Arquitectura de Integración para la Modernización de Sistemas Bancarios Basada en BIAN

Candidato: Jaime Alfredo Bonilla Pérez

2024-08-06

EJERCICIO DE INTEGRACIÓN

Descripción

Diseñar una arquitectura de integración para la modernización de sistemas bancarios.

Escenario

Un banco tradicional busca modernizar su infraestructura tecnológica para mejorar sus servicios digitales y cumplir con nuevas tendencias de industria y regulaciones. Se requiere integrar: a. Core bancario tradicional existente, más un nuevo core bancario digital. b. Nuevo sistema de banca web y banca móvil. c. Plataforma de servicios de pago. d. APIs para servicios de terceros (Open Finance). e. Sistema de gestión de riesgos. f. Sistema de prevención de fraudes

Tareas

1. Diseñar una arquitectura de integración de alto nivel.
2. Especificar patrones de integración y tecnologías a utilizar.
3. Abordar requisitos de seguridad, cumplimiento normativo, ley orgánica de protección de datos personales.
4. Proponer estrategia para garantizar alta disponibilidad y recuperación ante desastres.
5. Proponer estrategia de integración multicore.
6. Delinear un enfoque para la gestión de identidad y acceso en todos los sistemas.
7. Diseñar una estrategia de API internas y externas, bajo estándares de industria respecto a la mensajería.
8. Proponer un modelo de gobierno de APIs y microservicios.
9. Proponer un plan de migración gradual que minimice el riesgo operativo.

Consideraciones

- Garantice alta disponibilidad (HA) y tolerancia a fallos (DR), seguridad, monitoreo.
- Sí lo considera necesario, su diseño de integración puede contener elementos de infraestructura en nube como Azure o AWS; API Manager. Debe garantizar baja latencia.

Entregables

- Diagramas C4
- El aspirante podrá entregar los insumos necesarios para justificar la propuesta.

Arquitectura de Integración para la Modernización de Sistemas Bancarios Basada en BIAN

Introducción

¿Qué es este Documento?

Este documento describe la Arquitectura de Integración para la Modernización de Sistemas Bancarios basada en el modelo BIAN (Banking Industry Architecture Network). Proporciona un marco detallado para la implementación de una infraestructura tecnológica moderna que integra sistemas core tradicionales y digitales, plataformas de banca web y móvil, servicios de pago, APIs para servicios de terceros (Open Finance), sistemas de gestión de riesgos y prevención de fraudes.

¿Para quién está Destinado?

Este documento está destinado a:

- Arquitectos de Sistemas y Software: Profesionales responsables de diseñar e implementar soluciones tecnológicas para instituciones bancarias.
- Desarrolladores y Equipos de TI: Equipos técnicos que trabajan en la integración y modernización de sistemas bancarios.
- Gerentes de Proyecto y Líderes de Equipo: Líderes que gestionan proyectos de modernización tecnológica en el sector bancario.
- Auditores y Consultores de Seguridad: Profesionales encargados de asegurar el cumplimiento normativo y la seguridad de los sistemas bancarios.

- Tomadores de Decisiones en Instituciones Bancarias: Ejecutivos y directivos que necesitan comprender la infraestructura tecnológica para tomar decisiones informadas sobre la modernización y optimización de los sistemas bancarios.

Objetivos del Documento

1. Proporcionar una Visión Detallada:
    - Ofrecer una descripción completa de la arquitectura propuesta, incluyendo componentes clave, interacciones y tecnologías utilizadas.
2. Asegurar la Integración y la Escalabilidad:
    - Describir estrategias para integrar sistemas core tradicionales y digitales, garantizando escalabilidad y flexibilidad para futuras expansiones.
3. Cumplimiento Normativo y Seguridad:
    - Detallar medidas para asegurar el cumplimiento de normativas como la Ley Orgánica de Protección de Datos Personales (LOPD) y la implementación de prácticas robustas de seguridad.
4. Garantizar Alta Disponibilidad y Recuperación ante Desastres:
    - Proponer estrategias para asegurar que los sistemas sean resilientes y capaces de recuperarse rápidamente en caso de fallos o desastres.
5. Optimizar la Gestión de Identidades y Accesos:
    - Incluir un plan para implementar un sistema de gestión de identidades centralizado (IAM), uso de Single Sign-On (SSO) y políticas de acceso basadas en roles (RBAC).
6. Definir una Estrategia de APIs Internas y Externas:
    - Presentar una estrategia clara para la implementación de APIs internas y externas, utilizando estándares modernos y garantizando eficiencia y seguridad en la comunicación.

Estructura del Documento

1. Diseño de Arquitectura de Integración de Alto Nivel:
    - Descripción de la arquitectura general y sus componentes clave.
2. Patrones de Integración y Tecnologías a Utilizar:
    - Detalles sobre los patrones de integración y las tecnologías específicas que se utilizarán.
3. Requisitos de Seguridad y Cumplimiento Normativo:
    - Medidas y prácticas para asegurar la seguridad y el cumplimiento de normativas.
4. Estrategia para Garantizar Alta Disponibilidad y Recuperación ante Desastres:

    - Estrategias y herramientas para asegurar que los sistemas sean resilientes y capaces de recuperarse rápidamente.
5. Estrategia de Integración Multicore:
    - Plan para sincronizar y gestionar múltiples sistemas core.
6. Gestión de Identidad y Acceso:
    - Plan para implementar un sistema de gestión de identidades centralizado, SSO y RBAC.
7. Estrategia de API Internas y Externas:
    - Estrategia para la implementación de APIs internas y externas, garantizando eficiencia y seguridad en la comunicación.
8. Modelo de Gobierno de APIs y Microservicios:
    - Propuesta para establecer un modelo de gobierno eficaz para APIs y microservicios, asegurando su gestión y mantenimiento adecuado.
9. Plan de Migración Gradual:
    - Plan detallado para la migración gradual de componentes, asegurando una transición suave y minimizando el riesgo operativo.
10. Conclusión:

- Resumen y conclusiones del documento, destacando los beneficios y la importancia de la arquitectura propuesta.

---

1. Diseño de Arquitectura de Integración de Alto Nivel

Diseño de Arquitectura de Integración de Alto Nivel

El propósito de la Diseño de Arquitectura de Integración de Alto Nivel es proporcionar un análisis detallado y una propuesta de implementación de una **arquitectura de integración basada en el marco BIAN** para la **modernización de los sistemas bancarios**. Este enfoque busca justificar la necesidad de actualizar la infraestructura tecnológica del banco para mejorar la eficiencia operativa, cumplir con las normativas regulatorias y habilitar la integración con nuevas tecnologías y servicios, facilitando la transformación digital del banco.

Contexto

En respuesta a los cambios en el entorno regulatorio y a las demandas de una industria cada vez más digital, un banco busca modernizar su infraestructura tecnológica. Esta modernización implica la **integración de un core bancario tradicional** con un **nuevo core digital**, además de la implementación de sistemas de **banca web y móvil**, **plataformas de pago**, **APIs para servicios de terceros** en el contexto de **Open Finance**, y **sistemas de gestión de riesgos** y **prevención de fraudes** avanzados.

1.2. Objetivos Estratégicos

Alineación con la Estrategia Empresarial

La arquitectura de integración propuesta está alineada con los objetivos estratégicos del banco, ya que:

- **Facilita la Transformación Digital**: Modernizando la infraestructura, permitiendo al banco adaptarse a las crecientes demandas de los clientes y del mercado, enfocándose en la digitalización y los servicios móviles.
- **Mejora la Experiencia del Cliente**: A través de la optimización de procesos, tiempos de respuesta más rápidos y accesibilidad en múltiples plataformas, la arquitectura propuesta mejorará significativamente la experiencia del cliente.
- **Cumple con Normativas Actuales y Futuras**: Asegura que todos los sistemas bancarios cumplen con las regulaciones vigentes en cuanto a protección de datos y seguridad, facilitando la adaptación a nuevas normativas.

Metas y Objetivos

- **Eficiencia Operativa**: Reducir los tiempos de procesamiento en al menos un 30%, mejorando la eficiencia de los sistemas clave y minimizando errores operativos.
- **Expansión de Servicios Digitales**: Aumentar la oferta de servicios digitales en un 40%, habilitando nuevas funcionalidades que amplíen el acceso de los clientes a servicios innovadores.
- **Agilidad Organizacional**: Mejorar la capacidad de respuesta del banco a cambios regulatorios y del mercado, reduciendo el tiempo de respuesta a las demandas externas en un 50%.

1.3. Análisis de la Situación Actual (AS-IS)

Evaluación de la Situación Actual

Actualmente, “Banco Soluciones Financieras S.A.” opera con un **core bancario tradicional** que limita su capacidad para adaptarse a las demandas digitales y mejorar la integración tecnológica. Los sistemas de **banca web y móvil** son poco flexibles, lo que genera una experiencia de usuario inconsistente. Además, los **sistemas de pago** no están completamente integrados, lo que impacta negativamente en la eficiencia operativa y la satisfacción del cliente.

Identificación de Problemas y Oportunidades

- **Problemas**:
  - **Fragmentación de Sistemas**: Los sistemas tradicionales y digitales no están completamente integrados, lo que genera redundancias y reduce la eficiencia.
  - **Tecnología Obsoleta**: El sistema actual carece de la flexibilidad necesaria para integrar nuevas tecnologías y cumplir con las expectativas del cliente, así como con los requisitos regulatorios emergentes.
- **Oportunidades**:
  - **Mejora en la Experiencia del Cliente**: La modernización permitirá ofrecer una experiencia de usuario personalizada y fluida, mejorando significativamente la interacción en plataformas móviles y web.
  - **Optimización de Procesos**: Se espera que la modernización tecnológica reduzca los costos operativos en un 25%, incrementando la eficiencia mediante la automatización y la simplificación de los procesos.

1.4. Propuesta de la Arquitectura Futura (TO-BE)

La **arquitectura propuesta** se centra en la integración de un **core bancario digital** con el sistema tradicional, utilizando un enfoque modular basado en **microservicios** y **APIs RESTful**, permitiendo mayor flexibilidad y escalabilidad. Los principales componentes de la arquitectura son:

1.4.1. Descripción General

La arquitectura de integración propuesta se basa en el enfoque de **microservicios** siguiendo el marco **BIAN** (Banking Industry Architecture Network). **BIAN** proporciona un conjunto estandarizado de servicios y definiciones que permiten una **interoperabilidad eficiente** entre sistemas bancarios, tanto tradicionales como digitales, facilitando la adopción de nuevos servicios financieros y garantizando que los sistemas puedan escalar

y adaptarse a futuras innovaciones. Este enfoque garantiza que los diferentes componentes bancarios puedan comunicarse de manera fluida a través de **APIs** y eventos.

- **Core Bancario Digital**: Un nuevo core que permite la integración fluida con el core tradicional, proporcionando capacidades avanzadas como la escalabilidad y la flexibilidad necesarias para soportar la transformación digital.
- **Sistema de Banca Web y Móvil**: Plataformas digitales que permiten una mejor experiencia del cliente, utilizando tecnologías modernas como **React** o **Angular** para interfaces de usuario y microservicios en el backend.
- **Plataforma de Pagos y APIs de Open Finance**: Implementación de una plataforma de pagos en tiempo real y APIs abiertas para facilitar la integración con terceros, aumentando la gama de servicios financieros disponibles para los clientes.
- **Sistemas de Gestión de Riesgos y Prevención de Fraudes**: Implementación de soluciones avanzadas basadas en inteligencia artificial para mejorar la seguridad y el cumplimiento normativo, asegurando la protección de los datos y transacciones.

La arquitectura híbrida propuesta combina infraestructura **on-premises** (en las instalaciones físicas del banco) con la **nube** (por ejemplo, **AWS**) para optimizar tanto el rendimiento como la escalabilidad. Esto permite que los datos y sistemas críticos permanezcan protegidos dentro del entorno local, mientras que los servicios que requieren alta elasticidad y disponibilidad se ejecutan en la nube, ofreciendo:

- **Resiliencia**: Alta tolerancia a fallos y capacidad de recuperación rápida.
- **Elasticidad**: Escalabilidad inmediata según la demanda.
- **Optimización de costos**: Reducción de la infraestructura física y de los costos operativos.

1.4.2. Diagrama de Contexto

Este nivel de diagrama presenta una vista de alto nivel del sistema y su interacción con usuarios y otros sistemas externos. En este caso, muestra la interacción entre el sistema bancario y las entidades externas, como los clientes, servicios de terceros y plataformas de pago.

El **Diagrama de Contexto** presenta una vista macro de cómo los diferentes sistemas interactúan entre sí y con entidades externas. Este diagrama ofrece una visión clara de la interacción del sistema bancario con sus **clientes**,

servicios de terceros (Open Finance), y plataformas de **banca móvil y web**:

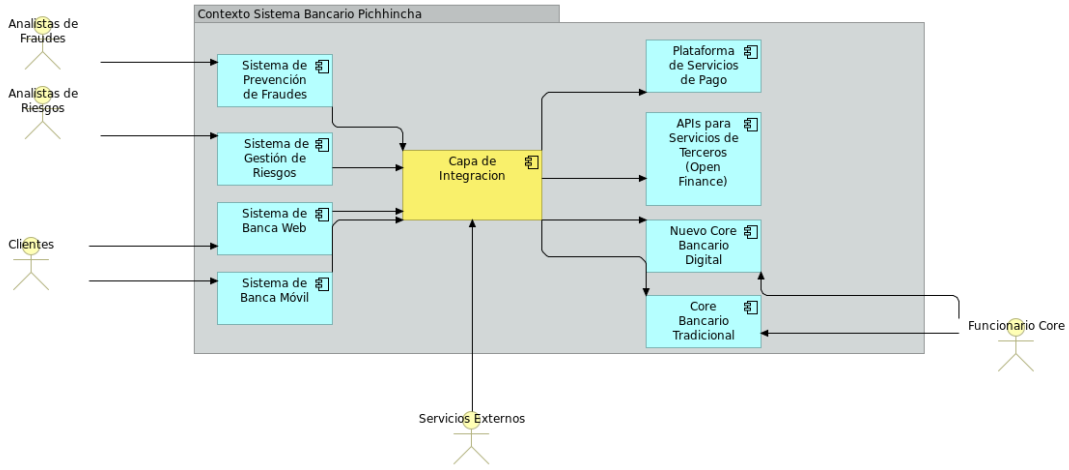

Diagrama de Contexto

Este diagrama sitúa al sistema bancario en el centro de las interacciones, asegurando que todos los flujos de información sean seguros, eficientes y cumplan con las normativas vigentes.

1.4.3. Diagrama de Contenedores

En el nivel de contenedores, se describe la estructura del sistema en términos de componentes clave, como aplicaciones, bases de datos y servicios, detallando cómo interactúan a través de interfaces y protocolos.

El **Diagrama de Contenedores** desglosa la arquitectura en **componentes principales**, detallando la relación entre las aplicaciones bancarias, bases de datos, sistemas core y servicios externos. Los contenedores clave de la arquitectura incluyen:

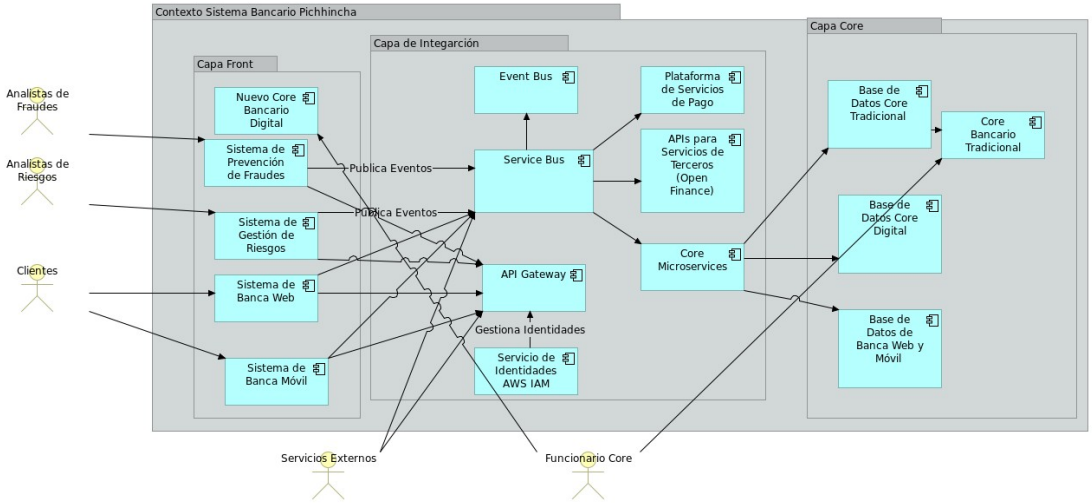

Diagrama de Contenedores

Este nivel de diseño proporciona una visión clara de cómo los diferentes módulos interactúan entre sí mediante protocolos estándar como **REST** y **gRPC** para garantizar comunicaciones rápidas y seguras.

1.4.4. Diagrama de Componentes

En el **Diagrama de Componentes**, cada contenedor descrito en el nivel anterior se desglosa en **componentes más pequeños** y específicos que gestionan las diferentes funciones del sistema. Cada uno de estos componentes se especializa en una tarea particular y se comunica con otros a través de interfaces bien definidas. Algunos de los componentes principales incluyen:

En este nivel, cada contenedor es desglosado en sus componentes internos, describiendo cómo interactúan entre sí para proporcionar las funcionalidades requeridas.

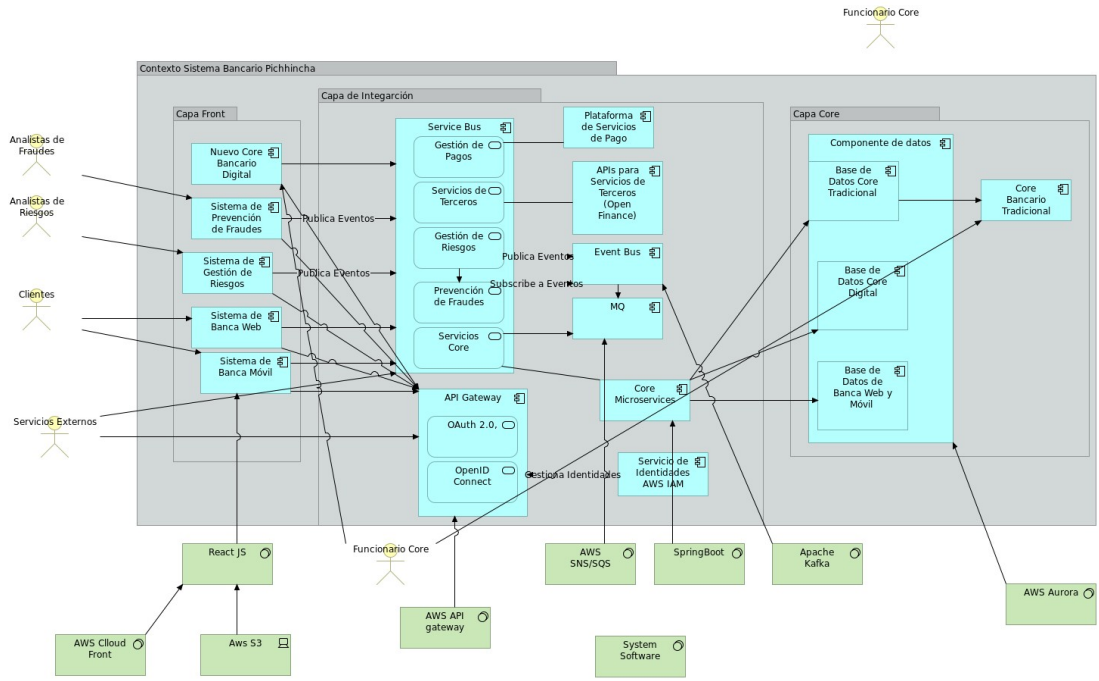

Diagrama de Componentes

Este nivel de detalle asegura que los componentes críticos del sistema bancario puedan desarrollarse, desplegarse y mantenerse de forma independiente, ofreciendo alta modularidad y facilidad de mantenimiento.

1.4.5. Diagrama de Clases

El **Diagrama de Clases** detalla la **estructura de datos** y las relaciones entre los objetos del sistema. Este diagrama es esencial para comprender cómo los datos fluyen a través de los diferentes componentes y cómo se almacenan y procesan. Algunos elementos clave del diagrama incluyen:

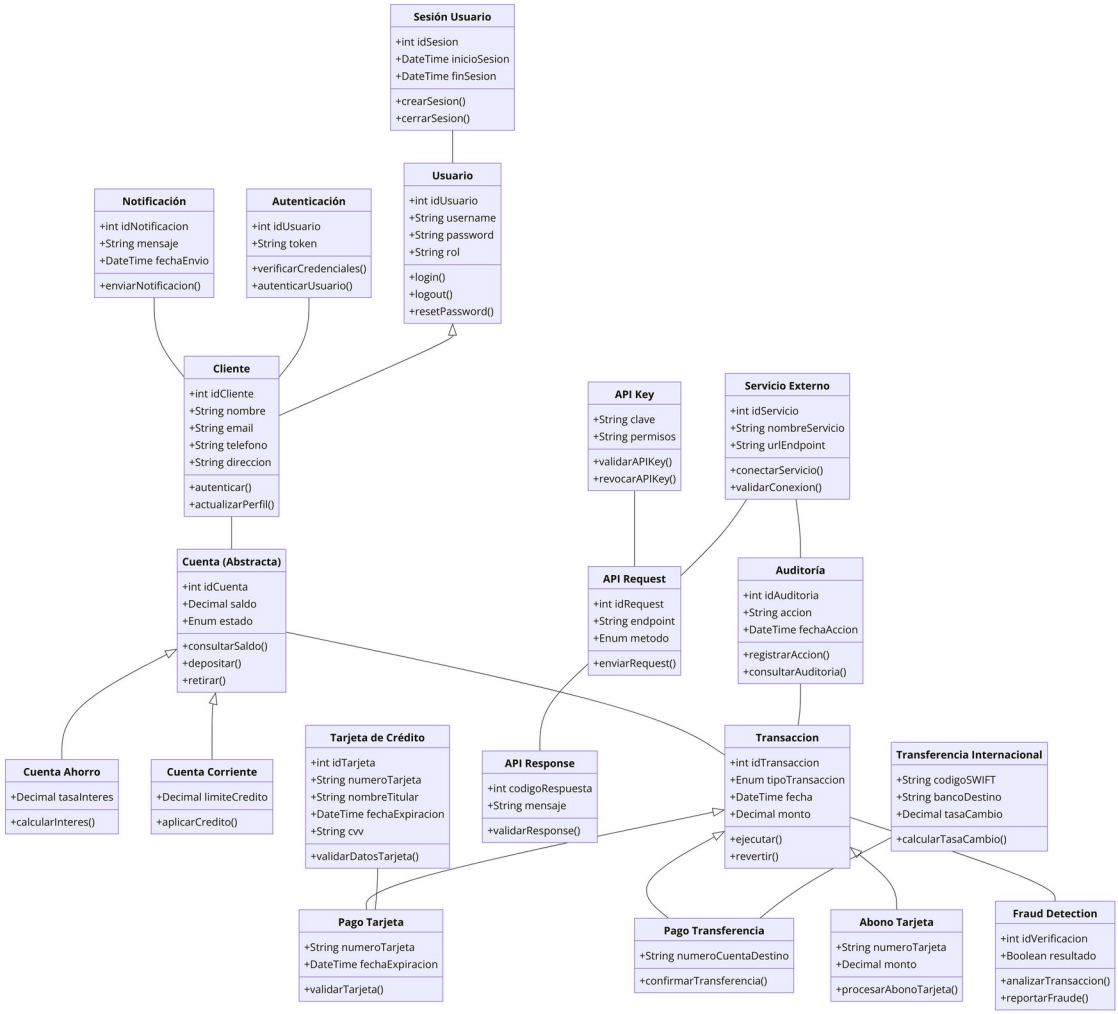

Diagrama de Clases

Este diagrama es fundamental para asegurar que los datos fluyan correctamente entre los sistemas y se mantenga la **integridad** y **consistencia** de la información.

1.4.6. Beneficios Esperados

- **Mejor Toma de Decisiones**: Con el acceso a datos en tiempo real y análisis avanzados, el banco podrá tomar decisiones más rápidas y basadas en información precisa.
- **Reducción de Costos**: La eficiencia operativa mejorada y la automatización reducirán los costos operativos en un 25%.
- **Satisfacción del Cliente**: La modernización mejorará significativamente la satisfacción del cliente, con un incremento estimado del 35% gracias a servicios más rápidos y accesibles.

1.5. Análisis de Costos y Retorno de la Inversión (ROI)

Estimación de Costos

- **Costos de Implementación**:
  - **Tecnología**: Adquisición de hardware, software y servicios en la nube, estimado en $30,500,000.
  - **Desarrollo e Integración**: Desarrollo de las plataformas y la integración de sistemas, estimado en $20,000,000.
  - **Capacitación**: Capacitación del personal en nuevas tecnologías, estimado en $15,000,000.
  - **Transformación Organizacional**: Costos relacionados con la reorganización de procesos internos, estimado en $7,000,000.
- **Costos Operativos**:
  - **Mantenimiento**: Gastos anuales en mantenimiento, estimado en $10,000,000.
  - **Licencias y Suscripciones**: Licencias de software y servicios en la nube, estimado en $6,000,000 anuales.

Retorno de la Inversión (ROI)

- **Beneficios Financieros**: Se proyecta un aumento de ingresos de $50,000,000 anuales gracias a la expansión de los servicios y la reducción de costos operativos en $10,000,000.

- **Costos Totales**: Con un costo de implementación de $72,500,000 y costos operativos anuales de $16,000,000, se proyecta un **ROI del 45%** dentro de los primeros tres años y un período de recuperación de inversión de aproximadamente 2.5 años.

1.6. Resumen

Resumen de Beneficios

La implementación de la nueva arquitectura ofrecerá una mayor eficiencia operativa, mejorará la experiencia del cliente y garantizará el cumplimiento normativo. Se espera una reducción significativa en los costos operativos y una mejora en la satisfacción del cliente, lo que contribuirá al crecimiento sostenible del banco.

Recomendación

Se recomienda proceder con la implementación de la arquitectura de integración basada en BIAN. La evaluación positiva de costos y beneficios, junto con las estrategias de mitigación de riesgos propuestas, respalda la viabilidad y el valor de este proyecto para el banco.

---

2. Patrones de Integración y Tecnologías a Utilizar

El propósito de este apartado es describir los **patrones de integración** y las **tecnologías** que se utilizarán para asegurar que los diferentes componentes del sistema bancario funcionen de manera cohesiva, eficiente y escalable. En el contexto de la modernización de sistemas bancarios, la integración de sistemas heterogéneos, como el core bancario tradicional, plataformas digitales, sistemas de terceros y servicios de pago, requiere un enfoque estratégico en la manera en que estos sistemas interactúan entre sí. Aquí, los patrones de integración definen **cómo los componentes se comunican**, **cómo gestionan los datos** y **cómo se garantiza la seguridad y la resiliencia del sistema**.

Los **patrones de integración** no solo facilitan la interoperabilidad y el flujo de datos entre sistemas dispares, sino que también **mejoran la escalabilidad, el rendimiento y la capacidad de recuperación**. Por otro lado, las **tecnologías** asociadas permiten implementar estos patrones de manera práctica y eficiente, optimizando la infraestructura de TI para el cumplimiento de los objetivos del banco.

Este enfoque de patrones y tecnologías asegura que la **arquitectura sea robusta, escalable y segura**, cumpliendo con los requisitos de modernización del banco y las demandas del entorno financiero actual.

Este punto es crucial para garantizar que las soluciones tecnológicas adoptadas sean capaces de: 1. **Soportar cargas de trabajo variables**. 2. **Mantener la consistencia de los datos** entre diferentes sistemas. 3. **Adaptarse a cambios futuros** en los requisitos del negocio o de la industria. 4. **Asegurar la seguridad y el cumplimiento normativo** en todo el ecosistema bancario.

2.1. Patrones de Integración Aplicados al Caso de Modernización de un Banco

Para la modernización del banco descrita en el escenario, se propone utilizar una serie de **patrones de integración** que facilitarán la interoperabilidad entre los sistemas tradicionales y digitales, permitiendo una **transición eficiente** y mejorando la **resiliencia**, **escalabilidad** y **seguridad** del sistema. A continuación, se detallan los patrones aplicados específicamente a las necesidades del banco en su proceso de modernización:

2.1.1. Event-Driven Architecture (EDA) Aplicada al Core Bancario

Descripción

La **arquitectura dirigida por eventos (EDA)** es un patrón ideal para integrar los **sistemas core bancarios tradicionales** con el **nuevo core digital**. En este caso, cualquier **evento crítico** (como la actualización de una cuenta, la creación de una transacción o una alerta de riesgo) es publicado como un **evento en tiempo real**, el cual es consumido por otros sistemas (microservicios) que necesitan responder a este cambio.

Aplicación al Caso

- **Sistema de Pagos en Tiempo Real**: En lugar de una integración directa y dependiente entre sistemas, el **sistema de pagos** del banco reaccionará a eventos generados por la plataforma de pagos o el core digital. Por ejemplo, cuando se autoriza un pago, este genera un evento que otros servicios, como la **gestión de cuentas** o **fraudes**, pueden procesar en tiempo real.

- **Actualización de Core Digital**: La EDA permite que los eventos generados por el **nuevo core digital** se sincronicen inmediatamente con el **core bancario tradicional**, asegurando que los datos estén actualizados en ambos sistemas sin necesidad de procesos de sincronización complejos.

Tecnologías Aplicadas

- **Apache Kafka** para el **streaming de eventos** en tiempo real entre los sistemas core y los microservicios asociados.

- **AWS SNS/SQS** para la mensajería asíncrona y desacoplada entre los sistemas de pago y la banca móvil.

Ventajas Específicas para el Banco

- **Escalabilidad**: Dado que los sistemas de pagos y core digital pueden procesar eventos de manera independiente, pueden escalar de manera autónoma sin sobrecargar los otros sistemas.

- **Flexibilidad**: Permite agregar nuevos servicios, como **alertas de fraude** o **servicios de riesgo**, sin afectar la operativa de los sistemas principales.

2.1.2. API Gateway para Exponer Servicios Bancarios Externos

Descripción

El uso de un **API Gateway** permite centralizar y gestionar las comunicaciones externas del banco, garantizando que todas las APIs del **core digital**, la **plataforma de pagos** y los **servicios móviles** se gestionen de manera eficiente y segura. El API Gateway actúa como una

puerta de entrada única, facilitando el enrutamiento, la autenticación y la seguridad.

Aplicación al Caso

- **Open Finance**: Las APIs que exponen servicios bancarios a terceros en el marco de **Open Finance** (por ejemplo, para compartir datos financieros o permitir pagos desde aplicaciones de terceros) estarán controladas a través de un **API Gateway**, lo que asegurará que se cumplan las políticas de seguridad y los controles de acceso adecuados.

- **Plataforma de Banca Web y Móvil**: La interacción entre las plataformas de **banca móvil** y **banca web** con los sistemas backend del banco estará mediada por el API Gateway, facilitando la administración de las solicitudes y protegiendo los sistemas internos de sobrecargas o accesos no autorizados.

Tecnologías Aplicadas

- **AWS API Gateway** para gestionar y escalar las APIs del banco, con soporte para autenticación mediante **OAuth 2.0** y **OpenID Connect**.

- **Kong Gateway** para la gestión de API de servicios internos y externos.

Ventajas Específicas para el Banco

- **Seguridad Centralizada**: Todas las solicitudes se validan en el API Gateway, lo que simplifica la gestión de políticas de seguridad y acceso.

- **Monitoreo y Control**: El banco puede monitorear y controlar todas las solicitudes a sus sistemas, incluyendo la gestión de **tasa de solicitudes (rate limiting)** y **autenticación centralizada**.

2.1.3. Service Bus para la Integración del Core Bancario y la Gestión de Riesgos

Descripción

El **Service Bus** facilita la **integración asincrónica** entre sistemas bancarios tradicionales y digitales, actuando como un **middleware** para la comunicación de mensajes y garantizando que las transacciones se manejen de manera eficiente y segura. Este patrón es clave en la integración de sistemas core con **sistemas de gestión de riesgos** y **prevención de fraudes**.

Aplicación al Caso

- **Integración entre el Core Bancario y el Sistema de Gestión de Riesgos**: El **Service Bus** se utilizará para coordinar la comunicación

entre el **core bancario tradicional**, el **nuevo core digital** y los sistemas de **gestión de riesgos**. Por ejemplo, cuando se detecta una transacción sospechosa, se envía un mensaje al sistema de riesgos para que lo procese sin interrumpir el flujo principal.

- **Sistema de Prevención de Fraudes**: Los mensajes de alerta generados por transacciones inusuales o comportamientos atípicos serán enviados al **Service Bus** y distribuidos a los sistemas de prevención de fraudes, que los procesarán de manera autónoma.

Tecnologías Aplicadas

- **AWS SQS** para la **entrega fiable de mensajes** entre los sistemas core y los sistemas de gestión de riesgos.
- **RabbitMQ** para la gestión de **mensajería asincrónica** en procesos de transacciones distribuidas.

Ventajas Específicas para el Banco

- **Desacoplamiento**: Los sistemas de core bancario y gestión de riesgos pueden operar de manera independiente, lo que facilita el mantenimiento y la escalabilidad.
- **Garantía de Entrega**: Asegura que todos los mensajes importantes, como las alertas de riesgo, se entreguen incluso en caso de fallos.

2.1.4. Microservicios para la Modernización Modular

Descripción

El patrón de **microservicios** se implementa para permitir que el banco divida sus sistemas en **componentes más pequeños**, que pueden ser desarrollados, desplegados y escalados de manera independiente. Cada microservicio maneja una parte específica de la funcionalidad bancaria, como la **gestión de cuentas**, **procesamiento de pagos** o **sistemas de riesgos**.

Aplicación al Caso

- **Core Bancario Modular**: El nuevo **core digital** estará compuesto por microservicios autónomos que manejan cada una de las funciones bancarias (como la gestión de cuentas, transacciones, pagos). Esto permite realizar **actualizaciones** y **despliegues** independientes, mejorando la capacidad de adaptación a nuevas regulaciones y demandas del mercado.
- **Plataforma de Pagos**: La plataforma de pagos será un conjunto de microservicios que gestionan todo el ciclo de vida de una transacción, desde la autenticación del usuario hasta la confirmación del pago.

Tecnologías Aplicadas

- **Docker** para el despliegue y contenedorización de microservicios.
- **Amazon ECS** y **Kubernetes** para la orquestación de microservicios.
- **Spring Boot** para el desarrollo de microservicios en Java.

Ventajas Específicas para el Banco

- **Escalabilidad Modular**: Cada componente del sistema puede escalarse de manera independiente, permitiendo que los sistemas críticos

tengan más recursos durante picos de demanda.

- **Despliegue Autónomo**: Los equipos de desarrollo pueden implementar y actualizar microservicios sin interrumpir el funcionamiento de otros sistemas.

2.1.5. Saga Pattern para Transacciones Distribuidas

Descripción

El **patrón Saga** es clave para gestionar **transacciones distribuidas** entre los diversos microservicios del banco. En lugar de bloquear recursos hasta que una transacción completa sea exitosa, **Saga** coordina las transacciones distribuidas y, en caso de fallo, realiza acciones compensatorias.

Aplicación al Caso

- **Gestión de Transacciones Bancarias**: Cuando un cliente realiza una transacción compleja que involucra múltiples servicios (por ejemplo, una transferencia bancaria que requiere validación de fondos, actualización de cuentas y confirmación de la transacción), el **patrón Saga** coordina las acciones de cada servicio y, en caso de error, ejecuta acciones compensatorias, como la reversión de una operación fallida.
- **Creación de Nuevas Cuentas**: La creación de una cuenta bancaria involucra múltiples servicios, como la verificación de identidad y la asignación de cuentas. Si algún paso falla, Saga permite que los pasos previos se reviertan de manera automática.

Tecnologías Aplicadas

- **Apache Camel** para la **orquestación de procesos** basados en eventos.
- **Spring Cloud Data Flow** para la gestión de transacciones distribuidas en arquitecturas de microservicios.

Ventajas Específicas para el Banco

- **Coordinación de Servicios**: Permite gestionar transacciones complejas entre diferentes servicios sin bloquear el sistema.
- **Resiliencia en Fallos**: Si un paso de la transacción falla, Saga garantiza que se ejecuten acciones compensatorias sin afectar al sistema global.

2.3. Conclusión

Este conjunto de patrones aplicados específicamente a la arquitectura del banco permitirá una **modernización eficiente**, asegurando una **integración fluida** entre los sistemas tradicionales y los nuevos sistemas digitales, mejorando la **escalabilidad** y la **resiliencia** de la infraestructura tecnológica.

3. Requisitos de Seguridad y Cumplimiento Normativo: Estrategia de Seguridad Aplicada a la Modernización del Banco

El objetivo de la **estrategia de seguridad** es asegurar la **confidencialidad**, **integridad** y **disponibilidad** de los datos sensibles del banco, como información financiera y personal de los clientes, en un entorno digital modernizado. Dado que la arquitectura propuesta implica la integración de sistemas core tradicionales y digitales, es crucial implementar medidas de seguridad robustas que mitiguen riesgos de **ciberataques**, accesos no autorizados y **filtraciones de datos**, cumpliendo también con las **normativas regulatorias** aplicables, como la **Ley Orgánica de Protección de Datos Personales (LOPD)**.

3.1. Implementación de OAuth 2.0 y OpenID Connect para Autenticación y Autorización

Aplicación al Caso Bancario

Dado que la modernización del banco implica la exposición de **APIs** tanto internas como externas a través de **Open Finance**, es fundamental implementar un esquema de autenticación y autorización sólido para proteger el acceso a los sistemas. Los usuarios, tanto internos como externos (terceros autorizados), necesitarán acceso a servicios críticos, como transacciones y gestión de cuentas.

OAuth 2.0

- **Protocolo de Autorización**: En este escenario, **OAuth 2.0** se implementará para permitir que terceros, como aplicaciones de pago o agregadores financieros, accedan a los **recursos del banco** (transacciones, saldos, datos de clientes) sin exponer las credenciales de los usuarios. Esto es clave en el contexto de **Open Finance**, donde se requiere una autorización basada en permisos específicos.

- **Roles y Permisos**: En el sistema bancario, se definirán roles para diferentes niveles de acceso:

    - **Usuarios finales**: Tendrán acceso restringido a sus propios datos de cuentas y transacciones.
    - **Terceros autorizados (Fintech)**: Acceso limitado a los recursos necesarios para la integración, como la verificación de identidad o la iniciación de pagos.
    - **Administradores internos**: Control total para gestionar las políticas de acceso, monitorear actividades y administrar recursos sensibles.

- **Autorización Granular**: En un banco, la **autorización granular** garantiza que solo los usuarios con los permisos correctos accedan a información confidencial, como datos financieros o personales. Por ejemplo, un tercero autorizado solo podrá acceder a información de transacciones específicas, pero no a los datos personales completos del cliente.

OpenID Connect

- **Protocolo de Autenticación**: **OpenID Connect** se implementará para gestionar la autenticación de los usuarios que acceden a las plataformas de **banca móvil** y **banca web**. OpenID Connect permite verificar la identidad de los usuarios mediante proveedores de identidad externos, mejorando la **seguridad** y asegurando que solo los usuarios autenticados puedan acceder a los recursos del banco.

- **SSO (Single Sign-On)**: Se implementará **SSO** para que los clientes puedan iniciar sesión una sola vez en las plataformas del banco (móvil, web, APIs) y acceder a todas las funcionalidades sin necesidad de autenticarse varias veces. Esto mejora la **experiencia del usuario** y facilita la gestión de identidades dentro del sistema bancario.

- **Proveedores de Identidad**: Para la gestión de autenticación segura, el banco integrará **AWS Cognito** como proveedor de identidad, lo que facilitará el manejo de usuarios tanto internos como externos en el contexto de **Open Finance**, permitiendo la integración con plataformas fintech mediante flujos seguros.

3.2. Cifrado de Datos en Tránsito y en Reposo

Cifrado Aplicado a la Seguridad Bancaria

El banco maneja datos altamente sensibles, desde información personal hasta detalles de transacciones financieras. La **integridad y confidencialidad** de estos datos debe estar protegida tanto cuando están en tránsito entre los sistemas como cuando están almacenados en bases de datos o archivos.

Cifrado de Datos en Tránsito

- **TLS (Transport Layer Security)**: Se implementará **TLS** en todas las comunicaciones entre **banca móvil**, **web**, y los sistemas backend del banco, garantizando que los datos como transacciones, credenciales de usuario, y consultas a las APIs estén protegidos frente a **ataques de intercepción** o **manipulación**. Todo el tráfico entre microservicios en la nube también será cifrado mediante TLS, asegurando que los datos internos no sean vulnerables durante su transmisión.

- **Certificados de Seguridad**: Se utilizarán **certificados SSL/TLS** gestionados a través de **AWS Certificate Manager (ACM)** para asegurar que las conexiones entre los usuarios y el banco estén verificadas y seguras. Esto incluye la gestión automatizada de los certificados, lo que garantiza su actualización sin interrupciones en el servicio.

Cifrado de Datos en Reposo

- **AWS Key Management Service (KMS)**: Los datos almacenados en bases de datos y sistemas de archivos serán cifrados utilizando **KMS**, asegurando que solo los sistemas autorizados puedan acceder a los datos sensibles. Esto será particularmente importante para la protección de datos de clientes almacenados en las bases de datos de **Amazon RDS** y los backups almacenados en **Amazon S3**.

- **Amazon S3 Encryption y RDS Encryption**: Todo el almacenamiento en **Amazon S3** y las bases de datos en **Amazon RDS** estarán configurados para cifrar automáticamente los datos en reposo utilizando claves gestionadas por KMS. Esto garantiza que incluso en el caso de un compromiso del almacenamiento físico, los datos permanezcan protegidos.

3.3. Firewalls y Políticas de Seguridad a Nivel de Red y Aplicación

Seguridad a Nivel de Red en la Arquitectura Bancaria

En un banco, la seguridad a nivel de red es crítica, dado que las infraestructuras están expuestas a amenazas constantes. Asegurar que solo el tráfico autorizado pueda interactuar con los sistemas bancarios es vital para proteger las transacciones y la integridad de los datos.

Firewalls a Nivel de Red

- **Security Groups y Network ACLs**: Se implementarán **Security Groups** para cada uno de los microservicios que conforman el core bancario digital y los sistemas de pago. Estos actúan como firewalls virtuales que controlan el tráfico entrante y saliente, permitiendo solo el acceso desde IPs o servicios autorizados. Las **Network ACLs** proporcionan una capa adicional de control a nivel de subred, lo que garantiza que incluso dentro de la infraestructura del banco solo el tráfico autorizado pueda moverse entre servicios.

Firewalls a Nivel de Aplicación

- **AWS WAF (Web Application Firewall)**: El **WAF** será una barrera adicional en las plataformas de banca móvil y web, protegiendo a los sistemas contra amenazas comunes como **inyecciones SQL** y

ataques de **cross-site scripting (XSS)**, que podrían comprometer la integridad de los datos o permitir accesos no autorizados.

Componentes de AWS VPC y Relacionados con Firewalls

- **VPC (Virtual Private Cloud)**: Toda la infraestructura del banco operará dentro de una **VPC** en AWS, lo que permite aislar los sistemas bancarios y protegerlos de accesos no autorizados. Los componentes como **subnets privadas**, **NAT gateways** y **Elastic IP addresses** garantizan que los sistemas internos sean accesibles solo a través de rutas seguras.

3.4. Monitoreo de Seguridad Continuo con Herramientas SIEM

Monitoreo Aplicado a la Seguridad Bancaria

Dado que las amenazas de seguridad evolucionan constantemente, es fundamental que el banco implemente un sistema de **monitoreo continuo** que permita detectar anomalías en tiempo real y tomar medidas preventivas o correctivas antes de que ocurra un incidente grave.

Amazon GuardDuty

- **Detección de Amenazas**: **Amazon GuardDuty** se utilizará para detectar actividades sospechosas y potencialmente maliciosas en el entorno AWS del banco. GuardDuty monitoreará los **logs de VPC**, **CloudTrail** y **DNS** para identificar patrones de ataque o acceso no autorizado en tiempo real, lo que permitirá una rápida respuesta a las amenazas.

AWS Security Hub

- **Panel Centralizado de Seguridad**: **AWS Security Hub** integrará los hallazgos de **GuardDuty**, **AWS Config**, y otros servicios de seguridad para proporcionar una **vista unificada** del estado de seguridad del banco. Esto permitirá al equipo de seguridad priorizar y gestionar las amenazas de manera eficiente en todas las cuentas y servicios del banco.

AWS CloudTrail y Amazon CloudWatch

- **Registro de Actividades**: **AWS CloudTrail** registrará todas las actividades en la infraestructura del banco, desde cambios en la configuración de los microservicios hasta solicitudes API. Esto es clave para realizar **auditorías de seguridad** y análisis forenses en caso de incidentes de seguridad.

- **Monitoreo de Rendimiento y Seguridad**: **Amazon CloudWatch** se configurará para monitorear el rendimiento de las aplicaciones críticas,

configurando alertas cuando se detecten anomalías o cuellos de botella. Las métricas de rendimiento también se correlacionarán con los eventos de seguridad para detectar posibles intentos de denegación de servicio.

3.5. Cumplimiento Normativo

Cumplimiento con la Ley Orgánica de Protección de Datos Personales (LOPD)

El banco estará plenamente alineado con las regulaciones de la

LOPD, asegurando la **protección de los datos personales** de los clientes mediante la implementación de **encriptación**, **anonimización** y **gestión del consentimiento**.

Protección de Datos Personales

- **Encriptación y Anonimización**: Los datos personales de los clientes se almacenarán en formato **cifrado**, garantizando que solo los usuarios autorizados tengan acceso a ellos. Además, se implementarán técnicas de **anonimización** y **pseudonimización** para los datos en procesos de análisis, reduciendo el riesgo de exposición en caso de violaciones de seguridad.

Consentimiento del Usuario y Notificación de Brechas

- **Gestión del Consentimiento**: A través de los sistemas de **gestión de identidades (IAM)** y **OpenID Connect**, el banco implementará mecanismos que permitan a los usuarios otorgar y revocar su consentimiento para el tratamiento de sus datos. Además, se establecerán procedimientos claros para notificar a los usuarios y las autoridades competentes en caso de una **brecha de seguridad** dentro de los plazos establecidos por la normativa.

3.6. Ethical Hacking como Estrategia de Seguridad Proactiva

El **Ethical Hacking**, o **hacking ético**, es una práctica que consiste en realizar **pruebas de intrusión controladas** para identificar vulnerabilidades en los sistemas del banco antes de que puedan ser explotadas por actores maliciosos. El objetivo es descubrir debilidades de seguridad que podrían ser aprovechadas para acceder a datos sensibles, interrumpir servicios críticos o comprometer la infraestructura del banco. En el contexto de la modernización bancaria, el ethical hacking será una herramienta clave para fortalecer la seguridad y garantizar el cumplimiento normativo.

3.6.1. Aplicación del Ethical Hacking en el Banco

Dado el entorno de **integración de sistemas core tradicionales y digitales**, el banco expondrá nuevas superficies de ataque, especialmente a través de sus **APIs**, **plataformas de banca móvil** y **web**, y **servicios de Open Finance**. El ethical hacking proporcionará una capa adicional de seguridad, asegurando que estas interfaces sean robustas frente a **ciberataques**.

3.6.2. Pruebas de Intrusión (Penetration Testing)

El **Penetration Testing**, o pruebas de intrusión, será una técnica esencial para evaluar la seguridad del banco. Esto se llevará a cabo mediante simulaciones de ataques reales, intentando vulnerar los sistemas de manera controlada para identificar puntos débiles y corregirlos antes de que sean explotados.

- **Alcance del Pentesting**: Las pruebas abarcarán todas las áreas críticas del banco, incluidas las APIs expuestas, la infraestructura en la nube, las bases de datos, las aplicaciones móviles y web, y los sistemas de pago.
- **Tipos de Pruebas**:
    - **Pruebas de Caja Negra**: Simulación de un ataque desde el exterior, sin conocimiento interno del sistema, para evaluar las defensas externas.
    - **Pruebas de Caja Blanca**: Evaluación con pleno acceso al código fuente y documentación para identificar vulnerabilidades internas más profundas.
    - **Pruebas de Caja Gris**: Combinación de las anteriores para simular un ataque de un actor con acceso limitado, como un empleado malintencionado o un usuario con privilegios restringidos.

3.6.3. Seguridad de APIs y Aplicaciones

En el contexto del **Open Finance**, donde terceros como fintechs tienen acceso a las APIs del banco, las pruebas de ethical hacking se centrarán en identificar vulnerabilidades relacionadas con:

- **Inyecciones SQL**: Asegurar que las APIs y aplicaciones no sean vulnerables a inyecciones SQL, que podrían permitir a un atacante acceder a bases de datos sensibles.
- **Cross-Site Scripting (XSS)**: Proteger las aplicaciones web del banco contra ataques XSS, que podrían comprometer la experiencia del usuario y la integridad de los datos.

- **Accesos No Autorizados**: Asegurar que los mecanismos de autenticación y autorización, como **OAuth 2.0** y **OpenID Connect**, están correctamente configurados para evitar accesos no autorizados.

3.6.4. Auditorías de Seguridad Basadas en Resultados de Hacking Ético

Los resultados de los tests de hacking ético serán utilizados para **auditorías de seguridad regulares**. Estas auditorías identificarán las vulnerabilidades más críticas y permitirán priorizar su remediación.

- **Corrección de Vulnerabilidades**: Implementar medidas correctivas en áreas donde se descubran vulnerabilidades significativas, como mejorar la seguridad de APIs o reforzar la autenticación en los sistemas de banca móvil.
- **Reportes y Cumplimiento Normativo**: El ethical hacking también ayudará a generar informes detallados que serán útiles para cumplir con regulaciones como la **Ley Orgánica de Protección de Datos Personales (LOPD)**, proporcionando evidencia de que se están tomando medidas proactivas para proteger los datos de los clientes.

3.6.5. Concienciación y Entrenamiento del Personal

Además de las pruebas técnicas, el ethical hacking puede revelar debilidades en los **procedimientos operativos** o en la **concienciación del personal** frente a las amenazas de ciberseguridad. Se propondrán programas de **capacitación** para asegurar que los empleados del banco puedan identificar amenazas como **phishing** o ataques dirigidos, y respondan adecuadamente.

3.6.6. Pruebas de Red y Firewalls

Los firewalls, tanto a nivel de red como de aplicación, serán sometidos a pruebas para asegurar que están correctamente configurados y no permiten tráfico no autorizado. El ethical hacking ayudará a verificar:

- **Configuración de Security Groups**: Confirmar que las políticas de acceso en los **Security Groups** y **ACLs** (Access Control Lists) de la nube están configuradas para permitir solo el tráfico necesario y bloquear posibles vectores de ataque.
- **Pruebas de WAF**: Probar la efectividad del **Web Application Firewall (WAF)** de AWS para detectar y mitigar ataques comunes como **inyecciones de código** o **ataques de fuerza bruta** en las plataformas web y móviles.

3.6.7. Monitoreo Continuo Basado en Ethical Hacking

El ethical hacking no es una actividad única, sino parte de un ciclo continuo de **mejora de seguridad**. Los sistemas del banco deberán ser evaluados regularmente, adaptando las pruebas a las **nuevas amenazas** emergentes.

- **Integración con SIEM**: Los resultados del ethical hacking se integrarán con el sistema de **monitoreo de seguridad SIEM** (Security Information and Event Management), lo que permitirá al equipo de seguridad **correlacionar incidentes de seguridad en tiempo real** y tomar decisiones proactivas.

3.6.8. Beneficios del Ethical Hacking

- **Identificación Proactiva de Vulnerabilidades**: Permite identificar debilidades antes de que sean explotadas, reduciendo el riesgo de incidentes de seguridad que podrían comprometer los datos de los clientes o la operación del banco.
- **Cumplimiento Normativo**: Apoya al banco en el cumplimiento de normativas regulatorias, proporcionando pruebas de que se han tomado medidas para proteger los sistemas y datos.
- **Mejora Continua de la Seguridad**: Fomenta una cultura de mejora continua en la seguridad, adaptándose a las amenazas emergentes y asegurando que el banco mantenga una postura sólida frente a ciberataques.

3.6.9. En Resumen

La inclusión de **ethical hacking** en la estrategia de seguridad del banco asegura que la infraestructura digital modernizada no solo cumpla con los requisitos de **seguridad** y **cumplimiento normativo**, sino que también esté preparada para enfrentar **amenazas avanzadas**. Esto refuerza la **confianza de los clientes** y protege los **activos críticos** del banco en un entorno de ciberseguridad cada vez más complejo.

3.6. Conclusión

Esta **estrategia de seguridad** asegura que la modernización del banco no solo mejora la eficiencia operativa y la experiencia del cliente, sino que también cumple con los **más altos estándares de seguridad** y las **normativas regulatorias**, protegiendo tanto a los clientes como a la infraestructura crítica del banco.

---

4. Estrategia para Garantizar Alta Disponibilidad y Recuperación ante Desastres

El propósito de la estrategia de **Alta Disponibilidad (HA)** y **Recuperación ante Desastres (DR)** es asegurar que los sistemas bancarios críticos estén siempre operativos y que cualquier fallo, ya sea a nivel de hardware, software o infraestructura, tenga un impacto mínimo en el servicio. En el contexto de la banca, donde las interrupciones pueden generar pérdidas financieras significativas, dañar la reputación del banco o resultar en el incumplimiento de regulaciones, es crucial que las operaciones se mantengan sin interrupciones.

Además, en caso de un desastre, ya sea natural o causado por fallos técnicos, la organización debe ser capaz de **recuperar rápidamente** las aplicaciones y datos esenciales para minimizar el tiempo de inactividad. Esto incluye la **replicación de datos**, la **automación de la recuperación** y la **preparación continua** mediante simulaciones y pruebas.

4.1. Alta Disponibilidad (HA)

Alta Disponibilidad (HA) se refiere a la capacidad de un sistema para mantenerse operativo durante largos periodos sin interrupciones. La meta es reducir el tiempo de inactividad planificado o no planificado, asegurando que los servicios bancarios puedan continuar funcionando incluso en escenarios de alta demanda o fallos técnicos.

Despliegues Multi-Región

- **AWS Multi-AZ (Availability Zones)**: Al desplegar servicios en varias **zonas de disponibilidad (AZ)** dentro de una misma región de AWS, se garantiza que si una zona experimenta fallos, las otras puedan mantener la operación sin interrupciones. Las **Multi-AZ** actúan como una medida de tolerancia a fallos local que reduce el riesgo de una interrupción significativa.

- **AWS Multi-Region**: Para eventos más catastróficos, como la caída de una región completa, el uso de **Multi-Región** asegura que los datos y servicios estén replicados en varias regiones geográficas. Esto es vital para mantener la continuidad del negocio en caso de desastres a gran escala.

Balanceo de Carga

- **AWS Elastic Load Balancing (ELB)**: ELB distribuye el tráfico entre múltiples instancias de la aplicación, asegurando que ninguna instancia se vea sobrecargada y que haya redundancia en caso de que una instancia falle. Este enfoque optimiza el rendimiento y la disponibilidad del servicio.

- **Auto Scaling Groups**: Los **grupos de autoescalado** permiten que el sistema ajuste automáticamente el número de instancias en función de la demanda, asegurando que los recursos sean proporcionales a la carga de trabajo, lo que aumenta tanto la eficiencia como la capacidad de recuperación.

Microservicios Desplegados en Contenedores

- **Amazon ECS (Elastic Container Service)**: Al desplegar los microservicios en **contenedores gestionados** a través de ECS, se asegura que los fallos en un contenedor o en una aplicación específica no afecten al resto del sistema. ECS maneja la orquestación automática y la recuperación de fallos, lo que garantiza que los microservicios estén siempre disponibles.

Base de Datos Alta Disponibilidad

- **Amazon RDS Multi-AZ**: Con **RDS Multi-AZ**, las bases de datos están replicadas automáticamente en diferentes zonas de disponibilidad, lo que garantiza que, si una base de datos o zona falla, las solicitudes se redirijan automáticamente a una réplica en una zona diferente sin afectar a la operación.

- **Amazon Aurora**: Aurora es una base de datos diseñada para ser altamente disponible, con replicación automática y un failover casi instantáneo, lo que asegura que las aplicaciones críticas de datos no sufran interrupciones prolongadas.

Monitoreo y Alertas

- **Amazon CloudWatch**: CloudWatch proporciona monitoreo constante de la salud de las aplicaciones y la infraestructura, permitiendo detectar problemas de rendimiento o fallos antes de que se conviertan en incidentes críticos. Las **alertas automáticas** notifican a los equipos de TI para que respondan rápidamente.

- **AWS X-Ray**: X-Ray rastrea las solicitudes de los usuarios a través de toda la arquitectura, detectando problemas de rendimiento, cuellos de botella y fallos. Esto permite a los equipos solucionar problemas antes de que impacten a los usuarios finales.

4.2. Recuperación ante Desastres (DR)

La **recuperación ante desastres (DR)** se centra en la capacidad de restaurar las operaciones de manera rápida y eficiente después de un incidente que provoque una interrupción significativa. La estrategia DR asegura que los sistemas puedan ser restaurados en cuestión de minutos u horas, dependiendo de la criticidad de los servicios, minimizando las pérdidas operativas y financieras.

Backups y Replicación

- **Amazon S3**: Almacenar copias de seguridad de bases de datos y aplicaciones en **Amazon S3** proporciona un lugar de almacenamiento seguro y escalable, con políticas de ciclo de vida para gestionar la retención y el archivado de datos, asegurando que los datos críticos estén siempre disponibles.

- **AWS Backup**: **AWS Backup** permite automatizar y centralizar la gestión de copias de seguridad para diferentes servicios de AWS, como **Amazon RDS**, **DynamoDB** y **EFS**, lo que asegura que todos los datos críticos estén respaldados y sean fácilmente restaurables.

Planes de Recuperación

- **AWS CloudFormation**: **CloudFormation** permite crear infraestructuras como código (IaC), lo que facilita la recreación rápida de entornos de recuperación mediante plantillas predefinidas. Esto asegura que los entornos de recuperación sean consistentes y estén disponibles cuando se necesiten.

- **AWS Elastic Disaster Recovery**: Este servicio permite replicar aplicaciones y datos críticos a otra región, con failover automatizado y pruebas de recuperación periódicas, lo que garantiza que los sistemas puedan ser restaurados rápidamente en caso de una interrupción catastrófica.

Failover y Conmutación por Error

- **Amazon Route 53**: Con **Route 53**, se pueden configurar políticas de failover DNS que redirijan automáticamente el tráfico a una región de recuperación si una región principal falla. Esto asegura que los usuarios finales experimenten interrupciones mínimas.

- **Pilot Light y Warm Standby**: Estrategias como **Pilot Light** y **Warm Standby** permiten mantener un entorno reducido en la región de recuperación, asegurando que solo sea necesario activar los servicios en caso de una interrupción, lo que optimiza costos y tiempos de recuperación.

Pruebas y Simulaciones

- **Game Days**: Realizar simulacros regulares de recuperación, conocidos como **Game Days**, permite al equipo probar los procedimientos de recuperación en situaciones simuladas, asegurando que estén preparados para responder de manera efectiva ante un incidente real.

- **AWS Fault Injection Simulator**: Probar fallos controlados mediante **Fault Injection Simulator** permite identificar y corregir posibles puntos de fallo en el sistema, mejorando la resiliencia general.

Automatización y Orquestación

- **Infraestructura como Código (IaC)**: Definir la infraestructura de recuperación mediante **IaC** asegura que los entornos puedan ser recreados de manera consistente y repetible en situaciones de desastre.
- **Automatización de Despliegues**: Implementar pipelines de CI/CD mediante **AWS CodePipeline** y **CodeDeploy** permite automatizar los despliegues de las aplicaciones y sus actualizaciones en los entornos de recuperación, reduciendo el riesgo de errores humanos y asegurando la rapidez en la restauración del sistema.

Conclusión

La **estrategia de Alta Disponibilidad (HA) y Recuperación ante Desastres (DR)** es fundamental para garantizar la **resiliencia operativa** de un sistema bancario, permitiendo que los servicios permanezcan activos incluso en situaciones de fallos técnicos o desastres naturales. El enfoque basado en múltiples regiones, el uso de tecnologías de monitoreo y replicación, junto con la automatización de la recuperación, asegura que el sistema sea capaz de resistir fallos y recuperarse rápidamente, protegiendo tanto los datos del cliente como la continuidad del negocio. ## 4. Estrategia para Garantizar Alta Disponibilidad y Recuperación ante Desastres

El propósito de la estrategia de **Alta Disponibilidad (HA)** y **Recuperación ante Desastres (DR)** es asegurar que los sistemas bancarios críticos del banco permanezcan operativos en todo momento, minimizando los impactos de cualquier interrupción, ya sea por fallos técnicos o desastres naturales. Esta estrategia es especialmente crítica en el sector bancario, donde las interrupciones de servicio pueden generar pérdidas financieras sustanciales, dañar la reputación del banco y derivar en incumplimientos regulatorios. El enfoque aquí se centra en mantener la **continuidad operativa**, **recuperar servicios esenciales** rápidamente en caso de fallos y **garantizar la protección de los datos**.

---

5. Estrategia de Integración Multicore

Propósito

El propósito de la **Estrategia de Integración Multicore** es asegurar que tanto el **Core Bancario Tradicional** como el **Nuevo Core Bancario**

Digital trabajen en conjunto de manera armoniosa y eficiente, proporcionando **sincronización de datos en tiempo real**, **flexibilidad operativa** y **resiliencia**. Esto es vital para garantizar que los sistemas heredados y los modernos operen en sinergia, sin comprometer la **integridad de los datos** ni la **experiencia del cliente**. La estrategia también permite la evolución independiente de cada core, maximizando la **escalabilidad** y la capacidad de responder rápidamente a los cambios del mercado.

5.1. Sincronización de Datos en Tiempo Real

La sincronización de datos entre el **core tradicional** y el **core digital** es esencial para mantener la coherencia de la información, permitiendo que las operaciones bancarias ocurran sin interrupciones ni inconsistencias. Esta sincronización asegura que cualquier transacción o actualización en un core se refleje inmediatamente en el otro.

Bus de Eventos

Apache Kafka como Bus de Eventos

Se utilizará **Apache Kafka** como la columna vertebral del sistema de **eventos**. Este bus de eventos manejará la sincronización entre los diferentes servicios de ambos cores, permitiendo una comunicación eficiente y escalable sin necesidad de conexiones directas entre ellos.

- **Publicación de Eventos**: Cuando se realice una operación importante en cualquiera de los cores, como la apertura de una nueva cuenta o la ejecución de una transacción, se generará un **evento** que se publicará en Kafka. Este evento se distribuirá a todos los sistemas que necesiten estar informados de dicha operación, permitiendo que las bases de datos y servicios de ambos cores estén sincronizados.
- **Suscripción a Eventos**: Los microservicios que gestionan las cuentas de clientes, transacciones y alertas de fraude se suscriben a estos eventos, garantizando que cualquier cambio importante sea registrado y procesado en tiempo real en todos los sistemas relevantes.

Integración con Event Sourcing

El uso del patrón de **Event Sourcing** permitirá reconstruir el estado del sistema a partir de los eventos almacenados. Cada evento representa un cambio en los datos, por lo que las bases de datos podrán ser actualizadas de manera continua, mejorando la resiliencia y facilitando auditorías detalladas en caso de discrepancias.

Data Streams

Kafka Streams para Procesamiento en Tiempo Real

Kafka Streams permitirá transformar y procesar los eventos generados por los cores en tiempo real. Esto es esencial para realizar operaciones complejas, como la reconciliación de datos entre ambos cores y la preparación de informes financieros de forma automática.

- **Procesamiento en Tiempo Real**: Por ejemplo, cuando se autoriza un pago, Kafka Streams puede transformar los eventos relacionados para incluir información adicional sobre el cliente o las características de la transacción antes de almacenarlos en el sistema de destino.
- **ETL para Grandes Volúmenes de Datos**: En casos donde se requiere sincronizar grandes volúmenes de datos históricos entre los sistemas, se implementarán pipelines de **ETL (Extract, Transform, Load)**. Esto asegurará que los datos históricos del **core bancario tradicional** se mantengan actualizados en el **nuevo core digital**.

Replicación de Base de Datos

AWS Database Migration Service (DMS)

Se implementará **AWS DMS** para garantizar la **replicación en tiempo real** entre las bases de datos del **core bancario tradicional** y el **core digital**. DMS permite realizar esta sincronización sin interrumpir las operaciones normales, facilitando la **consistencia** entre los dos sistemas sin generar tiempos de inactividad.

- **Sincronización de Datos Sensibles**: Por ejemplo, cualquier actualización en la información de los clientes, como cambios de dirección o de datos financieros, se replicará automáticamente entre ambas bases de datos, asegurando que los sistemas reflejen la misma información al instante.
- **Monitoreo Continuo**: DMS proporciona monitoreo continuo de la replicación, detectando y corrigiendo cualquier retraso en la sincronización de datos para evitar inconsistencias.

5.2. Uso de Microservicios para Abstracción y Flexibilidad

La **arquitectura de microservicios** proporciona una estructura modular para el sistema bancario, donde cada función crítica (como la gestión de pagos o la validación de identidad) se gestiona de manera independiente. Esto permite que los equipos de desarrollo trabajen de manera paralela en diferentes partes del sistema, mejorando la **agilidad** y reduciendo la dependencia entre componentes.

Descomposición Funcional

División en Microservicios Independientes

El **core bancario digital** estará descompuesto en **microservicios** que gestionan funciones específicas, como la **gestión de cuentas**, **procesamiento de pagos**, **prevención de fraudes** y **evaluación de riesgos**. Cada uno de estos microservicios operará de manera autónoma, lo que permite que puedan ser **actualizados, desplegados y escalados** de manera independiente sin afectar a otros.

API First Approach

La adopción de un enfoque **API First** asegura que cada microservicio exponga una API clara y bien documentada, facilitando la interacción entre los diferentes componentes del sistema, así como con terceros externos, como en el caso de **Open Finance**.

- **Contratos API Definidos**: Por ejemplo, la API para gestionar los pagos seguirá estándares REST, permitiendo que otros microservicios o aplicaciones de terceros se integren fácilmente con este servicio sin necesidad de conocer los detalles internos.

Orquestación de Servicios

Service Mesh para Gestión de Microservicios

Se implementará un **Service Mesh**, como **Istio**, para gestionar la comunicación entre los microservicios. Esto permitirá el **descubrimiento automático de servicios**, el **balanceo de carga** y la **implementación de políticas de seguridad** de forma centralizada.

- **Políticas de Seguridad**: Cada microservicio deberá autenticarse a través del **Service Mesh**, que controlará qué servicios pueden comunicarse entre sí. Esto protege los datos sensibles en tránsito entre microservicios, como las transacciones financieras.

- **Balanceo de Carga Dinámico**: En tiempos de alta demanda, como durante los picos de transacciones, el **Service Mesh** optimizará el balanceo de carga para evitar que cualquier microservicio quede sobrecargado.

Saga Pattern para Transacciones Distribuidas

El **Saga Pattern** se implementará para coordinar **transacciones distribuidas** que involucren múltiples microservicios. Cada paso de la transacción será gestionado por un microservicio, y en caso de error, se ejecutarán **transacciones compensatorias** para revertir cualquier cambio no deseado.

- **Ejemplo de Uso**: Cuando un cliente realiza una transferencia que involucra tanto el **core tradicional** como el **digital**, cada paso se realiza como parte de una saga. Si algún paso falla, el sistema revertirá automáticamente las operaciones ya ejecutadas, garantizando **consistencia eventual** sin bloquear recursos.

Escalabilidad y Resiliencia

Autoscaling para Microservicios

Mediante la configuración de **Auto Scaling Groups** en AWS ECS, los microservicios podrán escalar de manera automática en función de la demanda. Esto permitirá que el sistema responda de manera eficiente durante eventos de alto volumen de transacciones o cuando nuevas funcionalidades se introduzcan al sistema.

- **Ejemplo de Uso**: Si el sistema detecta un aumento en las transacciones durante el cierre del día, los microservicios encargados de procesar estas transacciones escalarán dinámicamente, asegurando que las operaciones continúen sin interrupciones.

Circuit Breaker Pattern para Gestión de Fallos

El uso del **Circuit Breaker Pattern** protegerá el sistema frente a **fallos en los microservicios**, evitando que los problemas en un servicio se propaguen a todo el sistema. Si un microservicio deja de responder, el patrón aislará ese componente y redirigirá las solicitudes a otras instancias o alternativas.

Gestión de Configuración y Secretos

AWS Systems Manager Parameter Store y AWS Secrets Manager

Todos los **secretos** y configuraciones críticas, como claves API y credenciales, serán gestionados a través de **AWS Secrets Manager** y **Parameter Store**. Estos servicios proporcionan almacenamiento seguro y auditable, asegurando que solo los microservicios autorizados tengan acceso a la información crítica.

Configuración como Código (CaC)

Para asegurar la coherencia en los despliegues, se implementará la **configuración como código (CaC)** utilizando herramientas como **AWS CloudFormation**. Esto permitirá desplegar, gestionar y modificar la infraestructura de microservicios de manera declarativa, evitando errores humanos y asegurando que los cambios en la infraestructura se apliquen de manera consistente.

5.3. Conclusión

La **estrategia de integración multicore** propuesta combina la sincronización de datos en tiempo real mediante buses de eventos y la flexibilidad de los microservicios, asegurando que el **core bancario tradicional** y el **nuevo core digital** operen en sinergia. Con una arquitectura desacoplada y altamente escalable, el banco podrá adaptarse rápidamente a las demandas del mercado, mantener la **integridad de los datos** y garantizar una **experiencia fluida** para los clientes. Esta estrategia es clave para apoyar la transformación digital del banco, manteniendo su competitividad y capacidad de innovación.

---

6. Gestión de Identidad y Acceso

Propósito

La **Gestión de Identidad y Acceso (IAM)** en un entorno bancario moderno busca garantizar que solo las personas y sistemas **autorizados** tengan acceso a los recursos adecuados, manteniendo un control **estricto y seguro** sobre la información sensible. En la modernización de los sistemas bancarios, que incluye tanto sistemas tradicionales como nuevos servicios digitales, es crucial que la gestión de identidades y accesos sea **escalable**, **centralizada** y alineada con los principios de **privilegio mínimo** y **cumplimiento normativo**. Este enfoque asegura una infraestructura resistente a amenazas y permite una administración de accesos eficiente en todos los niveles.

6.1. Implementación de un Sistema de Gestión de Identidades Centralizado (IAM)

Centralización de Identidades

AWS IAM para el Banco

La implementación de **AWS IAM** proporciona un sistema robusto para la **gestión centralizada de identidades** y **permisos**. Con este enfoque, el banco puede gestionar de manera **uniforme** quién tiene acceso a los recursos en la infraestructura de AWS y bajo qué condiciones, asegurando que cada operación esté completamente controlada.

- **Gestión de Usuarios y Roles**: **IAM** permite gestionar usuarios, grupos y roles para definir permisos específicos según las funciones

del usuario dentro del banco. Por ejemplo, un **rol de auditor** tendrá acceso restringido solo a los sistemas y reportes necesarios para realizar revisiones, mientras que un **administrador de sistemas** tendrá acceso más amplio a la infraestructura crítica.

- **Delegación Segura con Roles Temporales**: IAM permite la creación de **roles temporales**, permitiendo que servicios específicos accedan a recursos sin exponer credenciales. Por ejemplo, los servicios automatizados que actualizan saldos o validan pagos solo necesitarán acceso temporal a ciertas bases de datos, reduciendo el riesgo de accesos innecesarios o persistentes.

Integración con Otros Servicios de AWS

IAM se integra con servicios clave de AWS como **S3**, **RDS** y **Lambda**, lo que facilita la aplicación de **políticas granulares**. Esto asegura que las bases de datos sensibles, como la información de cuentas de clientes o transacciones financieras, solo sean accesibles por los servicios y personas que realmente lo necesitan, reduciendo el riesgo de acceso no autorizado.

Autenticación y Autorización

AWS Cognito para la Gestión de Usuarios Finales

AWS Cognito se utilizará para gestionar las identidades de los **usuarios finales**, permitiendo una **autenticación segura** tanto para las aplicaciones móviles como para las plataformas web del banco. Cognito soporta la **federación de identidades**, lo que facilita la integración con otros proveedores de identidad (como **Google**, **Facebook** o **Active Directory**).

- **Autenticación Multi-Factor (MFA)**: AWS Cognito ofrece **MFA** como una capa adicional de seguridad para la autenticación de los usuarios, lo que es crucial para las transacciones sensibles, como la **aprobación de transferencias** o el acceso a información de cuentas.

Federación de Identidades

En entornos híbridos, donde el banco puede operar con múltiples sistemas internos y externos, la **federación de identidades** con proveedores como **Active Directory** o **Auth0** permitirá unificar la gestión de accesos. Esto simplifica la administración de identidades y asegura que, aunque existan múltiples puntos de autenticación, todo el control se centralice en una **plataforma segura y auditable**.

6.2. Uso de Single Sign-On (SSO)

SSO para Usuarios Internos

AWS SSO para el Banco

AWS Single Sign-On (SSO) proporcionará a los empleados y administradores del banco un **acceso unificado** a todas las aplicaciones y servicios autorizados. Los empleados solo necesitarán un inicio de sesión único para acceder a sus recursos, simplificando el proceso de autenticación.

- **Federación con SAML y Active Directory**: Al integrar AWS SSO con **Active Directory** utilizando **SAML**, los empleados del banco podrán acceder a todos los sistemas de AWS utilizando sus **credenciales corporativas**. Esto reduce la complejidad de la autenticación, mejorando la seguridad al evitar la creación de credenciales adicionales para cada sistema.

Ventajas Operativas

La implementación de SSO no solo mejora la experiencia de usuario, sino que también **facilita la gestión de accesos**. Por ejemplo, cuando un empleado cambia de departamento o deja la organización, sus accesos a múltiples aplicaciones pueden ser gestionados de forma **centralizada** y **rápida**, minimizando los riesgos asociados con cuentas huérfanas o permisos desactualizados.

Simplificación de Autenticación

Reducción de la Complejidad en el Manejo de Contraseñas

Con **SSO**, se reduce la necesidad de múltiples contraseñas, minimizando la **fatiga de contraseñas** y el riesgo de que los usuarios utilicen contraseñas débiles o repetidas en múltiples aplicaciones.

Gestión de Accesos Centralizada

Todas las políticas de autenticación y autorización estarán centralizadas en una plataforma única, permitiendo a los administradores de TI del banco **controlar y auditar** los accesos de manera más eficiente. Esto es especialmente importante para cumplir con los **requisitos normativos** que exigen un control detallado de quién accede a qué sistemas y cuándo.

6.3. Políticas de Acceso Basadas en Roles (RBAC)

Definición de Roles y Políticas

RBAC para Control de Acceso Granular

El banco adoptará el modelo de **Control de Acceso Basado en Roles (RBAC)**, asignando **roles específicos** que reflejan las responsabilidades de los empleados o servicios automatizados. Por ejemplo:

- Un **analista de riesgo** tendrá acceso a los módulos que gestionan la evaluación de riesgos financieros, pero no tendrá acceso a datos de clientes individuales.
- Un **desarrollador** tendrá acceso a los entornos de desarrollo, pero no a los sistemas de producción.

Matriz de Roles y Permisos

Se definirá una **matriz clara de roles y permisos**, detallando qué acciones están permitidas para cada rol y en qué contexto. Por ejemplo, el acceso a la base de datos de clientes solo estará permitido para usuarios con el rol de **gestión de cuentas** y únicamente para realizar operaciones específicas.

Gestión de Permisos

Aplicación del Principio de Privilegio Mínimo

La implementación del **principio de privilegio mínimo** asegura que los usuarios solo puedan acceder a los recursos necesarios para desempeñar su función. Esto reduce la **superficie de ataque** y minimiza el impacto potencial en caso de que una cuenta sea comprometida.

Revisión Periódica de Permisos

Los permisos asignados serán revisados regularmente para garantizar que sean apropiados y estén actualizados. Esto incluye la revocación de accesos innecesarios, lo cual es crítico en un entorno bancario donde los empleados pueden cambiar de roles o departamentos con frecuencia.

Auditorías y Monitoreo

Auditoría Continua con AWS CloudTrail

AWS CloudTrail será utilizado para registrar todas las acciones y accesos en la infraestructura de AWS, incluyendo la creación, modificación y eliminación de recursos. Esto permitirá auditorías detalladas y revisiones forenses en caso de un incidente de seguridad.

Monitorización Activa y Alertas de Seguridad

El banco implementará sistemas de **alertas automáticas** que notificarán al equipo de seguridad sobre cualquier intento de acceso no autorizado o actividad inusual, como intentos de inicio de sesión fallidos o accesos fuera del horario laboral. Estos sistemas estarán respaldados por **Amazon CloudWatch** y **Amazon GuardDuty**.

Conclusión

La **gestión de identidad y acceso** del banco garantizará que los recursos críticos estén protegidos, que los accesos estén controlados de manera

estricta y que los sistemas cumplan con las normativas de seguridad. Con una combinación de **AWS IAM**, **Cognito**, **Single Sign-On** y **RBAC**, la organización podrá gestionar las identidades y accesos de manera escalable, auditada y eficiente. Esto no solo mejora la seguridad, sino que también facilita la administración diaria y asegura que el banco esté preparado para adaptarse a los cambios en su infraestructura tecnológica y regulatoria.

---

7. Estrategia de API Internas y Externas

Propósito

La estrategia de **APIs Internas y Externas** está diseñada para proporcionar una **arquitectura altamente interoperable** que optimice las comunicaciones internas entre sistemas y facilite la integración con socios externos y fintechs. En un contexto bancario en proceso de modernización, las APIs se posicionan como el **nexo de integración clave** entre los servicios bancarios core, los microservicios, y los ecosistemas externos de **Open Finance**, permitiendo a los desarrolladores internos y externos interactuar con el sistema de manera segura y eficiente. Esta estrategia no solo mejora la **agilidad** y **escalabilidad** del sistema, sino que también impulsa la **innovación** al permitir la creación de nuevos productos y servicios sobre la infraestructura bancaria.

7.1. APIs Internas

Las **APIs internas** forman la columna vertebral de la infraestructura tecnológica del banco, conectando microservicios y sistemas legados de manera fluida. Son cruciales para garantizar una **comunicación ágil y desacoplada** dentro de la arquitectura de microservicios, permitiendo una mejor **modularidad** y **mantenimiento** del sistema.

REST y GraphQL

- **RESTful APIs**: Las **APIs RESTful** son esenciales para establecer una comunicación clara y eficiente entre microservicios. Proporcionan un estándar sencillo y flexible para acceder a datos y ejecutar operaciones utilizando **verbos HTTP** (GET, POST, PUT, DELETE) y respuestas **JSON**. Esto es ideal para servicios como la gestión de cuentas, procesamiento de pagos, y gestión de usuarios. La simplicidad y la amplia adopción de REST aseguran que los equipos puedan desarrollar e integrar nuevos servicios rápidamente.

- **GraphQL**: Para mejorar la eficiencia en las consultas de datos, **GraphQL** permite solicitar solo los datos necesarios, evitando la sobrecarga de información y optimizando el rendimiento en tiempo

real. Esto es particularmente útil en operaciones complejas, como la consulta de múltiples fuentes de datos internas para informes financieros, gestión de riesgos o análisis de datos en tiempo real.

Seguridad y Autorización

- **OAuth 2.0 y OpenID Connect**: Implementar **OAuth 2.0** como estándar de **autorización** asegura que los accesos a las APIs estén controlados y sean seguros. **OpenID Connect**, utilizado para la **autenticación**, proporciona una capa adicional de seguridad para validar la identidad de los usuarios antes de que accedan a los servicios críticos. Este enfoque protege la información sensible y asegura que solo los servicios y usuarios autorizados interactúen con las APIs internas.

- **CORS (Cross-Origin Resource Sharing)**: Al implementar **CORS**, se restringe el acceso a las APIs internas solo a dominios confiables, mejorando la seguridad y previniendo ataques **XSS (Cross-Site Scripting)** y otras amenazas de inyección de código malicioso.

Monitoreo y Logging

- **Amazon CloudWatch**: **CloudWatch** monitorea las métricas de rendimiento de las APIs internas en tiempo real, lo que permite a los administradores detectar problemas como **alta latencia** o **errores de conexión** antes de que afecten a los usuarios. Esto asegura una operativa eficiente y una rápida resolución de problemas.

- **AWS X-Ray**: Con **X-Ray**, se puede trazar el recorrido de cada solicitud a través de los diferentes microservicios del sistema. Esto es crucial para identificar **cuellos de botella** o posibles fallos en la arquitectura, permitiendo una optimización continua y manteniendo un alto nivel de disponibilidad del sistema.

7.2. APIs Externas

Las **APIs externas** permiten que el banco exponga sus servicios y datos a socios comerciales, fintechs y otras plataformas dentro del ecosistema de **Open Finance**. Estas APIs son críticas para fomentar la innovación, permitir integraciones con terceros y mejorar la experiencia del cliente mediante la habilitación de nuevos servicios digitales.

Estándares de OpenAPI

- **Documentación con OpenAPI**: Para facilitar la integración con socios externos, todas las **APIs externas** deben seguir el estándar **OpenAPI**, lo que asegura que estén **bien documentadas** y sean fáciles de implementar. Esto permite a los desarrolladores externos comprender

rápidamente cómo interactuar con los servicios del banco, acelerando el proceso de integración.

- **Swagger**: **Swagger** es una herramienta que facilita la creación automática de documentación y la **validación de APIs**. Utilizando Swagger, el banco puede garantizar que las APIs externas sean consistentes y estén siempre actualizadas, reduciendo errores y mejorando la adopción por parte de terceros.

Conformidad con Normativas de Open Finance

- **Cumplimiento Regulatorio**: Las **APIs externas** deben cumplir con las normativas de **Open Finance**, lo que implica mantener altos estándares de **protección de datos** y **transparencia**. Esto incluye asegurar que los datos de los clientes se procesen de manera segura y cumpliendo con normativas locales e internacionales de protección de datos.

- **Acceso Seguro**: Implementar **OAuth 2.0** para la autenticación y autorización garantiza que solo usuarios y aplicaciones autorizadas puedan acceder a las APIs externas. Este enfoque asegura que los datos sensibles no sean comprometidos y que cada solicitud esté debidamente validada antes de procesarse.

Control de Acceso y Tasa de Solicitudes

- **Rate Limiting**: Para evitar **sobrecargas** en el sistema o posibles abusos por parte de aplicaciones externas, se implementarán políticas de **rate limiting**, que controlarán el número de solicitudes permitidas en un tiempo determinado. Esto asegura una distribución equitativa de los recursos y protege el sistema contra picos de demanda inesperados.

- **IP Whitelisting**: Para mejorar la seguridad, solo las **direcciones IP previamente autorizadas** podrán acceder a las APIs externas. Esto limita el acceso a socios comerciales de confianza y previene ataques desde ubicaciones no autorizadas.

7.3. Mensajería

La **mensajería** asíncrona es clave para permitir la comunicación entre los servicios internos y externos del banco, facilitando una integración **desacoplada** y resiliente. Los sistemas de mensajería permiten que los servicios se comuniquen entre sí sin necesidad de una conexión directa, mejorando la escalabilidad y la tolerancia a fallos.

Serialización de Mensajes

- **JSON**: El formato de **JSON** será el estándar para la **serialización de mensajes** en las APIs, dado su balance entre **simplicidad**,

compatibilidad y **legibilidad**. JSON es ideal para la mayoría de las comunicaciones internas y externas, ofreciendo facilidad de uso y flexibilidad.

- **Protobuf (Protocol Buffers)**: En servicios donde la **eficiencia** y **baja latencia** son críticas, como el procesamiento de pagos en tiempo real o la sincronización de datos entre sistemas core, se adoptará **Protobuf**. Este formato ofrece una mayor **compresión** y **velocidad de procesamiento** en comparación con JSON, lo que lo hace ideal para manejar grandes volúmenes de datos de manera eficiente.

Integración y Comunicación

- **Apache Kafka**: **Kafka** será utilizado como la plataforma principal de mensajería para la transmisión de **eventos** y **datos** entre microservicios. Kafka soporta grandes volúmenes de mensajes en tiempo real, lo que es crucial para mantener la consistencia y confiabilidad de las operaciones bancarias, particularmente en transacciones financieras críticas.
- **AWS SNS/SQS**: Para la mensajería **asíncrona** y **desacoplada**, se utilizarán **SNS** y **SQS** de AWS. SNS gestionará la publicación de eventos que deban ser procesados por múltiples servicios simultáneamente, mientras que **SQS** permitirá la gestión de colas de mensajes, asegurando que las solicitudes sean procesadas en el orden correcto y sin pérdida de información.

7.4. Conclusión

La estrategia de **APIs internas y externas** establece una arquitectura sólida que facilita la **interoperabilidad**, **seguridad**, y **escalabilidad** en el ecosistema bancario. Al implementar APIs RESTful, GraphQL, y sistemas de mensajería robustos como Kafka y AWS SNS/SQS, el banco puede responder rápidamente a las demandas del mercado, mejorar la experiencia del cliente y abrir nuevas oportunidades de negocio a través de **Open Finance**. Esta infraestructura no solo respalda la operación actual del banco, sino que también lo prepara para futuras innovaciones y expansiones.

---

8. Modelo de Gobierno de APIs y Microservicios

Propósito

El **Modelo de Gobierno de APIs y Microservicios** tiene como objetivo garantizar la **gestión efectiva**, **seguridad** y **alineación estratégica** de

los servicios internos y externos en un entorno bancario de alta complejidad. Dado el papel central que juegan las APIs y los microservicios en la infraestructura del banco, este modelo asegura que se cumplan las normativas de la industria, se maximice la **eficiencia operativa**, y se favorezca la **innovación** a través de una arquitectura flexible y escalable. La gobernanza asegura que todas las decisiones técnicas estén alineadas con los objetivos empresariales, y que los servicios puedan escalar de forma segura y controlada dentro del ecosistema bancario.

8.1. Estructura Organizacional y Roles

Comité de Gobierno de APIs y Microservicios

Este comité supervisa el ciclo de vida completo de las APIs y microservicios, garantizando que cada desarrollo cumpla con los estándares del banco y se alinee con las metas de negocio.

- **Funciones**:
  - **Definir políticas y estándares de desarrollo** para APIs y microservicios.
  - **Revisar y aprobar** nuevas APIs y actualizaciones importantes.
  - **Asegurar la alineación** de las APIs con la estrategia digital del banco.

Arquitectos Empresariales

Responsables de alinear las decisiones tecnológicas con los objetivos estratégicos del banco.

- **Funciones**:
  - **Diseñar la arquitectura global** de APIs y microservicios.
  - **Evaluar y proponer tecnologías** que garanticen la escalabilidad y seguridad.
  - **Supervisar la implementación** de patrones de integración y tecnologías clave, como el API Gateway y arquitecturas dirigidas por eventos.

Propietarios de APIs

Cada API crítica tendrá un propietario responsable de su ciclo de vida completo.

- **Funciones**:
  - **Asegurar el cumplimiento** con las políticas de seguridad y versiones.
  - **Supervisar el rendimiento y adopción** de la API, haciendo ajustes cuando sea necesario.

    - **Colaborar con equipos de desarrollo** para priorizar mejoras y solucionar problemas.

Desarrolladores de Microservicios

Los equipos de desarrollo son responsables de crear y mantener los microservicios dentro de los lineamientos arquitectónicos.

- **Funciones**:
    - **Desarrollar microservicios** alineados con las especificaciones arquitectónicas.
    - **Realizar pruebas automatizadas** y participar en revisiones de código.
    - **Trabajar en integración continua y despliegue continuo** (CI/CD) para asegurar una entrega rápida y segura.

8.2. Políticas y Estándares

Estándares de Desarrollo

- **REST y GraphQL**: Definir cuándo utilizar APIs **RESTful** o **GraphQL** para asegurar la interoperabilidad dentro de la arquitectura.
- **Principios SOLID y DDD**: Utilizar los principios de **SOLID** y **Domain-Driven Design (DDD)** para diseñar servicios modulares y escalables.

Políticas de Seguridad

- **Autenticación y Autorización**: Todas las APIs deben implementar **OAuth 2.0** para autorización y **OpenID Connect** para la autenticación.
- **Cifrado de Datos**: El cifrado en tránsito y en reposo es obligatorio para proteger la información sensible.

Políticas de Versionado

- **Versionado semántico (SemVer)**: Utilizar versiones claras para asegurar compatibilidad hacia atrás y documentar claramente los cambios y las migraciones.

8.3. Ciclo de Vida de las APIs y Microservicios

Desarrollo y Pruebas

- **Automatización de Pruebas**: Utilizar herramientas como **SwaggerHub**, **Postman**, **Gherkin** y **Cucumber** para pruebas continuas y validación de APIs.
- **Desarrollo Ágil**: Implementar **metodologías ágiles** para asegurar iteraciones rápidas y eficientes.

Despliegue y Gestión

- **AWS CodePipeline y CodeDeploy**: Automatizar el despliegue de APIs y microservicios mediante pipelines CI/CD, asegurando una integración fluida y minimizando riesgos en la producción.

Monitoreo y Auditoría

- **Amazon CloudWatch y AWS X-Ray**: Implementar estas herramientas para monitorear el rendimiento en tiempo real y realizar auditorías sobre el uso de las APIs y microservicios, asegurando el cumplimiento normativo y la seguridad.

8.4. Documentación y Comunicación

Repositorio Central de APIs

Un repositorio único donde se almacenen todas las APIs y sus documentaciones, accesibles para los equipos internos y socios externos.

- **Swagger y OpenAPI**: Asegurar que las APIs estén bien documentadas y actualizadas.

Comunicación Transparente

- **Informes de Rendimiento**: Publicar informes regulares sobre el rendimiento de las APIs y los microservicios, compartiendo esta información con los stakeholders relevantes.

8.5. Adaptación al Marco TOGAF

Arquitectura de Negocio (Business Architecture)

- **Gobernanza Alineada con los Objetivos Empresariales**: Definir claramente los roles y responsabilidades para la gobernanza de APIs, asegurando que todas las decisiones estén alineadas con la estrategia de negocio y que los servicios sean eficientes y sostenibles.

Arquitectura de Datos (Data Architecture)

- **Seguridad y Gestión de Datos**: Las APIs y microservicios deben cumplir con todas las normativas de seguridad y protección de datos. Implementar mecanismos de **autenticación**, **autorización** y **cifrado** de datos para asegurar la confidencialidad, integridad y disponibilidad de la información.

Arquitectura de Aplicaciones (Applications Architecture)

- **Patrones de Integración y API Gateway**: Implementar arquitecturas de **API Gateway** y arquitecturas **basadas en eventos** para permitir la comunicación asíncrona y desacoplada entre los microservicios, optimizando la integración entre sistemas.

Arquitectura Tecnológica (Technology Architecture)

- **Infraestructura Escalable**: Usar **Amazon ECS**, **AWS CloudFormation** y **Amazon CloudWatch** para asegurar que la infraestructura que soporta las APIs y microservicios sea **escalable** y **resiliente**, soportando el crecimiento futuro sin comprometer la seguridad ni la disponibilidad.

Es esencial para cualquier estrategia de **gobernanza de APIs y microservicios** no solo definir los **servicios** que se van a implementar, sino también documentar exhaustivamente los **flujos de integración**, **colas**, **adaptadores**, **conectores**, y otros componentes relacionados con la comunicación entre sistemas. La falta de documentación clara de estos flujos puede generar **ineficiencias** y **problemas de escalabilidad** a largo plazo, especialmente en una arquitectura bancaria compleja.

Aquí tienes la versión con la numeración corregida para reflejar el formato adecuado:

8.6. Documentación de Artefactos y Entregables de Integración

Propósito

El objetivo principal de esta sección es estructurar la **documentación de artefactos y entregables** en el contexto de la **gobernanza de APIs y microservicios** de un sistema bancario, siguiendo un marco como **TOGAF**. Esto asegura que todas las capas de la integración (APIs, microservicios, eventos, colas de mensajes, adaptadores) se documenten de manera clara y alineada con los principios de la arquitectura empresarial, permitiendo así una gestión eficiente, escalabilidad, y cumplimiento normativo. La distinción entre **entregables**, **artefactos** y **bloques de construcción** es clave para lograr una arquitectura organizada y trazable.

8.6.1. Entregables

Los **entregables** son documentos formales que se producen durante las fases de desarrollo y evolución de la arquitectura empresarial. Estos se entregan a las partes interesadas y contienen múltiples **artefactos** que describen los componentes clave de la integración.

*8.6.1.1. **Documento de Arquitectura de Integración***

- **Propósito**: Proporcionar una visión general de la arquitectura de integración que cubre todos los aspectos técnicos de los microservicios y APIs implementados, así como los flujos de datos y sistemas de mensajería.
- **Contenido**:

    - **Visión Estratégica**: Alineación de la integración con los objetivos del banco.
    - **Principios Arquitectónicos**: Principios de interoperabilidad, seguridad, y escalabilidad.
    - **Resumen de Artefactos y Bloques de Construcción**.
- **Artefactos Incluidos**: Diagramas de flujo de datos, diagramas de integración de servicios, especificaciones de APIs y modelos de mensajería.
- **Bloques de Construcción**: Servicios de integración, APIs REST/GraphQL, microservicios de negocio, conectores.
- **Formato**: Documento formal alineado con las guías de TOGAF para la arquitectura de integración.
- **Herramientas**: ArchiMate, Visio, Lucidchart.

*8.6.1.2. **Catálogo de Servicios y APIs***

- **Propósito**: Centralizar la documentación de todas las APIs internas y externas expuestas por el banco, con detalles técnicos, versiones y dependencias.
- **Contenido**:
    - **Descripción de APIs**: Para cada API, se detalla la funcionalidad, puntos finales (endpoints), parámetros, respuestas y políticas de seguridad.
    - **Versionado Semántico**: Historial de versiones, documentando cambios y actualizaciones.
    - **Dependencias entre Microservicios**: Relación entre APIs y microservicios que las consumen o exponen.
- **Artefactos Incluidos**: Documentación Swagger/OpenAPI para cada API.
- **Bloques de Construcción**: APIs REST/GraphQL, microservicios de negocio.
- **Formato**: Repositorio centralizado de APIs, siguiendo las mejores prácticas de TOGAF para gestión de catálogos.
- **Herramientas**: SwaggerHub, Postman, Confluence.

*8.6.1.3. **Especificación Técnica de Integración***

- **Propósito**: Detallar las especificaciones técnicas de los componentes de integración, incluyendo colas de mensajes, adaptadores y conectores.
- **Contenido**:
    - **Descripción técnica de adaptadores y conectores**.
    - **Configuración de colas de mensajes y buses de eventos** (Kafka, AWS SQS).
    - **Especificación de transformaciones de datos entre sistemas**.

- **Artefactos Incluidos**: Diagramas de componentes, esquemas de mensajería, especificaciones de adaptadores.
- **Bloques de Construcción**: Conectores, adaptadores, buses de mensajería.
- **Formato**: Documento de especificación técnica alineado con TOGAF.
- **Herramientas**: Kafka Schema Registry, MuleSoft, Talend.

*8.6.1.4. **Guía de Seguridad y Cumplimiento***

- **Propósito**: Garantizar que todos los mecanismos de integración, APIs y microservicios cumplen con los requisitos de seguridad y normativas regulatorias.
- **Contenido**:
  - **Autenticación y autorización**: Uso de OAuth 2.0 y OpenID Connect.
  - **Políticas de cifrado de datos en tránsito y en reposo**.
  - **Auditoría y trazabilidad de transacciones**.
- **Artefactos Incluidos**: Políticas de seguridad, diagramas de autenticación, esquemas de cifrado.
- **Bloques de Construcción**: APIs protegidas por OAuth 2.0, sistemas de cifrado TLS/KMS.
- **Formato**: Guía formal de seguridad alineada con TOGAF y regulaciones como GDPR.
- **Herramientas**: AWS IAM, AWS KMS, GuardDuty.

8.6.2. Artefactos

Los **artefactos** son componentes técnicos individuales que forman parte de los entregables. Estos proporcionan detalles sobre la implementación y operación de los bloques de construcción.

*8.6.2.1. **Diagrama de Integración de Servicios***

- **Descripción**: Representa las interacciones entre microservicios, APIs y sistemas externos.
- **Propósito**: Visualizar el flujo de datos y la interoperabilidad entre sistemas.
- **Bloques de Construcción**: Microservicios, APIs, conectores.
- **Herramientas**: C4 Model, ArchiMate.

*8.6.2.2. **Modelo de Datos de Mensajería***

- **Descripción**: Define el formato y estructura de los datos que se transmiten entre microservicios mediante colas y eventos.
- **Propósito**: Garantizar la consistencia en el intercambio de datos.
- **Bloques de Construcción**: Kafka, AWS SQS, transformadores de datos.
- **Herramientas**: Kafka Schema Registry, Protobuf, JSON Schema.

*8.6.2.3. **Esquema de Seguridad API***

- **Descripción**: Detalla cómo se aplica la seguridad a nivel de API, incluyendo autenticación y autorización.
- **Propósito**: Asegurar el control de acceso a los datos y servicios.
- **Bloques de Construcción**: OAuth 2.0, OpenID Connect, AWS IAM.
- **Herramientas**: Swagger, OpenAPI.

*8.6.2.4. **Matriz de Dependencias de Microservicios***

- **Descripción**: Identifica las relaciones entre los microservicios y las APIs que consumen o exponen.
- **Propósito**: Gestionar la interdependencia de los componentes.
- **Bloques de Construcción**: Microservicios, APIs.
- **Herramientas**: Jira, Confluence.

8.6.3. Bloques de Construcción

Los **bloques de construcción** son los componentes reutilizables y modulares que se implementan en el sistema. Estos se documentan a través de los entregables y artefactos.

*8.6.3.1. **APIs REST/GraphQL***

- **Descripción**: Interfaces expuestas por los microservicios que permiten la comunicación entre componentes internos y externos.
- **Propósito**: Facilitar la interoperabilidad y el consumo de servicios.
- **Artefactos relacionados**: Documentación OpenAPI, políticas de autenticación.
- **Herramientas**: Swagger, GraphQL Playground.

*8.6.3.2. **Bus de Mensajería (Kafka, AWS SQS)***

- **Descripción**: Mecanismos de mensajería asincrónica entre microservicios y otros sistemas.
- **Propósito**: Desacoplar los servicios y asegurar la transmisión confiable de eventos.
- **Artefactos relacionados**: Esquema de mensajería, diagramas de flujo de eventos.
- **Herramientas**: Kafka, AWS SQS.

*8.6.3.3. **Adaptadores y Conectores***

- **Descripción**: Componentes que permiten la integración de sistemas legados con microservicios modernos.
- **Propósito**: Habilitar la interoperabilidad entre tecnologías heterogéneas.
- **Artefactos relacionados**: Especificaciones de conectores, diagramas de integración.
- **Herramientas**: MuleSoft, Talend.

*8.6.3.4. **Microservicios de Negocio***

- **Descripción**: Bloques funcionales que encapsulan funcionalidades específicas del negocio, como la gestión de cuentas o pagos.
- **Propósito**: Modularizar las operaciones del sistema bancario para facilitar el mantenimiento y escalabilidad.
- **Artefactos relacionados**: Matrices de dependencias, diagramas de componentes.
- **Herramientas**: Kubernetes, Docker, Spring Boot.

8.7. Conclusión

Este modelo de **Gobernanza de APIs y Microservicios** asegura que el banco pueda operar con agilidad y seguridad, apoyando la innovación tecnológica y garantizando el cumplimiento normativo en todo momento.

La **Documentación de Artefactos y Entregables de Integración** es esencial para mantener la **interoperabilidad**, **seguridad**, y **escalabilidad** del sistema bancario moderno. Los formatos como **diagramas de secuencia**, **documentación Swagger**, **BPMN** y **diagramas de componentes** permiten capturar todos los aspectos de la interacción entre sistemas, desde la comunicación síncrona con APIs REST hasta la mensajería asincrónica y los adaptadores para sistemas legacy. Una documentación clara y precisa es la base para una operación eficiente y garantiza que los equipos puedan trabajar de manera **ágil** y **segura** en una arquitectura bancaria cada vez más compleja.

En una estrategia de gobernanza de APIs y microservicios dentro de un sistema bancario, es fundamental diferenciar claramente entre **entregables** (documentos formales que consolidan la arquitectura y las decisiones estratégicas), **artefactos** (componentes técnicos detallados dentro de los entregables) y **bloques de construcción** (componentes reutilizables que forman la base de la solución técnica). Este enfoque asegura que la arquitectura sea trazable, escalable y alineada con los objetivos estratégicos del banco, permitiendo una implementación efectiva y un mantenimiento eficiente a lo largo del ciclo de vida de la arquitectura.

---

9. Plan de Migración Gradual

9.1. Propósito

El **Plan de Migración Gradual** tiene como objetivo llevar al banco desde su infraestructura tradicional a una arquitectura moderna, basada en **microservicios** y **nube**, con un enfoque en la continuidad operativa, la seguridad y la optimización del rendimiento. La clave de esta migración es

su **carácter iterativo**, minimizando riesgos al migrar componentes de manera controlada, comenzando con los sistemas menos críticos y validando cada fase antes de proceder a la siguiente. Este enfoque asegura que la modernización sea **escalable**, **resiliente** y sin interrupciones significativas para el negocio.

9.2. Fase 1: Evaluación y Planificación

La primera fase se centra en la evaluación profunda de la situación actual del banco y la planificación detallada de la transición a la nueva arquitectura, asegurando una migración eficiente y segura.

9.2.1. Evaluación de Compatibilidad

- Identificar qué partes del sistema actual son **reutilizables** y qué componentes necesitan **migrarse o rediseñarse**. Esta evaluación incluye aplicaciones, bases de datos, infraestructura y servicios.

9.2.2. Mapeo de Dependencias

- Desarrollar un **mapa de dependencias** que permita entender las relaciones críticas entre los sistemas, identificando los componentes que deben ser migrados juntos para evitar interrupciones en las operaciones.

9.2.3. Planificación del Roadmap de Migración

- Crear un **roadmap** con las fases de migración, asignación de recursos y fechas clave. Las fases deben priorizar los componentes menos críticos y avanzar hacia los más sensibles.

9.2.4. Definición de Criterios de Éxito

- Establecer **criterios de éxito** para cada fase del proceso, asegurando que los requisitos de rendimiento, seguridad y funcionalidad se cumplan antes de continuar con la siguiente fase.

9.2.5. Pruebas Automatizadas con Gherkin y Cucumber

- Definir pruebas con **Gherkin** para evaluar la funcionalidad de cada componente y usar **Cucumber** para la ejecución automatizada de las pruebas durante todo el proceso de migración.

9.3. Fase 2: Migración Inicial de Componentes No Críticos

La segunda fase se enfoca en la migración de componentes **no críticos** para validar la infraestructura y realizar ajustes sin impactar las operaciones principales del banco.

9.3.1. Despliegue de Componentes No Críticos

- Migrar herramientas internas o servicios secundarios para **probar la infraestructura en la nube** y realizar ajustes sin riesgo.

9.3.2. Configuración de AWS CloudFormation

- Utilizar **AWS CloudFormation** para desplegar la infraestructura en la nube mediante **Infraestructura como Código (IaC)**, asegurando que todos los componentes como **VPCs**, **subredes**, **grupos de seguridad**, y **clústeres ECS** sean gestionados de manera consistente y segura.

9.3.3. Automatización y Gestión de Cambios

- Implementar la **gestión de cambios** a través de IaC para garantizar que las configuraciones puedan ser auditadas y revertidas fácilmente en caso de errores.

9.3.4. Herramientas de Amazon para Configuración como Código

- Utilizar las herramientas de AWS para la configuración y gestión de la infraestructura:
  - **AWS CloudFormation**: Para desplegar la infraestructura.
  - **AWS CodePipeline**: Para automatizar el pipeline de integración continua.
  - **AWS CodeBuild**: Para compilar y probar el código.
  - **AWS CodeDeploy**: Para desplegar aplicaciones y servicios de manera automática.

9.4. Fase 3: Migración de Componentes Críticos

En esta fase se migra la parte más **sensible y crítica** del sistema, garantizando la **disponibilidad** y **seguridad** de los componentes esenciales para las operaciones del banco.

9.4.1. Migración de Sistemas Core Bancarios

- Migrar los sistemas core bancarios con **replicación en tiempo real** y utilizando **AWS Multi-AZ** para asegurar que los datos estén disponibles incluso en casos de fallo en una zona de disponibilidad.

9.4.2. Despliegue de Microservicios en ECS

- Los microservicios que gestionan transacciones financieras, prevención de fraudes y otros sistemas críticos se desplegarán en **Amazon ECS**, garantizando **alta disponibilidad** y **escalabilidad**.

9.4.3. Integración de Kafka y Bases de Datos

- Asegurar la correcta integración de los microservicios con las bases de datos y el sistema de mensajería en tiempo real utilizando **Apache Kafka** para sincronizar los datos entre los diferentes cores bancarios.

9.4.4. Monitorización en Tiempo Real

- Configurar **Amazon CloudWatch** y **AWS X-Ray** para la monitorización constante del rendimiento del sistema y la detección de problemas en tiempo real.

9.4.5. Pruebas Continuas Automatizadas

- Continuar ejecutando **pruebas automatizadas** con Cucumber para garantizar que los componentes críticos migren de manera segura y eficiente.

9.5. Fase 4: Optimización y Monitoreo Continuo

Esta fase se centra en la **optimización del sistema** y el monitoreo constante para asegurar que todo funcione correctamente en producción, ajustando la capacidad y el rendimiento según sea necesario.

9.5.1. Optimización Basada en Métricas

- Utilizar los datos recopilados durante las fases anteriores para realizar **ajustes** en la infraestructura, mejorando la asignación de recursos y la eficiencia operativa.

9.5.2. Monitoreo Proactivo y Escalabilidad

- Implementar la **escalabilidad automática** mediante **Amazon CloudWatch**, permitiendo que los servicios se ajusten en función de la demanda y evitando posibles interrupciones por sobrecarga.

9.5.3. Auditorías de Seguridad

- Realizar auditorías de seguridad para asegurar que los controles de acceso, como los roles de **AWS IAM** y las políticas de seguridad, estén alineados con las normativas y estándares de protección de datos.

9.5.4. Automatización de Pruebas Continuas

- Ajustar y continuar ejecutando las **pruebas automatizadas** con Cucumber para validar todos los cambios realizados, asegurando que cualquier ajuste o nueva funcionalidad se valide antes de implementarse en producción.

9.6 Cronograma Detallado de Migración

Este cronograma de migración ha sido ajustado para un plazo de **3 meses**, comprimiendo las actividades de cada fase sin comprometer la seguridad o

la calidad del proceso. El enfoque sigue siendo iterativo, pero con un avance más rápido, orientado a completar la migración de manera eficiente en un plazo reducido. Se prioriza la migración segura y el monitoreo continuo para garantizar la estabilidad del sistema durante todo el proceso.

9.6.1. Fase 1: Evaluación y Planificación (Semanas 1-2)

Actividades Principales:

1. **Evaluación de Compatibilidad de Infraestructura** – 3 días
    - Análisis rápido de los componentes actuales para determinar qué sistemas son reutilizables o necesitan ser rediseñados.
    - Recursos involucrados: Arquitectos de TI, Ingenieros de Infraestructura.
2. **Mapeo de Dependencias** – 3 días
    - Creación de un mapa detallado de las dependencias críticas entre sistemas.
    - Recursos involucrados: Arquitectos de TI, Desarrolladores.
3. **Planificación del Roadmap de Migración** – 4 días
    - Establecer un roadmap claro con prioridades, fechas y recursos.
    - Recursos involucrados: Gerentes de Proyectos, Arquitectos Empresariales.
4. **Definición de Criterios de Éxito** – 2 días
    - Definir métricas claras de rendimiento, seguridad y funcionalidad.
    - Recursos involucrados: Arquitectos de TI, Gerentes de Calidad.
5. **Preparación de Pruebas Automatizadas con Gherkin y Cucumber** – 2 días
    - Definir y preparar las pruebas automatizadas para cada fase del proceso.
    - Recursos involucrados: Equipos de QA, Desarrolladores.

Duración Total: 2 semanas

9.6.2. Fase 2: Migración Inicial de Componentes No Críticos (Semanas 3-4)

Actividades Principales:

1. **Despliegue de Herramientas Internas y Servicios No Críticos** – 1 semana
    - Migrar componentes no críticos y validar la infraestructura en la nube.
    - Recursos involucrados: Desarrolladores, Administradores de Sistemas.
2. **Configuración de AWS CloudFormation para IaC** – 5 días

    - Desplegar infraestructura mediante **CloudFormation**, incluyendo VPCs, subredes, y clústeres ECS.
    - Recursos involucrados: Ingenieros de Infraestructura, Arquitectos de Nube.
3. **Ajuste de Configuraciones y Monitorización** – 4 días
    - Ajustar infraestructura y activar el monitoreo con **Amazon CloudWatch**.
    - Recursos involucrados: Ingenieros de Operaciones, Arquitectos de TI.
4. **Revisión y Validación de Pruebas Automatizadas** – 3 días
    - Validar las pruebas automáticas y corregir errores en la infraestructura.
    - Recursos involucrados: Equipos de QA, Desarrolladores.

Duración Total: 2 semanas

9.6.3. Fase 3: Migración de Componentes Críticos (Semanas 5-8)

Actividades Principales:

1. **Migración de Sistemas Core Bancarios** – 2 semanas
    - Migrar las funciones críticas del core bancario con replicación en tiempo real.
    - Recursos involucrados: Desarrolladores, Administradores de Base de Datos, Arquitectos Empresariales.
2. **Despliegue de Microservicios Críticos en ECS** – 1 semana
    - Desplegar microservicios críticos en **Amazon ECS** para operaciones esenciales.
    - Recursos involucrados: Ingenieros de Nube, Desarrolladores.
3. **Integración de Kafka y Bases de Datos** – 5 días
    - Configurar Kafka para sincronización de datos y eventos entre sistemas core.
    - Recursos involucrados: Administradores de Sistemas, Arquitectos de Datos.
4. **Monitorización Intensiva y Pruebas Continuas** – 5 días
    - Configurar y activar monitorización constante con **Amazon CloudWatch** y realizar pruebas automatizadas continuas.
    - Recursos involucrados: Equipos de QA, Administradores de Sistemas.
5. **Capacitación del Personal en Nuevos Sistemas** – 4 días
    - Capacitar al equipo en el uso de la nueva infraestructura y microservicios.
    - Recursos involucrados: Equipos de Formación y Soporte Técnico.

Duración Total: 4 semanas

9.6.4. Fase 4: Optimización y Monitoreo Continuo (Semanas 9-12)

Actividades Principales:

1. **Optimización de Desempeño Basada en Métricas** – 1 semana
    - Analizar las métricas recolectadas y realizar ajustes de rendimiento.
    - Recursos involucrados: Ingenieros de Rendimiento, Administradores de Sistemas.
2. **Auditorías de Seguridad y Cumplimiento** – 1 semana
    - Realizar auditorías de seguridad y revisar configuraciones de IAM, roles y accesos.
    - Recursos involucrados: Auditores de Seguridad, Arquitectos de TI.
3. **Automatización y Refuerzo de Pruebas** – 1 semana
    - Ajustar pruebas automatizadas para cualquier nueva funcionalidad y pruebas de regresión.
    - Recursos involucrados: Equipos de QA, Desarrolladores.
4. **Pruebas de Recuperación Ante Desastres** – 3 días
    - Realizar simulacros de recuperación para asegurar la continuidad operativa.
    - Recursos involucrados: Equipos de Respuesta a Incidentes, Ingenieros de Infraestructura.
5. **Optimización Final de Microservicios y Escalabilidad** – 1 semana
    - Realizar ajustes finales en la escalabilidad de microservicios, permitiendo que los sistemas se adapten a las demandas cambiantes.
    - Recursos involucrados: Arquitectos de Microservicios, Ingenieros de Operaciones.

Duración Total: 4 semanas

9.6.5. Resumen del Cronograma

Fase | Actividades Principales | Duración |
| | | |
Fase 1 | Evaluación y Planificación | 2 semanas |
Fase 2 | Migración de Componentes No Críticos | 2 semanas |
Fase 3 | Migración de Componentes Críticos | 4 semanas |
Fase 4 | Optimización y Monitoreo Continuo | 4 semanas |

9.6.6. Controles y Puntos de Revisión

Cada fase tendrá puntos de revisión semanal para validar el avance según los criterios de éxito establecidos. Un **comité de revisión de migración**,

compuesto por gerentes de proyecto, arquitectos empresariales, auditores de seguridad y líderes técnicos, realizará estas revisiones para garantizar que el proyecto se mantenga en los plazos y con la calidad esperada.

Total de Duración: **3 meses**

9.7. Conclusión

El **Plan de Migración Gradual** proporciona un enfoque sólido y seguro para llevar al banco desde una arquitectura tradicional hacia una moderna basada en **microservicios** y **nube**, minimizando los riesgos y maximizando la eficiencia. Cada fase del proceso está diseñada para garantizar la **continuidad operativa**, con herramientas automatizadas de AWS que facilitan el monitoreo, las pruebas y los despliegues. Este enfoque asegura que el banco pueda adaptarse rápidamente a las demandas cambiantes del mercado, manteniendo la **seguridad**, **escalabilidad** y **resiliencia** necesarias para su éxito a largo plazo.

---

10. Apendices

10.1. Glosario de Términos

Alta Disponibilidad (HA): Capacidad de un sistema para operar de forma continua durante un largo período de tiempo sin fallos, maximizando el tiempo de funcionamiento.

Amazon CloudWatch: Servicio de monitoreo de AWS que permite recopilar, visualizar y analizar métricas y logs en tiempo real de recursos en la nube.

Amazon ECS (Elastic Container Service): Servicio de orquestación de contenedores de AWS que permite desplegar, gestionar y escalar aplicaciones en contenedores Docker.

Amazon RDS (Relational Database Service): Servicio gestionado de bases de datos relacionales en AWS que permite la configuración, operación y escalado automatizado de bases de datos.

Apache Kafka: Plataforma distribuida de transmisión de eventos en tiempo real, que permite procesar grandes volúmenes de datos de manera eficiente mediante el uso de colas de mensajes.

API Gateway: Servicio que facilita la gestión, distribución y monitoreo de APIs, proporcionando control sobre el tráfico, autenticación y balanceo de carga.

APIs REST/GraphQL: Interfaz de programación que permite la comunicación entre sistemas a través de protocolos HTTP, con REST utilizando un formato de URL y GraphQL permitiendo consultas personalizadas.

Autenticación: Proceso de verificación de la identidad de un usuario o sistema antes de permitirle acceder a un recurso.

AWS IAM (Identity and Access Management): Servicio de AWS que permite controlar de forma segura el acceso a los recursos de AWS mediante la creación y gestión de usuarios, grupos y permisos.

AWS KMS (Key Management Service): Servicio de AWS que facilita la creación y gestión de claves de cifrado para proteger datos en reposo y en tránsito.

AWS Lambda: Servicio de cómputo sin servidor que ejecuta código en respuesta a eventos y gestiona automáticamente los recursos de cómputo.

AWS Multi-AZ (Availability Zones): Funcionalidad de AWS que replica servicios y bases de datos en diferentes zonas de disponibilidad para garantizar la redundancia y tolerancia a fallos.

AWS S3 (Simple Storage Service): Servicio de almacenamiento de objetos en la nube de AWS, diseñado para almacenar y recuperar cualquier cantidad de datos desde cualquier parte del mundo.

AWS SNS (Simple Notification Service): Servicio de notificaciones y mensajería en tiempo real que permite la entrega de mensajes entre sistemas distribuidos en diferentes regiones de AWS.

AWS SQS (Simple Queue Service): Servicio de colas de mensajes de AWS que permite la transmisión de mensajes entre aplicaciones, asegurando su entrega de forma fiable.

Balanceo de Carga: Técnica utilizada para distribuir el tráfico de red o las solicitudes entre varios servidores o instancias para evitar sobrecargas y asegurar alta disponibilidad.

BIAN (Banking Industry Architecture Network): Modelo de referencia estandarizado para la industria bancaria, que facilita la interoperabilidad y la integración de sistemas.

Cifrado en Tránsito: Proceso de protección de los datos mientras son transmitidos entre diferentes sistemas o componentes para evitar accesos no autorizados.

Cifrado en Reposo: Método para proteger los datos almacenados en discos o bases de datos, de modo que no puedan ser leídos sin las claves de cifrado adecuadas.

Control de Acceso Basado en Roles (RBAC): Modelo de gestión de acceso donde los permisos se asignan a usuarios en función de su rol dentro de la organización.

Core Bancario: Sistema principal de un banco que gestiona las transacciones financieras, cuentas de clientes, préstamos, pagos y otros procesos críticos del negocio bancario.

CORS (Cross-Origin Resource Sharing): Política de seguridad implementada en servidores web para controlar las solicitudes de recursos que provienen de dominios externos.

DevOps: Conjunto de prácticas y herramientas que combinan el desarrollo de software (Dev) y las operaciones de TI (Ops) para mejorar la entrega de aplicaciones y servicios de alta calidad mediante la automatización.

DMS (Database Migration Service): Servicio de AWS que facilita la migración de bases de datos entre entornos locales y la nube, o entre bases de datos de diferentes motores.

ETL (Extract, Transform, Load): Proceso de extracción, transformación y carga de datos desde múltiples fuentes a un sistema de destino, como un data warehouse.

Event-Driven Architecture (EDA): Arquitectura basada en eventos donde los servicios reaccionan a cambios en el sistema mediante la publicación y suscripción de eventos.

Grafana: Herramienta de visualización y monitoreo que se integra con diferentes bases de datos y sistemas para mostrar métricas en tiempo real a través de dashboards.

Infraestructura como Código (IaC): Práctica de gestión de infraestructura mediante archivos de configuración en lugar de realizar cambios manuales, asegurando consistencia y escalabilidad.

JSON (JavaScript Object Notation): Formato de datos ligero utilizado para intercambiar información entre sistemas de forma legible y estructurada.

Kafka Streams: API de procesamiento distribuido de eventos basada en Apache Kafka que permite realizar operaciones en tiempo real sobre flujos de datos.

Ley Orgánica de Protección de Datos Personales (LOPD): Ley que regula el tratamiento y la protección de los datos personales de las personas físicas en España y Europa.

Mensajería Asíncrona: Método de comunicación entre sistemas donde los mensajes se envían sin requerir una respuesta inmediata, permitiendo la ejecución en paralelo y la resiliencia.

Microservicios: Arquitectura de software donde una aplicación se construye a partir de pequeños servicios independientes que se comunican entre sí a través de APIs, facilitando su escalabilidad y mantenimiento.

OAuth 2.0: Protocolo estándar de autorización que permite otorgar acceso limitado a recursos sin necesidad de compartir credenciales entre el cliente y el servidor.

Open Finance: Marco de integración financiera que permite compartir información entre bancos, fintechs y otros actores mediante APIs abiertas, fomentando la interoperabilidad y la innovación.

OpenID Connect: Protocolo de autenticación basado en OAuth 2.0 que permite verificar la identidad de un usuario a través de un proveedor de identidad.

Patrón Saga: Patrón de diseño utilizado para gestionar transacciones distribuidas en microservicios, asegurando que cada paso de la transacción se ejecute de manera consistente y, en caso de fallos, se apliquen acciones compensatorias.

Rate Limiting: Técnica para limitar la cantidad de solicitudes que una API puede recibir en un tiempo determinado, evitando sobrecargas y asegurando un uso equitativo de los recursos.

Recuperación Ante Desastres (DR): Estrategia para restaurar rápidamente los sistemas críticos en caso de fallos o desastres, asegurando la continuidad operativa.

REST (Representational State Transfer): Arquitectura para la construcción de APIs que utiliza métodos HTTP y respuestas en formato JSON o XML para la interacción entre sistemas.

Swagger: Herramienta para la documentación y pruebas de APIs RESTful que facilita la comprensión e integración de las APIs.

TLS (Transport Layer Security): Protocolo de seguridad que garantiza la privacidad y la integridad de los datos transmitidos a través de internet mediante el cifrado.

TOGAF (The Open Group Architecture Framework): Marco de arquitectura empresarial que proporciona una metodología y herramientas para el diseño, planificación e implementación de arquitecturas de sistemas empresariales.

VPC (Virtual Private Cloud): Red privada virtual en AWS que permite aislar recursos en la nube, controlando el tráfico de red y la conectividad de manera segura.

10.2. Principios de Arquitectura para el proyecto.

A continuación se presenta la matriz de principios de arquietctura para el proyecto.

ID	Nombre del Principio	Descripción	Justificación	Implicaciones
P1	Alineación Estratégica	Todas las iniciativas de TI en el banco deben estar alineadas con los objetivos estratégicos financieros.	Asegura que las inversiones en TI apoyen los objetivos clave del banco, como mejorar la experiencia del cliente y reducir riesgos.	Evaluar todas las iniciativas de TI en función de su alineación con las metas estratégicas del banco.
P2	Seguridad	La seguridad de la información bancaria es una prioridad en todas las soluciones de TI.	Protege los datos sensibles de los clientes y asegura el cumplimiento con regulaciones bancarias.	Implementar controles de seguridad robustos y auditorías frecuentes para evitar brechas de seguridad.
P3	Flexibilidad y Adapt	Las arquitecturas de TI deben ser flexibles para adaptarse a	Permite al banco responder rápidamente a cambios	Diseñar soluciones modulares y escalables que puedan adaptarse a nuevas normativas y

ID	Nombre del Principio	Descripción	Justificación	Implicaciones
	abilidad	cambios regulatorios y del mercado bancario.	regulatorios, productos financieros nuevos, y demanda del cliente.	servicios bancarios.
P4	Uso de Estándares Abiertos	En la medida de lo posible, se deben utilizar estándares abiertos e interoperables en las soluciones bancarias.	Facilita la integración entre sistemas bancarios, reduce la dependencia de proveedores, y mejora la eficiencia.	Adoptar estándares como ISO 20022 para pagos y asegurar la compatibilidad con otros bancos e instituciones financieras.
P5	Reutilización de Componentes	Fomentar la reutilización de componentes de TI existentes en el banco cuando sea posible.	Reduce costos y tiempos de implementación en el banco, manteniendo la consistencia en las soluciones.	Aprovechar sistemas de core bancario y APIs existentes para nuevos proyectos.
P6	Escalabilidad	Las soluciones de TI del banco deben ser escalables para soportar el crecimiento de clientes y transacciones.	Asegura que la infraestructura de TI pueda crecer con el aumento de usuarios y volumen transaccional sin interrupciones.	Diseñar sistemas que soporten el crecimiento sin requerir grandes rediseños, especialmente para tiempos de alta demanda.
P7	Disponibilidad	Las soluciones de TI deben ser altamente disponibles, especialmente en operaciones críticas como pagos y banca en línea.	Garantiza que los servicios bancarios estén siempre disponibles para los clientes, incluso en horarios fuera de oficina.	Implementar redundancias, alta disponibilidad y planes de contingencia para asegurar continuidad en operaciones críticas.
P	Auto	Las tareas	Automatizar	Implementar RPA

ID	Nombre del Principio	Descripción	Justificación	Implicaciones
8	matización	repetitivas en el banco deben ser automatizadas para mejorar la eficiencia y reducir errores humanos.	procesos mejora la eficiencia y reduce los errores en procesos bancarios, como conciliaciones y auditorías.	(Automatización Robótica de Procesos) para tareas manuales repetitivas dentro de las áreas de operaciones bancarias.
P9	Calidad del Servicio	Las soluciones de TI deben cumplir con los acuerdos de niveles de servicio (SLA) para garantizar la experiencia del cliente bancario.	Mejora la experiencia del cliente y asegura la continuidad del negocio en servicios como banca móvil y pagos en tiempo real.	Establecer SLAs rigurosos y monitorear continuamente para asegurar tiempos de respuesta óptimos en servicios críticos.
P10	Interoperabilidad	Los sistemas de TI deben ser interoperables con otros sistemas bancarios nacionales e internacionales.	Facilita la integración y colaboración entre bancos y sistemas de pago internacionales, esenciales para servicios transfronterizos.	Usar estándares como SWIFT y SEPA para facilitar transferencias y pagos internacionales.
P11	Transparencia	Las decisiones y procesos de TI deben ser transparentes y documentados, especialmente en relación a la gestión de riesgos.	Facilita auditorías y fomenta la confianza interna y con los reguladores en el cumplimiento bancario.	Mantener un registro detallado de las decisiones y procesos, facilitando su accesibilidad a las partes interesadas internas y externas.
P12	Innovación	Fomentar la innovación continua en el desarrollo de productos y servicios bancarios	Mantiene al banco competitivo en el sector y asegura que adopte nuevas tecnologías	Invertir en laboratorios de innovación y equipos de desarrollo enfocados en nuevas tecnologías como blockchain y AI para optimizar los

ID	Nombre del Principio	Descripción	Justificación	Implicaciones
		digitales.	financieras (Fintech) rápidamente.	servicios.
P13	Gestión de Datos	Los datos bancarios deben ser gestionados como un activo crítico para la toma de decisiones estratégicas.	Asegura que los datos sean precisos, accesibles y que respalden la toma de decisiones, tanto operativas como estratégicas.	Implementar políticas de gobernanza de datos robustas y asegurar la integridad y precisión de la información financiera.
P14	Orientación al Cliente	Las soluciones de TI deben centrarse en mejorar la experiencia del cliente bancario.	Incrementa la satisfacción del cliente, su fidelidad y mejora la percepción del banco en el mercado.	Recoger y analizar continuamente feedback del cliente a través de canales como la banca en línea, móvil y presencial, y adaptar los servicios según sus necesidades.
P15	Responsabilidad Ambiental	Las operaciones de TI deben minimizar su impacto ambiental, incluyendo el uso eficiente de energía en los centros de datos.	Promueve la sostenibilidad y contribuye a una mejor imagen pública del banco ante los consumidores.	Implementar tecnologías de ahorro de energía y prácticas de TI verde en centros de datos y oficinas del banco.
P16	Rendimiento	Las soluciones de TI deben ofrecer un rendimiento adecuado para soportar operaciones bancarias críticas.	Mejora la eficiencia y asegura que los clientes puedan realizar transacciones sin interrupciones.	Realizar pruebas de estrés periódicas en sistemas críticos como el core bancario para garantizar tiempos de respuesta adecuados.
P17	Cumplimiento Norm	Las soluciones de TI del banco deben cumplir con las leyes y	Evita sanciones regulatorias y asegura el funcionamiento	Mantenerse al día con regulaciones como la GDPR (Protección de Datos) y PCI-DSS

ID	Nombre del Principio	Descripción	Justificación	Implicaciones
	ativo	regulaciones aplicables a nivel local e internacional.	ético del banco dentro de las normativas de banca.	(Seguridad de Tarjetas de Pago).
P18	Responsabilidad y Rendición de Cuentas	Cada miembro del equipo de TI debe ser responsable de sus acciones y resultados, especialmente en áreas sensibles como el manejo de datos de clientes.	Fomenta la responsabilidad y asegura que se tomen decisiones éticas y estratégicas.	Establecer roles y responsabilidades claras en proyectos y operaciones, y realizar revisiones periódicas de desempeño.
P19	Reducción de la Complejidad	Las soluciones de TI deben minimizar la complejidad para facilitar su mantenimiento y gestión.	Simplifica la gestión y operación de sistemas bancarios, permitiendo respuestas más rápidas a problemas y nuevas oportunidades.	Diseñar arquitecturas que sean modulares y fáciles de gestionar, evitando dependencias excesivas entre sistemas.
P20	Mejora Continua	Las soluciones de TI deben ser mejoradas continuamente para mantenerse al día con las necesidades del banco y las exigencias regulatorias.	Asegura que los sistemas se mantengan actualizados y preparados para cambios futuros, como nuevas regulaciones bancarias.	Implementar procesos de mejora continua en infraestructura y software, asegurando revisiones periódicas y actualizaciones.