Diseño de Arquitectura de Integración para la Modernización de Sistemas Bancarios

1. Introducción

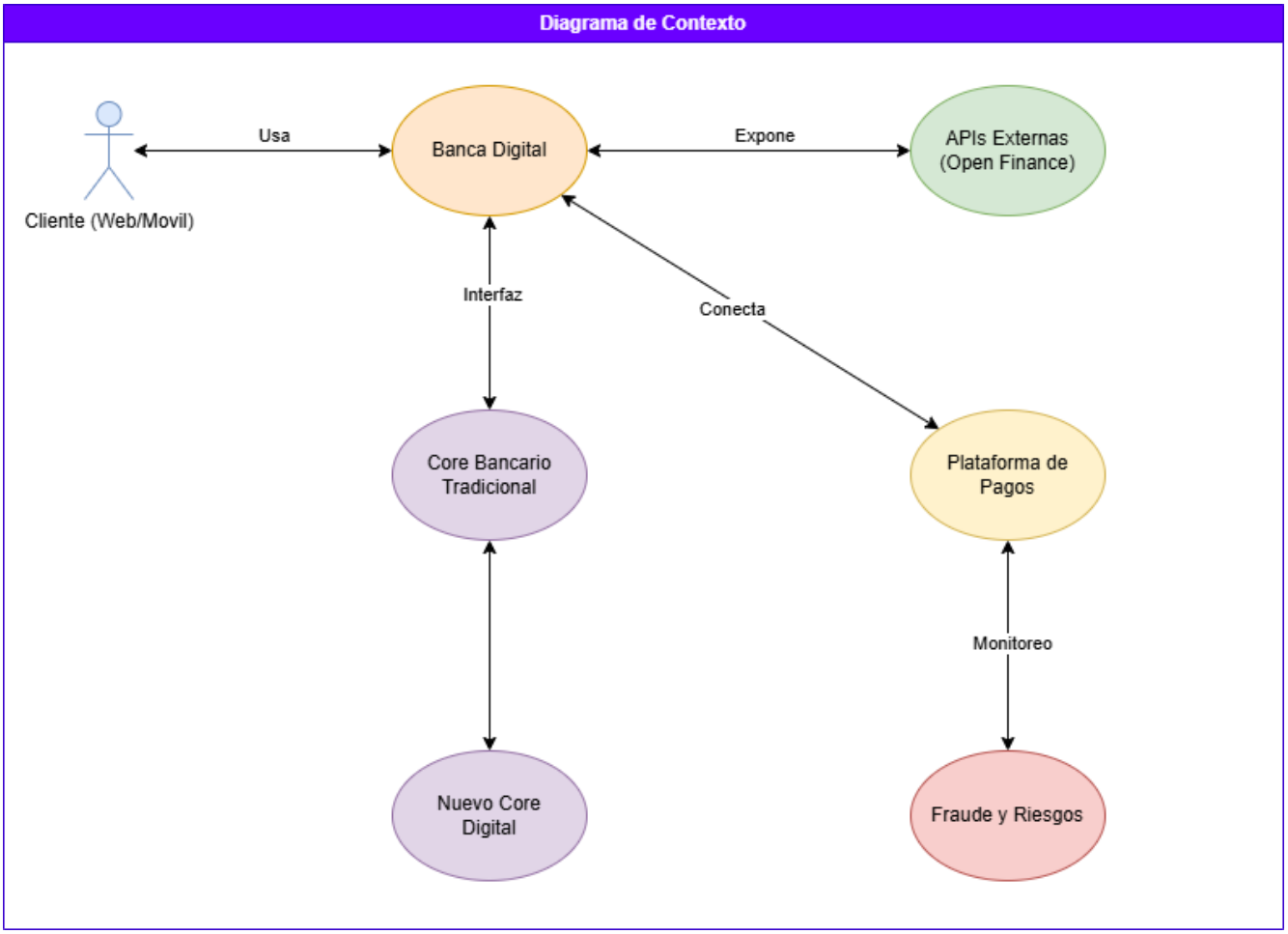
En el contexto de la modernización de la infraestructura tecnológica de un banco tradicional, se requiere diseñar una arquitectura de integración robusta y escalable que permita la coexistencia de un core bancario tradicional con un nuevo core digital, garantizando la interoperabilidad con otros sistemas clave. Esta solución cumple con altos estándares de seguridad, regulaciones del sector y garantiza la alta disponibilidad y la recuperación ante desastres.

2. Arquitectura de Integración de Alto Nivel

2.1. Modelo C4

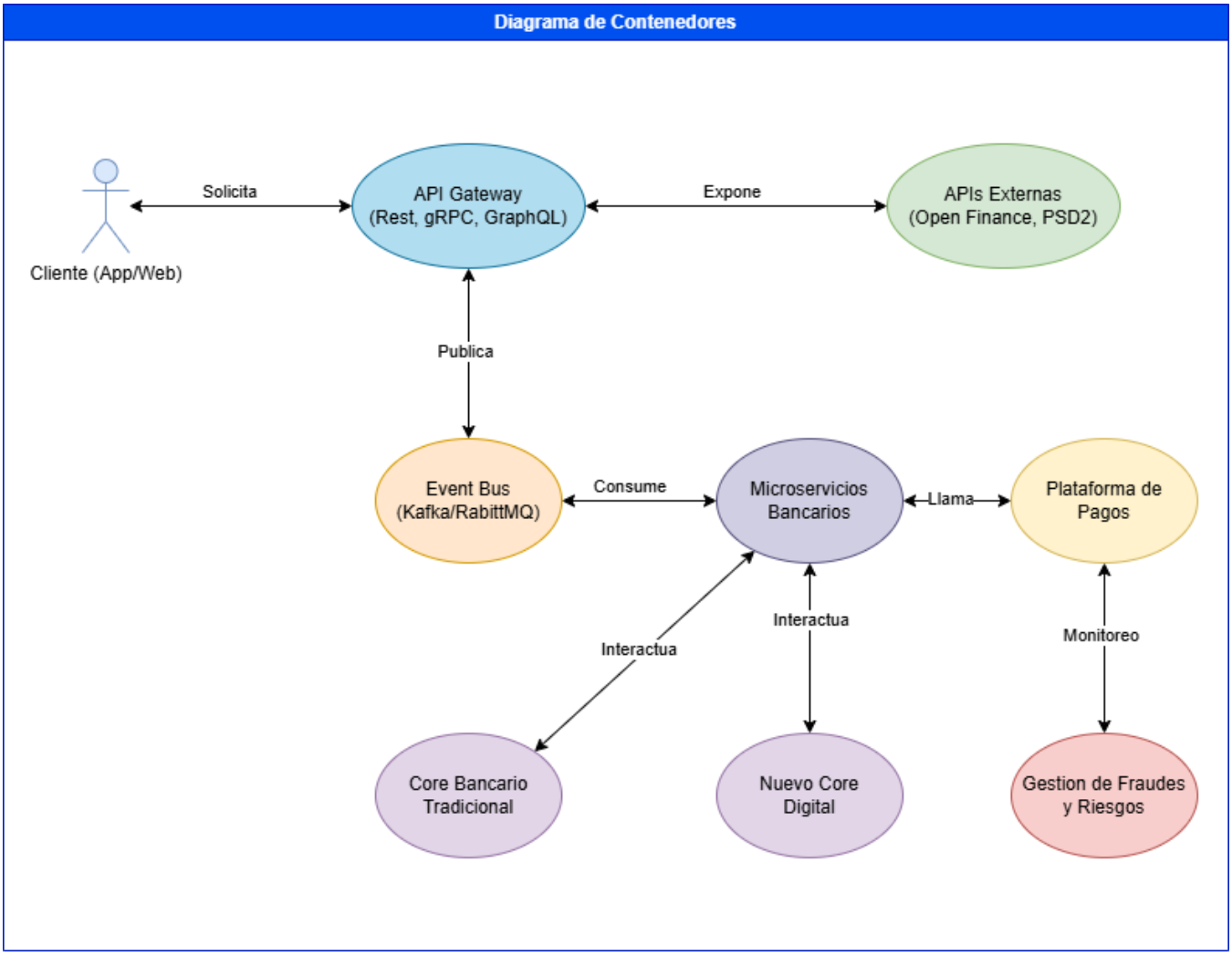
2.1.1. Diagrama de Contexto

Se representa la interacción entre los actores principales (usuarios, sistemas y terceros) con el ecosistema bancario.



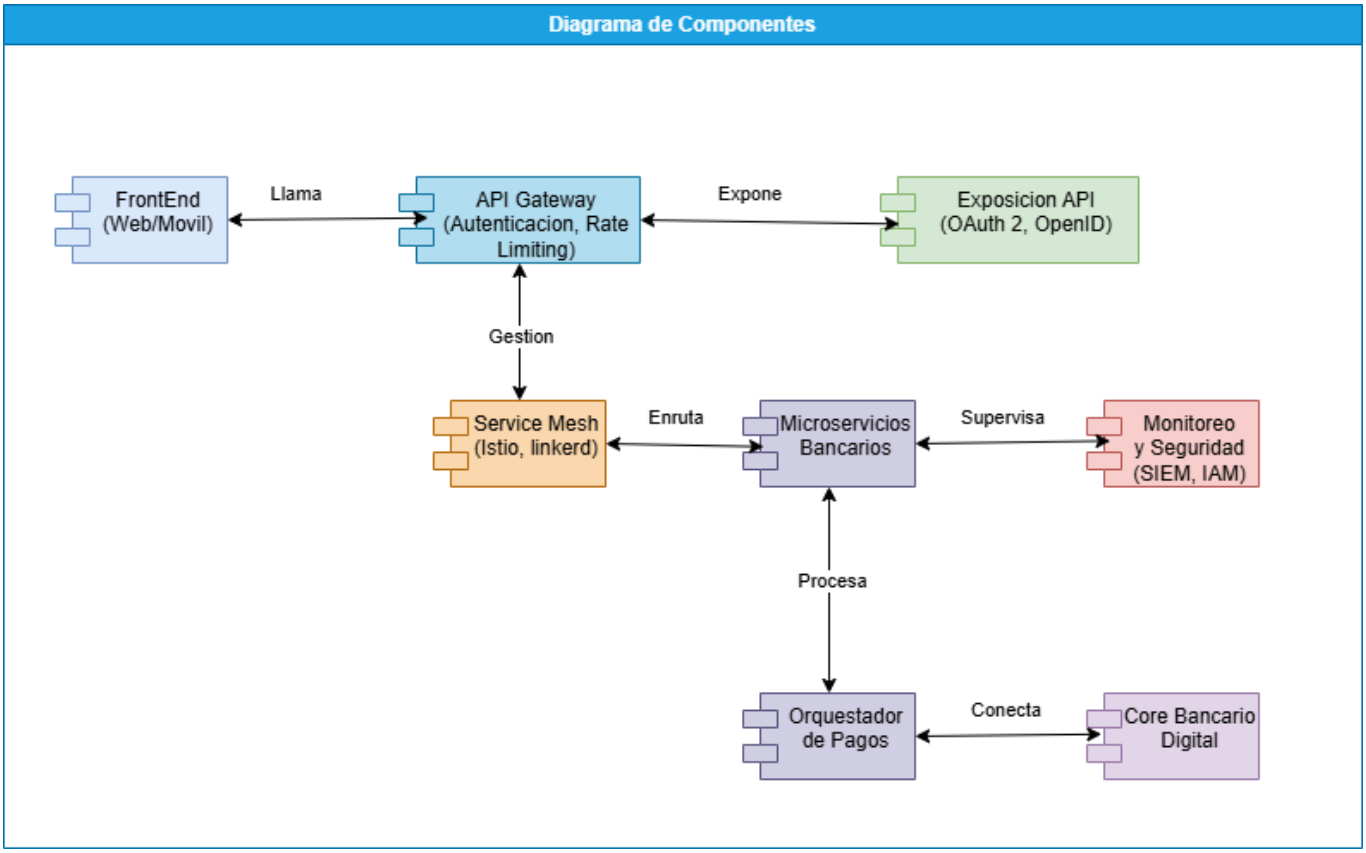
2.1.2. Diagrama de Contenedores

Muestra la organización de los principales sistemas y plataformas dentro de la arquitectura.



2.1.3. Diagrama de Componentes

Describe los módulos internos y su interacción dentro de los contenedores.



3. Patrones de Integración y Tecnologías

Se utilizarán los siguientes patrones:

- **Event-Driven Architecture (EDA)**: Para garantizar la comunicación asíncrona entre sistemas.
- **API Gateway y Microservicios**: Para gestionar la exposición de servicios internos y externos.
- **Message Broker (Kafka/RabbitMQ)**: Para el procesamiento de eventos y transacciones en tiempo real.

Las tecnologías incluyen:

- **Kubernetes** para la orquestación de contenedores.
- **AWS Lambda** para ejecución de funciones serverless.
- **API Manager (Apigee, Kong, AWS API Gateway)** para la gestión de APIs.
- **Frontend**: para el desarrollo de la interfaz de usuario en aplicaciones web y móviles, se emplearán React Native y Flutter. Estas tecnologías permiten una experiencia fluida y optimizada para múltiples plataformas (Web, Android e iOS), garantizando un desarrollo eficiente y mantenible con una única base de código.
- **Microservicios**: los microservicios estarán desarrollados en Java, utilizando el framework Quarkus. Esta elección se debe a su arquitectura no bloqueante, lo que permite un consumo eficiente de recursos, optimizando el rendimiento en entornos de nube. Además, Quarkus está diseñado para tiempos de arranque rápidos y bajo consumo de memoria, lo que lo hace ideal para despliegues en contenedores y plataformas serverless.

- **Integración y Despliegue Continuo (CI/CD):** para la gestión de infraestructura como código (IaC), se utilizará Terraform, dado que es una herramienta agnóstica del proveedor de nube, lo que permite una mayor flexibilidad y portabilidad en la administración y despliegue de recursos en múltiples entornos cloud.

4. Seguridad, Cumplimiento Normativo y Protección de Datos

- **Cumplimiento de regulaciones**: Se asegura cumplimiento con normativas como PSD2, GDPR y la Ley Orgánica de Protección de Datos Personales.
- **Autenticación y Autorización**: Uso de OpenID Connect y OAuth 2.0.
- **Cifrado de Datos**: AES-256 para datos en reposo y TLS 1.3 para datos en tránsito.
- **Monitoreo y Prevención de Fraudes**: Integración con herramientas SIEM y AI para análisis de comportamiento.

5. Estrategia de Alta Disponibilidad y Recuperación ante Desastres

- **Multi-región en Nube**: Replicación de datos y balanceo de carga global.
- **Backup y Replicación**: Estrategia 3-2-1 (tres copias, en dos medios distintos, una en sitio remoto).
- **Automatización de Failover**: Mediante mecanismos de detección y respuesta automática.

6. Estrategia de Integración Multicore

Se adoptará un enfoque de **Core Banking Middleware**, que actúa como un conector entre el core tradicional y el nuevo core digital, utilizando API Gateway y event sourcing para sincronización de datos en tiempo real.

7. Gestión de Identidad y Acceso

- **Single Sign-On (SSO)**: Integración con Identity Providers.
- **Autenticación Multifactor (MFA)**: Requerido para transacciones sensibles.
- **RBAC y ABAC**: Control de acceso basado en roles y atributos.

8. Estrategia de API Internas y Externas

- **APIs Internas**: Diseñadas en REST y gRPC para interoperabilidad.
- **APIs Externas**: Cumplimiento con estándares de Open Finance y regulaciones PSD2.
- **Mensajería**: Uso de WebSockets y eventos en Kafka para actualizaciones en tiempo real.

9. Modelo de Gobierno de APIs y Microservicios

- **Catalogación y Versionado**: Documentación con Swagger y versionado semántico.
- **Monitorización y Logging**: Integración con Prometheus y Grafana.
- **Rate Limiting y Seguridad**: Implementación de cuotas y protección contra ataques DoS.

10. Plan de Migración Gradual

- **Fase 1:** Evaluación y Pruebas Piloto.
- **Fase 2:** Implementación Híbrida (Coexistencia del Core Tradicional y Digital).
- **Fase 3:** Transición Completa a la Nueva Infraestructura.

- **Fase 4:** Desmantelamiento del Sistema Legado.

11. Conclusión

El diseño propuesto garantiza una transición escalonada y segura hacia una infraestructura moderna, optimizando la integración entre sistemas tradicionales y nuevos, cumpliendo con regulaciones y mejorando la experiencia del cliente. La arquitectura basada en eventos y microservicios permite flexibilidad y escalabilidad a futuro.