Nombre: Marlon Cedeño

Informe de Decisiones Arquitectónicas para Sistema de Integración Bancaria

1. Introducción

Este informe documenta las decisiones arquitectónicas adoptadas en el diseño del sistema de integración bancario, basado en los tres niveles del modelo C4:

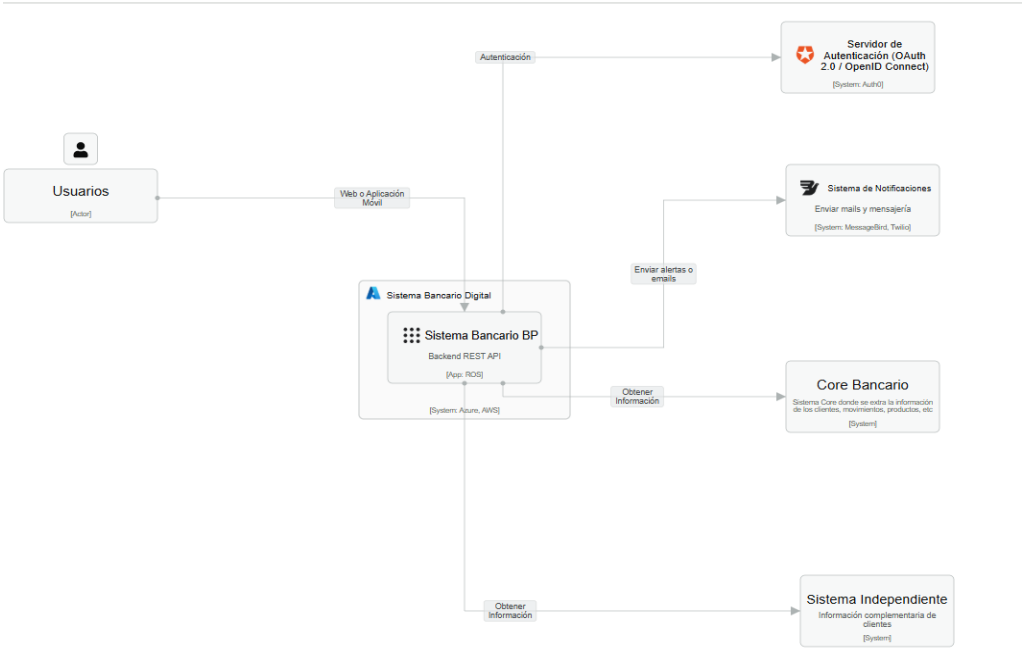
- Diagrama de Contexto
- Diagrama de Contenedores
- Diagrama de Componentes

Cada decisión está justificada teóricamente, indicando qué tecnologías se evaluaron, cuáles se seleccionaron, y por qué.

2. Diagrama de Contexto

Aunque no fue incluido gráficamente, se asume un nivel donde:

- El Usuario se conecta mediante Web/Mobile
- La autenticación se da vía OAuth
- El sistema integra servicios core (bancarios) y terceros



Enlace para ver mejor el diagrama:
https://s.icepanel.io/EIHIeEuC32s2vP/CwOf/landscape/diagrams/viewer?diagram=Fx1UKLaJj9&model=vQW0ej4nKB&overlay_tab=tags&x1=-2291.1&x2=1123.1&y1=-7.5&y2=1624.6

3. Diagrama de Contenedores

Contenedores

- Web App (Angular, vue Js)
- Mobile App (Flutter, React Native)
- API Gateway (Kong o Apigee)
- OAuth 2.0 Server
- Sistema Bancario BP (Go, Node Js)
- MongoDB Atlas (AWS, Azure)
- Mensajería asincrónica (NATS, Kafka)
- Servicios externos: Core Bancario y Sistema Independiente

Tecnologías Seleccionadas y Justificaciones

2.1. API Gateway: Api Managament

- **Alternativas evaluadas:** NGINX, Apigee, Azure API Gateway
- **Justificación:**
    1. Kong es altamente extensible con plugins, soporta autenticación, control de rate limiting y monitoreo nativamente.
    2. Está diseñado para arquitecturas de microservicios y se integra fácilmente con OAuth 2.0.

2.2. OAuth 2.0 Server:

- **Alternativas evaluadas:** Auth0, Okta, Firebase Auth
- **Justificación:**
    1. Es open source, soporta flujos de autenticación complejos (PKCE, SSO, MFA).
    2. Se integra fácilmente con aplicaciones web y móviles mediante OIDC y OAuth 2.0.

2.3. MongoDB Atlas

- **Alternativas evaluadas:** PostgreSQL, Sql Server, Couchbase
- **Justificación:**

1. MongoDB es ideal para esquemas flexibles y almacenamiento de documentos como logs de integraciones.
2. Multicloud
3. Posee alto rendimiento en lecturas y escrituras distribuidas.

2.4. NATS (mensajería)

- **Alternativas evaluadas:** Apache Kafka, RabbitMQ
- **Justificación:**
    1. NATS está diseñado para baja latencia y alta disponibilidad, ideal para microservicios.
    2. Su modelo pub/sub encaja con arquitectura orientada a eventos y es liviano en despliegue.

Diagrama:
https://s.icepanel.io/EIHIeEuC32s2vP/WR15/landscape/diagrams/viewer?diagram=87iu77uisf8&model=vQW0ej4nKB&overlay_tab=tags&x1=-3428.2&x2=1257.9&y1=-648.5&y2=1591.5

3. Diagrama de Componentes – Integration Service

Arquitectura Base

- **Arquitectura Hexagonal (Ports & Adapters)**
- **Patrón orientado a eventos**

Componentes Internos

- NATS Listener Adapter
- IntegrationUseCase
- MessageDispatcher
- CoreBankingClient
- ExternalSystemClient
- MongoDBAdapter

Justificaciones de Patrones y Tecnologías

3.1. Arquitectura Hexagonal

- **Alternativas evaluadas:** Arquitectura monolítica, MVC tradicional
- **Justificación:**
    1. Aisla la lógica de negocio del framework, permitiendo pruebas unitarias limpias.
    2. Facilita cambios en adaptadores (Mongo, REST, NATS)

3.2. Orientado a Eventos

- **Alternativas evaluadas:** Integración sincrónica REST, SOAP
- **Justificación:**
    1. Permite desacoplar servicios e integrar sistemas heterogéneos.
    2. Escala de manera natural, evitando cuellos de botella sincrónicos.

3.3. Lenguaje Go

- **Alternativas evaluadas:** Java, Node.js, Python
- **Justificación:**
    1. Go ofrece concurrencia eficiente y bajo uso de recursos.
    2. Excelente compatibilidad con sistemas distribuidos y mensajería.

Diagrama:
https://s.icepanel.io/EIHIeEuC32s2vP/3jn5/landscape/diagrams/viewer?diagram=YAb0

AKf346&model=yihtdja23ps&object=ihY2XFmaio&object_tab=details&overlay_tab=tags&x1=-818&x2=2445.5&y1=-320&y2=1240

Justificaciones Globales

- **Microservicios**: favorecen el despliegue independiente, escalar partes específicas (e.g., procesamiento de integraciones).
- **Docker + Kubernetes (K8s)** (infraestructura asumida): permite orquestación, autoescalado y resiliencia.

5. Protocolos de Comunicación

Componente origen	Destino	Protocolo	Justificación
Web/Mobile App	OAuth 2.0 Server	HTTPS (OIDC/OAuth)	Estándar moderno y seguro
OAuth Server	API Gateway	JWT	Seguridad mediante tokens
API Gateway	Microservicios	HTTP/REST o gRPC	Bajo acoplamiento, eficiencia (gRPC)
Microservicios	NATS Broker	NATS Protocol	Alta disponibilidad y baja latencia
Microservicios	MongoDB	MongoDB Wire Protocol	Driver nativo Go + BSON optimizado
Integration Service	Core Bancario / Externos	REST/gRPC	Flexibilidad + interoperabilidad

6. Conclusiones

La arquitectura propuesta permite integrar de forma segura, escalable y mantenible servicios bancarios y externos:

- Usa patrones modernos: eventos, hexagonal, microservicios
- Se apoya en herramientas maduras: NATS, MongoDB, Api Gateay
- Optimiza resiliencia y trazabilidad: Circuit Breakers, Retrys.