Patrones de Integración

Este archivo contiene toda la información sobre los patrones de integración:

Event-Driven Architecture

Implementamos Event-Driven Architecture para desacoplar servicios y permitir procesamiento asíncrono, mejorando la escalabilidad y resiliencia del sistema.

Características:

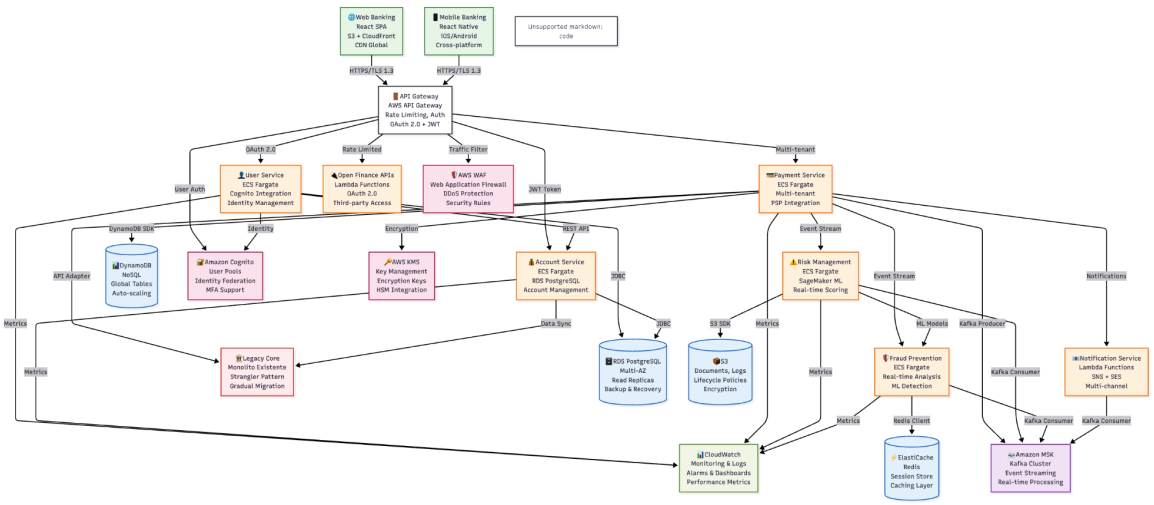
- Event Bus: Amazon MSK (Kafka) para event streaming
- Event Sources: Payment Service, Risk Service, Fraud Service, Account Service
- Event Consumers: Notification, Audit, Analytics, Compliance Services

Eventos Principales:

- payment-initiated: Iniciación de pagos con datos de transacción
- payment-completed: Finalización de pagos con estado y timestamp
- risk-evaluated: Evaluación de riesgo con score y recomendaciones
- fraud-detected: Detección de fraude con score y acción tomada

Beneficios:

- Desacoplamiento: Servicios independientes y escalables
- Resiliencia: Tolerancia a fallos de servicios individuales
- Escalabilidad: Procesamiento paralelo de eventos
- Auditabilidad: Trazabilidad completa de eventos



Patrones de Integración

API Gateway Pattern

API Gateway centraliza la gestión de APIs, proporcionando seguridad, rate limiting y monitoreo unificado para todos los clientes.

Características:

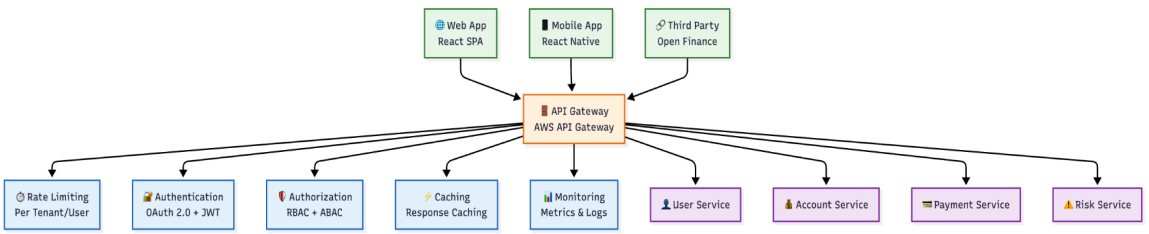
- Rate Limiting: Por tenant y plan (Basic: 100 req/min, Premium: 1000 req/min, Enterprise: 10000 req/min)
- Autenticación: OAuth 2.0 + OpenID Connect con MFA
- Autorización: RBAC + ABAC para control granular
- Caching: Response caching para mejorar performance
- Monitoreo: Métricas y logs centralizados

Flujos de Autenticación:

- OAuth 2.0: Authorization Code y Client Credentials
- OpenID Connect: Integración con identity provider
- MFA: SMS, TOTP, Biometric para operaciones críticas

Beneficios:

- Seguridad Centralizada: Un punto de control para todas las APIs
- Rate Limiting: Protección contra abuso y garantía de SLAs
- Monitoreo: Visibilidad completa del tráfico de APIs
- Versionado: Gestión de versiones de APIs



Patrones de Integración

Circuit Breaker Pattern

Circuit Breaker protege el sistema de fallos en cascada cuando servicios externos fallan, proporcionando degradación elegante.

Estados del Circuit Breaker:

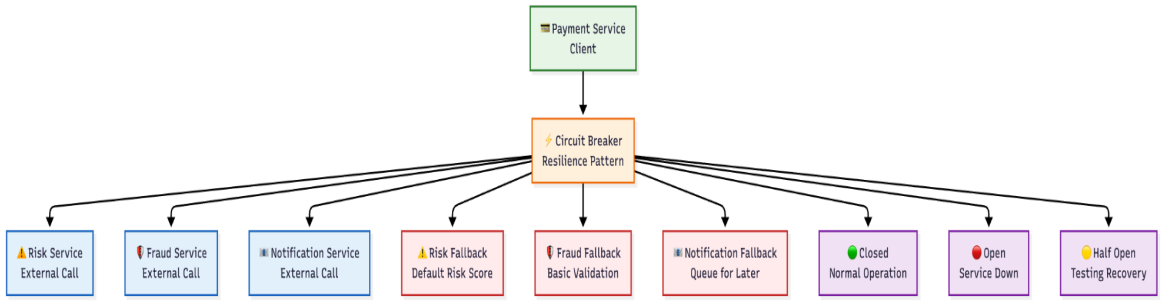
- Closed: Operación normal, todas las llamadas pasan
- Open: Servicio fallando, llamadas bloqueadas
- Half-Open: Probando recuperación del servicio

Implementación:

- Fallback Services: Servicios de respaldo para cada dependencia crítica
- Risk Fallback: Score de riesgo por defecto cuando Risk Service falla
- Fraud Fallback: Validación básica cuando Fraud Service falla
- Notification Fallback: Cola para procesamiento posterior

Beneficios:

- Resilencia: Prevención de fallos en cascada
- Degradación Elegante: Servicio continúa con funcionalidad limitada
- Recuperación Automática: Detección automática de recuperación de servicios
- Monitoreo: Alertas cuando circuitos se abren



Patrones de Integración

Saga Pattern

Saga Pattern maneja transacciones distribuidas complejas, asegurando consistencia eventual con compensación automática en caso de fallos.

Flujo de Saga de Pago:

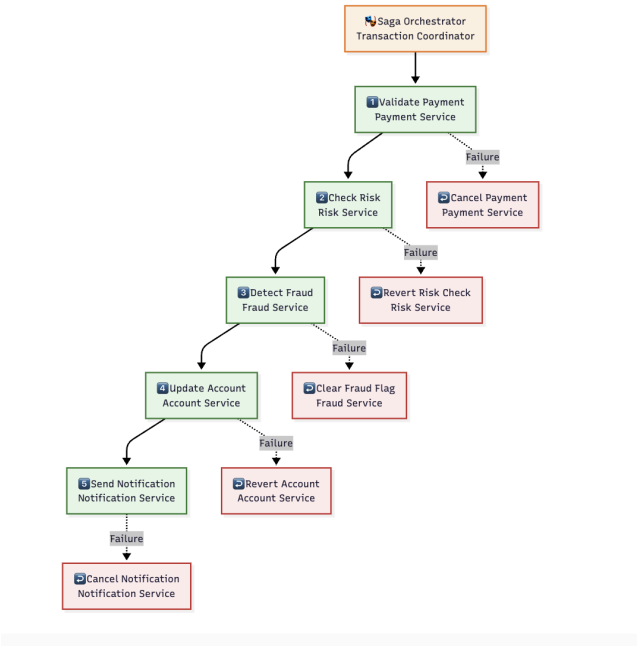
- Validate Payment: Validación inicial del pago
- Check Risk: Evaluación de riesgo
- Detect Fraud: Detección de fraude
- Update Account: Actualización de cuenta
- Send Notification: Envío de notificación

Compensación Automática:

- Si cualquier paso falla, se ejecutan pasos de compensación en orden inverso
- Cancel Payment: Reversión de validación
- Revert Risk Check: Limpieza de evaluación de riesgo
- Clear Fraud Flag: Eliminación de flags de fraude
- Revert Account: Reversión de cambios en cuenta
- Cancel Notification: Cancelación de notificaciones

Beneficios:

- Consistencia: Garantiza consistencia eventual en transacciones distribuidas
- Recuperación: Compensación automática en caso de fallos
- Auditabilidad: Trazabilidad completa de transacciones
- Flexibilidad: Fácil modificación de flujos de negocio



Patrones de Integración

Patrones de Comunicación

Síncrona (Request-Response)

- REST APIs: Para operaciones que requieren respuesta inmediata
- GraphQL: Para consultas flexibles y eficientes
- Protocolos: HTTPS/TLS 1.3, OAuth 2.0 + JWT

Asíncrona (Event-Driven)

- Message Queues: Kafka para eventos de negocio
- Webhooks: Para notificaciones a sistemas externos
- Event Streaming: Procesamiento en tiempo real

Patrones de Resiliencia

Retry Pattern

- Max Attempts: 3 intentos con backoff exponencial
- Backoff: 1 segundo de delay inicial
- Circuit Breaker: Integrado con retry para evitar llamadas innecesarias

Timeout Pattern

- Timeout: 5 segundos para operaciones críticas
- Graceful Degradation: Fallback cuando timeout se alcanza
- Monitoring: Alertas cuando timeouts ocurren frecuentemente

Bulkhead Pattern

- Thread Pools: Separación de recursos por tipo de operación
- Payment Executor: Pool dedicado para procesamiento de pagos

Patrones de Seguridad

API Key Pattern

- Authentication: Validación de API keys por tenant
- Rate Limiting: Límites específicos por API key
- Monitoring: Tracking de uso por API key

JWT Token Pattern

- Token Generation: JWT con claims específicos
- Expiration: 24 horas de validez
- Signature: HS512 para integridad
- Claims: Subject, Issued At, Expiration

Patrones de Integración

Patrones de Monitoreo

Health Check Pattern

- Liveness Probe: Verificación de que el servicio está vivo
- Readiness Probe: Verificación de que el servicio está listo
- Dependencies: Verificación de dependencias críticas

Metrics Pattern

- Business Metrics: Contadores de pagos, transacciones
- Technical Metrics: Latencia, throughput, error rate
- Custom Metrics: Métricas específicas de negocio