Caracas, 02 de septiembre de 2024

Arquitectura DEVSU

Arquitectura de Integración para la Modernización de Sistemas Bancarios

El objetivo es diseñar una arquitectura moderna que permita al banco integrar de manera efectiva sus sistemas tradicionales con nuevos servicios digitales, al tiempo que se asegura la escalabilidad, disponibilidad, cumplimiento normativo y recuperación ante desastres. La propuesta utilizará una infraestructura híbrida, combinando elementos locales con servicios en la nube (como Azure o AWS), y adoptará un enfoque centrado en eventos y APIs.

1. Diagrama C4 (Modelo hasta Componentes)

El modelo C4 desglosa la arquitectura en cuatro niveles:

- **Diagrama de Contexto**: Vista general de los actores y sistemas.
- **Diagrama de Contenedores**: Detalle de las aplicaciones principales, bases de datos y sistemas conectados.
- **Diagrama de Componentes**: Desglose de las aplicaciones en sus partes internas.

1.1. Diagrama de Contexto

El banco moderniza su infraestructura con un enfoque de microservicios que integra:

- **Core Bancario Tradicional**: Sistema heredado que mantiene los datos y procesos bancarios clave.
- **Core Bancario Digital**: Nuevo core para soportar servicios modernos, ágil y diseñado para digital banking.
- **Banca Web y Móvil**: Plataforma para clientes finales, que interactúa con APIs públicas y privadas.
- **Plataforma de Servicios de Pago**: Integrada con pasarelas de pago y servicios de pago de terceros.
- **APIs para Servicios de Terceros (Open Finance)**: API Gateway que expone servicios bancarios a terceros conforme a estándares de Open Banking.
- **Sistema de Gestión de Riesgos**: Analiza las transacciones y los datos bancarios para evaluar riesgos.
- **Sistema de Prevención de Fraudes**: Detecta y mitiga transacciones sospechosas en tiempo real.

```
graph TB
U[Usuario] -->|Interactúa| BW[Banca Web y Móvil]
subgraph "Sistema Bancario Modernizado"
```

```
BW -->|Usa| APIG[API Gateway]
APIG -->|Integra| CBD[Core Bancario Digital]
APIG -->|Integra| CBT[Core Bancario Tradicional]
APIG -->|Expone| OF[APIs para Open Finance]
APIG -->|Usa| PSP[Plataforma de Servicios de Pago]
APIG -->|Consulta| SGR[Sistema de Gestión de Riesgos]
APIG -->|Verifica| SPF[Sistema de Prevención de Fraudes]
end
T[Terceros] -->|Integra| OF
PP[Pasarelas de Pago] -->|Integra| PSP
R[Reguladores] -->|Supervisa| OF
```

1.2. Diagrama de Contenedores

El siguiente diagrama muestra cómo se dividen los sistemas en contenedores lógicos:

- **Frontend**:
    - Web y App Móvil (React Native/Angular, Swift)
    - API Gateway
    - Capa de servicios
- **Backend**:
    - Microservicios de Negocio (Gestión de clientes, pagos, transferencias)
    - Microservicios de Riesgos y Fraudes
    - Core Bancario Tradicional (AS/400, mainframe, legado)
    - Core Bancario Digital (arquitectura moderna en contenedores)
- **Integración**:
    - Bus de Mensajes (Event Hub/Kafka)
    - API Gateway (gestión interna y externa de APIs)
- **Infraestructura**:
    - Plataforma en la Nube (Azure/AWS)
    - Kubernetes/OpenShift (para despliegue de microservicios)
    - Almacenamiento persistente y bases de datos (SQL, NoSQL)

```
graph TB
subgraph Frontend
WA[Web App\nReact/Angular]
MA[Mobile App\nReact Native/Swift]
AG[API Gateway]
SL[Capa de Servicios]
end

subgraph Backend
subgraph Microservicios
MN[Microservicios de Negocio\nGestión de clientes, pagos, transferencias]
```

```
MR[Microservicios de\nRiesgos y Fraudes]
end
CBT[Core Bancario Tradicional\nAS/400, mainframe, legado]
CBD[Core Bancario Digital\nArquitectura moderna en contenedores]
end

subgraph Integración
BM[Bus de Mensajes\nEvent Hub/Kafka]
AGI[API Gateway\nGestión interna y externa de APIs]
end

subgraph Infraestructura
CN[Plataforma en la Nube\nAzure/AWS]
KO[Kubernetes/OpenShift]
DB[(Almacenamiento y Bases de Datos\nSQL, NoSQL)]
end

WA --> AG
MA --> AG
AG --> SL
SL --> AGI
AGI --> Microservicios
AGI --> CBT
AGI --> CBD
Microservicios --> BM
CBT --> BM
CBD --> BM
Microservicios --> DB
CBT --> DB
CBD --> DB
Microservicios -.-> KO
CBD -.-> KO
KO -.-> CN
DB -.-> CN
```

1.3. Diagrama de Componentes

Los componentes internos incluyen:

- **Microservicios de Banca**:
    - API de Cuentas
    - API de Pagos
    - API de Gestión de Usuarios
    - API de Gestión de Transacciones
- **API Gateway**:

        - Autenticación y autorización
        - Monitoreo de APIs
        - Transformación y ruteo de solicitudes
    - **Sistema de Mensajería**:
        - Bus de Mensajes (Kafka/EventHub)
        - Tópicos de eventos (pagos, transferencias, alertas de fraude)
    - **Base de Datos**:
        - Bases de datos distribuidas (PostgreSQL, MongoDB)
        - Almacenamiento para el Core Tradicional

```
graph TB
subgraph "Microservicios de Banca"
AC[API de Cuentas]
AP[API de Pagos]
AGU[API de Gestión de Usuarios]
AGT[API de Gestión de Transacciones]
end

subgraph "API Gateway"
AA[Autenticación y autorización]
MA[Monitoreo de APIs]
TR[Transformación y ruteo de solicitudes]
end

subgraph "Sistema de Mensajería"
BM[Bus de Mensajes\nKafka/EventHub]
subgraph "Tópicos de eventos"
TP[Pagos]
TT[Transferencias]
TAF[Alertas de fraude]
end
end

subgraph "Base de Datos"
BD[Bases de datos distribuidas\nPostgreSQL, MongoDB]
ACT[Almacenamiento para\nel Core Tradicional]
end

AA --> AC
AA --> AP
AA --> AGU
AA --> AGT
MA --> AC
MA --> AP
```

MA --> AGU
MA --> AGT
TR --> AC
TR --> AP
TR --> AGU
TR --> AGT

AC --> BM
AP --> BM
AGU --> BM
AGT --> BM

BM --> TP
BM --> TT
BM --> TAF

AC -.-> BD
AP -.-> BD
AGU -.-> BD
AGT -.-> BD
AC -.-> ACT
AP -.-> ACT
AGU -.-> ACT
AGT -.-> ACT

2. Patrones de Integración y Tecnología

- **Bus de Mensajes** (Apache Kafka, Azure Event Hubs): El sistema adoptará una arquitectura basada en eventos para lograr desacoplamiento entre los sistemas y asegurar la baja latencia. Los eventos de transacciones bancarias, pagos, alertas de fraude, y riesgo serán transmitidos a través del bus de mensajes.
- **API Gateway** (Apigee, Azure API Management, Kong): Gestión centralizada de APIs, tanto internas como externas, incluyendo autenticación, control de acceso, límites de tasa, y transformación de mensajes.
- **Microservicios** (Kubernetes/OpenShift): Cada funcionalidad clave (pagos, transferencias, gestión de usuarios) será implementada como un microservicio desplegado en contenedores orquestados con Kubernetes.
- **Base de Datos Distribuida**: Bases de datos como PostgreSQL o MongoDB en la nube

permitirán manejar grandes volúmenes de datos y consultas concurrentes con alta disponibilidad.

Tecnología Propuesta:

- **Nube:** Azure o AWS para alojar los nuevos servicios y componentes.
- **Base de Datos:** PostgreSQL o MongoDB para los nuevos sistemas, y una conexión a la base de datos del core bancario existente.
- **Contenedores:** Docker para contenerizar los microservicios.
- **Orquestación:** Kubernetes para gestionar los contenedores y la infraestructura en la nube.
- **API Manager:** Azure API Management o Kong para gestionar las APIs.
- **Mensajería:** Apache Kafka o RabbitMQ.

3. Seguridad y Cumplimiento Normativo

- **Seguridad de Datos y Cumplimiento (LGPD, GDPR)**:
    - **Cifrado** de datos en reposo y en tránsito mediante TLS/SSL.
    - Implementación de **políticas de anonimización y pseudonimización** de datos personales.
    - **Auditoría y monitoreo** de acceso a datos sensibles.
    - **Autenticación multifactor** (MFA) y protocolos de seguridad como OAuth 2.0 y OpenID Connect para garantizar la protección de las APIs.
- **Protección Contra Amenazas**: Firewalls de aplicaciones web (WAF), detección de intrusiones y sistemas de prevención (IDS/IPS) y DDoS protection.

4. Alta Disponibilidad y Recuperación ante Desastres

- **Despliegue en la Nube Híbrida** (Azure/AWS y On-Premises): Utilización de una infraestructura en la nube híbrida con redundancia geográfica para garantizar alta disponibilidad.
- **Estrategia de Disaster Recovery**:
    - Replicación de datos en múltiples zonas de disponibilidad.
    - Uso de servicios en la nube para backups automáticos y failover de bases de datos y servicios clave.

5. Estrategia de Integración Multicore

- **Orquestación y Exposición de APIs**: Ambos sistemas core, tanto el tradicional como el digital, estarán integrados a través de microservicios que expondrán APIs para interactuar con las plataformas de front-end y otros sistemas de backend. El API Gateway asegurará una capa uniforme para que los consumidores interactúen con ambos cores de manera transparente.
- **Sincronización de Datos**: Se implementarán mecanismos de sincronización de datos en tiempo real o en batch para asegurar la consistencia entre el core bancario tradicional y el nuevo core digital.

6. Gestión de Identidad y Acceso

- **Single Sign-On (SSO)**: Implementación de un sistema de SSO con estándares como OAuth 2.0 y OpenID Connect para una gestión centralizada de la identidad.
- **Gestión de Roles y Permisos**: Control de acceso basado en roles (RBAC) en todos los sistemas.

7. Estrategia de API Internas y Externas

- **APIs Internas**: Utilizadas para integrar los microservicios internos (pagos, cuentas, usuarios). Se aplicarán políticas de seguridad estrictas como OAuth 2.0 para acceso entre servicios.
- **APIs Externas** (Open Banking): Conformidad con los estándares de Open Banking (PSD2), asegurando que las APIs expuestas a terceros estén correctamente autenticadas y monitorizadas.

8. Modelo de Gobierno de APIs y Microservicios

- **Registro Centralizado de APIs**: Todas las APIs deberán estar registradas y documentadas en un API Gateway, con políticas de versionado y depreciación controladas.
- **Política de Auditoría y Trazabilidad**: Uso de un sistema de logging centralizado (ELK Stack, Azure Monitor) para rastrear todas las interacciones de las APIs y asegurar trazabilidad.

9. Plan de Migración Gradual

- **Enfoque Faseado**: La migración al nuevo core digital y a los sistemas de microservicios se realizará de forma gradual, comenzando con servicios no críticos y moviendo progresivamente las cargas de trabajo críticas al nuevo sistema.
- **Estrategia de Paralelización**: Se mantendrá el core bancario tradicional en paralelo con el nuevo core digital hasta que todos los sistemas estén completamente migrados y validados.

Conclusión

Este diseño propuesto garantiza una arquitectura moderna que permite la integración eficiente de los sistemas tradicionales con las nuevas plataformas digitales, asegura el cumplimiento normativo, y se basa en principios de escalabilidad, seguridad, y recuperación ante desastres. Con la adopción de microservicios y APIs, el banco podrá mejorar su agilidad y responder más rápidamente a las demandas del mercado y los requerimientos regulatorios.

Comentario:

URL del repositorio

https://github.com/luisnarvaez19/devsu