SOLUTION ARCHITECT BP -DEVSU

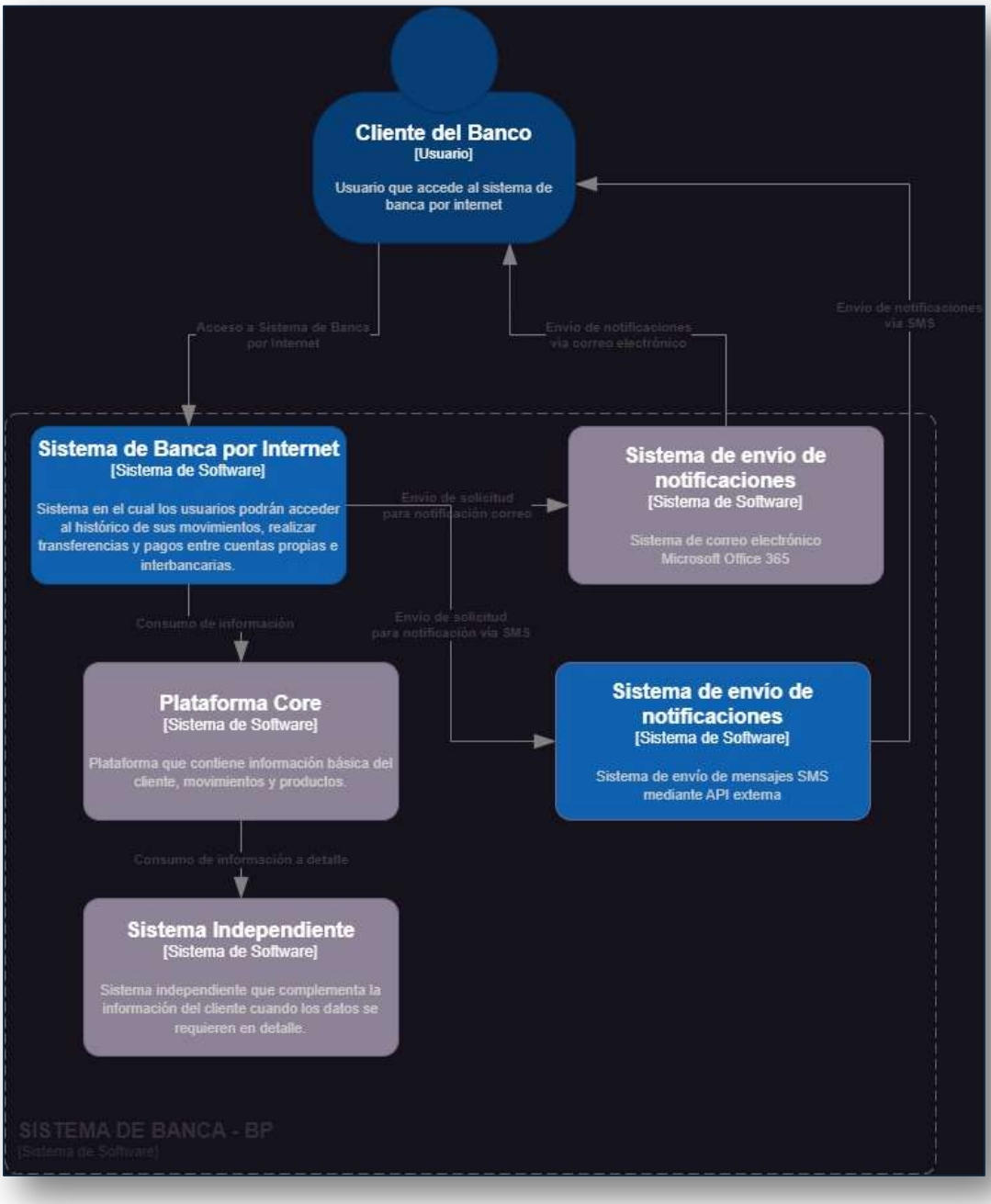
Contenido

Alejandro Morocho alejandro.morocho@gmail.com

1. DISEÑO DE SISTEMA DE BANCA POR INTERNET – MODELO C4

De acuerdo con la descripción del ejercicio práctico, y conforme el rol de arquitecto de soluciones contratado por la entidad llamada BP, se requiere presentar el diseño de 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; a continuación se muestra la estructura desarrollado de acuerdo al modelo C4

1.1 NIVEL DE CONTEXTO



De acuerdo con lo incluido en el Modelo de Contexto, se puede identificar que se representa al sistema de banca por internet como un único sistema que interactúa con diversos actores y sistemas externos; en lo que corresponde a una descripción general se muestra a continuación, el detalle correspondiente:

Usuarios: Acceden a la banca a través del Sistema de Banca por Internet

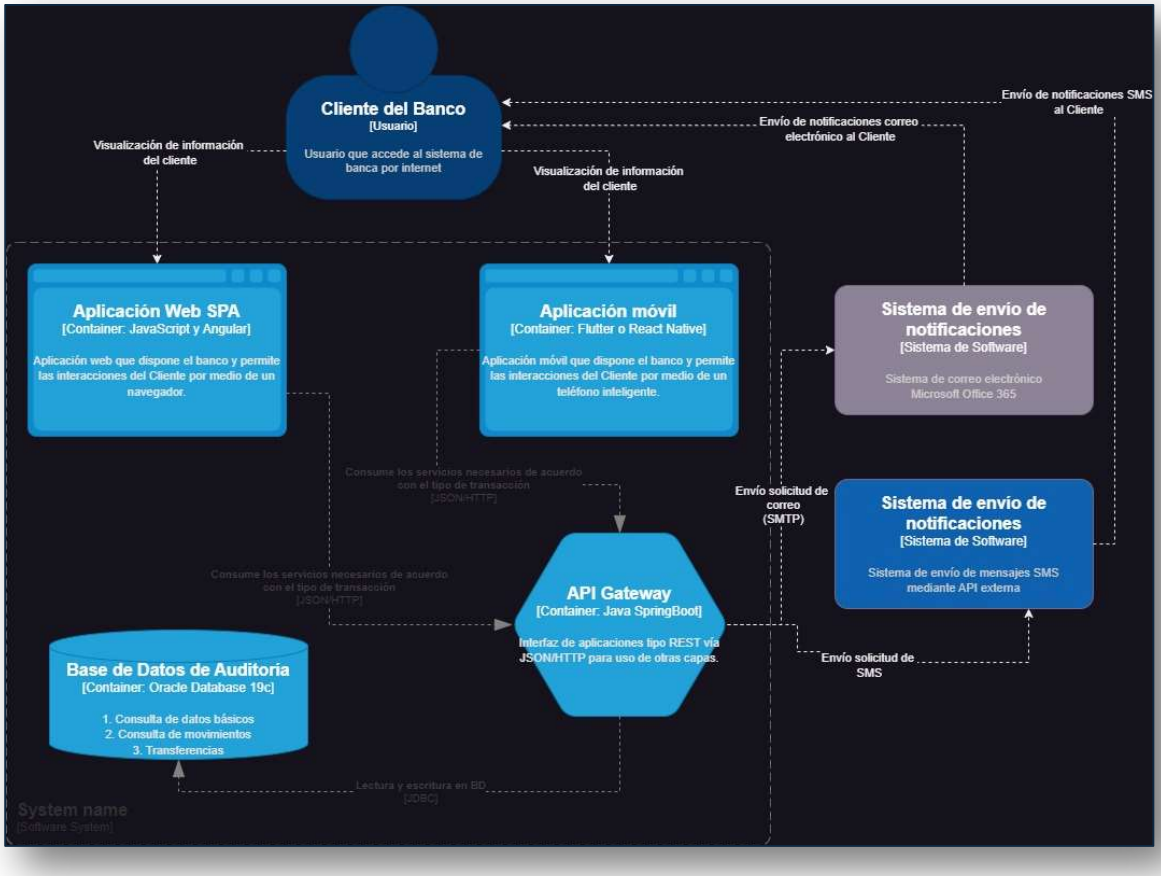
Sistema de Banca por Internet: Permite a los usuarios acceder al histórico de movimientos, realizar transferencias y pagos interbancarios o entre cuentas propias.

Plataforma Core: Proporciona los datos principales del cliente (movimientos, productos, cuentas).

Sistema Independiente: Proporciona información adicional del cliente cuando se requieren más detalles.

Servicios de Notificaciones: Se utilizan para enviar notificaciones de transacciones mediante, al menos, dos proveedores externos o internos.

1.2 NIVEL DE CONTENEDORES



Con relación al detalle correspondiente al Modelo de Contenedores se describe a continuación los principales elementos de software del sistema y su interacción:

Contenedores del sistema:
Aplicación Web SPA (Single Page Application):
Aplicación web responsiva desarrollada en framework Angular o también puede utilizarse React.
Funcionalidades: Acceso al historial de movimientos, transferencias y pagos.
Autenticación: OAuth 2.0 para gestionar sesiones y autenticación.
Justificación de uso: React y Angular son ideales para SPA por su rendimiento y la capacidad de crear interfaces dinámicas. React ofrece mayor simplicidad y un ecosistema más maduro, mientras que Angular ofrece herramientas integradas más robustas para proyectos grandes.

Aplicación Móvil (Framework Multiplataforma):
Aplicación móvil multiplataforma desarrollada en Flutter o React Native.
Funcionalidades: Incluye las mismas características que la SPA, con integración de reconocimiento facial para el onboarding.
Autenticación: OAuth 2.0 para acceso seguro.
Justificación de la tecnología: Flutter ofrece excelente rendimiento y personalización de UI, mientras que React Native es más popular y tiene una curva de aprendizaje más sencilla para desarrolladores web.

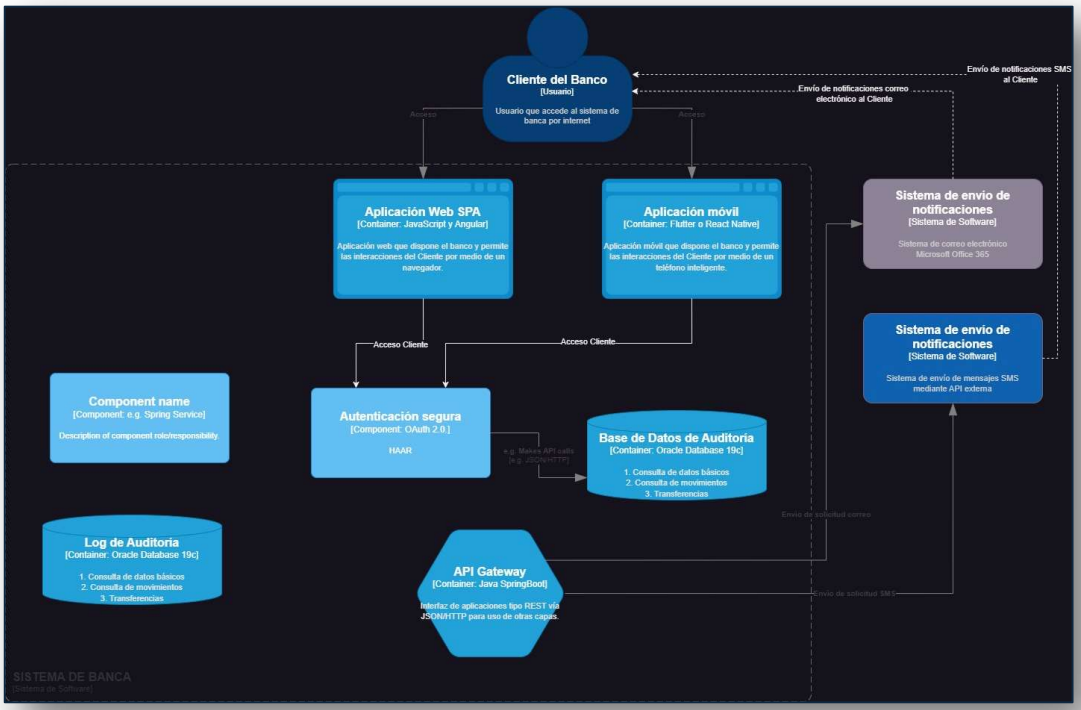
API Gateway:
Descripción: Este es el punto de entrada para todas las solicitudes al sistema. Gestiona la autenticación y la autorización mediante OAuth 2.0 y enruta las solicitudes a los servicios backend correspondientes.
Funcionalidades: Enrutamiento de solicitudes HTTP y gestión de tokens OAuth.

Servicio de Autenticación (OAuth 2.0):
Descripción: Gestiona la autenticación de usuarios utilizando el flujo Authorization Code Flow de OAuth 2.0, asegurando tokens de acceso y refresco.
Funcionalidades: Verifica la autenticidad de los usuarios antes de permitirles acceder a las funcionalidades del sistema.

Sistema de Notificaciones:
Descripción: Gestiona las notificaciones enviadas a los usuarios sobre movimientos y transacciones mediante al menos dos proveedores de servicios (internos o externos).
Tecnologías: Microsoft 365, Amazon SNS, SendGrid para enviar SMS, correos electrónicos, o notificaciones push.

Base de Datos de Auditoría:
Descripción: Registra todas las acciones de los usuarios para cumplimiento normativo y análisis de comportamiento.
Tecnologías: Oracle 19c.

1.3 NIVEL DE COMPONENTES



Con relación al detalle correspondiente al Modelo de Componentes se describe a continuación los principales elementos de software del sistema y su interacción:

Contenedor "SPA":

a. Componente de Autenticación:
   - Gestiona el flujo OAuth 2.0 para obtener y renovar tokens de acceso.

b. Componente de Movimientos:
   - Realiza las solicitudes al microservicio de movimientos a través del API Gateway.

c. Componente de Transferencias:
   - Maneja la interfaz para realizar transferencias entre cuentas propias e interbancarias.

d. Componente de Notificaciones:
   - Gestiona la recepción y visualización de notificaciones de movimientos.

Contenedor "Aplicación Móvil":

a) Componente de Onboarding:
   - Integra el reconocimiento facial usando servicios como Amazon Rekognition o Azure Face API.

b) Componente de Autenticación:
   - Gestiona el acceso a la app mediante OAuth 2.0, integrando métodos como PIN, contraseña o huella dactilar.

CONSIDERACIONES PARA DISEÑO

1. DETALLE FLUJO DE AUTENTICACIÓN

De acuerdo con la necesidad de recomendación sobre cuál es el mejor flujo de autenticación que se debería utilizar según el estándar Oauth 2.0, se menciona que será considerado el flujo Authorization Code Flow.

Es importante señalar que este flujo brinda seguridad para aplicaciones web y aplicaciones móviles que pueden gestionar un servidor backend seguro, el funcionamiento del mismo se detalla a continuación:

1. El cliente (aplicación) redirige al usuario al servidor de autorización (autorización del proveedor) con una solicitud para obtener un código de autorización.
2. El usuario ingresa sus credenciales en el servidor de autorización y concede el permiso a la aplicación.
3. El servidor de autorización redirige al cliente con un código de autorización temporal.
4. El cliente intercambia el código de autorización por un token de acceso (y opcionalmente un token de actualización) enviando el código junto con las credenciales del cliente (ID y secreto) al servidor de autorización.
5. El servidor de autorización responde con un token de acceso que puede ser utilizado para acceder a recursos protegidos en nombre del usuario.

Las ventajas consideradas para la utilización del flujo Authorization Code Flow son:

- **Alta seguridad:** Las credenciales del usuario no son expuestas directamente al cliente; en cambio, se utiliza un código de autorización temporal.
- **Protección contra ataques como el MITM (man-in-the-middle):** Se recomienda utilizar este flujo junto con PKCE (Proof Key for Code Exchange) para aplicaciones móviles o SPAs (Single Page Applications) que no pueden almacenar de forma segura un secreto de cliente.

- **Actualización de token:** Permite la renovación de tokens a través de un token de actualización, evitando la necesidad de reautenticación constante.

2. RECONOCIMIENTO FACIAL – INGRESO AL SISTEMA

A partir del Onboarding y para que el nuevo usuario pueda ingresar al sistema mediante usuario y clave, huella o algún otro método, se especificará dentro de la arquitectura, el uso de FaceSDK; la cual es una aplicación desarrollada por Luxand y diseñada para que los programadores puedan construir aplicaciones web con reconocimiento facial e identificación biométrica.

Face SDK es un conjunto de componentes de software (bibliotecas de códigos) para el desarrollo de soluciones de reconocimiento facial de cualquier escala, desde una simple aplicación móvil (Android, iOS) hasta una solución de servidor profesional. Permite la detección y seguimiento de rostros, además de la estimación de atributos faciales (género, edad, emociones) y ajuste de puntos de referencia faciales.

3. PATRONES DE DISEÑO – PERSISTENCIA DE INFORMACIÓN

Para la implementación de la base de datos de auditoría, además de un mecanismo de persistencia de información para clientes frecuentes dentro del sistema propuesto, una alternativa basada en patrones de diseño que se puede utilizar es: la combinación de los patrones Event Sourcing y CQRS (Command Query Responsibility Segregation).

El patrón **Event Sourcing** se centra en capturar todas las modificaciones en los datos del sistema como una secuencia de eventos. En lugar de almacenar el estado actual de los datos, se almacena una serie de eventos que representan cada cambio que ha ocurrido en el sistema.

Método de aplicación

- Registro de acciones del cliente: Cada acción del cliente se captura y almacena como un evento en la base de datos de auditoría. Por ejemplo, una transferencia de fondos, una consulta de saldo, o un inicio de sesión se registran como eventos individuales.

- Reconstrucción del estado: El estado actual de un cliente o de sus transacciones puede ser reconstruido en cualquier momento al reproducir los eventos desde el comienzo hasta el presente.
- Auditoría detallada: Debido a que se almacena cada evento, se tiene un historial completo y detallado de todas las acciones, lo que es esencial para auditorías.

Ventajas:

- Historial completo: Cada cambio en el sistema está registrado, lo que permite una auditoría completa.
- Reversibilidad: Es posible deshacer acciones aplicando los eventos en orden inverso.
- Integridad y trazabilidad: Se puede rastrear cómo se ha llegado a un estado específico, proporcionando trazabilidad total.

El patrón CQRS (Command Query Responsibility Segregation) separa las operaciones de lectura (query) y escritura (command) en diferentes modelos, optimizando cada uno para su función específica.

Método de aplicación

- Módulo de comandos: Este módulo maneja las operaciones que modifican el estado del sistema, como realizar transferencias, pagos, o cualquier acción del cliente. Cada comando genera un evento que es almacenado en la base de datos de auditoría (implementación de Event Sourcing).
- Módulo de consultas: Este módulo se encarga de las operaciones de lectura. Debido a la naturaleza de Event Sourcing, las consultas pueden reconstruir el estado del sistema o del cliente a partir de los eventos, o consultar directamente vistas materializadas (pre-calculadas) que se mantienen en una base de datos optimizada para lecturas rápidas.

Ventajas:

- Escalabilidad: La separación entre lectura y escritura permite escalar cada uno de manera independiente.
- Optimización: Las consultas se pueden optimizar usando bases de datos especializadas para lecturas, como bases de datos NoSQL o vistas materializadas.

- Flexibilidad: Facilita la adaptación a requerimientos cambiantes, ya que los modelos de lectura y escritura pueden evolucionar de manera independiente.

Método de integración de los componentes

1. Base de Datos de Eventos (Event Store):
   - Actúa como la base de datos principal para almacenar todos los eventos generados por las acciones de los clientes.
2. Command Handlers:
   - Procesan los comandos y generan eventos que se almacenan en el Event Store.
3. Projections (Vistas Materializadas):
   - Utilizadas para construir representaciones actuales de los datos (como el estado de la cuenta del cliente o su historial de transacciones) a partir de los eventos, permitiendo consultas rápidas y eficientes.
4. Audit Log:
   - Un sistema de logs que puede ser construido como una proyección especializada de los eventos, proporcionando un registro de auditoría de todas las acciones del cliente.
5. Caching Layer (para clientes frecuentes):
   - Se puede implementar un mecanismo de cacheo para los clientes frecuentes, utilizando los eventos almacenados para mantener actualizada la información relevante en memoria o en un datastore optimizado para acceso rápido.

La combinación de Event Sourcing y CQRS es una alternativa robusta para la implementación del sistema de auditoría y persistencia de información en el sistema implementado. Este enfoque no solo permite capturar y auditar todas las acciones del cliente, sino que también optimiza la escalabilidad, la integridad de los datos y la capacidad de respuesta del sistema.

4. ELEMENTOS NORMATIVOS

Elementos normativos

Conforme la necesidad de implementación de un sistema de banca por internet en nuestro país, es fundamental considerar diversas normativas y regulaciones para garantizar la conformidad legal, la seguridad, y la protección de los datos de los usuarios. A continuación, se detallan las principales consideraciones normativas y regulatorias:

a) Ley Orgánica de Protección de Datos Personales (LOPDP)

- Consentimiento del Usuario: Los bancos deben obtener el consentimiento explícito de los usuarios para el tratamiento de sus datos personales.
- Derecho de los Titulares de los Datos: Se debe garantizar que los usuarios puedan ejercer sus derechos de acceso, rectificación, cancelación, oposición, portabilidad, y supresión de sus datos personales.
- Protección de Datos Sensibles: Los datos financieros y otros datos sensibles deben ser tratados con medidas de seguridad adicionales.
- Notificación de Brechas de Seguridad: En caso de una violación de la seguridad que afecte los datos personales, las instituciones financieras están obligadas a notificar a la Superintendencia de Bancos y a los afectados.

b) Ley Orgánica de la Economía Popular y Solidaria y del Sector Financiero Popular y Solidario

- Autorización de la Superintendencia de Bancos: La implementación de servicios financieros electrónicos, incluyendo la banca por internet, requiere la autorización de la Superintendencia de Bancos.
- Transparencia en las Operaciones: Las instituciones financieras deben garantizar la transparencia en las operaciones, proporcionando información clara sobre las condiciones y términos de los productos y servicios ofrecidos en línea.

c) Normativa de la Superintendencia de Bancos del Ecuador

- Regulación de Canales Electrónicos: Existen regulaciones específicas que establecen los estándares de seguridad y operatividad para los canales electrónicos, incluyendo la banca por internet.
- Gestión de Riesgos Tecnológicos: Las instituciones financieras deben implementar un marco de gestión de riesgos tecnológicos que incluya la identificación, evaluación, y mitigación de riesgos asociados a la banca por internet.
- Plan de Contingencia y Continuidad del Negocio: Se requiere un plan de contingencia que garantice la continuidad de los servicios financieros electrónicos en caso de fallos técnicos o desastres.

d) Reglamento de Gestión de Seguridad de la Información

- Cifrado de Datos: El uso de cifrado para la transmisión y almacenamiento de datos es obligatorio para proteger la información sensible de los usuarios.
- Autenticación Fuerte: Se deben implementar mecanismos de autenticación robustos, como el uso de contraseñas fuertes, autenticación multifactor (MFA), y biometría (ejemplo: reconocimiento facial).
- Monitoreo y Detección de Incidentes: El reglamento exige un monitoreo continuo de los sistemas para detectar y responder a incidentes de seguridad en tiempo real.

e) Normativa de Prevención de Lavado de Activos y Financiamiento del Terrorismo

- Conozca a Su Cliente (KYC): Los bancos deben implementar procesos de verificación de identidad y monitoreo de transacciones para prevenir el lavado de activos y el financiamiento del terrorismo.
- Reportes de Operaciones Sospechosas (ROS): Las instituciones financieras deben reportar transacciones sospechosas a la Unidad de Análisis Financiero y Económico (UAFE).

f) Normas Internacionales

- Normas PCI-DSS: Para el manejo seguro de datos de tarjetas de crédito y débito en transacciones en línea.
- ISO/IEC 27001: Para la gestión de seguridad de la información.
- RGPD (si corresponde): Aplicación de ciertas normativas internacionales de protección de datos si el sistema involucra a ciudadanos de la UE.

g) Normativa sobre Firma Electrónica

- Ley de Comercio Electrónico, Firmas Electrónicas y Mensajes de Datos: En Ecuador, la firma electrónica tiene el mismo valor legal que una firma manuscrita y debe ser implementada en los sistemas de banca por internet para la validación de transacciones importantes.

h) Regulación de Comercio Electrónico

- Protección al Consumidor: Las regulaciones sobre comercio electrónico deben ser observadas, especialmente en lo referente a la protección de los derechos del consumidor en transacciones online.

i) Requerimientos de Interoperabilidad

- Interoperabilidad Bancaria: Los sistemas de banca por internet deben cumplir con los estándares de interoperabilidad establecidos por la Superintendencia de Bancos y otras autoridades para facilitar transacciones interbancarias seguras y eficientes.

j) Consideraciones de Localización de Datos

- Almacenamiento de Datos en Territorio Nacional: Dependiendo de las regulaciones, podría ser obligatorio que ciertos datos financieros o personales se almacenen en servidores ubicados dentro del territorio ecuatoriano.

Conforme todo lo señalado respecto a normativas y regulaciones es esencial su análisis y aplicación para evitar sanciones y garantizar la seguridad y confianza de los usuarios en el sistema de banca por internet. Es importante señalar la necesidad de que las instituciones financieras realicen auditorías de conformidad periódicas y mantengan un diálogo continuo con las autoridades regulatorias.

5. ARQUITECTURA PLANTEADA

Las consideraciones de Arquitectura que se han establecido son las siguientes:

- Alta disponibilidad (HA): Despliegue en múltiples zonas de disponibilidad.
- Tolerancia a fallos y Recuperación de Desastres (DR): Réplicas de datos y backups automáticos.
- Seguridad: Uso de OAuth 2.0, cifrado de datos en tránsito y reposo.
- Monitoreo: Uso de herramientas como AWS CloudWatch o Azure Monitor para monitorear la infraestructura y los servicios.
- Seguridad y monitoreo: Uso de herramientas como AWS CloudWatch, Azure Monitor, o Datadog para la monitorización y CloudTrail para auditoría.
- Auto-healing: Implementación de estrategias de recuperación automática ante fallos de servicios críticos.
- Baja latencia: Despliegue de servicios en regiones geográficas cercanas a los clientes.

6. COSTOS APROXIMADOS

6.1 Infraestructura en la Nube (Azure o AWS)

Almacenamiento: Se necesitarán bases de datos para auditar acciones del cliente, persistir información y soportar transacciones. Los costos pueden variar según el tipo de base de datos (Oracle para nuestro caso) y el almacenamiento necesario.

Costos estimados:

- RDS o Azure SQL: Dependerá del tamaño de la base de datos (referencia $0.10/GB/mes para almacenamiento básico).

Almacenamiento para auditoría: Servicios como Amazon S3 o Azure Blob Storage suelen ser más económicos para datos no críticos.

Cómputo: Los servidores que ejecutan las aplicaciones backend y el API Gateway tendrán costos variables dependiendo de su capacidad de procesamiento y el tráfico.

Costos estimados:

- EC2 en AWS o Azure VMs: A partir de $0.02/hora para instancias básicas, pero puede incrementarse según los requisitos de escalabilidad y alta disponibilidad.

Considera un auto-escalado que permite ajustar dinámicamente el número de instancias según el tráfico.

Bases de Datos de Auditoría:

Servicios gestionados como Amazon DynamoDB o Cosmos DB en Azure son opciones ideales para registrar todas las acciones del cliente con alta escalabilidad. Los precios varían según la cantidad de lecturas/escrituras por segundo (ejemplo $1.25 por millón de lecturas en DynamoDB).

API Gateway: Un componente crítico para enrutar las solicitudes de los usuarios a los servicios adecuados.

Costos estimados:

- AWS: $3.50 por millón de solicitudes.

- Azure API Management: Desde $0.03 por 10,000 transacciones.

6.2 Costos de Notificaciones

Servicios externos de notificaciones: Es necesario contar con al menos dos servicios para enviar notificaciones por correo electrónico, SMS, o push notifications.

Costos estimados:

- Amazon SNS o Azure Notification Hubs: $0.50 por millón de notificaciones push.
- Twilio o SendGrid: Para SMS o correos electrónicos, el costo puede variar según el volumen (e.g., $0.0075 por SMS en Twilio o $0.10 por 1,000 correos en SendGrid).

Es importante considerar que si la infraestructura del cliente dispone de Microsoft 365, se omitiría el rubro definido por notificaciones de correo.

6.3 Desarrollo de Aplicaciones Frontend (SPA y Móvil)

SPA (Single Page Application): El desarrollo de una aplicación web de página única incluye el costo de programadores web con experiencia en frameworks como React o Angular.

Costos de desarrollo:

- Tiempo estimado: 3-6 meses.
- Costos promedio: $40,000 - $60,000 dependiendo de la complejidad.

Aplicación Móvil (Flutter o React Native): El costo de desarrollar una aplicación móvil multiplataforma incluye el diseño de interfaces de usuario, integración con APIs y optimización para múltiples dispositivos.

Costos de desarrollo:

- Tiempo estimado: 4-8 meses.
- Costos promedio: $50,000 - $80,000 para una aplicación de banca completa.

Mantenimiento: Los costos de actualización y soporte también deben considerarse a largo plazo.

6.4 Costos de Seguridad y Autenticación

OAuth 2.0 y Onboarding Biométrico: Implementar un sistema de autenticación robusto basado en OAuth 2.0 y utilizar reconocimiento facial implica costos adicionales.

Autenticación (Auth0, Okta, Amazon Cognito):

- Auth0 y Okta ofrecen planes desde $23/mes por cada 1,000 usuarios activos.
- Amazon Cognito: Los primeros 50,000 usuarios activos son gratuitos, luego $0.0055 por usuario mensual.

Servicios de reconocimiento facial:

- AWS Rekognition o Azure Face API: Costos entre $1 y $3 por cada 1,000 solicitudes de reconocimiento facial.

6.5 Monitoreo y Seguridad

Herramientas de monitoreo: Se requiere monitorear la infraestructura, la disponibilidad de los servicios y posibles amenazas de seguridad.

Costos estimados:

- AWS CloudWatch o Azure Monitor: Desde $0.30 por GB de datos monitoreados.
- Seguridad (firewall, DDoS protection, etc.): AWS Shield ($3,000/mes para Shield Advanced) o Azure DDoS Protection ($2,944/mes).

6.6 Alta Disponibilidad (HA) y Recuperación ante Desastres (DR)

Zonas de Disponibilidad: Para garantizar HA, es necesario desplegar la infraestructura en múltiples zonas geográficas.

Costos estimados:

- Costos adicionales por replicación de bases de datos y servidores en zonas adicionales pueden incrementar un 20-30% el costo total de la infraestructura.

Backup y recuperación:

- Amazon RDS Backup o Azure Backup: Entre $0.02 y $0.10 por GB almacenado mensualmente.

6.7 Licenciamiento y Herramientas de Desarrollo

Herramientas de desarrollo: considerando el uso de herramientas de terceros, como GitHub o Jira, para gestionar el proyecto, es necesario considerar suscripciones.

Costos estimados: GitHub y Jira ofrecen planes de $7-15 por usuario/mes.

6.8 Costos de Personal y Mantenimiento

Equipo de desarrollo:

- Programadores: $30,000 - $50,000/año por programador.
- DevOps: $40,000 - $70,000/año.
- Mantenimiento: Los costos de mantenimiento y actualizaciones recurrentes de software deben calcularse en torno al 20-30% del costo de desarrollo anual.

6.9 Total Estimado del Proyecto (año 1)

- Infraestructura: $30,000 - $70,000 dependiendo de la cantidad de usuarios y uso de la nube.
- Desarrollo Frontend (SPA y móvil): $90,000 - $140,000.
- Seguridad y monitoreo: $10,000 - $30,000.
- Autenticación: $5,000 - $20,000 según el número de usuarios.
- Mantenimiento y personal: $70,000 - $120,000 anuales.
- El costo total estimado del proyecto para el primer año podría estar entre $200,000 y $350,000, con variaciones según la infraestructura, el número de usuarios, y las características específicas del sistema.