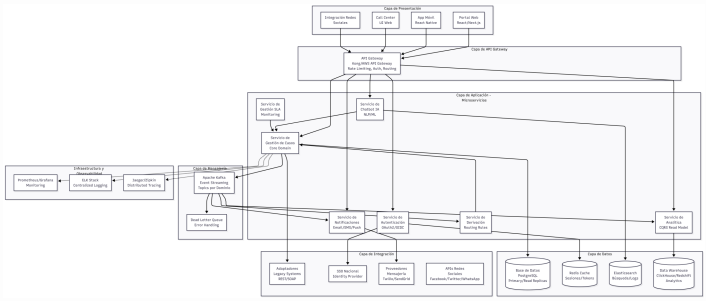
1. Diagrama de Arquitectura Detallado - Conecta360

Visión General

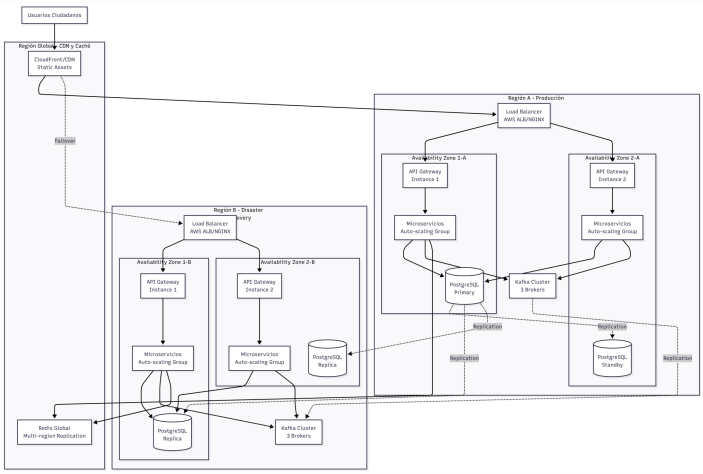
La arquitectura de Conecta360 está diseñada siguiendo los principios de **microservicios**, **event-driven architecture** y **arquitectura en capas**, asegurando escalabilidad, resiliencia y mantenibilidad.

Diagrama de Arquitectura - Vista de Niveles (C4 Model Level 1)



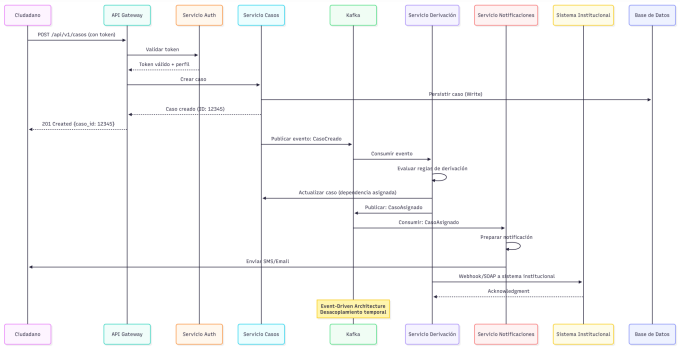
images/C4ModelLevel1.png

Diagrama de Arquitectura - Vista de Despliegue (Multi-Región)



images/vista-despliegue-multi-region.png

Flujo de Procesamiento de un Caso



images/flujo-de-procesamiento-de-un-caso.png

Capas del Sistema

1. Capa de Presentación

- **Portal Web**: React/Next.js con SSR para SEO

- **App Móvil**: React Native (iOS/Android)
- **Call Center UI**: Aplicación web interna para operadores
- **Widget Embebible**: Para integración en sitios de dependencias

2. Capa de API Gateway

- **Función**: Punto único de entrada, enrutamiento, autenticación, rate limiting
- **Tecnología**: Kong, AWS API Gateway o NGINX Plus
- **Características**:
  - Rate limiting por IP/usuario
  - Autenticación OAuth2/OIDC
  - Transformación de requests/responses
  - Circuit breaker patterns
  - Request/Response logging

3. Capa de Aplicación (Microservicios)

3.1 Servicio de Autenticación y Autorización

- **Responsabilidad**: Gestión de identidades, SSO, tokens JWT
- **Stack**: Node.js/Python con OAuth2 server

3.2 Servicio de Gestión de Casos (Core Domain)

- **Responsabilidad**: CRUD de casos, workflow, estados
- **Stack**: Java/Spring Boot o .NET Core
- **Patrón**: Domain-Driven Design (DDD)

3.3 Servicio de Notificaciones

- **Responsabilidad**: Email, SMS, push notifications
- **Stack**: Node.js con workers asíncronos

3.4 Servicio de Chatbot IA

- **Responsabilidad**: Clasificación inicial, routing inteligente
- **Stack**: Python con NLP (BERT/GPT-based models)

3.5 Servicio de Derivación

- **Responsabilidad**: Reglas de negocio para asignar casos
- **Stack**: Java/.NET con rule engine (Drools/Ortools)

3.6 Servicio de Gestión SLA

- **Responsabilidad**: Monitoreo de tiempos, alertas
- **Stack**: Go/Rust para alta concurrencia

3.7 Servicio de Analítica (CQRS Read Model)

- **Responsabilidad**: Dashboards, reportes, KPIs
- **Stack**: Node.js con consultas optimizadas

4. Capa de Integración

- **Adaptadores Legacy**: REST/SOAP/gRPC adapters para sistemas antiguos
- **SSO Nacional**: Integración con identity provider gubernamental
- **Proveedores Mensajería**: Twilio, SendGrid, FCM
- **APIs Redes Sociales**: Facebook Graph API, Twitter API, WhatsApp Business API

5. Capa de Mensajería Asíncrona

- **Apache Kafka**: Event streaming para comunicación desacoplada
- **Topics principales**:
  - `casos.creados`
  - `casos.asignados`
  - `casos.actualizados`
  - `casos.cerrados`

- `notificaciones.enviadas`
- `sla.violados`

6. Capa de Datos

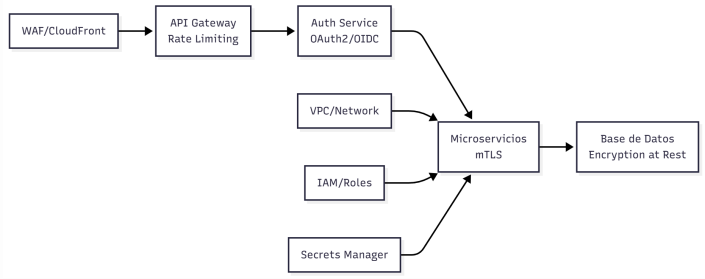
- **PostgreSQL**: Base de datos principal (ACID compliance)
  - Primary-Replica setup multi-región
  - Particionamiento por dependencia y fecha
- **Redis**: Cache distribuido (sesiones, tokens, datos frecuentes)
- **Elasticsearch**: Búsqueda full-text, logs centralizados
- **ClickHouse/Redshift**: Data warehouse para analítica

7. Infraestructura y Observabilidad

- **Monitoring**: Prometheus + Grafana (métricas)
- **Logging**: ELK Stack (Elasticsearch, Logstash, Kibana)
- **Tracing**: Jaeger/Zipkin (distributed tracing)
- **Alerting**: PagerDuty/OpsGenie

Seguridad y Compliance

Seguridad en Capas



images/seguridad-de-capas.png

Consideraciones de Seguridad

1. **Encriptación en tránsito**: TLS 1.3 en todas las comunicaciones
2. **Encriptación en reposo**: AES-256 para bases de datos
3. **Segregación de datos**: Aislamiento lógico por dependencia (multitenancy)
4. **Auditoría**: Log completo de todas las operaciones (quién, qué, cuándo)
5. **Control de acceso**: RBAC con roles granulares (ciudadano, operador, supervisor, admin)
6. **Compliance**: GDPR-like, protección de datos personales

Escalabilidad

Estrategia de Escalado

- **Horizontal**: Auto-scaling basado en CPU/memoria/request rate
- **Vertical**: Instancias optimizadas por carga de trabajo
- **Cache**: Redis para reducir carga en base de datos
- **CDN**: CloudFront para assets estáticos
- **Read Replicas**: Distribución de lecturas en múltiples réplicas

Capacidad Estimada

- **500,000 solicitudes/día** = ~5,787 req/s (peak)
- **Pico estimado**: 20,000 req/s (día de alta demanda)
- **Replicas necesarias**: ~40-50 instancias de microservicios (con balanceo)

Resiliencia y Alta Disponibilidad

Patrones Aplicados

1. **Circuit Breaker**: Evita cascading failures
2. **Retry con exponential backoff**: Para operaciones transitorias
3. **Bulkhead**: Aislamiento de recursos críticos
4. **Health Checks**: Endpoints `/health` y `/ready`
5. **Graceful Shutdown**: Cierre ordenado sin pérdida de datos
6. **Database Connection Pooling**: Optimización de conexiones

Disaster Recovery

- **RTO (Recovery Time Objective)**: < 1 hora
- **RPO (Recovery Point Objective)**: < 15 minutos
- **Backup Strategy**: Diarios incrementales + backups completos semanales
- **Failover automático**: Route 53 health checks con DNS failover

Consideraciones de Rendimiento

Optimizaciones

1. **CQRS**: Separación de modelos de lectura/escritura
2. **Database Indexing**: Índices estratégicos en campos frecuentes
3. **Pagination**: Todos los endpoints de listado con paginación
4. **Compression**: Gzip/Brotli para responses grandes
5. **Async Processing**: Operaciones pesadas en background workers
6. **Connection Pooling**: Gestión eficiente de conexiones DB

SLA de Rendimiento

- **P95 Response Time**: < 1.5s (requerimiento cumplido)
- **P99 Response Time**: < 3s
- **Availability**: 99.9% (≈ 8.76 horas de downtime/año)

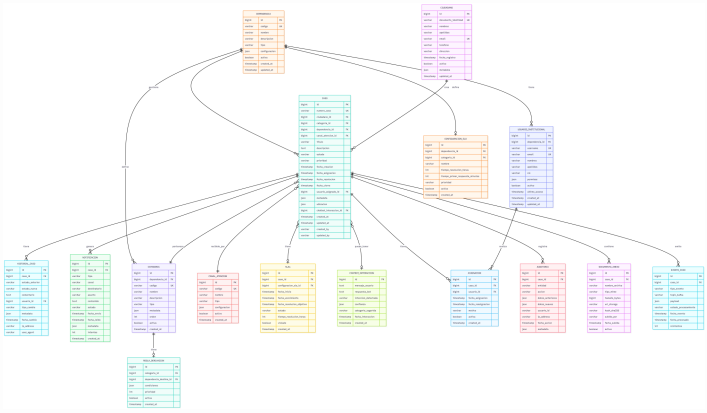
Siguiente: Ver Base de Datos

2. Modelo de Base de Datos - Conecta360

Visión General

El modelo de datos está diseñado para soportar **multitenancy** (múltiples instituciones), **auditoría completa**, **escalabilidad horizontal** y **alto rendimiento de consultas**.

Diagrama Entidad-Relación (Modelo Conceptual)



images/diagrama-entidad-relacion.png

Esquema Relacional Detallado

Tablas Principales

1. `ciudadanos`

Almacena información de ciudadanos registrados en el sistema.

```
CREATE TABLE ciudadanos (
    id BIGSERIAL PRIMARY KEY,
    documento_identidad VARCHAR(50) UNIQUE NOT NULL,
    nombres VARCHAR(200) NOT NULL,
    apellidos VARCHAR(200) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    telefono VARCHAR(20),
    direccion TEXT,
    fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    activo BOOLEAN DEFAULT TRUE,
    metadata JSONB, -- Datos adicionales flexibles
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    INDEX idx_ciudadanos_documento (documento_identidad),
    INDEX idx_ciudadanos_email (email),
    INDEX idx_ciudadanos_activo (activo)
);

-- Particionamiento por fecha_registro para escalabilidad
CREATE TABLE ciudadanos_2024 PARTITION OF ciudadanos
FOR VALUES FROM ('2024-01-01') TO ('2025-01-01');
```

2. `casos` (Core Table)

Tabla central que almacena todos los casos ciudadanos.

```
CREATE TABLE casos (
    id BIGSERIAL PRIMARY KEY,
    numero_caso VARCHAR(50) UNIQUE NOT NULL, -- Formato: CRV-2024-00012345
    ciudadano_id BIGINT NOT NULL REFERENCES ciudadanos(id),
    categoria_id BIGINT NOT NULL REFERENCES categorias(id),
    dependencia_id BIGINT NOT NULL REFERENCES dependencias(id),
    canal_atencion_id BIGINT NOT NULL REFERENCES canales_atencion(id),
    titulo VARCHAR(500) NOT NULL,
    descripcion TEXT NOT NULL,
    estado VARCHAR(50) NOT NULL, -- PENDIENTE, ASIGNADO, EN_PROCESO, RESUE
    prioridad VARCHAR(20) NOT NULL, -- BAJA, MEDIA, ALTA, URGENTE
    fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    fecha_asignacion TIMESTAMP,
    fecha_resolucion TIMESTAMP,
    fecha_cierre TIMESTAMP,
    usuario_asignado_id BIGINT REFERENCES usuarios_institucionales(id),
    metadata JSONB, -- Campos flexibles por dependencia
    ubicacion JSONB, -- {lat, lng, direccion, barrio}
    chatbot_interaccion_id BIGINT REFERENCES chatbot_interacciones(id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    created_by VARCHAR(100),
    updated_by VARCHAR(100),

    CONSTRAINT fk_caso_ciudadano FOREIGN KEY (ciudadano_id) REFERENCES ciu
    CONSTRAINT fk_caso_categoria FOREIGN KEY (categoria_id) REFERENCES cat
    CONSTRAINT fk_caso_dependencia FOREIGN KEY (dependencia_id) REFERENCES
    CONSTRAINT fk_caso_canal FOREIGN KEY (canal_atencion_id) REFERENCES ca

    INDEX idx_casos_numero (numero_caso),
    INDEX idx_casos_ciudadano (ciudadano_id),
    INDEX idx_casos_estado (estado),
    INDEX idx_casos_dependencia (dependencia_id),
    INDEX idx_casos_fecha_creacion (fecha_creacion),
    INDEX idx_casos_asignado (usuario_asignado_id),
    INDEX idx_casos_estado_fecha (estado, fecha_creacion),
    INDEX idx_casos_metadata_gin (metadata) USING GIN, -- Búsqueda en JSON
    INDEX idx_casos_ubicacion_gin (ubicacion) USING GIN
) PARTITION BY RANGE (fecha_creacion);

-- Particionamiento mensual para optimizar consultas y mantenimiento
CREATE TABLE casos_2024_01 PARTITION OF casos
FOR VALUES FROM ('2024-01-01') TO ('2024-02-01');
CREATE TABLE casos_2024_02 PARTITION OF casos
FOR VALUES FROM ('2024-02-01') TO ('2024-03-01');
-- ... (más particiones según necesidad)
```

3. `dependencias`

Instituciones gubernamentales que gestionan casos.

```
CREATE TABLE dependencias (
    id BIGSERIAL PRIMARY KEY,
    codigo VARCHAR(50) UNIQUE NOT NULL, -- Ej: MIN-SALUD, MUN-ALCALDIA
    nombre VARCHAR(200) NOT NULL,
    descripcion TEXT,
    tipo VARCHAR(50), -- MINISTERIO, MUNICIPALIDAD, ORGANISMO_AUTONOMO
    configuracion JSONB, -- Configuración específica por dependencia
    activa BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    INDEX idx_dependencias_codigo (codigo),
    INDEX idx_dependencias_activa (activa)
);
```

4. `categorias`

Categorización de casos por tipo de solicitud.

```
CREATE TABLE categorias (
    id BIGSERIAL PRIMARY KEY,
    dependencia_id BIGINT NOT NULL REFERENCES dependencias(id),
    codigo VARCHAR(50) NOT NULL,
    nombre VARCHAR(200) NOT NULL,
    descripcion TEXT,
    tipo VARCHAR(50), -- INCIDENCIA, SOLICITUD, QUEJA, RECLAMO
    metadata JSONB,
    orden INT DEFAULT 0,
    activa BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT uk_categoria_dependencia_codigo UNIQUE (dependencia_id, cod

    INDEX idx_categorias_dependencia (dependencia_id),
    INDEX idx_categorias_activa (activa)
);
```

5. `historial_casos`

Registro completo de cambios de estado y acciones sobre casos.

```
CREATE TABLE historial_casos (
    id BIGSERIAL PRIMARY KEY,
    caso_id BIGINT NOT NULL REFERENCES casos(id),
    estado_anterior VARCHAR(50),
    estado_nuevo VARCHAR(50),
    comentario TEXT,
    usuario_id BIGINT REFERENCES usuarios_institucionales(id),
    tipo_cambio VARCHAR(50), -- CAMBIO_ESTADO, ASIGNACION, REASIGNACION, C
    metadata JSONB,
    fecha_cambio TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    ip_address VARCHAR(45),
    user_agent VARCHAR(500),

    INDEX idx_historial_caso (caso_id),
    INDEX idx_historial_fecha (fecha_cambio),
    INDEX idx_historial_usuario (usuario_id)
) PARTITION BY RANGE (fecha_cambio);
```

6. `notificaciones`

Registro de todas las notificaciones enviadas a ciudadanos.

```sql
CREATE TABLE notificaciones (
    id BIGSERIAL PRIMARY KEY,
    caso_id BIGINT NOT NULL REFERENCES casos(id),
    tipo VARCHAR(50) NOT NULL, -- CREACION, ASIGNACION, ACTUALIZACION, CIE
    canal VARCHAR(20) NOT NULL, -- EMAIL, SMS, PUSH, WHATSAPP
    destinatario VARCHAR(255) NOT NULL,
    asunto VARCHAR(500),
    contenido TEXT NOT NULL,
    estado VARCHAR(50) DEFAULT 'PENDIENTE', -- PENDIENTE, ENVIADO, FALLIDO
    fecha_envio TIMESTAMP,
    fecha_leido TIMESTAMP,
    metadata JSONB,
    intentos INT DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    INDEX idx_notificaciones_caso (caso_id),
    INDEX idx_notificaciones_estado (estado),
    INDEX idx_notificaciones_fecha (fecha_envio),
    INDEX idx_notificaciones_destinatario (destinatario)
) PARTITION BY RANGE (fecha_envio);
```

7. `usuarios_institucionales`

Funcionarios y operadores que gestionan casos.

```sql
CREATE TABLE usuarios_institucionales (
    id BIGSERIAL PRIMARY KEY,
    dependencia_id BIGINT NOT NULL REFERENCES dependencias(id),
    username VARCHAR(100) UNIQUE NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    nombres VARCHAR(200) NOT NULL,
    apellidos VARCHAR(200) NOT NULL,
    rol VARCHAR(50) NOT NULL, -- OPERADOR, SUPERVISOR, ADMINISTRADOR
    permisos JSONB, -- Permisos granulares por funcionalidad
    activo BOOLEAN DEFAULT TRUE,
    ultimo_acceso TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    INDEX idx_usuarios_username (username),
    INDEX idx_usuarios_email (email),
    INDEX idx_usuarios_dependencia (dependencia_id),
    INDEX idx_usuarios_activo (activo),
    INDEX idx_usuarios_rol (rol)
);
```

8. `slas`

Seguimiento de Service Level Agreements por caso.

```sql
CREATE TABLE slas (
    id BIGSERIAL PRIMARY KEY,
    caso_id BIGINT NOT NULL REFERENCES casos(id),
    configuracion_sla_id BIGINT REFERENCES configuraciones_sla(id),
    fecha_inicio TIMESTAMP NOT NULL,
    fecha_vencimiento TIMESTAMP NOT NULL,
    fecha_resolucion_objetivo TIMESTAMP NOT NULL,
    estado VARCHAR(50) DEFAULT 'VIGENTE', -- VIGENTE, CUMPLIDO, VIOLADO
    tiempo_resolucion_horas INT NOT NULL,
    violado BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    INDEX idx_slas_caso (caso_id),
    INDEX idx_slas_vencimiento (fecha_vencimiento),
    INDEX idx_slas_violado (violado),
    INDEX idx_slas_estado (estado)
);
```

Tablas de Integración y Eventos

9. `eventos_casos`

Registro de eventos para publicación en Kafka (event sourcing pattern).

```sql
CREATE TABLE eventos_casos (
    id BIGSERIAL PRIMARY KEY,
    caso_id BIGINT NOT NULL REFERENCES casos(id),
    tipo_evento VARCHAR(100) NOT NULL, -- caso.creado, caso.asignado, caso
    topic_kafka VARCHAR(100) NOT NULL,
    payload JSONB NOT NULL,
    estado_procesamiento VARCHAR(50) DEFAULT 'PENDIENTE', -- PENDIENTE, PR
    fecha_evento TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    fecha_procesado TIMESTAMP,
    reintentos INT DEFAULT 0,

    INDEX idx_eventos_caso (caso_id),
    INDEX idx_eventos_procesamiento (estado_procesamiento),
    INDEX idx_eventos_fecha (fecha_evento)
) PARTITION BY RANGE (fecha_evento);
```

10. `auditoria`

Tabla de auditoría completa para compliance y trazabilidad.

```sql
CREATE TABLE auditoria (
    id BIGSERIAL PRIMARY KEY,
    caso_id BIGINT REFERENCES casos(id),
    entidad VARCHAR(100) NOT NULL,
    accion VARCHAR(50) NOT NULL, -- CREATE, UPDATE, DELETE, ASSIGN, CLOSE
    datos_anteriores JSONB,
    datos_nuevos JSONB,
    usuario_id VARCHAR(100),
    ip_address VARCHAR(45),
    fecha_accion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    metadata JSONB,

    INDEX idx_auditoria_caso (caso_id),
    INDEX idx_auditoria_entidad (entidad),
    INDEX idx_auditoria_fecha (fecha_accion),
    INDEX idx_auditoria_usuario (usuario_id)
) PARTITION BY RANGE (fecha_accion);
```

Modelo de Datos - Consideraciones de Diseño

1. Multitenancy

- **Aislamiento lógico**: Columna `dependencia_id` en todas las tablas relevantes
- **Filtrado automático**: Views o row-level security (RLS) en PostgreSQL
- **Método**: Soft multitenancy (más simple, suficiente para este caso)

2. Particionamiento

- **Razón**: Escalabilidad y mantenimiento
- **Estrategia**: Particionamiento por rango de fecha en tablas de alto volumen
  - `casos`: Por mes
  - `historial_casos`: Por mes
  - `notificaciones`: Por mes
  - `eventos_casos`: Por mes
  - `auditoria`: Por mes

3. Índices Estratégicos

- **Índices B-tree**: Para consultas de igualdad y rangos
- **Índices GIN**: Para búsquedas en campos JSONB
- **Índices compuestos**: Para consultas frecuentes multi-columna

4. JSONB para Flexibilidad

- Uso de `JSONB` en campos `metadata` para:
  - Adaptarse a necesidades específicas por dependencia
  - Evolución del esquema sin migraciones costosas
  - Almacenamiento eficiente y búsqueda indexada

5. Normalización vs Desnormalización

- **Normalizado**: Relaciones principales (evita redundancia)
- **Desnormalizado**: Campos calculados en `casos` para consultas rápidas
  - Ejemplo: `usuario_asignado_id` redundante pero útil para performance

6. Naming Conventions

- **Formato**: `snake_case` para tablas y columnas
- **Primary Keys**: Siempre `id` (BIGSERIAL)
- **Foreign Keys**: Formato `{tabla}_id`
- **Timestamps**: `created_at`, `updated_at`, `deleted_at` (soft delete)

Estrategia de Migración y Versionado

Versionado de Esquema

- Herramienta: **Flyway** o **Liquibase**
- Migraciones incrementales versionadas
- Rollback strategy documentado

Ejemplo de Migración

```
-- V1__create_initial_schema.sql
-- V2__add_indexes.sql
-- V3__partition_casos_table.sql
-- ...
```

Consideraciones de Performance

Query Optimization

1. **Read Replicas**: Todas las consultas de lectura a réplicas
2. **Connection Pooling**: PgBouncer o similar
3. **Materialized Views**: Para reportes complejos (refresh programado)
4. **Caching Layer**: Redis para queries frecuentes

Ejemplo de Query Optimizada

```
-- Consulta de casos con joins optimizados
SELECT
    c.id,
    c.numero_caso,
    c.titulo,
    c.estado,
    ciu.nombres || ' ' || ciu.apellidos AS ciudadano,
    dep.nombre AS dependencia,
    cat.nombre AS categoria
FROM casos c
INNER JOIN ciudadanos ciu ON c.ciudadano_id = ciu.id
INNER JOIN dependencias dep ON c.dependencia_id = dep.id
INNER JOIN categorias cat ON c.categoria_id = cat.id
WHERE c.dependencia_id = $1
  AND c.estado = $2
  AND c.fecha_creacion >= $3
ORDER BY c.fecha_creacion DESC
LIMIT $4 OFFSET $5;
```

Seguridad de Datos

Encriptación

- **At Rest**: AES-256 (PostgreSQL TDE o disk-level encryption)
- **In Transit**: TLS 1.3
- **Backups**: Encriptados con claves separadas

Row-Level Security (RLS)

```sql
-- Ejemplo de política RLS para multitenancy
CREATE POLICY dependencia_isolation ON casos
    FOR ALL
    TO aplicacion_role
    USING (dependencia_id = current_setting('app.current_dependencia_id'):
```

Siguiente: Ver Decisiones Tecnológicas

3. Justificación de Decisiones Tecnológicas - Conecta360

Visión General

Este documento justifica las decisiones tecnológicas clave del proyecto Conecta360, evaluando alternativas y explicando por qué se eligieron determinadas tecnologías, frameworks y patrones arquitectónicos.

Stack Tecnológico Seleccionado

Lenguajes de Programación

Java/Spring Boot (Microservicios Core)

Justificación: - **Ecosistema maduro**: Spring Boot ofrece autoconfiguración, actuaores, testing integrado - **Performance**: JVM optimizada para aplicaciones empresariales - **Seguridad**: Spring Security robusto con OAuth2/OIDC - **Observabilidad**: Integración nativa con Micrometer, Actuator - **Comunidad**: Amplio soporte y documentación

Alternativas consideradas: - **.NET Core**: Similar en características, pero menor ecosistema en entornos gubernamentales de la región - **Node.js**: Excelente para I/O, pero menos adecuado para lógica de negocio compleja - **Go**: Performance excelente, pero menor ecosistema empresarial

Uso en Conecta360: - Servicio de Gestión de Casos (core domain) - Servicio de Derivación (reglas complejas) - Servicio de Gestión SLA

Node.js/Express (Servicios Asíncronos)

Justificación: - **I/O no bloqueante**: Ideal para servicios de notificaciones, webhooks, APIs externas - **Ecosistema npm**: Librerías para integraciones (Twilio, SendGrid, etc.) - **Escalabilidad**: Excelente para alta concurrencia de operaciones I/O - **Rapidez de desarrollo**: Desarrollo rápido para servicios simples

Alternativas consideradas: - **Python/FastAPI**: Similar performance, pero menor ecosistema para integraciones web - **Ruby/Rails**: Menor performance en alta concurrencia

Uso en Conecta360: - Servicio de Notificaciones - Servicio de Autenticación (OAuth2 server) - Servicio de Analítica (CQRS read model)

Python (Servicios de IA/NLP)

Justificación: - **Ecosistema ML/AI**: TensorFlow, PyTorch, Transformers, spaCy - **Librerías NLP**: NLTK, spaCy para procesamiento de lenguaje natural - **Integración modelos**: Fácil integración con modelos pre-entrenados (BERT, GPT) - **Rapidez prototipo**: Desarrollo rápido de modelos ML

Uso en Conecta360: - Servicio de Chatbot IA (clasificación inicial, routing inteligente)

Alternativas consideradas: - **Java con DL4J**: Menor ecosistema de modelos pre-entrenados - **Rust con candle**: Más complejo de desarrollar

---

Frameworks y Librerías

Spring Boot (Java)

Características clave: - Spring Cloud Gateway para API routing - Spring Data JPA para acceso a datos - Spring Security para autenticación/autorización - Spring Cloud Stream para Kafka - Resilience4j para circuit breakers

Express.js (Node.js)

Características clave: - Express para servidores HTTP - Bull/Agenda para job queues - Axios para HTTP clients - Passport.js para OAuth2

FastAPI (Python)

Características clave: - FastAPI para APIs async - Pydantic para validación - Transformers para modelos NLP - Celery para background tasks

---

Bases de Datos

PostgreSQL (Base de Datos Principal)

Justificación: - **ACID compliance**: Transacciones robustas para datos críticos - **JSONB**: Soporte nativo para campos flexibles (metadata) - **Particionamiento**: Partitioning nativo para escalabilidad - **Full-text search**: Búsqueda de texto completo integrada - **Row-Level Security (RLS)**: Para multitenancy y seguridad - **Performance**: Excelente para operaciones OLTP - **Open source**: Sin costos de licencia

Alternativas consideradas: - **MySQL/MariaDB**: Menor soporte para JSONB y particionamiento avanzado - **Oracle**: Costos de licencia prohibitivos - **SQL Server**: Costos de licencia y menor soporte multiplataforma - **MongoDB**: Sin ACID guarantees necesarios, menor necesidad de flexibilidad documental total

Configuración: - Primary-Replica setup multi-región - Read replicas para distribución de carga - Particionamiento mensual en tablas de alto volumen - Connection pooling (PgBouncer)

Redis (Caché y Sesiones)

Justificación: - **Performance**: Alta velocidad para operaciones de lectura - **Estructuras de datos**: Sets, hashes, sorted sets para casos de uso complejos - **Pub/Sub**: Para notificaciones en tiempo real - **Tiempo de vida**: TTL automático para expiración de sesiones - **Replicación**: Redis Sentinel para alta disponibilidad

Alternativas consideradas: - **Memcached**: Menor funcionalidad, solo clave-valor simple - **Hazelcast**: Más complejo, mejor para distributed computing

Uso en Conecta360: - Caché de sesiones de usuarios - Caché de tokens OAuth2 - Caché de datos frecuentes (dependencias, categorías) - Rate limiting tokens

---

Elasticsearch (Búsqueda y Logging)

Justificación: - **Búsqueda full-text**: Búsqueda avanzada en casos, historial - **Agregaciones**: Para dashboards y reportes complejos - **Escalabilidad**: Sharding automático para grandes volúmenes - **Logging**: Integración con ELK Stack

Alternativas consideradas: - **OpenSearch**: Fork de Elasticsearch, similar funcionalidad - **Solr**: Similar, pero menor ecosistema

Uso en Conecta360: - Búsqueda avanzada de casos - Centralized logging (con Logstash, Kibana) - Análisis de interacciones de chatbot

---

ClickHouse/Amazon Redshift (Data Warehouse)

Justificación: - **Columnar storage**: Optimizado para queries analíticas - **Escalabilidad**: Alta capacidad para grandes volúmenes - **Performance**: Queries complejas en segundos/minutos - **Integración BI**: Conexión directa con Power BI, Tableau

Alternativas consideradas: - **BigQuery**: Similar, pero vendor lock-in con Google Cloud -

Snowflake: Costos más altos - **PostgreSQL con extensiones**: Menor performance para analítica

Uso en Conecta360: - Data warehouse para reportes históricos - Exportación a Power BI - Análisis de tendencias y KPIs

---

Mensajería Asíncrona

Apache Kafka

Justificación: - **Event streaming**: Perfecto para event-driven architecture - **Durabilidad**: Messages persistidos en disco con replicación - **Escalabilidad**: Horizontal scaling con partitions - **Ordering**: Garantía de orden por partition key - **Throughput**: Alto throughput (millones de mensajes/segundo) - **Ecosistema**: Kafka Connect, Kafka Streams

Alternativas consideradas: - **RabbitMQ**: Mejor para message queuing tradicional, menor throughput - **Amazon SQS**: Vendor lock-in, menor flexibilidad - **NATS**: Más simple, pero menor funcionalidad enterprise

Configuración en Conecta360: - **Topics principales:** - `casos.creados` - `casos.asignados` - `casos.actualizados` - `casos.cerrados` - `notificaciones.enviadas` - `sla.violados` - **Replication factor**: 3 (para alta disponibilidad) - **Partitions**: 6-12 por topic (según volumen) - **Retention**: 7 días (configurable)

Patrones aplicados: - **Event Sourcing**: Eventos como fuente de verdad - **CQRS**: Separación de lectura/escritura - **Saga Pattern**: Para transacciones distribuidas

---

API Gateway

Kong / AWS API Gateway

Justificación: - **Ruteo centralizado**: Punto único de entrada para todos los servicios - **Rate limiting**: Protección contra abuso - **Autenticación**: OAuth2/OIDC integration - **Transformación**: Request/response transformation - **Monitoreo**: Métricas y logging centralizados - **Plugins**: Ecosistema extensible

Alternativas consideradas: - **NGINX Plus**: Similar funcionalidad, menor ecosistema de plugins - **Istio**: Más complejo, orientado a service mesh completo - **Traefik**: Más simple, menor funcionalidad enterprise

Kong (Recomendado para on-premise/hybrid): - Open source core - Plugin ecosystem - Database-less mode para alta disponibilidad - Rate limiting avanzado

AWS API Gateway (Recomendado para cloud full): - Serverless (pago por uso) - Integración nativa con AWS services - API versioning automático - WebSocket support

---

Infraestructura y Nube

Arquitectura Multi-Cloud / Hybrid Cloud

Justificación: - **Redundancia**: Evitar vendor lock-in, resiliencia ante fallos de proveedor - **Compliance**: Algunos datos pueden requerir estar en-región - **Costos**: Optimización de costos por workload

Opciones evaluadas: 1. **AWS (Amazon Web Services)** - Pros: Madurez, servicios gestionados, global reach - Contras: Costos pueden ser altos, vendor lock-in potencial

2. **Azure Government**
   - Pros: Compliance gubernamental, integración con Office 365
   - Contras: Menor presencia en Latinoamérica
3. **Google Cloud Platform (GCP)**
   - Pros: ML/AI services excelentes, pricing competitivo
   - Contras: Menor adopción enterprise en la región

Recomendación: AWS con estrategia de salida - Usar servicios estándar (evitar servicios propietarios) - Multi-región deployment - Considerar Azure para workloads específicos de compliance

---

Contenedores y Orquestación

Kubernetes (K8s)

Justificación: - **Auto-scaling**: Horizontal Pod Autoscaling basado en métricas - **Service discovery**: DNS interno, load balancing automático - **Rolling updates**: Actualizaciones sin downtime - **Health checks**: Liveness y readiness probes - **Resource management**: CPU/memoria limits y requests

Alternativas consideradas: - **Docker Swarm**: Más simple, menor funcionalidad - **Nomad**: Menor adopción, ecosistema más pequeño - **EKS/GKE/AKS**: Managed Kubernetes (recomendado para producción)

Configuración: - **EKS (AWS) o GKE (GCP)**: Managed Kubernetes service - **Namespaces**: Separación por ambiente (dev, staging, prod) - **HPA**: Auto-scaling basado en CPU/memoria/request rate - **Service Mesh (Opcional)**: Istio o Linkerd para observabilidad avanzada

---

Docker

Justificación: - **Containers**: Empaquetado consistente de aplicaciones - **Multi-stage builds**: Optimización de imágenes - **Estandarización**: Desarrollo, staging y producción idénticos

---

Observabilidad

Prometheus + Grafana

Justificación: - **Prometheus**: Time-series database, pull-based metrics - **Grafana**: Visualización de dashboards - **Ecosistema**: Integración con Kubernetes, exporters estándar - **Open source**: Sin costos de licencia

Alternativas consideradas: - **Datadog**: Costos altos para gran escala - **New Relic**: Costos altos, menor control - **CloudWatch**: Vendor lock-in con AWS

ELK Stack (Elasticsearch, Logstash, Kibana)

Justificación: - **Logging centralizado**: Todos los logs en un solo lugar - **Búsqueda avanzada**: Full-text search en logs - **Visualización**: Kibana para análisis de logs - **Open source**: Elasticsearch open source version

---

Jaeger / Zipkin

Justificación: - **Distributed tracing**: Seguimiento de requests a través de servicios - **Performance analysis**: Identificar cuellos de botella - **Open source**: Sin costos

---

Patrones Arquitectónicos Aplicados

1. Microservicios

Justificación: - **Escalabilidad independiente**: Cada servicio escala según necesidad - **Tecnología heterogénea**: Diferentes stacks por servicio - **Deployment independiente**: Releases sin afectar otros servicios - **Aislamiento de fallos**: Fallo de un servicio no afecta a otros

Desafíos mitigados: - **Complejidad de deployment**: Kubernetes y CI/CD automatizado - **Service discovery**: Kubernetes DNS y API Gateway - **Distributed transactions**: Event-driven architecture con eventual consistency

---

2. Event-Driven Architecture

Justificación: - **Desacoplamiento temporal**: Servicios no necesitan estar disponibles

simultáneamente - **Escalabilidad**: Consumers pueden escalar independientemente - **Flexibilidad**: Fácil agregar nuevos consumers de eventos - **Resiliencia**: Event replay en caso de fallos

Implementación: - **Kafka como event store**: Eventos como fuente de verdad - **Event sourcing**: Historial completo de cambios - **CQRS**: Separación de modelos de lectura/escritura

---

3. CQRS (Command Query Responsibility Segregation)

Justificación: - **Optimización de lectura**: Modelos optimizados para consultas - **Optimización de escritura**: Modelos optimizados para transacciones - **Escalabilidad independiente**: Read models pueden tener múltiples réplicas - **Analítica**: Read models específicos para reportes

Implementación: - **Write model**: PostgreSQL (ACID compliance) - **Read models**: PostgreSQL replicas, Elasticsearch (búsqueda), ClickHouse (analítica) - **Sincronización**: Kafka events para mantener consistencia eventual

---

4. API Gateway Pattern

Justificación: - **Punto único de entrada**: Simplifica clientes - **Cross-cutting concerns**: Auth, rate limiting, logging centralizados - **Versioning**: Múltiples versiones de APIs simultáneas - **Transformation**: Adaptación de requests/responses

---

5. Circuit Breaker Pattern

Justificación: - **Resiliencia**: Evita cascading failures - **Graceful degradation**: Fallback responses cuando servicios fallan - **Rápida recuperación**: Detecta cuando servicios vuelven a estar disponibles

Implementación: - **Resilience4j** (Java) o **opossum** (Node.js) - Configuración por servicio

---

6. Saga Pattern

Justificación: - **Transacciones distribuidas**: Operaciones que involucran múltiples servicios - **Compensación**: Rollback en caso de fallos - **Orquestación o coreografía**: Event-driven sagas

Ejemplo en Conecta360: - Crear caso → Asignar a dependencia → Enviar notificación → Registrar en sistema institucional - Si falla cualquier paso, se compensan los anteriores

---

Seguridad

OAuth2 / OpenID Connect (OIDC)

Justificación: - **Estándar industria**: Ampliamente adoptado - **Seguridad**: Tokens JWT con expiración - **SSO**: Single Sign-On con identity provider nacional - **Autorización granular**: Scopes y claims

Implementación: - **Authorization Server**: Spring Authorization Server o Keycloak - **Resource Servers**: Cada microservicio valida tokens - **Token storage**: Redis para tokens refresh

---

Cifrado

- **En tránsito**: TLS 1.3 en todas las comunicaciones
- **En reposo**: AES-256 (PostgreSQL TDE o disk-level encryption)
- **Secrets**: AWS Secrets Manager o HashiCorp Vault

Evaluación de Alternativas (Matriz)

Base de Datos

Tecnología	Pros	Contras	Decisión
PostgreSQL	ACID, JSONB, particionamiento, open source	Requiere tuning para alta escala	□ Seleccionado
MySQL	Simple, amplia adopción	Menor soporte JSONB avanzado	□ Rechazado
MongoDB	Flexibilidad esquema, horizontal scaling	Sin ACID, menor necesidad documental	□ Rechazado
Oracle	Performance, enterprise features	Costos licencia, vendor lock-in	□ Rechazado

Mensajería

Tecnología	Pros	Contras	Decisión
Apache Kafka	Event streaming, alta throughput, durabilidad	Curva de aprendizaje, operación compleja	□ Seleccionado
RabbitMQ	Simple, message queuing tradicional	Menor throughput, menor escalabilidad	□ Rechazado
Amazon SQS	Gestionado, serverless	Vendor lock-in, menor funcionalidad	□□ Alternativa cloud
NATS	Simple, baja latencia	Menor funcionalidad enterprise	□ Rechazado

API Gateway

Tecnología	Pros	Contras	Decisión
Kong	Open source, extensible, maduro	Operación requiere expertise	□ Seleccionado (on-premise)
AWS API Gateway	Gestionado, serverless, integración AWS	Vendor lock-in, costos a escala	□ Seleccionado (cloud)
NGINX Plus	Performance, simple	Menor ecosistema plugins	□□ Alternativa
Istio	Service mesh completo	Complejidad, overhead	□ Rechazado (demasiado complejo)

Decisiones de Infraestructura

Cloud vs On-Premise

Recomendación: Cloud-native con opción hybrid

Justificación: - **Escalabilidad**: Auto-scaling en cloud es más sencillo - **Costos**: Pay-as-you-go reduce CAPEX - **Disaster Recovery**: Multi-región en cloud es más económico - **Compliance**: Algunos datos pueden requerir on-premise, estrategia hybrid

Estrategia: - **Fase 1**: Cloud completo (AWS/GCP) - **Fase 2**: Evaluar hybrid si hay requerimientos de compliance específicos - **Backup strategy**: On-premise backup para datos críticos si es requerido

---

Kubernetes Managed vs Self-Managed

Recomendación: Managed Kubernetes (EKS/GKE)

Justificación: - **Reducción de complejidad**: Managed control plane - **SLA garantizado**: 99.95% uptime SLA del control plane - **Updates automáticos**: Patches y versiones gestionadas - **Costos**: TCO menor que self-managed a largo plazo

---

Consideraciones de Costos

Estimación Anual (aproximada, 500K req/día)

Componente	Costo Anual Estimado
Infraestructura Cloud (EC2/EKS)	$150K - $300K
Base de Datos (RDS/Aurora)	$50K - $100K
Kafka (MSK o self-managed)	$30K - $60K
Monitoring y Logging	$20K - $40K
CDN y Bandwidth	$10K - $30K
Total Estimado	**$260K - $530K**

Nota: Costos pueden variar significativamente según región, uso real y optimizaciones

Estrategias de Optimización de Costos

1. **Reserved Instances**: Compromiso 1-3 años para workloads estables
2. **Spot Instances**: Para workloads tolerantes a fallos (staging, batch jobs)
3. **Auto-scaling**: Reducir instancias en horas de bajo uso
4. **Data tiering**: Datos antiguos a storage más económico (S3 Glacier)
5. **Caching**: Reducir carga en base de datos (costos de queries)

Roadmap de Adopción Tecnológica

Fase 1: Fundación (Mes 1-3)

- Infraestructura base (Kubernetes, CI/CD)
- Base de datos (PostgreSQL con replicación)
- API Gateway (Kong o AWS API Gateway)
- Servicios core (Casos, Autenticación)

Fase 2: Integración (Mes 4-6)

- Kafka para mensajería asíncrona
- Servicios de integración (Notificaciones, Derivación)
- Observabilidad (Prometheus, ELK)

Fase 3: Optimización (Mes 7-9)

- Chatbot IA (Python/FastAPI)
- Servicio de Analítica (CQRS read models)
- Data Warehouse (ClickHouse/Redshift)

Fase 4: Escalabilidad (Mes 10-12)

- Multi-región deployment
- Optimizaciones de performance
- Disaster recovery completo

Conclusión

Las decisiones tecnológicas están alineadas con los requerimientos de: - **Escalabilidad**: 500K solicitudes diarias - **Disponibilidad**: 99.9% SLA - **Rendimiento**: <1.5s response time - **Seguridad**: Cifrado, autenticación robusta - **Integración**: Sistemas legacy heterogéneos

El stack seleccionado es **open source en su mayoría**, evitando vendor lock-in y reduciendo costos de licencia, mientras mantiene flexibilidad para evolucionar según necesidades futuras.

Siguiente: Ver APIs RESTful

4. Diseño de APIs RESTful - Conecta360

Visión General

Este documento define las APIs RESTful principales del sistema Conecta360, siguiendo los principios REST y las mejores prácticas de diseño de APIs.

Especificación OpenAPI/Swagger

Especificación Completa (OpenAPI 3.0)

```yaml
openapi: 3.0.3
info:
  title: Conecta360 API
  description: |
    Sistema Integral de Atención Ciudadana - Conecta360

    API RESTful para gestión de casos, ciudadanos, dependencias y servicio

    **Autenticación:** OAuth2 / OpenID Connect (Bearer Token)
    **Versioning:** URL-based (v1, v2, etc.)
    **Rate Limiting:** 1000 requests/hour por usuario autenticado
  version: 1.0.0
  contact:
    name: API Support
    email: api-support@conecta360.gov.cv
  license:
    name: Propietario - Gobierno de Costa Verde

servers:
  - url: https://api.conecta360.gov.cv/v1
    description: Servidor de producción
  - url: https://api-staging.conecta360.gov.cv/v1
    description: Servidor de staging

tags:
  - name: Casos
    description: Gestión de casos ciudadanos
  - name: Ciudadanos
    description: Gestión de información de ciudadanos
  - name: Categorías
    description: Categorías de casos por dependencia
  - name: Dependencias
    description: Gestión de dependencias gubernamentales
  - name: Notificaciones
    description: Gestión de notificaciones
  - name: SLAs
    description: Seguimiento de Service Level Agreements
  - name: Analítica
    description: Dashboards y reportes

security:
  - bearerAuth: []

paths:
  /casos:
    get:
      summary: Listar casos
      description: |
        Obtiene una lista paginada de casos. Filtros disponibles por estad
        Los resultados están ordenados por fecha de creación (más reciente
      tags:
        - Casos
      parameters:
        - $ref: '#/components/parameters/PageNumber'
        - $ref: '#/components/parameters/PageSize'
        - $ref: '#/components/parameters/EstadoFilter'
        - $ref: '#/components/parameters/DependenciaFilter'
        - $ref: '#/components/parameters/CategoriaFilter'
        - $ref: '#/components/parameters/FechaDesdeFilter'
```

```yaml
      - $ref: '#/components/parameters/FechaHastaFilter'
      - $ref: '#/components/parameters/BusquedaFilter'
    responses:
      '200':
        description: Lista de casos obtenida exitosamente
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/PaginatedCasos'
            examples:
              success:
                value:
                  data:
                    - numero_caso: "CRV-2024-00012345"
                      titulo: "Bache en calle principal"
                      estado: "ASIGNADO"
                      prioridad: "ALTA"
                      fecha_creacion: "2024-01-15T10:30:00Z"
                      dependencia:
                        id: 1
                        nombre: "Ministerio de Obras Públicas"
                      categoria:
                        id: 5
                        nombre: "Infraestructura Vial"
      '401':
        $ref: '#/components/responses/UnauthorizedError'
      '429':
        $ref: '#/components/responses/RateLimitError'

  post:
    summary: Crear nuevo caso
    description: |
      Crea un nuevo caso ciudadano. El caso se crea inicialmente en esta
    tags:
      - Casos
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/CasoCreateRequest'
          examples:
            incidencia_vial:
              value:
                categoria_id: 5
                dependencia_id: 1
                canal_atencion_id: 1
                titulo: "Bache en calle principal"
                descripcion: "Hay un bache grande en la calle principal
                prioridad: "ALTA"
                ubicacion:
                  lat: -12.046374
                  lng: -77.042793
                  direccion: "Calle Principal, Barrio Los Rosales"
                  barrio: "Los Rosales"
                metadata:
                  foto_url: "https://storage.example.com/fotos/bache-123
                  sugerencia: "Recomendado reparar en horario nocturno"
    responses:
      '201':
        description: Caso creado exitosamente
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/CasoResponse'
            examples:
              success:
                value:
                  id: 12345
                  numero_caso: "CRV-2024-00012345"
                  titulo: "Bache en calle principal"
                  estado: "PENDIENTE"
                  prioridad: "ALTA"
                  fecha_creacion: "2024-01-15T10:30:00Z"
                  mensaje: "Su caso ha sido registrado. Número de caso:
      '400':
        $ref: '#/components/responses/BadRequestError'
      '401':
        $ref: '#/components/responses/UnauthorizedError'
```

```yaml
      '422':
        $ref: '#/components/responses/ValidationError'

  /casos/{numero_caso}:
    get:
      summary: Obtener caso por número
      description: |
        Obtiene la información completa de un caso específico, incluyendo
      tags:
        - Casos
      parameters:
        - name: numero_caso
          in: path
          required: true
          schema:
            type: string
            pattern: '^CRV-\d{4}-\d{8}$'
          example: "CRV-2024-00012345"
      responses:
        '200':
          description: Caso encontrado
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/CasoDetalleResponse'
        '404':
          $ref: '#/components/responses/NotFoundError'

    patch:
      summary: Actualizar caso
      description: |
        Actualiza parcialmente un caso. Solo los campos proporcionados ser
        Requiere permisos de OPERADOR o superior para la dependencia asign
      tags:
        - Casos
      parameters:
        - name: numero_caso
          in: path
          required: true
          schema:
            type: string
          example: "CRV-2024-00012345"
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/CasoUpdateRequest'
      responses:
        '200':
          description: Caso actualizado exitosamente
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/CasoResponse'
        '403':
          $ref: '#/components/responses/ForbiddenError'
        '404':
          $ref: '#/components/responses/NotFoundError'

  /casos/{numero_caso}/historial:
    get:
      summary: Obtener historial de caso
      description: |
        Obtiene el historial completo de cambios y acciones realizadas sob
      tags:
        - Casos
      parameters:
        - name: numero_caso
          in: path
          required: true
          schema:
            type: string
          example: "CRV-2024-00012345"
        - $ref: '#/components/parameters/PageNumber'
        - $ref: '#/components/parameters/PageSize'
      responses:
        '200':
          description: Historial obtenido exitosamente
```

```yaml
        description: Historial obtenido exitosamente
        content:
          application/json:
            schema:
              type: object
              properties:
                data:
                  type: array
                  items:
                    $ref: '#/components/schemas/HistorialItem'
                pagination:
                  $ref: '#/components/schemas/Pagination'

  /casos/{numero_caso}/comentarios:
    post:
      summary: Agregar comentario a caso
      description: |
        Agrega un comentario o actualización al caso. El comentario quedar
      tags:
        - Casos
      parameters:
        - name: numero_caso
          in: path
          required: true
          schema:
            type: string
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              required:
                - comentario
              properties:
                comentario:
                  type: string
                  minLength: 5
                  maxLength: 2000
                interno:
                  type: boolean
                  default: false
                  description: Si es true, el comentario solo es visible p
      responses:
        '201':
          description: Comentario agregado exitosamente
        '404':
          $ref: '#/components/responses/NotFoundError'

  /casos/{numero_caso}/cerrar:
    post:
      summary: Cerrar caso
      description: |
        Cierra un caso, cambiando su estado a CERRADO. Requiere estado act
      tags:
        - Casos
      parameters:
        - name: numero_caso
          in: path
          required: true
          schema:
            type: string
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              required:
                - motivo_cierre
              properties:
                motivo_cierre:
                  type: string
                  enum: [RESUELTO, CANCELADO, DUPLICADO, INVALIDO]
                comentario:
                  type: string
      responses:
        '200':
          description: Caso cerrado exitosamente
        '400':
```

```yaml
      400:
        $ref: '#/components/responses/BadRequestError'

/ciudadanos:
  get:
    summary: Buscar ciudadanos
    description: |
      Busca ciudadanos por documento de identidad, email o nombre. Requi
    tags:
      - Ciudadanos
    parameters:
      - name: documento_identidad
        in: query
        schema:
          type: string
      - name: email
        in: query
        schema:
          type: string
      - name: nombre
        in: query
        schema:
          type: string
    responses:
      '200':
        description: Lista de ciudadanos encontrados
        content:
          application/json:
            schema:
              type: object
              properties:
                data:
                  type: array
                  items:
                    $ref: '#/components/schemas/CiudadanoResponse'

  post:
    summary: Registrar ciudadano
    description: |
      Registra un nuevo ciudadano en el sistema. Si el ciudadano ya exis
    tags:
      - Ciudadanos
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/CiudadanoCreateRequest'
    responses:
      '201':
        description: Ciudadano registrado exitosamente
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/CiudadanoResponse'
      '409':
        description: Ciudadano ya existe

/ciudadanos/{id}/casos:
  get:
    summary: Obtener casos de un ciudadano
    description: |
      Obtiene todos los casos asociados a un ciudadano específico.
    tags:
      - Ciudadanos
      - Casos
    parameters:
      - name: id
        in: path
        required: true
        schema:
          type: integer
      - $ref: '#/components/parameters/PageNumber'
      - $ref: '#/components/parameters/PageSize'
    responses:
      '200':
        description: Lista de casos del ciudadano
        content:
          application/json:
            schema:
```

```yaml
            schema:
              $ref: '#/components/schemas/PaginatedCasos'

  /categorias:
    get:
      summary: Listar categorías
      description: |
        Obtiene la lista de categorías disponibles, opcionalmente filtrada
      tags:
        - Categorías
      parameters:
        - name: dependencia_id
          in: query
          schema:
            type: integer
      responses:
        '200':
          description: Lista de categorías
          content:
            application/json:
              schema:
                type: object
                properties:
                  data:
                    type: array
                    items:
                      $ref: '#/components/schemas/CategoriaResponse'

  /dependencias:
    get:
      summary: Listar dependencias
      description: |
        Obtiene la lista de todas las dependencias gubernamentales activas
      tags:
        - Dependencias
      responses:
        '200':
          description: Lista de dependencias
          content:
            application/json:
              schema:
                type: object
                properties:
                  data:
                    type: array
                    items:
                      $ref: '#/components/schemas/DependenciaResponse'

  /analytics/dashboard:
    get:
      summary: Dashboard central
      description: |
        Obtiene métricas y KPIs globales para el dashboard central. Requie
      tags:
        - Analítica
      parameters:
        - name: fecha_desde
          in: query
          schema:
            type: string
            format: date
        - name: fecha_hasta
          in: query
          schema:
            type: string
            format: date
        - name: dependencia_id
          in: query
          schema:
            type: integer
      responses:
        '200':
          description: Métricas del dashboard
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/DashboardResponse'

  /notificaciones:
    get:
```

```yaml
    get:
      summary: Obtener notificaciones del usuario
      description: |
        Obtiene las notificaciones del usuario autenticado.
      tags:
        - Notificaciones
      parameters:
        - name: leido
          in: query
          schema:
            type: boolean
        - $ref: '#/components/parameters/PageNumber'
        - $ref: '#/components/parameters/PageSize'
      responses:
        '200':
          description: Lista de notificaciones
          content:
            application/json:
              schema:
                type: object
                properties:
                  data:
                    type: array
                    items:
                      $ref: '#/components/schemas/NotificacionResponse'

components:
  securitySchemes:
    bearerAuth:
      type: http
      scheme: bearer
      bearerFormat: JWT
      description: Token JWT obtenido del endpoint de autenticación OAuth2

  parameters:
    PageNumber:
      name: page
      in: query
      schema:
        type: integer
        minimum: 1
        default: 1
      description: Número de página

    PageSize:
      name: size
      in: query
      schema:
        type: integer
        minimum: 1
        maximum: 100
        default: 20
      description: Tamaño de página

    EstadoFilter:
      name: estado
      in: query
      schema:
        type: string
        enum: [PENDIENTE, ASIGNADO, EN_PROCESO, RESUELTO, CERRADO, CANCELA
      description: Filtrar por estado

    DependenciaFilter:
      name: dependencia_id
      in: query
      schema:
        type: integer
      description: Filtrar por dependencia

    CategoriaFilter:
      name: categoria_id
      in: query
      schema:
        type: integer
      description: Filtrar por categoría

    FechaDesdeFilter:
      name: fecha_desde
      in: query
      schema:
```

```yaml
      schema:
        type: string
        format: date-time
      description: Filtrar casos desde fecha

    FechaHastaFilter:
      name: fecha_hasta
      in: query
      schema:
        type: string
        format: date-time
      description: Filtrar casos hasta fecha

    BusquedaFilter:
      name: q
      in: query
      schema:
        type: string
      description: Búsqueda full-text en título y descripción

  schemas:
    CasoCreateRequest:
      type: object
      required:
        - categoria_id
        - dependencia_id
        - canal_atencion_id
        - titulo
        - descripcion
      properties:
        categoria_id:
          type: integer
          example: 5
        dependencia_id:
          type: integer
          example: 1
        canal_atencion_id:
          type: integer
          example: 1
        titulo:
          type: string
          minLength: 5
          maxLength: 500
          example: "Bache en calle principal"
        descripcion:
          type: string
          minLength: 10
          maxLength: 5000
          example: "Hay un bache grande en la calle principal del barrio L
        prioridad:
          type: string
          enum: [BAJA, MEDIA, ALTA, URGENTE]
          default: MEDIA
        ubicacion:
          $ref: '#/components/schemas/Ubicacion'
        metadata:
          type: object
          additionalProperties: true
          example:
            foto_url: "https://storage.example.com/fotos/bache-123.jpg"
            sugerencia: "Recomendado reparar en horario nocturno"

    CasoUpdateRequest:
      type: object
      properties:
        estado:
          type: string
          enum: [PENDIENTE, ASIGNADO, EN_PROCESO, RESUELTO, CERRADO, CANCE
        prioridad:
          type: string
          enum: [BAJA, MEDIA, ALTA, URGENTE]
        usuario_asignado_id:
          type: integer
        comentario:
          type: string

    CasoResponse:
      type: object
      properties:
        id:
```

```yaml
      id:
        type: integer
        example: 12345
      numero_caso:
        type: string
        example: "CRV-2024-00012345"
      titulo:
        type: string
      estado:
        type: string
        enum: [PENDIENTE, ASIGNADO, EN_PROCESO, RESUELTO, CERRADO, CANCE
      prioridad:
        type: string
        enum: [BAJA, MEDIA, ALTA, URGENTE]
      fecha_creacion:
        type: string
        format: date-time
      fecha_asignacion:
        type: string
        format: date-time
      fecha_resolucion:
        type: string
        format: date-time
      dependencia:
        $ref: '#/components/schemas/DependenciaMinimal'
      categoria:
        $ref: '#/components/schemas/CategoriaMinimal'
      ciudadano:
        $ref: '#/components/schemas/CiudadanoMinimal'

  CasoDetalleResponse:
    allOf:
      - $ref: '#/components/schemas/CasoResponse'
      - type: object
        properties:
          descripcion:
            type: string
          ubicacion:
            $ref: '#/components/schemas/Ubicacion'
          metadata:
            type: object
            additionalProperties: true
          usuario_asignado:
            type: object
            properties:
              id:
                type: integer
              nombres:
                type: string
              apellidos:
                type: string
          historial_count:
            type: integer
          notificaciones_count:
            type: integer
          sla:
            $ref: '#/components/schemas/SLAResponse'

  Ubicacion:
    type: object
    properties:
      lat:
        type: number
        format: float
        example: -12.046374
      lng:
        type: number
        format: float
        example: -77.042793
      direccion:
        type: string
        example: "Calle Principal, Barrio Los Rosales"
      barrio:
        type: string
        example: "Los Rosales"
      ciudad:
        type: string
        example: "San José"

  PaginatedCases:
```

```yaml
PaginatedCasos:
  type: object
  properties:
    data:
      type: array
      items:
        $ref: '#/components/schemas/CasoResponse'
    pagination:
      $ref: '#/components/schemas/Pagination'

Pagination:
  type: object
  properties:
    page:
      type: integer
    size:
      type: integer
    total:
      type: integer
    total_pages:
      type: integer
    has_next:
      type: boolean
    has_previous:
      type: boolean

HistorialItem:
  type: object
  properties:
    id:
      type: integer
    fecha_cambio:
      type: string
      format: date-time
    estado_anterior:
      type: string
    estado_nuevo:
      type: string
    tipo_cambio:
      type: string
      enum: [CAMBIO_ESTADO, ASIGNACION, REASIGNACION, COMENTARIO]
    comentario:
      type: string
    usuario:
      type: object
      properties:
        id:
          type: integer
        nombres:
          type: string
        apellidos:
          type: string

CiudadanoCreateRequest:
  type: object
  required:
    - documento_identidad
    - nombres
    - apellidos
    - email
  properties:
    documento_identidad:
      type: string
      pattern: '^\d{9,11}$'
      example: "123456789"
    nombres:
      type: string
      minLength: 2
      maxLength: 200
    apellidos:
      type: string
      minLength: 2
      maxLength: 200
    email:
      type: string
      format: email
    telefono:
      type: string
    direccion:
      type: string
```

```yaml
      type: string

    CiudadanoResponse:
      type: object
      properties:
        id:
          type: integer
        documento_identidad:
          type: string
        nombres:
          type: string
        apellidos:
          type: string
        email:
          type: string
        telefono:
          type: string
        fecha_registro:
          type: string
          format: date-time

    CiudadanoMinimal:
      type: object
      properties:
        id:
          type: integer
        nombres:
          type: string
        apellidos:
          type: string
        documento_identidad:
          type: string

    CategoriaResponse:
      type: object
      properties:
        id:
          type: integer
        codigo:
          type: string
        nombre:
          type: string
        descripcion:
          type: string
        tipo:
          type: string
        dependencia:
          $ref: '#/components/schemas/DependenciaMinimal'

    CategoriaMinimal:
      type: object
      properties:
        id:
          type: integer
        codigo:
          type: string
        nombre:
          type: string

    DependenciaResponse:
      type: object
      properties:
        id:
          type: integer
        codigo:
          type: string
          example: "MIN-SALUD"
        nombre:
          type: string
        descripcion:
          type: string
        tipo:
          type: string
          enum: [MINISTERIO, MUNICIPALIDAD, ORGANISMO_AUTONOMO]

    DependenciaMinimal:
      type: object
      properties:
        id:
          type: integer
```

```yaml
      codigo:
        type: string
      nombre:
        type: string

  SLAResponse:
    type: object
    properties:
      id:
        type: integer
      estado:
        type: string
        enum: [VIGENTE, CUMPLIDO, VIOLADO]
      fecha_inicio:
        type: string
        format: date-time
      fecha_vencimiento:
        type: string
        format: date-time
      fecha_resolucion_objetivo:
        type: string
        format: date-time
      tiempo_resolucion_horas:
        type: integer
      violado:
        type: boolean
      tiempo_restante_horas:
        type: integer

  NotificacionResponse:
    type: object
    properties:
      id:
        type: integer
      tipo:
        type: string
        enum: [CREACION, ASIGNACION, ACTUALIZACION, CIERRE]
      canal:
        type: string
        enum: [EMAIL, SMS, PUSH, WHATSAPP]
      asunto:
        type: string
      contenido:
        type: string
      estado:
        type: string
        enum: [PENDIENTE, ENVIADO, FALLIDO, LEIDO]
      fecha_envio:
        type: string
        format: date-time
      fecha_leido:
        type: string
        format: date-time

  DashboardResponse:
    type: object
    properties:
      kpis:
        type: object
        properties:
          casos_totales:
            type: integer
          casos_pendientes:
            type: integer
          casos_resueltos:
            type: integer
          tiempo_promedio_resolucion_horas:
            type: number
          cumplimiento_sla_porcentaje:
            type: number
          satisfaccion_promedio:
            type: number
      casos_por_estado:
        type: object
        additionalProperties:
          type: integer
      casos_por_dependencia:
        type: array
        items:
```

```yaml
        items:
          type: object
          properties:
            dependencia:
              type: string
            total:
              type: integer
      tendencias:
        type: array
        items:
          type: object
          properties:
            fecha:
              type: string
              format: date
            casos_creados:
              type: integer
            casos_resueltos:
              type: integer

  ErrorResponse:
    type: object
    properties:
      error:
        type: object
        properties:
          codigo:
            type: string
          mensaje:
            type: string
          detalles:
            type: array
            items:
              type: object
              properties:
                campo:
                  type: string
                mensaje:
                  type: string
          timestamp:
            type: string
            format: date-time
          ruta:
            type: string

responses:
  BadRequestError:
    description: Solicitud inválida
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/ErrorResponse'
        example:
          error:
            codigo: "BAD_REQUEST"
            mensaje: "Los datos proporcionados no son válidos"
            detalles:
              - campo: "titulo"
                mensaje: "El título debe tener al menos 5 caracteres"

  UnauthorizedError:
    description: No autenticado o token inválido
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/ErrorResponse'
        example:
          error:
            codigo: "UNAUTHORIZED"
            mensaje: "Token de autenticación inválido o expirado"

  ForbiddenError:
    description: No tiene permisos para realizar esta acción
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/ErrorResponse'
        example:
          error:
            codigo: "FORBIDDEN"
```

```yaml
              codigo: "FORBIDDEN"
              mensaje: "No tiene permisos para actualizar este caso"

    NotFoundError:
      description: Recurso no encontrado
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/ErrorResponse'
          example:
            error:
              codigo: "NOT_FOUND"
              mensaje: "Caso no encontrado"

    ValidationError:
      description: Error de validación
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/ErrorResponse'
          example:
            error:
              codigo: "VALIDATION_ERROR"
              mensaje: "Los datos proporcionados no pasaron la validación"
              detalles:
                - campo: "email"
                  mensaje: "El formato del email no es válido"

    RateLimitError:
      description: Límite de solicitudes excedido
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/ErrorResponse'
          example:
            error:
              codigo: "RATE_LIMIT_EXCEEDED"
              mensaje: "Ha excedido el límite de solicitudes. Intente nuev
              timestamp: "2024-01-15T10:30:00Z"
```

Ejemplos de Uso

Crear un caso (POST /casos)

Request:

```bash
curl -X POST "https://api.conecta360.gov.cv/v1/casos" \
  -H "Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9..." \
  -H "Content-Type: application/json" \
  -d '{
    "categoria_id": 5,
    "dependencia_id": 1,
    "canal_atencion_id": 1,
    "titulo": "Bache en calle principal",
    "descripcion": "Hay un bache grande en la calle principal del barrio L
    "prioridad": "ALTA",
    "ubicacion": {
      "lat": -12.046374,
      "lng": -77.042793,
      "direccion": "Calle Principal, Barrio Los Rosales",
      "barrio": "Los Rosales"
    },
    "metadata": {
      "foto_url": "https://storage.example.com/fotos/bache-123.jpg",
      "sugerencia": "Recomendado reparar en horario nocturno"
    }
  }'
```

Response:

```
{
  "id": 12345,
  "numero_caso": "CRV-2024-00012345",
  "titulo": "Bache en calle principal",
  "estado": "PENDIENTE",
  "prioridad": "ALTA",
  "fecha_creacion": "2024-01-15T10:30:00Z",
  "dependencia": {
    "id": 1,
    "codigo": "MIN-OBRAS",
    "nombre": "Ministerio de Obras Públicas"
  },
  "categoria": {
    "id": 5,
    "codigo": "INFRA-VIAL",
    "nombre": "Infraestructura Vial"
  },
  "mensaje": "Su caso ha sido registrado. Número de caso: CRV-2024-0001234
}
```


Obtener caso por número (GET /casos/{numero_caso})

Request:

```
curl -X GET "https://api.conecta360.gov.cv/v1/casos/CRV-2024-00012345" \
  -H "Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9..."
```

Response:


```
{
  "id": 12345,
  "numero_caso": "CRV-2024-00012345",
  "titulo": "Bache en calle principal",
  "descripcion": "Hay un bache grande en la calle principal...",
  "estado": "ASIGNADO",
  "prioridad": "ALTA",
  "fecha_creacion": "2024-01-15T10:30:00Z",
  "fecha_asignacion": "2024-01-15T10:35:00Z",
  "dependencia": {
    "id": 1,
    "nombre": "Ministerio de Obras Públicas"
  },
  "categoria": {
    "id": 5,
    "nombre": "Infraestructura Vial"
  },
  "ciudadano": {
    "id": 5432,
    "nombres": "Juan",
    "apellidos": "Pérez",
    "documento_identidad": "123456789"
  },
  "usuario_asignado": {
    "id": 987,
    "nombres": "María",
    "apellidos": "González"
  },
  "ubicacion": {
    "lat": -12.046374,
    "lng": -77.042793,
    "direccion": "Calle Principal, Barrio Los Rosales",
    "barrio": "Los Rosales"
  },
  "sla": {
    "estado": "VIGENTE",
    "fecha_vencimiento": "2024-01-22T10:30:00Z",
    "tiempo_resolucion_horas": 168,
    "tiempo_restante_horas": 142,
    "violado": false
  },
  "historial_count": 3,
  "notificaciones_count": 2
}
```


Listar casos con filtros (GET /casos)

Request:

```
curl -X GET "https://api.conecta360.gov.cv/v1/casos?estado=ASIGNADO&depend
  -H "Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9..."
```

Response:

```
{
  "data": [
    {
      "numero_caso": "CRV-2024-00012345",
      "titulo": "Bache en calle principal",
      "estado": "ASIGNADO",
      "prioridad": "ALTA",
      "fecha_creacion": "2024-01-15T10:30:00Z",
      "dependencia": {
        "id": 1,
        "nombre": "Ministerio de Obras Públicas"
      },
      "categoria": {
        "id": 5,
        "nombre": "Infraestructura Vial"
      }
    }
  ],
  "pagination": {
    "page": 1,
    "size": 20,
    "total": 156,
    "total_pages": 8,
    "has_next": true,
    "has_previous": false
  }
}
```

Principios de Diseño REST

1. Recursos como Sustantivos

- `/casos` (no `/crearCaso`)
- `/ciudadanos` (no `/obtenerCiudadanos`)

2. Métodos HTTP Semánticos

- `GET`: Lectura (idempotente)
- `POST`: Creación
- `PATCH`: Actualización parcial
- `PUT`: Actualización completa (si aplica)
- `DELETE`: Eliminación (soft delete)

3. Códigos de Estado HTTP

- `200`: Éxito
- `201`: Creado
- `400`: Solicitud inválida
- `401`: No autenticado
- `403`: Sin permisos
- `404`: No encontrado
- `422`: Error de validación
- `429`: Rate limit excedido
- `500`: Error interno del servidor

4. Versionado

- **URL-based**: `/v1/casos`, `/v2/casos`
- **Header-based**: `Accept: application/vnd.conecta360.v1+json` (alternativa)

5. Paginación

- Query parameters: `?page=1&size=20`
- Headers de respuesta con información de paginación

6. Filtrado y Búsqueda

- Query parameters: `?estado=ASIGNADO&dependencia_id=1`
- Búsqueda full-text: `?q=bache`

7. Formato de Respuesta

- JSON consistente con estructura `{data: [], pagination: {}}`
- Errores en formato estándar: `{error: {codigo, mensaje, detalles}}`

Siguiente: Ver APIs Asíncronas

5. Diseño de APIs Asíncronas - Conecta360

Visión General

Este documento define la arquitectura de mensajería asíncrona de Conecta360 usando Apache Kafka, siguiendo el estándar AsyncAPI para describir eventos y canales de comunicación.

Especificación AsyncAPI

AsyncAPI 2.6.0 Specification

```yaml
asyncapi: 2.6.0
info:
  title: Conecta360 Event Streaming API
  version: 1.0.0
  description: |
    Sistema Integral de Atención Ciudadana - Conecta360

    API de Eventos Asíncronos usando Apache Kafka para comunicación entre

    **Protocolo:** Apache Kafka
    **Formato de mensaje:** JSON
    **Versionado:** Por topic (ej: casos.creados.v1, casos.creados.v2)
  contact:
    name: API Support
    email: api-support@conecta360.gov.cv
  license:
    name: Propietario - Gobierno de Costa Verde

servers:
  kafka-prod:
    url: kafka.conecta360.gov.cv:9092
    protocol: kafka
    description: Servidor Kafka de producción
  kafka-staging:
    url: kafka-staging.conecta360.gov.cv:9092
    protocol: kafka
    description: Servidor Kafka de staging

defaultContentType: application/json

channels:
  casos.creados:
    description: |
      Evento publicado cuando se crea un nuevo caso ciudadano.
      Consumidores: Servicio de Derivación, Servicio de Notificaciones, Se
    publish:
      operationId: publicarCasoCreado
      summary: Publicar evento de caso creado
      message:
        $ref: '#/components/messages/CasoCreado'
    subscribe:
      operationId: consumirCasoCreado
      summary: Consumir evento de caso creado
      message:
        $ref: '#/components/messages/CasoCreado'

  casos.asignados:
```

```yaml
  description: |
    Evento publicado cuando un caso es asignado a una dependencia o usua
    Consumidores: Servicio de Notificaciones, Servicio de SLA, Servicio
  publish:
    operationId: publicarCasoAsignado
    summary: Publicar evento de caso asignado
    message:
      $ref: '#/components/messages/CasoAsignado'
  subscribe:
    operationId: consumirCasoAsignado
    summary: Consumir evento de caso asignado
    message:
      $ref: '#/components/messages/CasoAsignado'

casos.actualizados:
  description: |
    Evento publicado cuando se actualiza un caso (cambio de estado, prio
    Consumidores: Servicio de Notificaciones, Servicio de Analítica
  publish:
    operationId: publicarCasoActualizado
    summary: Publicar evento de caso actualizado
    message:
      $ref: '#/components/messages/CasoActualizado'
  subscribe:
    operationId: consumirCasoActualizado
    summary: Consumir evento de caso actualizado
    message:
      $ref: '#/components/messages/CasoActualizado'

casos.cerrados:
  description: |
    Evento publicado cuando un caso es cerrado (resuelto, cancelado, dup
    Consumidores: Servicio de Notificaciones, Servicio de SLA, Servicio
  publish:
    operationId: publicarCasoCerrado
    summary: Publicar evento de caso cerrado
    message:
      $ref: '#/components/messages/CasoCerrado'
  subscribe:
    operationId: consumirCasoCerrado
    summary: Consumir evento de caso cerrado
    message:
      $ref: '#/components/messages/CasoCerrado'

notificaciones.enviadas:
  description: |
    Evento publicado cuando se envía una notificación (email, SMS, push)
    Consumidores: Servicio de Analítica (tracking de notificaciones)
  publish:
    operationId: publicarNotificacionEnviada
    summary: Publicar evento de notificación enviada
    message:
      $ref: '#/components/messages/NotificacionEnviada'
  subscribe:
    operationId: consumirNotificacionEnviada
    summary: Consumir evento de notificación enviada
    message:
      $ref: '#/components/messages/NotificacionEnviada'

sla.violados:
  description: |
    Evento publicado cuando se viola un SLA (tiempo de resolución excedi
    Consumidores: Servicio de Alertas, Servicio de Analítica
  publish:
    operationId: publicarSlaViolado
    summary: Publicar evento de SLA violado
    message:
      $ref: '#/components/messages/SlaViolado'
  subscribe:
    operationId: consumirSlaViolado
    summary: Consumir evento de SLA violado
    message:
      $ref: '#/components/messages/SlaViolado'

sla.cumplidos:
  description: |
    Evento publicado cuando un SLA es cumplido exitosamente.
    Consumidores: Servicio de Analítica
  publish:
```

```yaml
    publish:
      operationId: publicarSlaCumplido
      summary: Publicar evento de SLA cumplido
      message:
        $ref: '#/components/messages/SlaCumplido'
    subscribe:
      operationId: consumirSlaCumplido
      summary: Consumir evento de SLA cumplido
      message:
        $ref: '#/components/messages/SlaCumplido'

  ciudadanos.registrados:
    description: |
      Evento publicado cuando se registra un nuevo ciudadano.
      Consumidores: Servicio de Analítica
    publish:
      operationId: publicarCiudadanoRegistrado
      summary: Publicar evento de ciudadano registrado
      message:
        $ref: '#/components/messages/CiudadanoRegistrado'
    subscribe:
      operationId: consumirCiudadanoRegistrado
      summary: Consumir evento de ciudadano registrado
      message:
        $ref: '#/components/messages/CiudadanoRegistrado'

  dependencias.configuradas:
    description: |
      Evento publicado cuando se actualiza la configuración de una depende
      Consumidores: Servicio de Derivación, Servicio de SLA
    publish:
      operationId: publicarDependenciaConfigurada
      summary: Publicar evento de dependencia configurada
      message:
        $ref: '#/components/messages/DependenciaConfigurada'
    subscribe:
      operationId: consumirDependenciaConfigurada
      summary: Consumir evento de dependencia configurada
      message:
        $ref: '#/components/messages/DependenciaConfigurada'

components:
  messages:
    CasoCreado:
      name: CasoCreado
      title: Caso Creado
      summary: Evento que indica que se ha creado un nuevo caso
      contentType: application/json
      traits:
        - $ref: '#/components/messageTraits/CommonHeaders'
      payload:
        $ref: '#/components/schemas/CasoCreadoPayload'
      examples:
        - payload:
            evento_id: "evt_123456789"
            evento_tipo: "casos.creados"
            evento_version: "1.0"
            timestamp: "2024-01-15T10:30:00Z"
            origen: "microservicio-casos"
            datos:
              caso_id: 12345
              numero_caso: "CRV-2024-00012345"
              ciudadano_id: 5432
              categoria_id: 5
              dependencia_id: 1
              canal_atencion_id: 1
              titulo: "Bache en calle principal"
              estado: "PENDIENTE"
              prioridad: "ALTA"
              fecha_creacion: "2024-01-15T10:30:00Z"
              ubicacion:
                lat: -12.046374
                lng: -77.042793
                direccion: "Calle Principal, Barrio Los Rosales"
            metadata:
              request_id: "req_abc123"
              usuario_id: 5432
              ip_address: "192.168.1.100"

    CasoAsignado:
```

```yaml
CasoAsignado:
  name: CasoAsignado
  title: Caso Asignado
  summary: Evento que indica que un caso ha sido asignado
  contentType: application/json
  traits:
    - $ref: '#/components/messageTraits/CommonHeaders'
  payload:
    $ref: '#/components/schemas/CasoAsignadoPayload'
  examples:
    - payload:
        evento_id: "evt_123456790"
        evento_tipo: "casos.asignados"
        evento_version: "1.0"
        timestamp: "2024-01-15T10:35:00Z"
        origen: "microservicio-derivacion"
        datos:
          caso_id: 12345
          numero_caso: "CRV-2024-00012345"
          dependencia_id: 1
          dependencia_nombre: "Ministerio de Obras Públicas"
          usuario_asignado_id: 987
          usuario_asignado_nombre: "María González"
          fecha_asignacion: "2024-01-15T10:35:00Z"
          motivo_asignacion: "Derivación automática por categoría"
          estado_anterior: "PENDIENTE"
          estado_nuevo: "ASIGNADO"
        metadata:
          request_id: "req_abc124"
          usuario_id: 987

CasoActualizado:
  name: CasoActualizado
  title: Caso Actualizado
  summary: Evento que indica que un caso ha sido actualizado
  contentType: application/json
  traits:
    - $ref: '#/components/messageTraits/CommonHeaders'
  payload:
    $ref: '#/components/schemas/CasoActualizadoPayload'
  examples:
    - payload:
        evento_id: "evt_123456791"
        evento_tipo: "casos.actualizados"
        evento_version: "1.0"
        timestamp: "2024-01-15T11:00:00Z"
        origen: "microservicio-casos"
        datos:
          caso_id: 12345
          numero_caso: "CRV-2024-00012345"
          cambios:
            - campo: "estado"
              valor_anterior: "ASIGNADO"
              valor_nuevo: "EN_PROCESO"
            - campo: "prioridad"
              valor_anterior: "ALTA"
              valor_nuevo: "URGENTE"
          fecha_actualizacion: "2024-01-15T11:00:00Z"
          comentario: "Caso escalado por riesgo de seguridad vial"
        metadata:
          request_id: "req_abc125"
          usuario_id: 987

CasoCerrado:
  name: CasoCerrado
  title: Caso Cerrado
  summary: Evento que indica que un caso ha sido cerrado
  contentType: application/json
  traits:
    - $ref: '#/components/messageTraits/CommonHeaders'
  payload:
    $ref: '#/components/schemas/CasoCerradoPayload'
  examples:
    - payload:
        evento_id: "evt_123456792"
        evento_tipo: "casos.cerrados"
        evento_version: "1.0"
        timestamp: "2024-01-18T14:30:00Z"
        origen: "microservicio-casos"
```

```
          datos:
            caso_id: 12345
            numero_caso: "CRV-2024-00012345"
            motivo_cierre: "RESUELTO"
            estado_final: "CERRADO"
            fecha_resolucion: "2024-01-18T14:00:00Z"
            fecha_cierre: "2024-01-18T14:30:00Z"
            tiempo_total_horas: 88
            comentario_cierre: "Bache reparado exitosamente. Ciudadano n
            satisfaccion_ciudadano: 5
          metadata:
            request_id: "req_abc126"
            usuario_id: 987

  NotificacionEnviada:
    name: NotificacionEnviada
    title: Notificación Enviada
    summary: Evento que indica que se ha enviado una notificación
    contentType: application/json
    traits:
      - $ref: '#/components/messageTraits/CommonHeaders'
    payload:
      $ref: '#/components/schemas/NotificacionEnviadaPayload'
    examples:
      - payload:
          evento_id: "evt_123456793"
          evento_tipo: "notificaciones.enviadas"
          evento_version: "1.0"
          timestamp: "2024-01-15T10:35:30Z"
          origen: "microservicio-notificaciones"
          datos:
            notificacion_id: 7890
            caso_id: 12345
            numero_caso: "CRV-2024-00012345"
            tipo: "ASIGNACION"
            canal: "SMS"
            destinatario: "+50612345678"
            estado: "ENVIADO"
            fecha_envio: "2024-01-15T10:35:30Z"
            proveedor: "Twilio"
            costo_estimado: 0.05
          metadata:
            request_id: "req_abc127"

  SlaViolado:
    name: SlaViolado
    title: SLA Violado
    summary: Evento que indica que un SLA ha sido violado
    contentType: application/json
    traits:
      - $ref: '#/components/messageTraits/CommonHeaders'
    payload:
      $ref: '#/components/schemas/SlaVioladoPayload'
    examples:
      - payload:
          evento_id: "evt_123456794"
          evento_tipo: "sla.violados"
          evento_version: "1.0"
          timestamp: "2024-01-22T10:31:00Z"
          origen: "microservicio-sla"
          datos:
            sla_id: 567
            caso_id: 12345
            numero_caso: "CRV-2024-00012345"
            configuracion_sla_id: 10
            tiempo_resolucion_horas: 168
            tiempo_transcurrido_horas: 168.02
            fecha_vencimiento: "2024-01-22T10:30:00Z"
            fecha_violacion: "2024-01-22T10:31:00Z"
            dependencia_id: 1
            prioridad: "ALTA"
            accion_requerida: "Escalar a supervisor"
          metadata:
            request_id: "req_abc128"

  SlaCumplido:
    name: SlaCumplido
    title: SLA Cumplido
    summary: Evento que indica que un SLA ha sido cumplido
```

```yaml
    contentType: application/json
    traits:
      - $ref: '#/components/messageTraits/CommonHeaders'
    payload:
      $ref: '#/components/schemas/SlaCumplidoPayload'
    examples:
      - payload:
          evento_id: "evt_123456795"
          evento_tipo: "sla.cumplidos"
          evento_version: "1.0"
          timestamp: "2024-01-18T14:00:00Z"
          origen: "microservicio-sla"
          datos:
            sla_id: 567
            caso_id: 12345
            numero_caso: "CRV-2024-00012345"
            configuracion_sla_id: 10
            tiempo_resolucion_horas: 168
            tiempo_real_horas: 88
            fecha_resolucion: "2024-01-18T14:00:00Z"
            cumplimiento_porcentaje: 152.27
          metadata:
            request_id: "req_abc129"

  CiudadanoRegistrado:
    name: CiudadanoRegistrado
    title: Ciudadano Registrado
    summary: Evento que indica que se ha registrado un nuevo ciudadano
    contentType: application/json
    traits:
      - $ref: '#/components/messageTraits/CommonHeaders'
    payload:
      $ref: '#/components/schemas/CiudadanoRegistradoPayload'
    examples:
      - payload:
          evento_id: "evt_123456796"
          evento_tipo: "ciudadanos.registrados"
          evento_version: "1.0"
          timestamp: "2024-01-15T10:28:00Z"
          origen: "microservicio-ciudadanos"
          datos:
            ciudadano_id: 5432
            documento_identidad: "123456789"
            email: "juan.perez@example.com"
            fecha_registro: "2024-01-15T10:28:00Z"
            canal_registro: "WEB"
          metadata:
            request_id: "req_abc122"

  DependenciaConfigurada:
    name: DependenciaConfigurada
    title: Dependencia Configurada
    summary: Evento que indica que se ha actualizado la configuración de
    contentType: application/json
    traits:
      - $ref: '#/components/messageTraits/CommonHeaders'
    payload:
      $ref: '#/components/schemas/DependenciaConfiguradaPayload'
    examples:
      - payload:
          evento_id: "evt_123456797"
          evento_tipo: "dependencias.configuradas"
          evento_version: "1.0"
          timestamp: "2024-01-15T09:00:00Z"
          origen: "microservicio-dependencias"
          datos:
            dependencia_id: 1
            codigo: "MIN-OBRAS"
            cambios:
              - campo: "configuracion.sla_tiempo_resolucion_horas"
                valor_anterior: 168
                valor_nuevo: 120
            fecha_actualizacion: "2024-01-15T09:00:00Z"
          metadata:
            request_id: "req_abc121"
            usuario_id: 1001

messageTraits:
  CommonHeaders:
```

```yaml
      headers:
        type: object
        properties:
          evento_id:
            type: string
            description: ID único del evento
            example: "evt_123456789"
          evento_tipo:
            type: string
            description: Tipo del evento
            example: "casos.creados"
          evento_version:
            type: string
            description: Versión del esquema del evento
            example: "1.0"
          timestamp:
            type: string
            format: date-time
            description: Timestamp del evento (ISO 8601)
            example: "2024-01-15T10:30:00Z"
          origen:
            type: string
            description: Microservicio que generó el evento
            example: "microservicio-casos"
          correlation_id:
            type: string
            description: ID de correlación para rastrear requests
            example: "req_abc123"

  schemas:
    EventoBase:
      type: object
      required:
        - evento_id
        - evento_tipo
        - evento_version
        - timestamp
        - origen
        - datos
      properties:
        evento_id:
          type: string
          description: ID único del evento (UUID)
          example: "evt_123456789"
        evento_tipo:
          type: string
          description: Tipo del evento
          example: "casos.creados"
        evento_version:
          type: string
          description: Versión del esquema del evento
          pattern: '^\d+\.\d+$'
          example: "1.0"
        timestamp:
          type: string
          format: date-time
          description: Timestamp del evento (ISO 8601)
          example: "2024-01-15T10:30:00Z"
        origen:
          type: string
          description: Microservicio que generó el evento
          example: "microservicio-casos"
        datos:
          type: object
          description: Datos específicos del evento
        metadata:
          type: object
          description: Metadatos adicionales (opcional)
          properties:
            request_id:
              type: string
            usuario_id:
              type: integer
            ip_address:
              type: string
            user_agent:
              type: string

    CasoCreadoPayload:
```

```yaml
    allOf:
      - $ref: '#/components/schemas/EventoBase'
      - type: object
        properties:
          datos:
            type: object
            required:
              - caso_id
              - numero_caso
              - ciudadano_id
              - categoria_id
              - dependencia_id
              - estado
              - fecha_creacion
            properties:
              caso_id:
                type: integer
              numero_caso:
                type: string
                pattern: '^CRV-\d{4}-\d{8}$'
              ciudadano_id:
                type: integer
              categoria_id:
                type: integer
              dependencia_id:
                type: integer
              canal_atencion_id:
                type: integer
              titulo:
                type: string
              estado:
                type: string
                enum: [PENDIENTE]
              prioridad:
                type: string
                enum: [BAJA, MEDIA, ALTA, URGENTE]
              fecha_creacion:
                type: string
                format: date-time
              ubicacion:
                type: object
                properties:
                  lat:
                    type: number
                    format: float
                  lng:
                    type: number
                    format: float
                  direccion:
                    type: string
                  barrio:
                    type: string

  CasoAsignadoPayload:
    allOf:
      - $ref: '#/components/schemas/EventoBase'
      - type: object
        properties:
          datos:
            type: object
            required:
              - caso_id
              - numero_caso
              - dependencia_id
              - fecha_asignacion
              - estado_nuevo
            properties:
              caso_id:
                type: integer
              numero_caso:
                type: string
              dependencia_id:
                type: integer
              dependencia_nombre:
                type: string
              usuario_asignado_id:
                type: integer
              usuario_asignado_nombre:
                type: string
```

```yaml
              type: string
            fecha_asignacion:
              type: string
              format: date-time
            motivo_asignacion:
              type: string
            estado_anterior:
              type: string
            estado_nuevo:
              type: string
              enum: [ASIGNADO]

    CasoActualizadoPayload:
      allOf:
        - $ref: '#/components/schemas/EventoBase'
        - type: object
          properties:
            datos:
              type: object
              required:
                - caso_id
                - numero_caso
                - fecha_actualizacion
              properties:
                caso_id:
                  type: integer
                numero_caso:
                  type: string
                cambios:
                  type: array
                  items:
                    type: object
                    properties:
                      campo:
                        type: string
                      valor_anterior:
                        type: string
                      valor_nuevo:
                        type: string
                fecha_actualizacion:
                  type: string
                  format: date-time
                comentario:
                  type: string

    CasoCerradoPayload:
      allOf:
        - $ref: '#/components/schemas/EventoBase'
        - type: object
          properties:
            datos:
              type: object
              required:
                - caso_id
                - numero_caso
                - motivo_cierre
                - estado_final
                - fecha_cierre
              properties:
                caso_id:
                  type: integer
                numero_caso:
                  type: string
                motivo_cierre:
                  type: string
                  enum: [RESUELTO, CANCELADO, DUPLICADO, INVALIDO]
                estado_final:
                  type: string
                  enum: [CERRADO]
                fecha_resolucion:
                  type: string
                  format: date-time
                fecha_cierre:
                  type: string
                  format: date-time
                tiempo_total_horas:
                  type: number
                  format: float
                comentario_cierre:
                  type: string
```

```yaml
              type: string
            satisfaccion_ciudadano:
              type: integer
              minimum: 1
              maximum: 5

    NotificacionEnviadaPayload:
      allOf:
        - $ref: '#/components/schemas/EventoBase'
        - type: object
          properties:
            datos:
              type: object
              required:
                - notificacion_id
                - caso_id
                - tipo
                - canal
                - estado
                - fecha_envio
              properties:
                notificacion_id:
                  type: integer
                caso_id:
                  type: integer
                numero_caso:
                  type: string
                tipo:
                  type: string
                  enum: [CREACION, ASIGNACION, ACTUALIZACION, CIERRE]
                canal:
                  type: string
                  enum: [EMAIL, SMS, PUSH, WHATSAPP]
                destinatario:
                  type: string
                estado:
                  type: string
                  enum: [ENVIADO, FALLIDO]
                fecha_envio:
                  type: string
                  format: date-time
                proveedor:
                  type: string
                  example: "Twilio"
                costo_estimado:
                  type: number
                  format: float

    SlaVioladoPayload:
      allOf:
        - $ref: '#/components/schemas/EventoBase'
        - type: object
          properties:
            datos:
              type: object
              required:
                - sla_id
                - caso_id
                - numero_caso
                - tiempo_resolucion_horas
                - tiempo_transcurrido_horas
                - fecha_violacion
              properties:
                sla_id:
                  type: integer
                caso_id:
                  type: integer
                numero_caso:
                  type: string
                configuracion_sla_id:
                  type: integer
                tiempo_resolucion_horas:
                  type: integer
                tiempo_transcurrido_horas:
                  type: number
                  format: float
                fecha_vencimiento:
                  type: string
                  format: date-time
```

```yaml
            fecha_violacion:
              type: string
              format: date-time
            dependencia_id:
              type: integer
            prioridad:
              type: string
            accion_requerida:
              type: string

    SlaCumplidoPayload:
      allOf:
        - $ref: '#/components/schemas/EventoBase'
        - type: object
          properties:
            datos:
              type: object
              required:
                - sla_id
                - caso_id
                - numero_caso
                - tiempo_resolucion_horas
                - tiempo_real_horas
                - fecha_resolucion
              properties:
                sla_id:
                  type: integer
                caso_id:
                  type: integer
                numero_caso:
                  type: string
                configuracion_sla_id:
                  type: integer
                tiempo_resolucion_horas:
                  type: integer
                tiempo_real_horas:
                  type: number
                  format: float
                fecha_resolucion:
                  type: string
                  format: date-time
                cumplimiento_porcentaje:
                  type: number
                  format: float

    CiudadanoRegistradoPayload:
      allOf:
        - $ref: '#/components/schemas/EventoBase'
        - type: object
          properties:
            datos:
              type: object
              required:
                - ciudadano_id
                - documento_identidad
                - fecha_registro
              properties:
                ciudadano_id:
                  type: integer
                documento_identidad:
                  type: string
                email:
                  type: string
                  format: email
                fecha_registro:
                  type: string
                  format: date-time
                canal_registro:
                  type: string
                  enum: [WEB, MOVIL, CALL_CENTER, VENTANILLA]

    DependenciaConfiguradaPayload:
      allOf:
        - $ref: '#/components/schemas/EventoBase'
        - type: object
          properties:
            datos:
              type: object
              required:
```

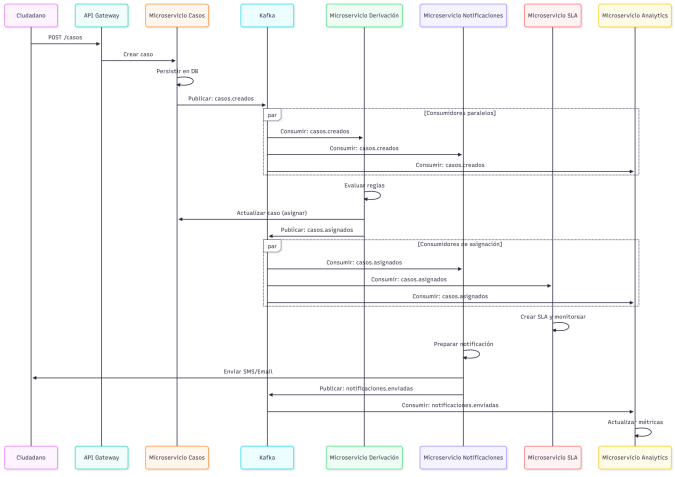
```
    - dependencia_id
    - codigo
    - fecha_actualizacion
  properties:
    dependencia_id:
      type: integer
    codigo:
      type: string
    cambios:
      type: array
      items:
        type: object
        properties:
          campo:
            type: string
          valor_anterior:
            type: string
          valor_nuevo:
            type: string
    fecha_actualizacion:
      type: string
      format: date-time
```

Configuración de Kafka Topics

Estrategia de Particionamiento

```yaml
Topics:
  casos.creados:
    partitions: 6
    replication_factor: 3
    retention_ms: 604800000  # 7 días
    partition_key: caso_id  # Para garantizar orden por caso

  casos.asignados:
    partitions: 6
    replication_factor: 3
    retention_ms: 604800000
    partition_key: caso_id

  casos.actualizados:
    partitions: 12
    replication_factor: 3
    retention_ms: 604800000
    partition_key: caso_id

  casos.cerrados:
    partitions: 6
    replication_factor: 3
    retention_ms: 2592000000  # 30 días
    partition_key: caso_id

  notificaciones.enviadas:
    partitions: 12
    replication_factor: 3
    retention_ms: 2592000000  # 30 días
    partition_key: notificacion_id

  sla.violados:
    partitions: 3
    replication_factor: 3
    retention_ms: 2592000000  # 30 días
    partition_key: sla_id

  sla.cumplidos:
    partitions: 3
    replication_factor: 3
    retention_ms: 2592000000
    partition_key: sla_id

  ciudadanos.registrados:
    partitions: 3
    replication_factor: 3
    retention_ms: 604800000
    partition_key: ciudadano_id

  dependencias.configuradas:
    partitions: 3
    replication_factor: 3
    retention_ms: 604800000
    partition_key: dependencia_id
```

Flujo de Eventos - Ejemplo Completo



images/async-api-flujo-de-eventos.png

Patrones Aplicados

1. Event Sourcing

- Eventos como fuente de verdad
- Historial completo de cambios
- Replay de eventos para reconstruir estado

2. CQRS (Command Query Responsibility Segregation)

- Write model: PostgreSQL (transaccional)
- Read models: PostgreSQL replicas, Elasticsearch, ClickHouse
- Sincronización vía eventos Kafka

3. Saga Pattern

- Transacciones distribuidas usando eventos
- Compensación en caso de fallos
- Ejemplo: Crear caso → Asignar → Notificar → Registrar en sistema institucional

4. Outbox Pattern

- Eventos publicados desde base de datos transaccional
- Evita perdida de eventos en caso de fallos
- Implementación: Tabla `eventos_casos` + CDC (Change Data Capture) o polling

5. Dead Letter Queue (DLQ)

- Mensajes fallidos enviados a DLQ
- Reintentos automáticos con exponential backoff
- Alertas cuando mensajes van a DLQ

Consideraciones de Implementación

Idempotencia

- **Event ID único**: Cada evento tiene `evento_id` (UUID)
- **Consumer tracking**: Consumidores trackean eventos procesados
- **Replay seguro**: Reprocesar eventos sin duplicar efectos

Orden de Eventos

- **Partition key**: `caso_id` garantiza orden por caso
- **Consumer groups**: Un consumer group procesa en orden
- **Parallel processing**: Múltiples consumer groups para paralelismo

Manejo de Errores

- **Retry automático**: 3-5 reintentos con exponential backoff
- **DLQ**: Mensajes fallidos después de reintentos
- **Circuit breaker**: Detener consumo si hay fallos masivos
- **Alertas**: Notificaciones cuando eventos fallan

Versionado de Eventos

- **Estrategia**: Schema evolution (forward/backward compatible)
- **Versión en payload**: `evento_version` campo
- **Migración gradual**: Soporte múltiples versiones durante transición

---

Siguiente: Ver Supuestos y Riesgos

6. Supuestos, Riesgos y Mitigaciones - Conecta360

Supuestos Documentados

Supuestos de Negocio

1. Volumen de Casos

Supuesto: - Volumen promedio: 500,000 solicitudes diarias (≈ 5,787 req/s promedio) - Picos estimados: 20,000 req/s en días de alta demanda - Crecimiento anual estimado: 15-20%

Justificación: - Basado en población de 10+ millones de habitantes - Considerando múltiples canales de atención - Proyección conservadora de adopción ciudadana

Impacto si es incorrecto: - **Subestimación**: Requerirá escalado horizontal adicional (costos) - **Sobreestimación**: Infraestructura subutilizada (costos innecesarios)

Mitigación: - Arquitectura diseñada para escalar horizontalmente - Auto-scaling basado en métricas reales - Monitoreo continuo de volumen y ajuste de capacidad

---

2. Integración con Sistemas Legacy

Supuesto: - 70% de dependencias tienen sistemas con APIs REST/SOAP disponibles - 30% requerirán desarrollo de adaptadores o middleware - Sistemas legacy soportarán integración mediante: - REST APIs (preferido) - SOAP Web Services - Archivos planos (CSV/XML) por batch - Webhooks donde sea posible

Justificación: - Estimación basada en experiencia con sistemas gubernamentales - Algunas dependencias tendrán sistemas muy antiguos sin APIs

Impacto si es incorrecto: - **Mayor complejidad**: Más sistemas sin APIs requerirán adaptadores complejos - **Mayor tiempo de implementación**: Desarrollo adicional de middleware

Mitigación: - Análisis detallado de sistemas legacy en fase de discovery - Desarrollo de adaptadores genéricos reutilizables - Estrategia de batch processing para sistemas sin tiempo real

---

3. Adopción Ciudadana

Supuesto: - 60% de adopción en el primer año - 80% de adopción en el segundo año - Crecimiento gradual con campañas de divulgación

Justificación: - Basado en experiencias similares en otros países - Considerando resistencia al cambio inicial

Impacto si es incorrecto: - **Menor adopción**: Menor volumen real, infraestructura subutilizada - **Mayor adopción**: Posible necesidad de escalado adicional

Mitigación: - Plan de divulgación y capacitación ciudadana - Diseño de UX intuitivo y accesible - Soporte multi-canal para facilitar adopción

---

4. Disponibilidad de Personal Técnico

Supuesto: - Equipo técnico con experiencia en: - Microservicios y arquitecturas distribuidas - Kafka y mensajería asíncrona - Kubernetes y DevOps - Desarrollo cloud-native

Justificación: - Necesario para mantener y operar el sistema - Puede requerir capacitación adicional

Impacto si es incorrecto: - **Falta de expertise**: Curva de aprendizaje prolongada - **Rotación de personal**: Pérdida de conocimiento

Mitigación: - Documentación exhaustiva del sistema - Plan de capacitación técnica - Contratos con consultores expertos si es necesario - Knowledge transfer sessions

---

Supuestos Técnicos

5. Infraestructura Cloud

Supuesto: - Infraestructura cloud disponible (AWS/GCP/Azure) - Red de alta velocidad entre regiones - Servicios gestionados disponibles (RDS, EKS, MSK)

Justificación: - Arquitectura diseñada para cloud-native - Requiere servicios cloud modernos

Impacto si es incorrecto: - **On-premise requerido**: Requerirá adaptación significativa - **Servicios limitados**: Desarrollo adicional de componentes

Mitigación: - Diseño modular que permite on-premise con adaptaciones - Uso de contenedores (portabilidad) - Abstraction layer para servicios gestionados

---

6. Latencia de Red

Supuesto: - Latencia promedio entre regiones: < 50ms - Latencia entre servicios en misma región: < 5ms - Ancho de banda suficiente para volumen estimado

Justificación: - Requerido para tiempos de respuesta < 1.5s - Necesario para replicación multi-región

Impacto si es incorrecto: - **Mayor latencia**: Afectará tiempos de respuesta - **Ancho de banda insuficiente**: Cuellos de botella

Mitigación: - CDN para assets estáticos - Caché distribuido (Redis) para reducir latencia - Optimización de queries y responses

---

7. Disponibilidad de Servicios Externos

Supuesto: - Proveedores de mensajería (Twilio, SendGrid): SLA 99.9% - SSO Nacional: Disponible 24/7 - APIs de redes sociales: Disponibilidad razonable

Justificación: - Dependencias externas críticas para operación

Impacto si es incorrecto: - **Fallos de servicios externos**: Afectará notificaciones - **Degradación de servicio**: Funcionalidades parciales

Mitigación: - Circuit breakers para servicios externos - Retry automático con exponential backoff - Fallback mechanisms (ej: email si SMS falla) - Cola de mensajes para reintentos

---

Riesgos Identificados y Mitigaciones

Riesgos Técnicos

R1: Fallos en Kafka (Alto)

Descripción: Kafka es crítico para la comunicación entre microservicios. Un fallo puede afectar múltiples funcionalidades.

Probabilidad: Media
Impacto: Alto
Severidad: Alta

Mitigaciones: 1. **Replicación**: Replication factor de 3 (tolerancia a fallo de 2 brokers) 2. **Multi-región**: Kafka clusters en múltiples regiones 3. **Monitoreo**: Alertas proactivas de salud de brokers 4. **Outbox pattern**: Eventos persistidos en DB antes de publicación 5. **Dead Letter Queue**: Mensajes fallidos capturados y procesados manualmente 6. **Backup y recovery**: Estrategia de backup de topics críticos 7. **Circuit breaker**: Aislamiento de fallos de Kafka

Plan de Contingencia: - Failover automático a región secundaria - Modo degradado: Operaciones críticas directas a DB (sin eventos) - Escalación inmediata al equipo técnico

---

R2: Problemas de Performance en Base de Datos (Alto)

Descripción: PostgreSQL puede convertirse en cuello de botella con 500K solicitudes

diarias.

Probabilidad: Media
Impacto: Alto
Severidad: Alta

Mitigaciones: 1. **Read replicas**: Distribución de lecturas en múltiples réplicas 2. **Connection pooling**: PgBouncer para optimizar conexiones 3. **Particionamiento**: Tablas particionadas por fecha (mensual) 4. **Índices estratégicos**: Optimización de queries frecuentes 5. **Caché**: Redis para datos frecuentes (dependencias, categorías) 6. **CQRS**: Separación de modelos de lectura/escritura 7. **Query optimization**: Análisis y optimización continua 8. **Auto-scaling**: Escalado vertical/horizontal según carga

Plan de Contingencia: - Throttling de requests si DB está sobrecargada - Degradación temporal de features no críticas - Escalado de instancias DB en minutos

---

R3: Pérdida de Datos (Crítico)

Descripción: Pérdida de datos de casos ciudadanos sería crítica para el gobierno.

Probabilidad: Baja
Impacto: Crítico
Severidad: Crítica

Mitigaciones: 1. **Backups automáticos**: - Incrementales diarios - Completos semanales - Retención de 90 días 2. **Replicación**: Multi-región con replicación síncrona/asíncrona 3. **Point-in-time recovery**: Capacidad de restaurar a cualquier punto en tiempo 4. **Testing de restauración**: Pruebas regulares de restore 5. **Auditoría**: Log completo de todas las operaciones 6. **Versionado**: Event sourcing permite reconstruir estado

Plan de Contingencia: - Procedimiento documentado de restauración - RTO < 1 hora, RPO < 15 minutos - Equipo de respuesta disponible 24/7

R4: Vulnerabilidades de Seguridad (Alto)

Descripción: Sistema gubernamental con datos personales ciudadanos es objetivo de ataques.

Probabilidad: Media-Alta
Impacto: Alto
Severidad: Alta

Mitigaciones: 1. **Seguridad en capas**: - WAF (Web Application Firewall) - API Gateway con rate limiting - Autenticación OAuth2/OIDC - mTLS entre microservicios 2. **Cifrado**: - TLS 1.3 en tránsito - AES-256 en reposo 3. **Acceso**: - RBAC granular - Segregación por dependencia - Audit logging completo 4. **Monitoreo**: - SIEM (Security Information and Event Management) - Detección de anomalías - Alertas de seguridad 5. **Testing**: - Penetration testing regular - Security audits - Dependency scanning 6. **Compliance**: - Cumplimiento normativo de protección de datos - Certificaciones de seguridad

Plan de Contingencia: - Procedimiento de respuesta a incidentes - Equipo de seguridad dedicado - Comunicación a ciudadanos afectados (si aplica) - Escalación a autoridades competentes

---

R5: Problemas de Integración con Sistemas Legacy (Medio)

Descripción: Sistemas legacy pueden no tener APIs o tener problemas de estabilidad.

Probabilidad: Alta
Impacto: Medio
Severidad: Media

Mitigaciones: 1. **Adaptadores genéricos**: Desarrollo de adaptadores reutilizables 2. **Batch processing**: Integración por lotes para sistemas sin tiempo real 3. **Circuit breakers**: Aislamiento de fallos de sistemas externos 4. **Retry logic**: Reintentos automáticos con exponential backoff 5. **Timeouts**: Timeouts apropiados para evitar bloqueos 6. **Fallback mechanisms**: Modo degradado cuando integraciones fallan 7. **Testing exhaustivo**: Pruebas con sistemas legacy reales

Plan de Contingencia: - Modo offline: Operaciones sin integración (sincronización

posterior) - Alertas para integraciones fallidas - Soporte técnico para dependencias con problemas

---

R6: Escalabilidad Insuficiente (Medio)

Descripción: El sistema no escala adecuadamente ante picos de demanda inesperados.

Probabilidad: Baja
Impacto: Medio
Severidad: Media

Mitigaciones: 1. **Auto-scaling**: Horizontal Pod Autoscaling (HPA) basado en métricas 2. **Load balancing**: Distribución eficiente de carga 3. **Caching**: Redis para reducir carga en servicios backend 4. **CDN**: CloudFront para assets estáticos 5. **Database optimization**: Read replicas, connection pooling 6. **Performance testing**: Load testing regular 7. **Capacity planning**: Monitoreo y proyección de capacidad

Plan de Contingencia: - Escalado manual de recursos críticos - Throttling de requests no críticos - Degradación temporal de features opcionales

---

Riesgos de Negocio

R7: Resistencia al Cambio Organizacional (Alto)

Descripción: Funcionarios y ciudadanos pueden resistirse a adoptar el nuevo sistema.

Probabilidad: Media-Alta
Impacto: Alto
Severidad: Alta

Mitigaciones: 1. **Change management**: Plan de gestión del cambio 2. **Capacitación**: Programas de entrenamiento para funcionarios 3. **Divulgación**: Campañas de comunicación ciudadana 4. **UX intuitivo**: Diseño centrado en usuario 5. **Soporte**: Canales de soporte disponibles 6. **Feedback**: Mecanismos de retroalimentación y mejora continua 7. **Fase de transición**: Migración gradual, soporte dual durante transición

Plan de Contingencia: - Equipo de soporte ampliado durante rollout - Material de capacitación adicional - Extensiones de plazo si es necesario

---

R8: Cumplimiento Normativo (Alto)

Descripción: Sistema debe cumplir con normativas de protección de datos y seguridad.

Probabilidad: Media
Impacto: Alto
Severidad: Alta

Mitigaciones: 1. **Análisis legal**: Revisión con asesores legales 2. **Privacy by design**: Protección de datos desde el diseño 3. **Auditoría**: Logs completos para cumplimiento 4. **Consentimiento**: Gestión apropiada de consentimientos ciudadanos 5. **Data retention**: Políticas de retención de datos 6. **Right to be forgotten**: Capacidad de eliminar datos personales 7. **Certificaciones**: Certificaciones de seguridad requeridas

Plan de Contingencia: - Revisiones legales regulares - Actualización de políticas según normativas - Consulta con autoridades regulatorias

---

R9: Costos Superiores a Presupuesto (Medio)

Descripción: Costos de infraestructura cloud pueden exceder presupuesto inicial.

Probabilidad: Media
Impacto: Medio
Severidad: Media

Mitigaciones: 1. **Budget monitoring**: Monitoreo continuo de costos 2. **Cost optimization**: - Reserved instances para workloads estables - Spot instances para workloads tolerantes a fallos - Auto-scaling para reducir instancias en bajo uso - Data tiering (datos antiguos a storage económico) 3. **Right-sizing**: Optimización de tamaños de

instancias 4. **Caching**: Reducir carga en servicios pagados por uso 5. **Cost alerts**: Alertas cuando costos exceden umbrales

Plan de Contingencia: - Revisión trimestral de costos - Optimización continua basada en uso real - Consideración de on-premise para workloads estables si es más económico

---

R10: Dependencias de Proveedores (Medio)

Descripción: Vendor lock-in con proveedores cloud o servicios externos.

Probabilidad: Media
Impacto: Medio
Severidad: Media

Mitigaciones: 1. **Multi-cloud strategy**: Estrategia multi-cloud cuando sea posible 2. **Abstraction layers**: Capas de abstracción para servicios gestionados 3. **Open source**: Priorización de tecnologías open source 4. **Portabilidad**: Uso de contenedores (Kubernetes) para portabilidad 5. **Contratos**: Cláusulas de salida en contratos con proveedores 6. **Estrategia de salida**: Plan documentado para migración si es necesario

Plan de Contingencia: - Migración gradual a alternativas si es necesario - Mantener capacidades de migración en todo momento

---

Riesgos de Proyecto

R11: Retrasos en Implementación (Medio)

Descripción: Proyecto puede retrasarse por complejidad técnica o dependencias externas.

Probabilidad: Media
Impacto: Medio
Severidad: Media

Mitigaciones: 1. **Metodología ágil**: Sprints cortos con entregas incrementales 2. **MVP primero**: Valor entregado desde el inicio 3. **Gestión de dependencias**: Identificación temprana de dependencias críticas 4. **Buffer de tiempo**: Contingencia en cronograma 5. **Priorización**: Features críticas primero 6. **Comunicación**: Comunicación regular con stakeholders

Plan de Contingencia: - Revisión de alcance si es necesario - Extensiones de plazo negociables - Priorización de features críticas

---

R12: Falta de Expertise Técnico (Medio)

Descripción: Equipo técnico puede no tener suficiente experiencia en tecnologías seleccionadas.

Probabilidad: Media
Impacto: Medio
Severidad: Media

Mitigaciones: 1. **Capacitación**: Plan de entrenamiento técnico 2. **Consultores**: Contratos con consultores expertos 3. **Documentación**: Documentación exhaustiva del sistema 4. **Code reviews**: Revisión de código para transferencia de conocimiento 5. **Pair programming**: Programación en parejas para aprendizaje 6. **Comunidad**: Participación en comunidades técnicas

Plan de Contingencia: - Contratos con consultores externos - Capacitación intensiva si es necesario - Extensión de plazos para aprendizaje

---

Matriz de Riesgos

Riesgo	Probabilidad	Impacto	Severidad	Prioridad
R3: Pérdida de Datos	Baja	Crítico	Crítica	**1**
R1: Fallos en Kafka	Media	Alto	Alta	**2**
R2: Performance DB	Media	Alto	Alta	**2**
R4: Vulnerabilidades	Media-Alta	Alto	Alta	**2**
R7: Resistencia Cambio	Media-Alta	Alto	Alta	**3**
R8: Cumplimiento Normativo	Media	Alto	Alta	**3**
R5: Integración Legacy	Alta	Medio	Media	**4**
R6: Escalabilidad	Baja	Medio	Media	**4**
R9: Costos	Media	Medio	Media	**5**
R10: Vendor Lock-in	Media	Medio	Media	**5**
R11: Retrasos	Media	Medio	Media	**5**
R12: Falta Expertise	Media	Medio	Media	**5**

Plan de Monitoreo de Riesgos

1. **Revisión mensual**: Matriz de riesgos actualizada mensualmente
2. **Alertas**: Alertas automáticas para riesgos críticos
3. **Reporting**: Reportes trimestrales a stakeholders
4. **Mitigaciones activas**: Seguimiento de acciones de mitigación
5. **Actualización**: Nuevos riesgos identificados y documentados

---

Conclusión

Los riesgos identificados están documentados con probabilidades e impactos estimados. Las mitigaciones propuestas incluyen medidas técnicas, organizacionales y de proceso. Los riesgos críticos (pérdida de datos, fallos críticos, seguridad) tienen planes de contingencia detallados.

Priorización de Mitigaciones: 1. Seguridad de datos y backups 2. Alta disponibilidad y resiliencia 3. Performance y escalabilidad 4. Integración con sistemas legacy 5. Gestión del cambio organizacional

---

Siguiente: Ver Despliegue y Mantenimiento

7. Consideraciones de Despliegue y Mantenimiento - Conecta360

Visión General

Este documento describe la estrategia de despliegue, operación y mantenimiento a largo plazo del sistema Conecta360, asegurando alta disponibilidad, escalabilidad continua y evolución del sistema.

Estrategia de Despliegue

Ambientes

1. Desarrollo (Development)

Propósito: Desarrollo activo y pruebas unitarias

Configuración: - Kubernetes namespace: `dev` - Infraestructura mínima (1-2 nodos) - Base de datos: Single instance PostgreSQL - Kafka: Single broker (o Kafka local) - Sin replicación (para reducir costos) - Auto-scaling deshabilitado

Características: - Deploy automático desde ramas de desarrollo - Datos de prueba sintéticos - Logs en consola local - Sin monitoreo complejo

---

2. Testing/QA (Quality Assurance)

Propósito: Pruebas funcionales e integración

Configuración: - Kubernetes namespace: `qa` - Infraestructura similar a producción (escalada 1:4) - Base de datos: Primary + 1 Read Replica - Kafka: Cluster de 3 brokers - Replicación habilitada

Características: - Deploy automático desde rama `qa` o `testing` - Datos de prueba más realistas - Monitoreo básico (Prometheus, Grafana) - Pruebas de carga periódicas

---

3. Staging/Pre-Producción

Propósito: Pruebas finales antes de producción, capacitación

Configuración: - Kubernetes namespace: `staging` - Infraestructura idéntica a producción (escalada 1:2) - Base de datos: Primary + Read Replicas (multi-región) - Kafka: Cluster completo con replicación - Todas las integraciones reales (con limites)

Características: - Deploy desde rama `staging` o releases candidates - Datos anonimizados de producción - Monitoreo completo (mismo que producción) - Pruebas de desastre recovery

---

4. Producción (Production)

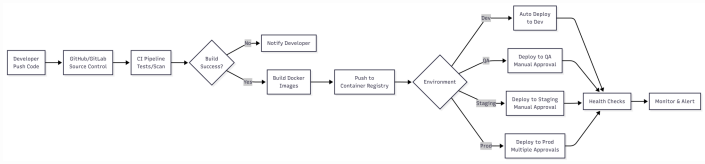
Propósito: Sistema en operación real

Configuración: - Kubernetes namespace: `prod` - Multi-región: Región A (principal) + Región B (DR) - Base de datos: Primary + múltiples Read Replicas por región - Kafka: Cluster de 3 brokers por región con replicación cross-region - CDN global para assets estáticos - Auto-scaling completo

Características: - Deploy controlado desde rama `main` con aprobaciones - Datos reales de ciudadanos - Monitoreo completo y alertas 24/7 - Backup y disaster recovery activos - SSL/TLS end-to-end

---

Pipeline CI/CD



images/pipeline-ci-cd.png

Etapas del Pipeline

1. Source Control - Git flow: `main`, `develop`, `feature/*`, `hotfix/*` - Branch protection: Requiere PR reviews y CI checks - Commit hooks: Linting, format checks

2. Continuous Integration (CI)

```yaml
# Ejemplo .github/workflows/ci.yml
name: CI Pipeline
on:
  pull_request:
  push:
    branches: [main, develop]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Run unit tests
        run: npm test
      - name: Run integration tests
        run: npm run test:integration
      - name: Security scanning
        run: npm audit && snyk test

  build:
    needs: test
    runs-on: ubuntu-latest
    steps:
      - name: Build Docker image
        run: docker build -t conecta360/service:${{ github.sha }} .
      - name: Push to registry
        run: docker push conecta360/service:${{ github.sha }}

  deploy-dev:
    needs: build
    if: github.ref == 'refs/heads/develop'
    runs-on: ubuntu-latest
    steps:
      - name: Deploy to Dev
        run: kubectl set image deployment/service service=conecta360/servi
```

3. Continuous Deployment (CD) - **Dev**: Auto-deploy en cada push a `develop` - **QA**: Auto-deploy con aprobación de QA lead - **Staging**: Deploy manual con aprobación de tech lead - **Production**: Blue-Green o Canary deployment con múltiples aprobaciones

---

Estrategias de Despliegue

1. Blue-Green Deployment

Descripción: Mantener dos ambientes idénticos (Blue y Green), cambiar tráfico instantáneamente

Ventajas: - Rollback instantáneo - Sin downtime - Pruebas en ambiente idéntico antes de switch

Uso: Producción para releases mayores

Implementación:

```yaml
# Kubernetes Service con selectores intercambiables
apiVersion: v1
kind: Service
metadata:
  name: casos-service
spec:
  selector:
    version: blue  # Cambiar a 'green' para nuevo deploy
  ports:
    - port: 80
```

Proceso: 1. Desplegar nueva versión en ambiente Green (paralelo a Blue) 2. Ejecutar smoke tests en Green 3. Cambiar selector de Service a Green 4. Monitorear métricas 5. Si hay problemas, revertir a Blue (instantáneo) 6. Si todo OK, mantener Green y preparar próximo Blue

---

2. Canary Deployment

Descripción: Desplegar nueva versión a un pequeño porcentaje de tráfico, incrementar

gradualmente

Ventajas: - Detección temprana de problemas - Rollback fácil - Menor impacto si hay issues

Uso: Producción para releases menores o actualizaciones frecuentes

Implementación:

```yaml
# Istio VirtualService para canary
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: casos-vs
spec:
  hosts:
    - casos.conecta360.gov.cv
  http:
    - match:
        - headers:
            canary:
              exact: "true"
      route:
        - destination:
            host: casos-service
            subset: canary
      weight: 10  # 10% tráfico
    - route:
        - destination:
            host: casos-service
            subset: stable
      weight: 90  # 90% tráfico
```

Proceso: 1. Desplegar canary (10% tráfico) 2. Monitorear métricas por 30 minutos 3. Si OK, incrementar a 25%, luego 50%, luego 100% 4. Si hay problemas, reducir a 0% (rollback)

---

3. Rolling Update (Default Kubernetes)

Descripción: Actualizar pods gradualmente, reemplazando uno por uno

Ventajas: - Simple, automático - Sin downtime si hay múltiples réplicas

Uso: Desarrollo, QA, releases menores

Configuración:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: casos-deployment
spec:
  replicas: 5
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1        # Pods adicionales durante update
      maxUnavailable: 1  # Pods que pueden estar down
  template:
    spec:
      containers:
      - name: casos
        image: conecta360/casos:v1.2.0
```

Configuración de Infraestructura como Código (IaC)

Terraform para Infraestructura Cloud

```
# Ejemplo: main.tf
provider "aws" {
  region = "us-east-1"
}

module "eks_cluster" {
  source = "./modules/eks"

  cluster_name    = "conecta360-prod"
  node_group_size = "t3.large"
  min_nodes       = 3
  max_nodes       = 10
  desired_nodes   = 5
}

module "rds_postgres" {
  source = "./modules/rds"

  instance_class      = "db.r5.xlarge"
  allocated_storage   = 500
  multi_az            = true
  backup_retention    = 30
  deletion_protection = true
}

module "kafka_msk" {
  source = "./modules/msk"

  cluster_name        = "conecta360-kafka"
  instance_type       = "kafka.m5.large"
  broker_count        = 3
  replication_factor  = 3
}
```

Helm Charts para Kubernetes

```yaml
# valores-prod.yaml
replicaCount: 5

image:
  repository: conecta360/casos
  tag: "v1.2.0"
  pullPolicy: Always

resources:
  requests:
    cpu: 500m
    memory: 1Gi
  limits:
    cpu: 2000m
    memory: 4Gi

autoscaling:
  enabled: true
  minReplicas: 5
  maxReplicas: 20
  targetCPUUtilizationPercentage: 70
  targetMemoryUtilizationPercentage: 80

env:
  - name: DB_HOST
    valueFrom:
      secretKeyRef:
        name: db-credentials
        key: host
  - name: KAFKA_BROKERS
    value: "kafka-1:9092,kafka-2:9092,kafka-3:9092"
```

Operación y Mantenimiento

Monitoreo y Observabilidad

Stack de Monitoreo

1. Métricas (Prometheus + Grafana) - **Recolección**: Prometheus scraping cada 15s - **Visualización**: Grafana dashboards - **Retención**: 30 días de métricas

Métricas Críticas: - Request rate (req/s) - Response time (P50, P95, P99) - Error rate (%) - CPU/Memoria por servicio - Database connections - Kafka lag - SLA compliance rate

Dashboards: - Dashboard Global: Visión general del sistema - Dashboard por Servicio: Métricas detalladas por microservicio - Dashboard de Infraestructura: CPU, memoria, red, disk - Dashboard de Negocio: Casos por estado, tiempos de resolución, SLA

---

2. Logging (ELK Stack) - **Recolección**: Filebeat en cada pod → Logstash → Elasticsearch - **Visualización**: Kibana - **Retención**: 90 días de logs

Logs Estructurados:

```
{
  "timestamp": "2024-01-15T10:30:00Z",
  "level": "INFO",
  "service": "microservicio-casos",
  "trace_id": "abc123",
  "span_id": "def456",
  "message": "Caso creado exitosamente",
  "caso_id": 12345,
  "numero_caso": "CRV-2024-00012345",
  "usuario_id": 5432,
  "request_id": "req_xyz789"
}
```

Búsqueda en Kibana: - Búsqueda por trace_id para seguir requests - Búsqueda por caso_id para auditoría - Alertas basadas en patrones de logs (errores, excepciones)

---

3. Distributed Tracing (Jaeger/Zipkin) - **Propósito**: Seguir requests a través de múltiples servicios - **Sampling**: 100% en producción (o adaptativo según carga)

Información Capturada: - Tiempo de cada operación - Servicios involucrados - Llamadas HTTP/gRPC entre servicios - Queries a base de datos - Llamadas a Kafka

---

Alertas y Notificaciones

Niveles de Severidad:

Critical (Pager Duty - On-call 24/7) - Servicios down - Database down - Kafka cluster down - Error rate > 10% - SLA violation masivo - Security breach

High (Email + Slack) - Error rate > 5% - Response time P95 > 3s - Database connection pool exhaustion - Kafka lag > 1000 mensajes - Disk space > 80%

Medium (Slack) - CPU > 80% por más de 10 minutos - Memory > 80% por más de 10 minutos - Warnings en logs

Low (Slack channel de info) - Deployments exitosos - Health checks OK después de fallos

Ejemplo de Alerta (Prometheus + Alertmanager):

```
groups:
  - name: conecta360_alerts
    rules:
      - alert: HighErrorRate
        expr: rate(http_requests_total{status=~"5.."}[5m]) > 0.05
        for: 5m
        labels:
          severity: high
        annotations:
          summary: "Error rate above 5%"
          description: "Service {{ $labels.service }} has error rate of {{

      - alert: DatabaseDown
        expr: up{job="postgres"} == 0
        for: 1m
        labels:
          severity: critical
        annotations:
          summary: "PostgreSQL database is down"
```

Mantenimiento Preventivo

1. Actualizaciones y Parches

Política de Actualizaciones: - **Security patches**: Aplicar dentro de 48 horas - **Minor updates**: Mensual (ventana de mantenimiento) - **Major updates**: Trimestral con plan detallado

Proceso: 1. **Evaluación**: Revisar changelog y breaking changes 2. **Testing**: Probar en QA/Staging por 1 semana 3. **Planificación**: Ventana de mantenimiento comunicada 4. **Ejecución**: Blue-Green deployment en producción 5. **Validación**: Smoke tests post-deployment 6. **Monitoreo**: 24 horas de monitoreo intensivo

---

2. Backup y Restauración

Estrategia de Backups:

Base de Datos (PostgreSQL): - **Incrementales**: Cada 6 horas (retención 7 días) - **Completos diarios**: A las 2:00 AM (retención 30 días) - **Completos semanales**: Domingos (retención 90 días) - **Backup antes de deployments**: Automático

Backup de Configuraciones: - Kubernetes configs: Git (versionado) - Secrets: HashiCorp Vault o AWS Secrets Manager - Terraform state: S3 con versionado

Testing de Restauración: - **Mensual**: Restaurar backup de staging en ambiente aislado - **Trimestral**: Disaster recovery drill completo - **Documentación**: Procedimientos documentados y actualizados

RTO/RPO Objetivos: - **RTO (Recovery Time Objective)**: < 1 hora - **RPO (Recovery Point Objective)**: < 15 minutos

---

3. Limpieza y Archivado

Datos Antiguos: - **Casos cerrados > 2 años**: Archivados a S3 Glacier - **Logs > 90 días**: Compressed y movidos a S3 - **Eventos Kafka > 30 días**: Compactados o eliminados - **Auditoría**: Retención según normativa (7 años recomendado)

Proceso Automatizado:

```
# Ejemplo: Script de archivado mensual
def archivar_casos_antiguos():
    fecha_limite = datetime.now() - timedelta(days=730)
    casos = obtener_casos_cerrados_anteriores_a(fecha_limite)

    for caso in casos:
        # Exportar a formato comprimido
        datos = exportar_caso_completo(caso)
        # Subir a S3 Glacier
        s3.upload_to_glacier(f"archivos/{caso.numero_caso}.tar.gz", datos)
        # Eliminar de base de datos principal
        eliminar_caso(caso.id)
        # Registrar en índice de archivos
        registrar_archivo(caso.numero_caso, "s3://glacier/...")
```

---

4. Performance Tuning Continuo

Análisis Regular: - **Semanal**: Revisión de queries lentas en PostgreSQL - **Mensual**: Análisis de uso de índices - **Trimestral**: Optimización de microservicios (profiling)

Herramientas: - PostgreSQL `pg_stat_statements` para queries lentas - APM tools (New Relic, Datadog, o similar) - Profiling de código (Java Flight Recorder, Node.js clinic.js)

Optimizaciones Comunes: - Agregar índices faltantes - Optimizar queries N+1 - Ajustar connection pools - Tuning de JVM (heap size, GC) - Cache misses analysis

---

5. Seguridad Continuada

Auditorías de Seguridad: - **Mensual**: Vulnerability scanning (dependencias) - **Trimestral**: Penetration testing externo - **Anual**: Security audit completo

Actividades Regulares: - Rotación de secrets (cada 90 días) - Revisión de permisos RBAC (trimestral) - Análisis de logs de seguridad (SIEM) - Actualización de certificados SSL/TLS

Compliance: - Revisión de cumplimiento normativo (semestral) - Actualización de políticas según nuevas normativas - Reportes de compliance a autoridades

---

Escalabilidad y Capacity Planning

Análisis de Crecimiento

Métricas a Monitorear: - Casos creados por día (tendencia) - Usuarios activos (crecimiento) - Tráfico de API (req/s) - Uso de recursos (CPU, memoria, disk, network) - Costos de infraestructura

Proyecciones: - **Mensual**: Revisar tendencias y proyectar 3 meses - **Trimestral**: Proyección anual con ajustes - **Anual**: Plan de capacidad para siguiente año

Escalado Proactivo: - Auto-scaling para picos de tráfico - Escalado manual anticipado para eventos conocidos (ej: campañas gubernamentales) - Provisioning adicional antes de fechas de alta demanda

---

Cost Optimization

Revisión Trimestral: 1. Análisis de uso de recursos 2. Identificación de instancias subutilizadas 3. Right-sizing recommendations 4. Evaluación de Reserved Instances vs On-Demand 5. Optimización de storage (data tiering)

Estrategias: - **Reserved Instances**: Para workloads estables (1-3 años) - **Spot Instances**: Para workloads tolerantes a fallos - **Auto-scaling**: Reducir instancias en horas de bajo uso - **Data tiering**: Datos antiguos a storage económico - **Cache optimization**: Reducir carga en servicios pagados

---

Gestión de Incidencias

Runbooks (Procedimientos Operativos)

Ejemplo: Runbook para “Caso no se asigna automáticamente”

1. **Síntoma**: Casos quedan en estado PENDIENTE por > 10 minutos
2. **Diagnóstico**:

```
# Verificar servicio de derivación
kubectl logs -n prod deployment/microservicio-derivacion --tail=100

# Verificar Kafka consumers
kafka-consumer-groups.sh --bootstrap-server kafka:9092 \
  --group derivacion-group --describe

# Verificar reglas de derivación en DB
psql -c "SELECT * FROM reglas_derivacion WHERE activa = true;"
```

3. **Acciones**:
   - Si servicio down: `kubectl rollout restart deployment/microservicio-derivacion`
   - Si Kafka lag: Escalar consumers o reiniciar servicio
   - Si reglas incorrectas: Revisar y corregir configuración
4. **Verificación**: Crear caso de prueba y verificar asignación
5. **Escalación**: Si problema persiste > 15 min, escalar a on-call

Runbooks Disponibles: - Database connection pool exhaustion - Kafka broker down - High error rate en microservicio - SSL certificate expiration - Disk space full - Service memory leak

---

On-Call Rotation

Estructura: - **Primary on-call**: Disponible 24/7, responde en < 15 minutos - **Secondary on-call**: Backup si primary no responde - **Rotación**: Semanal (lunes a lunes)

Herramientas: - PagerDuty o OpsGenie para alertas - Escalación automática si no hay respuesta - Runbooks accesibles desde móvil

---

Documentación Operativa

Documentación Requerida

1. Arquitectura - Diagramas actualizados - Descripción de componentes - Flujos de datos - Integraciones externas

2. Procedimientos - Deployment procedures - Rollback procedures - Backup/restore procedures - Disaster recovery plan

3. Troubleshooting - Runbooks para incidencias comunes - Known issues y workarounds - Contactos de soporte (proveedores externos)

4. Cambios - Changelog de releases - Breaking changes documentados - Migration guides

---

Plan de Mantenimiento a Largo Plazo

Evolución del Sistema

Roadmap Técnico (3-5 años)

Año 1: Estabilización - Optimización de performance - Refinamiento de procesos operativos - Expansión de integraciones - Mejora de UX basada en feedback

Año 2: Mejoras Avanzadas - Machine Learning para routing inteligente - Chatbot más sofisticado (GPT-based) - Mobile app nativa optimizada - Integración con más canales (WhatsApp, Telegram)

Año 3-5: Innovación - Predictive analytics (predecir demandas) - IoT integration (sensores ciudadanos) - Blockchain para trazabilidad inmutable (si es requerido) - AI avanzada para resolución automática de casos simples

---

Actualizaciones Tecnológicas

Ciclo de Vida de Tecnologías: - **LTS (Long Term Support)**: Preferir versiones LTS - **Deprecation timeline**: Planificar migraciones con 12-18 meses de anticipación - **Security EOL**: Actualizar antes de End of Life

Estrategia de Actualización: - Evaluar nuevas tecnologías anualmente - Prototipos en ambientes no productivos - Migraciones graduales cuando sea posible - Documentar decisiones de no actualizar (cuando aplique)

Sostenibilidad

Técnica

- **Code quality**: Code reviews, testing coverage > 80%
- **Documentación**: Mantenida y actualizada
- **Debt management**: Identificar y priorizar technical debt
- **Refactoring**: Refactoring continuo para mantener código limpio

Organizacional

- **Knowledge sharing**: Sessions regulares de conocimiento
- **Training**: Capacitación continua del equipo
- **Documentation**: Conocimiento documentado, no solo en personas
- **Succession planning**: No depender de una sola persona

Financiera

- **Cost monitoring**: Monitoreo continuo de costos
- **ROI analysis**: Medir retorno de inversión
- **Budget planning**: Planificación anual de presupuesto
- **Optimization**: Optimización continua de costos

Conclusión

El sistema Conecta360 está diseñado para ser **operable, mantenible y evolutivo** a largo plazo. Las estrategias de despliegue, monitoreo, mantenimiento y evolución descritas aseguran que el sistema pueda:

1. **Operar confiablemente** 24/7 con alta disponibilidad
2. **Escalar** según demanda creciente
3. **Mantenerse** de forma eficiente y económica
4. **Evolucionar** con nuevas tecnologías y requerimientos

La clave está en **automatización**, **monitoreo proactivo**, **documentación actualizada** y **mejora continua**.

Fin de la Documentación Arquitectónica