que puede operar en la IT On Premise del 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)

Justificación
- *Para la aplicación SPA se menciona como alternativas Angular o React que permiten una implementación mediante librerías JS/TS* - *Para la aplicación móvil, se propone el uso de Flutter o Reac Native como alternativa que corren en Android, iOS principalmente, que si bien es cierto no son lenguajes nativos ofrecen un excelente rendimiento.*

Autor: David Córdova Fecha: 02-Sept-2024

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.

Justificación
- *A nivel de seguridad, las aplicaciones pueden utilizar alguno de los flujos provistos por **OAuth 2.0**, en el caso de SPA y banca móvil, se puede utilizar:* *Implicit Grant, esta opción es un tanto insegura porque expone el token de acceso* *Authorization Code Grant with PKCE, basado en **RFC7636**, esta opción es la recomendada y utiliza básicamente un esquema de llave pública y privada que es utilizada por la aplicación SPA*

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.

Justificación
- *Se plantea el uso de FacePhi como un producto que permite el manejo del proceso de Onboarding de manera segura y la integración mediante APIs*

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

¡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!

Consideraciones.

A) 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.

Justificación
- ***Pagos*** *Tener en cuenta la implementación de tramas que cumplan con el estándar **ISO20022** para el intercambio de mensajes y que maneja una estructura XML, en lugar de la norma anterior **ISO8583***
- ***Aplicativo SPA*** *Tener en cuenta el uso de **OWASP** para validación de data que se ingresa para las distintas transacciones financieras, además de otras validaciones (inyección de código, XSS), uso de https para el aplicativo web, por mencionar algunas*
- ***Paso a históricos*** *Para fines regulatorios, tener en cuenta mantener un histórico por 7 años de los registros financieros realizados por los clientes, en caso de reclamos, se pueda justificar, atender y solucionar en base a la información guardada, para eso se tiene que tener en cuenta mover la información a una base histórica cada cierto tiempo desde las bases principales, y luego de pasado ese período es posible el paso a cintas.*
- ***Seguridad Data en Tránsito y en reposo*** *En el proceso de on boarding para los clientes nuevos que se registran en la banca móvil se propone el uso de FacePhi , la información capturada de imágenes del rostro del cliente o de sus documentos por ejemplo cédula de identidad, se puede por ejemplo considerar las normas **NIST IR 8034 , NIST SP-800-122*** *Se propone el uso de **https** para la data en transito* *Para la data en reposo que se considera sensible como números de cuenta, tarjeta, es posible implementar algún enmascaramiento o encriptación usando estándares aprobados por el cliente*

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

Justificación
- ***Alta Disponibildad (HA)*** *Para alta disponibilidad **(HA)** de algunos de los componentes de la solución, específicamente para el sitio web (SPA), se puede incluir, balanceo en las APIs que consume a través de **AWS API Gateway**, incluir servicios como **AWS Route53***
- ***Alta Disponibilidad (HA) Backend y tolerancia a fallos*** *Para el backend se puede hacer uso de Kubernetes con el servicio **AWS ECS** configurado como un clúster tipo **AWS Fargate**, que nos permita un manejo de auto escalamiento y un manejo adecuado en caso de errores, en el caso de AWS Fargate es gestionado por Amazon.* *Adicional en frente de AWS ECS colocamos un Balanceador de Aplicación*
- ***Réplica de Base de Datos*** *En lo que tiene que ver con el Core Bancario, se puede tener una réplica de la base de datos principal x ejemplo contando con una base de datos noche y otra del día.* *Para los nodos de aplicación del Core Bancario (Bancs), se puede pensar en una configuración tipo clúster activo, pasivo.*
- ***Tolerancia a Fallos*** *Tolerancia a fallos, se puede pensar en aplicar para los microservicios un patrón tipo **Circuit Breaker, Retry,** para no bloquear la aplicación que los consume, por ejemplo, puede ser útil en una transferencia o en un pago, que el cliente puede querer realizar varios intentos, a pesar de no tener una respuesta inicial, para eso se puede manejar **Resilience4J** para SpringBoot o dentro de AWS StepFunctions, configurar reintentos o el mismo Circuit Breaker* Se propone además el uso de una cola de mensajes a través de **AWS SQS** que nos permite incluir resiliencia en el uso de mensajes entre los componentes y para mensajería **AWS SNS**
- ***Disaster Recovery*** *Para el DR, se puede considerar una solución DCA que se active cuando el data center principal haya tenido problemas en su funcionamiento, nos permite levantar los*

servidores con data consistente, y con sus IP's originales en la mayoría de los casos y con herramientas que permiten copiar la data de base de datos entre los datacenter

c) 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.

Justificación
Para la resolución del reto, y con miras a tener alta disponibilidad, tolerancia a fallos, , auto scaling, en varias AZ en una o varias regiones dependiendo el tipo de carga que se tenga y dependiendo del tipo de servicio de AWS *Existen igualmente servicios que son gestionados por completo por AWS, lo que nos permite cumplir con el requerimiento* *A continuación, un listado de los servicios propuestos para la solución:* *-**AWS Route53**: Para rutear las conexiones desde las aplicaciones a las APIs* *-**AWS API Gateway:** Para gestion de las APIs requeridas en la solucion* *-**AWS Step Functions:** Como una máquina de estados que gestiona las transacciones financieras* *-**AWS Lambda Functions:** Para interactuar con AWS Step function y AWS SQS* *-**AWS DynamoDB** : Como base no SQL y como servicio administrado por Amazon para guardar templates para las notificaciones* *-**AWS SQS** - Para el manejo de colas tanto para la transaccionalidad como para notificaciones* *-**AWS SNS** - Para envió de notificaciones* *-**AWS ECS y AWS Fargate:** Para contenerizar los micro servicios en AWS ECS y aprovisionamiento con Fargate a cargo de AWS* *-**AWS Cloud Watch, Cloud Trail, X-Ray**: Para visualizar rendimiento, eventos y trazabilidad en el consumo de las APIs respectivamente* *-**AWS Direct Connect**: Para conectar los servicios de la nube de AWS con los componentes On Premise* ***-AWS Application Load Balancer*** *Otros Componentes Complementarios VPC, NAT, EndPoints ,*

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

Justificación

- *Se puede asumir por ejemplo que el microservicio de transferencias, sea reutilizable por los aplicativos web y móvil, considerando identificar los canales o aplicaciones desde donde se originan las transacciones y enviar esos valores en los campos de entrada o header del micro.*

- *Para el caso de Lambda function a pesar que visualmente en el diagrama se propone en la transferencia directa, interbancaria y en pagos, el uso de lambda para notificaciones, por ejemplo, las mismas son reutilizables.*

- *En la solución se utilizan servicios que cumplan con el criterio de responsabilidad única, y se busca bajo acoplamiento en las integraciones propuestas*

Modelo C4

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.

En este caso se utiliza como herramienta draw.io para generar los diagramas

Diagrama de Contexto

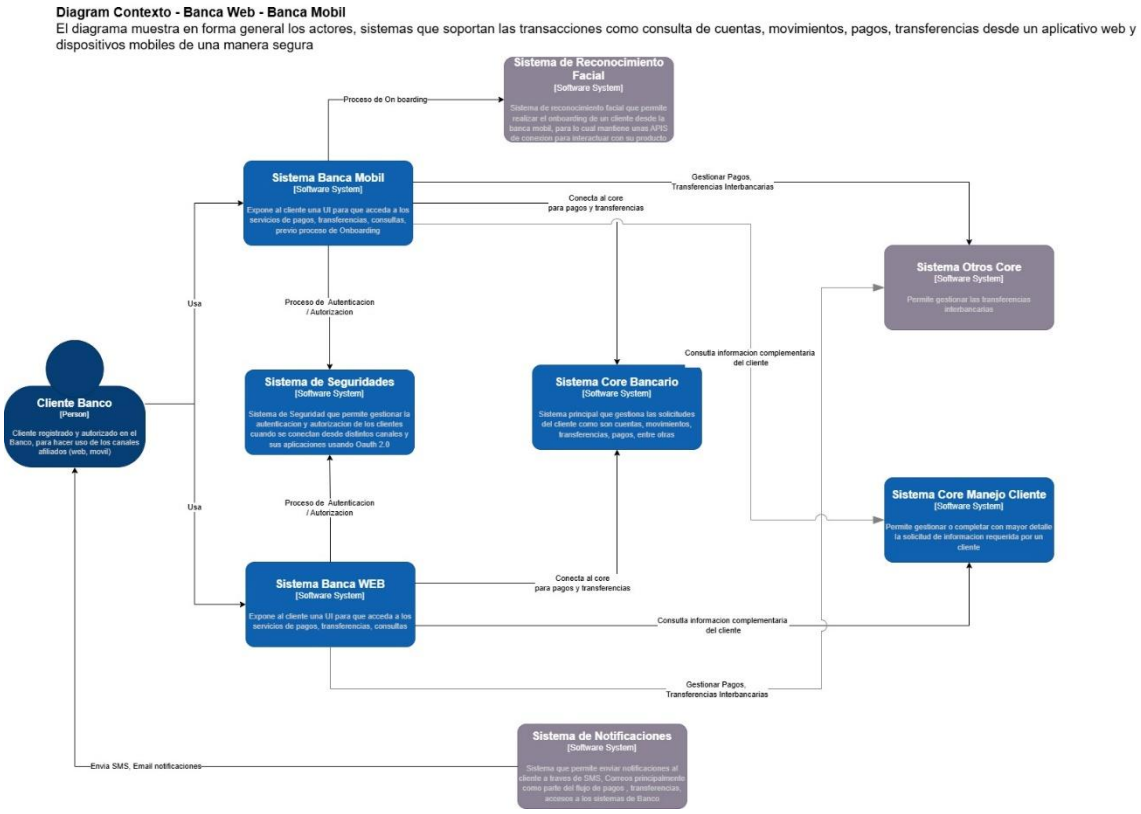
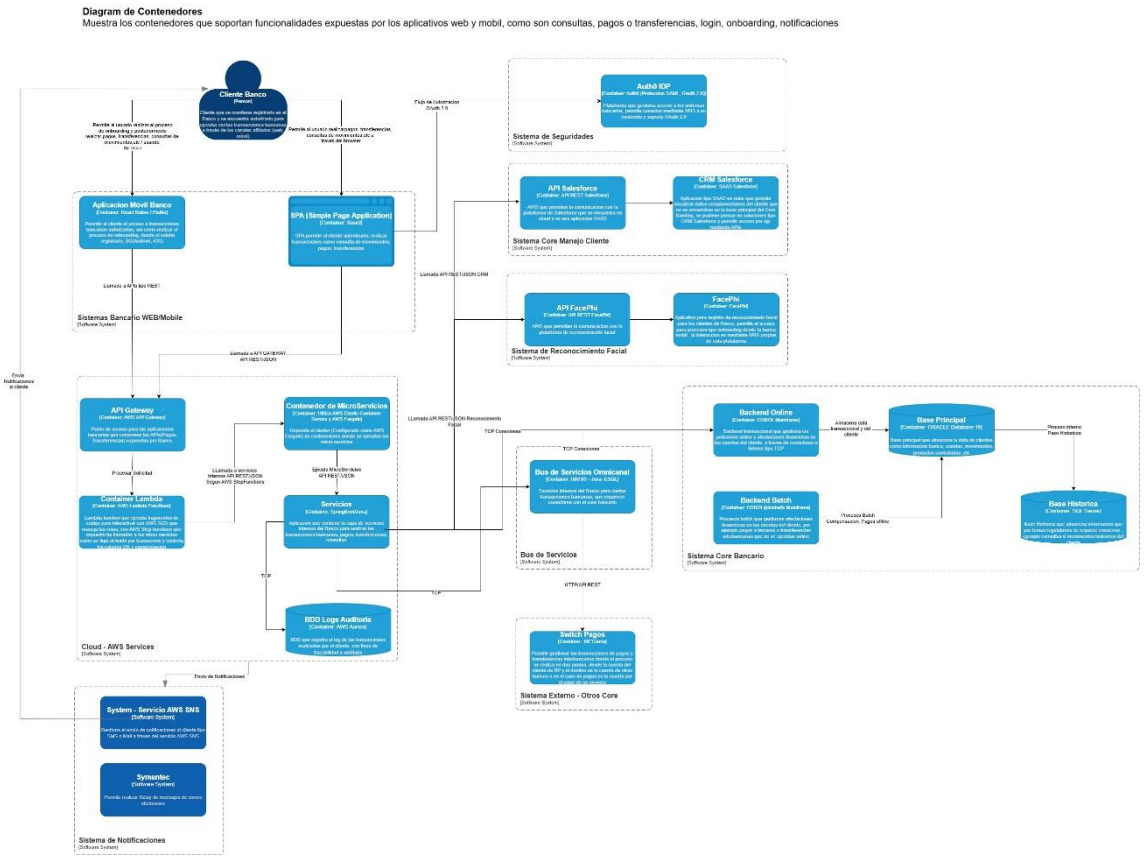


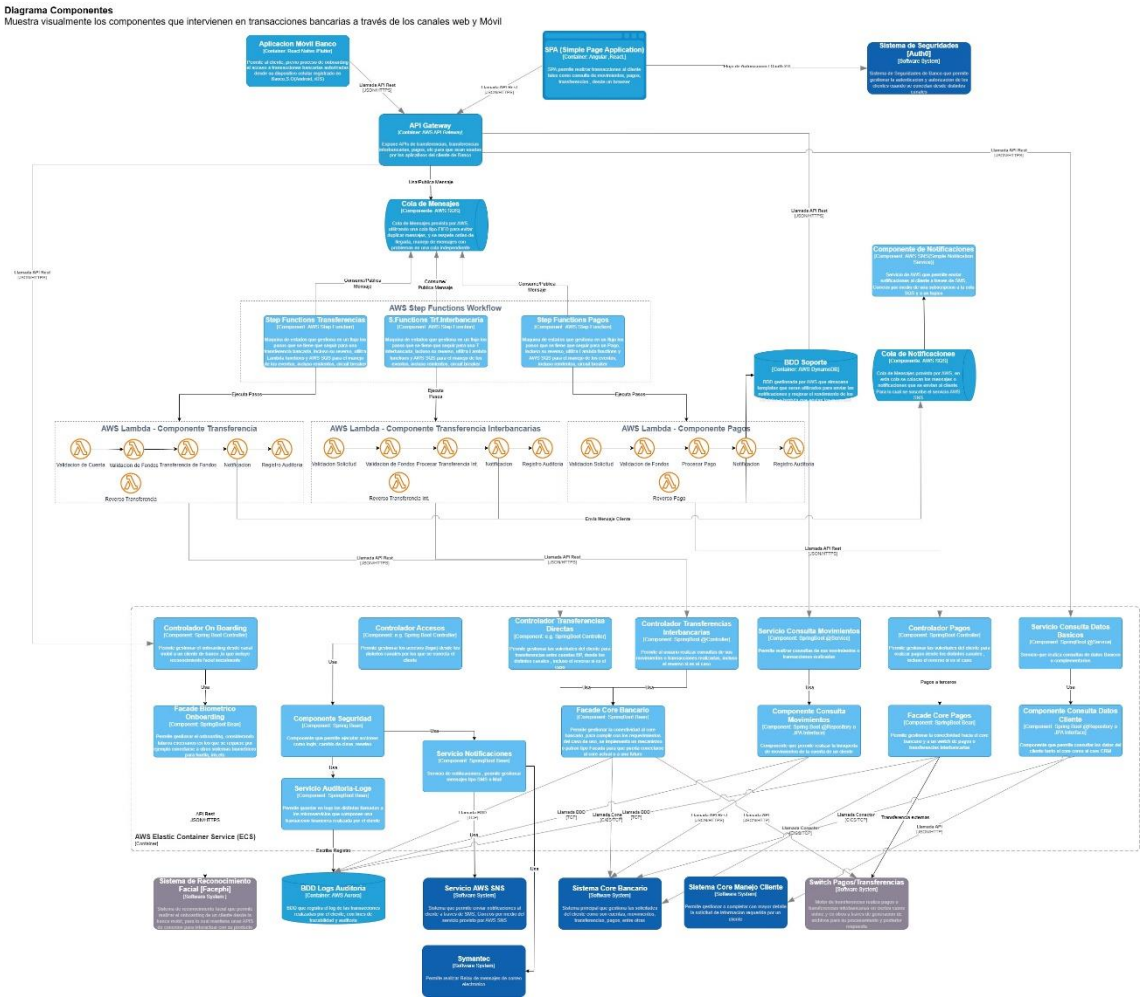
Diagrama de Contenedores



Autor: David Córdova Fecha: 02-Sept-2024

Diagrama de Componentes

Se propone una Arquitectura de Microservicios, y con un enfoque en el uso de servicios en la nube de AWS, tanto gestionados por la organización, como los que son gestionados directamente por AWS



Patrones de Arquitectura:

- Patrón Gateway gestionado por **AWS API Gateway** quien recibe las distintas peticiones de los canales a las API´s que se exponen y rutea internamente a los micro servicios correspondientes
- Aprovechando que se incluye el uso de AWS Step Function, AWS Lambda, AWS SQS, se plantea el manejo de un patrón **SAGA** para las transacciones financieras, lo que nos permite reaccionar proactivamente y ejecutar transacciones de compensación en caso de error en las transacciones financieras.
- Patrones **Circuit Breaker** y **Reintentos (Retry)** gestionado desde AWS Step Function y desde los microservicios en SpringBoot con Resilience4J

- Se podría sugerir el uso del patrón **CQRS** para la consulta de movimientos aprovechando que tenemos una base de data histórica, que toma como origen la base transaccional del Core banking

**Diagrama de Infraestructura*

El diagrama de Infraestructura (no pertenece al modelo C4) muestra un modelo *hibrido (AWS y On Premise)*