****

**UNIVERSIDAD PRIVADA DE TACNA**

**FACULTAD DE INGENIERÍA**

**Escuela Profesional de Ingeniería de Sistemas**

**Proyecto “Sistema Web y Móvil para la detección de enfermedades respiratorias en Tacna en 2025”**

Curso: Construcción de Software I

Docente: Alberto Flor Rodríguez

Integrantes:

***Chávez Linares, Cesar Fabian (2019063854)***

**Tacna – Perú**

**2025**

| CONTROL DE VERSIONES | | | | | |
| --- | --- | --- | --- | --- | --- |
| Versión | Hecha por | Revisada por | Aprobada por | Fecha | Motivo |
| 1.0 | CCL | AFR | AFR | 08/09/2025 | Versión Original |

*“Sistema Web y Móvil para la detección de enfermedades respiratorias en Tacna en 2025”*

Documento de Arquitectura de Software

Versión 1.0

| CONTROL DE VERSIONES | | | | | |
| --- | --- | --- | --- | --- | --- |
| Versión | Hecha por | Revisada por | Aprobada por | Fecha | Motivo |
| 1.0 | CCL | AFR | AFR | 08/09/2025 | Versión Original |

ÍNDICE GENERAL

1. Introducción [4](#_heading=h.w9zv38cnbbxb)

1.1 Propósito [4](#_heading=h.6qy35rlhf689)

1.2 Alcance [4](#_heading=h.4pezbb6mqxd9)

1.3 Definición, siglas y abreviaturas [4](#_heading=h.hny2p2mwsktn)

1.4 Referencias [4](#_heading=h.d5cnwq1sgqup)

1.5 Visión General [5](#_heading=h.1keh9da1ucxf)

2. Representación Arquitectónica [5](#_heading=h.km0ylng40vqw)

2.1 Escenarios [5](#_heading=h.z7fj738unfmw)

2.2 Vista Lógica [6](#_heading=h.4nu48qwjdb51)

2.3 Vista del Proceso [6](#_heading=h.de3m0meyw7e9)

2.4 Vista del desarrollo [6](#_heading=h.hkdi2fdni0lp)

2.5 Vista Física [6](#_heading=h.dyvjqhij2mya)

3. Objetivos y limitaciones arquitectónicas [7](#_heading=h.vg60b2v6q900)

3.1 Disponibilidad [7](#_heading=h.wo3v7txz27uu)

3.2 Seguridad [7](#_heading=h.tbbldi2ajt67)

3.3 Adaptabilidad [7](#_heading=h.up9n6k7rovbp)

3.4 Rendimiento [7](#_heading=h.a7sdp5nuegml)

4. Análisis de Requerimientos [8](#_heading=h.8oxne7ksb2x9)

4.1 Requerimientos funcionales [7](#_heading=h.wo3v7txz27uu)

4.2 Requerimientos no funcionales [7](#_heading=h.tbbldi2ajt67)

5. Vistas de Caso de Uso [9](#_heading=h.f6bdszs17709)

6. Vista Lógica [17](#_heading=h.hhp9aatshwvv)

6.1 Diagrama Contextual [17](#_heading=h.vplvr5x2h8mr)

7. Vista de Procesos [18](#_heading=h.5ox6z8gx2nkv)

7.1 Diagrama de Proceso Actual [18](#_heading=h.d7zrafq0cwgr)

7.2 Diagrama de Proceso Propuesto [18](#_heading=h.2e34plv3edla)

8. Vista de Despliegue [19](#_heading=h.btvv6i2i9bce)

8.1 Diagrama de Contenedor [19](#_heading=h.x6mmr6jbmd18)

9. Vista de Implementación [20](#_heading=h.vffyeb9ktyc4)

9.1 Diagrama de Componentes [20](#_heading=h.praszbh3h6ek)

10. Vista de Datos [22](#_heading=h.b3gygk34s27k)

10.1 Diagrama Entidad Relación [22](#_heading=h.8im5sa9wxcme)

11. Calidad [24](#_heading=h.3ruodebtfnr9)

11.1 Escenario de Seguridad 25

11.2 Escenario de Usabilidad 26

11.3 Escenario de Adaptabilidad 27

11.4 Escenario de Disponibilidad 28

11.5 Otro Escenario 28

**1. Introducción**

**1.1 Propósito**

Este documento define la arquitectura de software para el sistema RespiCare-Tacna, una plataforma web y móvil diseñada para la detección temprana de enfermedades respiratorias en la región de Tacna mediante técnicas de Procesamiento de Lenguaje Natural (PLN).

El propósito principal es proporcionar una guía completa para desarrolladores, arquitectos de software y stakeholders, describiendo las decisiones arquitectónicas, patrones de diseño y estructuras del sistema que permitirán cumplir con los objetivos de negocio establecidos.

**1.2 Alcance**

La arquitectura abarca el diseño completo del sistema RespiCare-Tacna, incluyendo:

**Incluye:**

* Aplicación web para profesionales de salud y administradores
* Aplicación móvil multiplataforma para pacientes y usuarios finales
* Servicios de backend para procesamiento de PLN y gestión de datos
* Integración con sistemas de información de salud existentes
* Módulos de análisis, reportes y notificaciones
* Infraestructura de seguridad y privacidad de datos médicos

**No Incluye:**

* Integración directa con equipos médicos especializados
* Sistemas de telemedicina en tiempo real
* Prescripción automatizada de medicamentos
* Gestión de historias clínicas hospitalarias externas

**1.3 Definición, siglas y abreviaturas**

| **Término** | **Definición** |
| --- | --- |
| PLN | Procesamiento de Lenguaje Natural - Técnica de IA para interpretar texto humano |
| DIRESA | Dirección Regional de Salud de Tacna |
| API | Application Programming Interface - Interfaz de programación de aplicaciones |
| JWT | JSON Web Token - Estándar para autenticación |
| GDPR | General Data Protection Regulation - Reglamento de protección de datos |
| PWA | Progressive Web Application - Aplicación web progresiva |
| ML | Machine Learning - Aprendizaje automático |
| SaaS | Software as a Service - Software como servicio |
| CDN | Content Delivery Network - Red de distribución de contenido |

**1.4 Referencias**

* IEEE Std 1471-2000: Recommended Practice for Architecture Description of Software-Intensive Systems
* ISO/IEC 25010:2011: Systems and software Quality Requirements and Evaluation (SQuaRE)
* Norma Técnica de Salud para la Gestión de la Historia Clínica - MINSA Perú
* Reglamento de la Ley N° 29733, Ley de Protección de Datos Personales
* Guía de Interoperabilidad de Salud Digital - MINSA

**1.5 Visión General**

El sistema RespiCare-Tacna implementa una arquitectura de microservicios distribuida que permite escalabilidad, mantenibilidad y alta disponibilidad. La arquitectura se basa en principios de diseño orientado a servicios (SOA) y patrones de arquitectura moderna en la nube.

El sistema está diseñado para procesar descripciones de síntomas en lenguaje natural, analizarlos mediante modelos de PLN especializados en terminología médica respiratoria, y proporcionar evaluaciones de riesgo junto con recomendaciones personalizadas para los usuarios.

**2. Representación Arquitectónica**

**2.1 Escenarios**

La arquitectura del sistema RespiCare-Tacna está diseñada para soportar los siguientes escenarios principales:

Escenario 1: Evaluación de Síntomas en Tiempo Real

* Un paciente describe síntomas respiratorios en lenguaje natural
* El sistema procesa la información mediante PLN
* Se genera una evaluación de riesgo inmediata
* Se proporcionan recomendaciones basadas en el nivel de riesgo

Escenario 2: Consulta Médica Asistida

* Un médico accede al historial de evaluaciones del paciente
* Revisa las evaluaciones preliminares del sistema PLN
* Toma decisiones informadas basadas en datos adicionales
* Actualiza el perfil de riesgo del paciente

Escenario 3: Análisis Epidemiológico Regional

* Administradores consultan datos agregados y anonimizados
* Generan reportes de tendencias de enfermedades respiratorias
* Identifican patrones epidemiológicos en la región de Tacna
* Apoyan la toma de decisiones en salud pública

**2.2 Vista Lógica**

La vista lógica del sistema se organiza en capas arquitectónicas que separan las responsabilidades:

**Capa de Presentación:**

* Interfaces de usuario web (React.js)
* Aplicación móvil multiplataforma (React Native)
* API Gateway para enrutamiento y autenticación

**Capa de Servicios de Negocio:**

* Servicio de Gestión de Usuarios
* Servicio de Análisis PLN
* Servicio de Evaluación de Riesgo
* Servicio de Contenido Educativo
* Servicio de Notificaciones

**Capa de Datos:**

* Base de datos principal (PostgreSQL)
* Cache distribuido (Redis)
* Almacenamiento de archivos (AWS S3)
* Data Lake para análisis (AWS Data Lake)

**2.3 Vista del Proceso**

Los procesos del sistema siguen patrones asíncronos para garantizar responsividad:

**Proceso Principal de Evaluación:**

* Recepción de síntomas del usuario
* Preprocesamiento de texto en cola de mensajes
* Procesamiento PLN asíncrono
* Cálculo de score de riesgo
* Generación de recomendaciones
* Notificación al usuario y actualización del historial

**Proceso de Alertas Médicas:**

* Evaluación continua de scores de riesgo
* Identificación de casos de alta prioridad
* Activación de alertas automáticas
* Notificación a profesionales de salud
* Seguimiento de respuesta y escalamiento

**2.4 Vista del desarrollo**

La arquitectura de desarrollo se basa en principios de DevOps y CI/CD:

**Organización del Código:**

* Repositorios separados por servicio (microservicios)
* Uso de contenedores Docker para portabilidad
* Pipelines automatizados de integración continua
* Despliegue automatizado con Kubernetes

**Tecnologías de Desarrollo:**

* Backend: Node.js con TypeScript, Python para PLN
* Frontend Web: React.js con TypeScript
* Mobile: React Native
* Base de datos: PostgreSQL, Redis
* Message Queue: Apache Kafka
* Monitoreo: Prometheus + Grafana

**2.5 Vista Física**

La infraestructura física utiliza servicios en la nube para garantizar escalabilidad:

**Infraestructura Cloud (AWS):**

* Load Balancers para distribución de carga
* Auto Scaling Groups para escalabilidad automática
* Multi-AZ deployment para alta disponibilidad
* CDN (CloudFront) para optimización de contenido
* VPC con subredes públicas y privadas para seguridad

**Distribución Geográfica:**

* Región primaria: AWS South America (São Paulo)
* Backup y DR en AWS US East (N. Virginia)
* Edge locations para CDN distribuidos globalmente

**3. Objetivos y limitaciones arquitectónicas**

**3.1 Disponibilidad**

Objetivo: 99.9% de disponibilidad (menos de 8.76 horas de downtime al año)

**Estrategias implementadas:**

* Despliegue multi-zona de disponibilidad
* Load balancing con health checks automáticos
* Base de datos con replicación maestro-esclavo
* Implementación de circuit breakers para tolerancia a fallos
* Monitoreo proactivo con alertas automáticas

**Limitaciones:**

* Mantenimientos programados requieren ventanas de 2-4 horas mensuales
* Dependencia de conectividad a internet del usuario final
* Tiempo de recuperación de desastres: máximo 4 horas (RTO)

**3.2 Seguridad**

**Objetivos de Seguridad:**

* Cumplimiento con normativas de protección de datos médicos
* Cifrado end-to-end de datos sensibles
* Autenticación multifactor para usuarios administrativos
* Auditoría completa de accesos y modificaciones

**Medidas implementadas:**

* Cifrado AES-256 para datos en reposo
* TLS 1.3 para comunicaciones en tránsito
* Tokenización JWT con expiración automática
* Implementación de OAuth 2.0 con PKCE
* Segregación de redes con VPC y security groups
* Logging centralizado con SIEM

**Limitaciones:**

* Usuarios finales dependen de la seguridad de sus dispositivos
* Necesidad de balancear usabilidad con seguridad
* Cumplimiento regulatorio requiere auditorías externas periódicas

**3.3 Adaptabilidad**

**Objetivos de Adaptabilidad:**

* Capacidad de agregar nuevos modelos de PLN sin interrupción
* Soporte para múltiples idiomas y dialectos regionales
* Configuración flexible de reglas de negocio
* Integración con sistemas de salud existentes mediante APIs estándar

**Arquitectura Modular:**

* Microservicios independientes con APIs bien definidas
* Uso de patrones Adapter para integración externa
* Configuración externalizada mediante variables de entorno
* Plugin architecture para extensiones futuras

**Limitaciones:**

* Cambios en modelos de PLN requieren reentrenamiento
* Integración con sistemas legacy puede requerir desarrollos específicos
* Modificaciones regulatorias pueden requerir cambios arquitectónicos

**3.4 Rendimiento**

**Objetivos de Rendimiento:**

* Tiempo de respuesta < 2 segundos para evaluaciones PLN
* Soporte para 10,000 usuarios concurrentes
* Procesamiento de 1,000 evaluaciones por minuto
* Tiempo de carga inicial de aplicación < 3 segundos

**Optimizaciones implementadas:**

* Cache distribuido con Redis para datos frecuentes
* CDN para contenido estático y assets de aplicación
* Lazy loading en interfaces de usuario
* Conexiones de base de datos con pooling
* Compresión de APIs con gzip
* Optimización de queries con índices apropiados

**Limitaciones:**

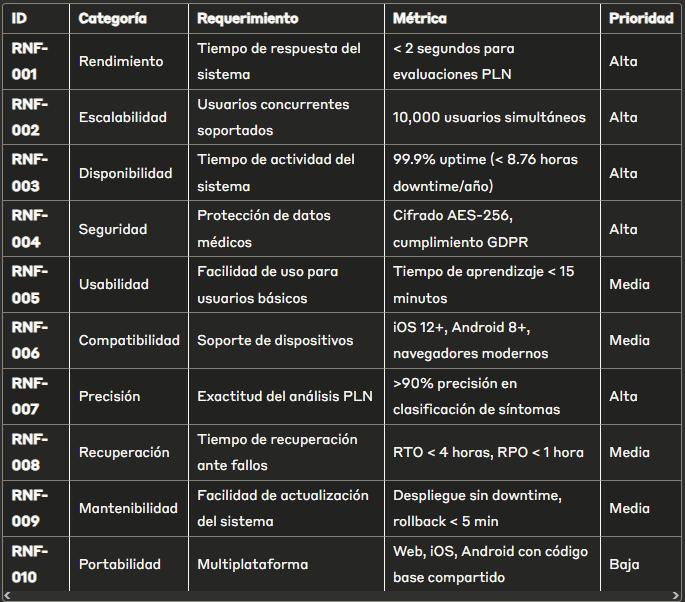
* Rendimiento degradado durante picos de tráfico extremo
* Latencia adicional en regiones geográficamente distantes
* Modelos de PLN complejos pueden requerir mayor tiempo de procesamiento

**4. Análisis de Requerimientos**

**4.1 Requerimientos funcionales**

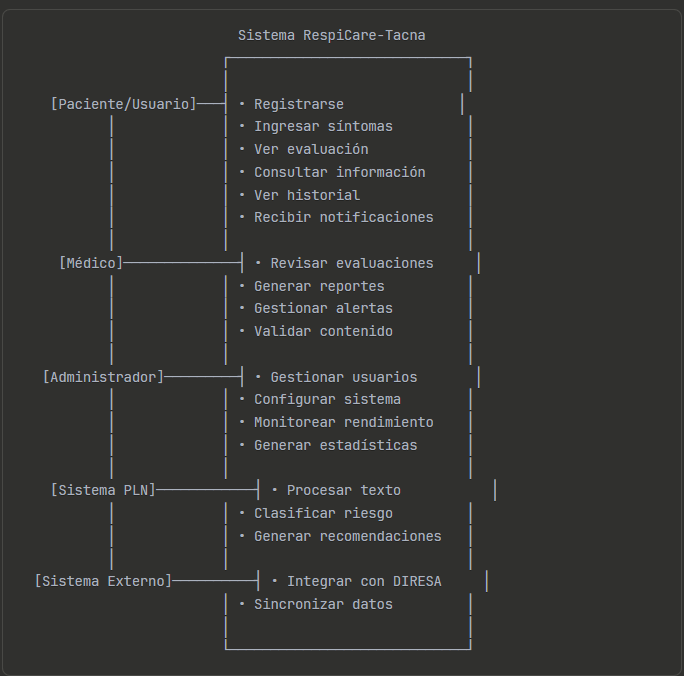
| ID | Requerimiento | Descripción | Prioridad |
| --- | --- | --- | --- |
| RF-001 | Gestión de Usuarios | Sistema completo de registro, autenticación y gestión de perfiles con roles diferenciados (Paciente, Médico, Administrador) | Alta |
| RF-002 | Análisis de Síntomas PLN | Procesamiento de lenguaje natural para evaluar descripciones de síntomas respiratorios en texto libre con precisión >90% | Alta |
| RF-003 | Sistema de Alertas | Generación automática de alertas médicas basadas en evaluaciones de riesgo con clasificación en Bajo, Medio y Alto | Alta |
| RF-004 | Módulo Educativo | Provisión de contenido educativo personalizado sobre salud respiratoria validado por profesionales médicos | Media |
| RF-005 | Generación de Reportes | Creación de reportes epidemiológicos y estadísticas de uso del sistema para análisis de salud pública | Media |
| RF-006 | Historial Médico | Almacenamiento seguro y consulta del historial de evaluaciones del usuario con capacidad de exportación | Media |
| RF-007 | Geolocalización | Localización de centros de salud cercanos basada en GPS del usuario con información de servicios disponibles | Baja |
| RF-008 | Notificaciones Push | Sistema de notificaciones para seguimiento, recordatorios y alertas médicas en tiempo real | Baja |
| RF-009 | Chat de Emergencia | Canal directo de comunicación con personal médico para casos de emergencia respiratoria | Media |
| RF-010 | Exportación de Datos | Funcionalidad para exportar historial médico personal en formato PDF para consultas médicas externas | Baja |

**4.2 Requerimientos no funcionales**



**5. Vistas de Caso de Uso**

**Diagrama General de Casos de Uso**



**Casos de Uso Detallados**

**CU-001: Análisis de Síntomas Respiratorios**

* Actor Principal: Paciente/Usuario
* Actores Secundarios: Sistema PLN
* Precondiciones: Usuario autenticado

**Flujo Principal:**

* Usuario describe síntomas en lenguaje natural
* Sistema valida y preprocesa el texto
* Servicio PLN analiza síntomas y extrae entidades médicas
* Sistema calcula score de riesgo
* Se generan recomendaciones personalizadas
* Usuario recibe evaluación con nivel de riesgo

Postcondiciones: Evaluación registrada en historial

**CU-002: Gestión de Alertas Médicas**

* Actor Principal: Sistema
* Actores Secundarios: Médico, Paciente
* Precondiciones: Evaluación de alto riesgo detectada

**Flujo Principal:**

* Sistema identifica caso de riesgo alto
* Genera alerta médica automática
* Notifica a médico asignado
* Envía recomendación urgente al paciente
* Registra alerta para seguimiento

**CU-003: Generación de Reportes Epidemiológicos**

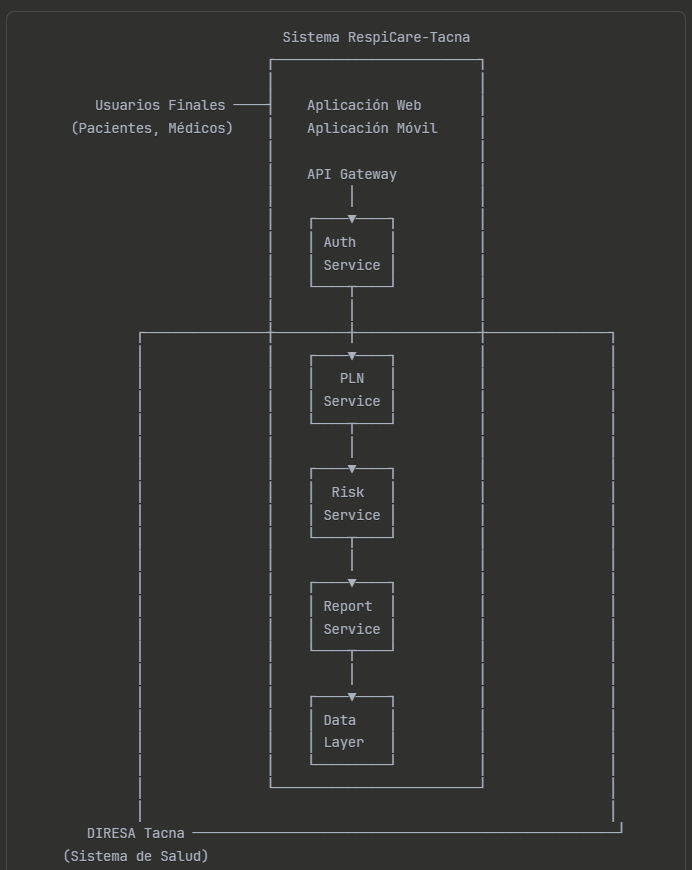
* Actor Principal: Administrador/Médico
* Precondiciones: Datos suficientes en sistema

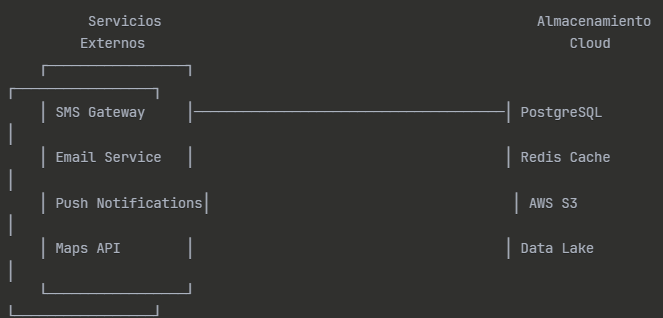
**Flujo Principal:**

* Usuario define parámetros de reporte
* Sistema consulta datos anonimizados
* Genera análisis estadístico
* Presenta visualizaciones interactivas
* Permite exportación en múltiples formatos

**6. Vista Lógica**

**6.1 Diagrama Contextual**





**Arquitectura de Microservicios**

La arquitectura lógica se organiza en los siguientes servicios principales:

**1. Authentication Service**

* Gestión de usuarios y sesiones
* Implementación de JWT y OAuth 2.0
* Control de acceso basado en roles

**2. NLP Processing Service**

* Análisis de texto con modelos especializados
* Extracción de entidades médicas
* Clasificación de síntomas

**3. Risk Assessment Service**

* Cálculo de scores de riesgo
* Generación de recomendaciones
* Activación de alertas

**4. Notification Service**

* Gestión de notificaciones push
* Envío de emails y SMS
* Logging de comunicaciones

**5. Reporting Service**

* Generación de reportes estadísticos
* Análisis epidemiológico
* Exportación de datos

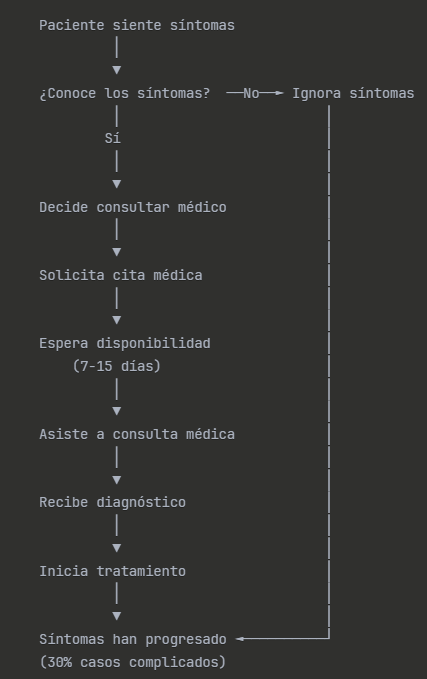
**6. Content Management Service**

* Gestión de contenido educativo
* Versionado de información médica
* Personalización de contenido

**7. Vista de Procesos**

**7.1 Diagrama de Proceso Actual**

Proceso Tradicional de Atención Respiratoria:

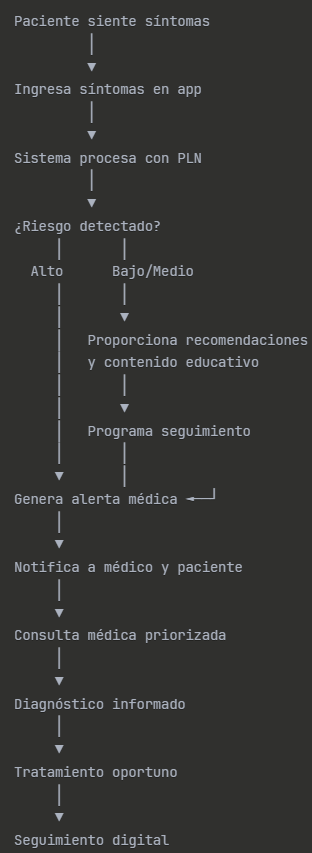


**Problemas Identificados:**

* Tiempo promedio de atención: 7-15 días
* 30% de abandono por demora en citas
* 70% de pacientes sin conocimiento de síntomas
* Detección tardía aumenta complicaciones
* Sobrecarga del sistema de salud

**7.2 Diagrama de Proceso Propuesto**

**Proceso Optimizado con RespiCare-Tacna:**



**Beneficios del Proceso Propuesto:**

* Evaluación inmediata (< 2 minutos)
* Educación preventiva continua
* Triaje automatizado inteligente
* Reducción de consultas no urgentes en 40%
* Detección temprana mejora pronóstico

**Procesos de Negocio Críticos**

**Proceso de Evaluación PLN:**

* Ingreso de Síntomas: Usuario describe síntomas en texto libre
* Preprocesamiento: Limpieza y normalización del texto
* Análisis PLN: Extracción de entidades médicas y clasificación
* Cálculo de Riesgo: Aplicación de algoritmos de scoring
* Generación de Respuesta: Recomendaciones personalizadas
* Registro: Almacenamiento en historial del usuario

**Proceso de Alertas Automáticas:**

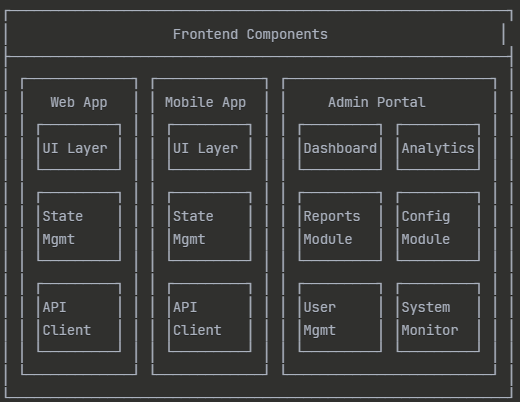
* Monitoreo Continuo: Evaluación de scores de riesgo
* Detección de Umbral: Identificación de casos críticos
* Activación de Alerta: Generación automática de notificación
* Enrutamiento: Asignación a personal médico disponible
* Seguimiento: Tracking de respuesta y escalamiento

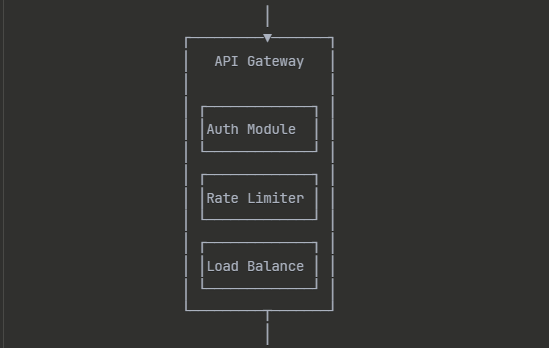
**8. Vista de Despliegue**

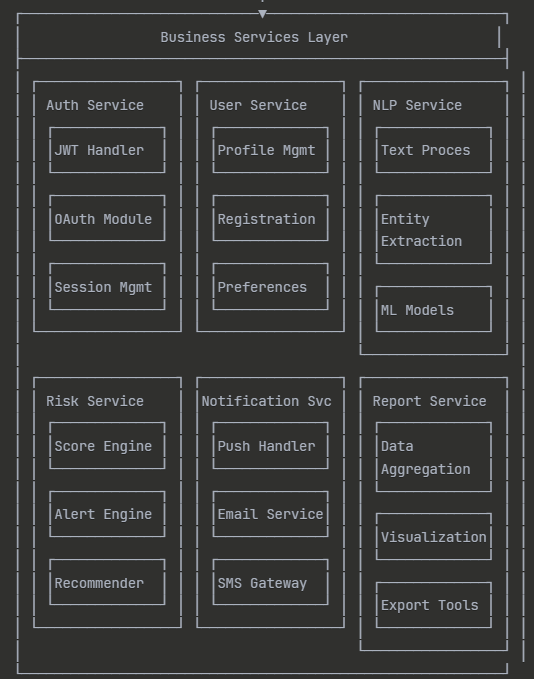
**8.1 Diagrama de Contenedor**

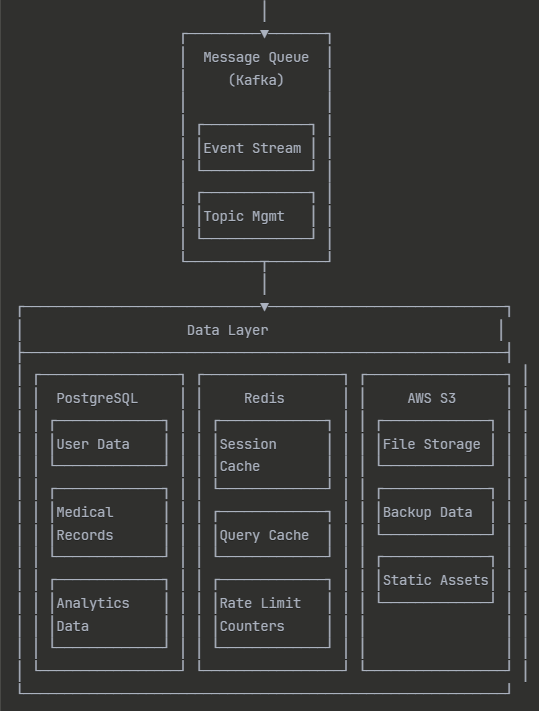
9. Vista de Implementación

9.1 Diagrama de Componentes









**Componentes Principales**

**Frontend Components:**

* UI Layer: Componentes reutilizables de interfaz usuario (React/React Native)
* State Management: Redux para manejo de estado global
* API Client: Axios con interceptors para manejo de autenticación
* Routing: React Router para navegación (web), React Navigation (mobile)
* Form Validation: Formik + Yup para validación de formularios

**Business Logic Components:**

* Authentication Module: JWT validation, OAuth 2.0, session management
* NLP Processing Engine: spaCy + custom medical models
* Risk Assessment Engine: Machine learning models para clasificación
* Notification Engine: Multi-channel notification system
* Report Generator: Data visualization con Chart.js/D3.js

**Data Access Components:**

* Database Adapters: Sequelize ORM para PostgreSQL
* Cache Manager: Redis client con clustering support
* File Handler: AWS SDK para S3 operations
* Message Broker: Kafka client para event streaming

**Patrones Arquitectónicos Implementados**

**1. Microservices Pattern**

* Servicios independientes con responsabilidades específicas
* Comunicación via REST APIs y message queues
* Deployment independiente de cada servicio

**2. CQRS (Command Query Responsibility Segregation)**

* Separación entre operaciones de lectura y escritura
* Optimización específica para cada tipo de operación
* Escalabilidad independiente de queries y commands

**3. Event-Driven Architecture**

* Comunicación asíncrona mediante eventos
* Desacoplamiento entre servicios
* Processing pipeline para análisis PLN

**4. Repository Pattern**

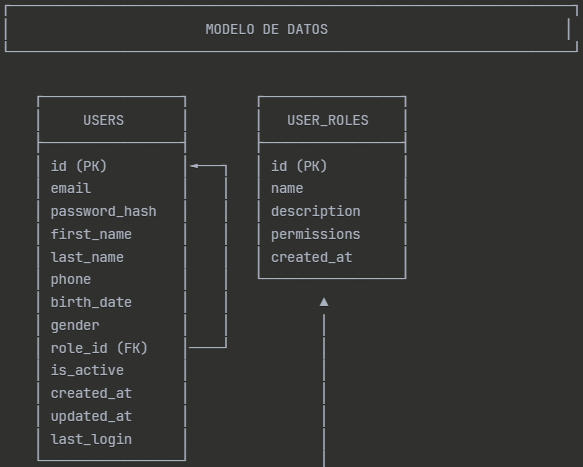
* Abstracción de acceso a datos
* Facilita testing con mock repositories
* Soporte para múltiples fuentes de datos

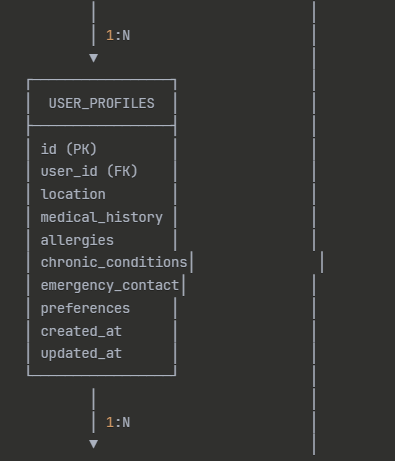
**5. Circuit Breaker Pattern**

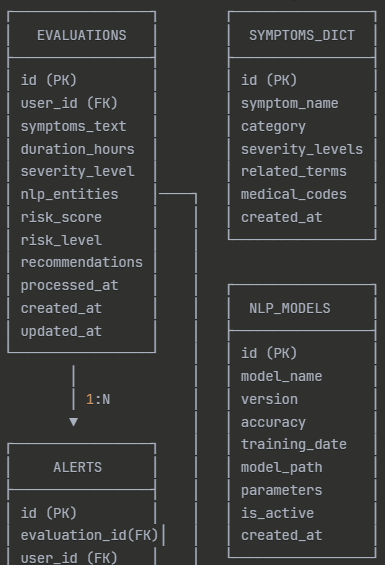
* Tolerancia a fallos en servicios externos
* Fallback strategies para alta disponibilidad
* Monitoring de health de servicios

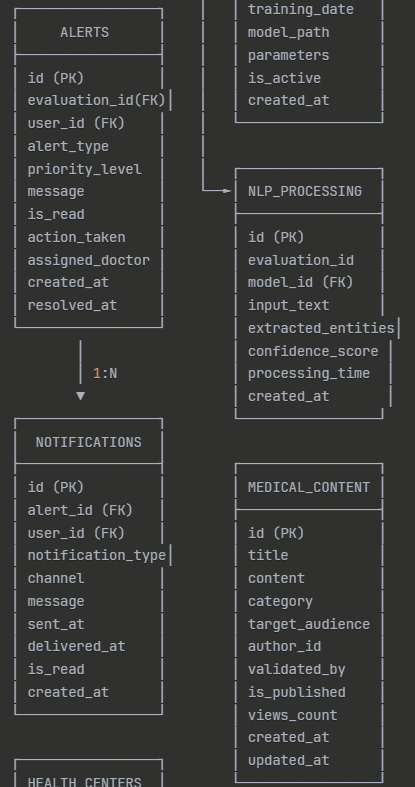
**10. Vista de Datos**

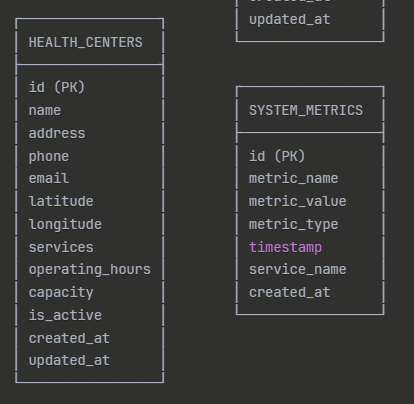
**10.1 Diagrama Entidad Relación**











**Especificación Detallada de Entidades**

**USERS Table:**

CREATE TABLE users (

id SERIAL PRIMARY KEY,

email VARCHAR(255) UNIQUE NOT NULL,

password\_hash VARCHAR(255) NOT NULL,

first\_name VARCHAR(100) NOT NULL,

last\_name VARCHAR(100) NOT NULL,

phone VARCHAR(20),

birth\_date DATE,

gender CHAR(1) CHECK (gender IN ('M', 'F', 'O')),

role\_id INTEGER REFERENCES user\_roles(id),

is\_active BOOLEAN DEFAULT true,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

last\_login TIMESTAMP

);

-- Índices para optimización

CREATE INDEX idx\_users\_email ON users(email);

CREATE INDEX idx\_users\_role ON users(role\_id);

CREATE INDEX idx\_users\_active ON users(is\_active);

**EVALUATIONS Table:**

CREATE TABLE evaluations (

id SERIAL PRIMARY KEY,

user\_id INTEGER REFERENCES users(id) ON DELETE CASCADE,

symptoms\_text TEXT NOT NULL,

duration\_hours INTEGER,

severity\_level INTEGER CHECK (severity\_level BETWEEN 1 AND 10),

nlp\_entities JSONB, -- Stored as JSON for flexibility

risk\_score DECIMAL(5,3) CHECK (risk\_score BETWEEN 0 AND 1),

risk\_level VARCHAR(10) CHECK (risk\_level IN ('LOW', 'MEDIUM', 'HIGH')),

recommendations TEXT,

processed\_at TIMESTAMP,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Índices para análisis y consultas frecuentes

CREATE INDEX idx\_evaluations\_user ON evaluations(user\_id);

CREATE INDEX idx\_evaluations\_risk ON evaluations(risk\_level);

CREATE INDEX idx\_evaluations\_date ON evaluations(created\_at);

CREATE INDEX idx\_evaluations\_nlp ON evaluations USING GIN (nlp\_entities);

**ALERTS Table:**

CREATE TABLE alerts (

id SERIAL PRIMARY KEY,

evaluation\_id INTEGER REFERENCES evaluations(id),

user\_id INTEGER REFERENCES users(id),

alert\_type VARCHAR(50) NOT NULL,

priority\_level INTEGER CHECK (priority\_level BETWEEN 1 AND 5),

message TEXT NOT NULL,

is\_read BOOLEAN DEFAULT false,

action\_taken TEXT,

assigned\_doctor INTEGER REFERENCES users(id),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

resolved\_at TIMESTAMP

);

CREATE INDEX idx\_alerts\_user ON alerts(user\_id);

CREATE INDEX idx\_alerts\_priority ON alerts(priority\_level);

CREATE INDEX idx\_alerts\_unread ON alerts(is\_read) WHERE is\_read = false;

**Estrategias de Optimización de Base de Datos**

**1. Particionamiento:**

-- Particionamiento por fecha para tabla de evaluaciones

CREATE TABLE evaluations\_2025\_01 PARTITION OF evaluations

FOR VALUES FROM ('2025-01-01') TO ('2025-02-01');

CREATE TABLE evaluations\_2025\_02 PARTITION OF evaluations

FOR VALUES FROM ('2025-02-01') TO ('2025-03-01');

**2. Vistas Materializadas para Reportes:**

CREATE MATERIALIZED VIEW mv\_daily\_stats AS

SELECT

DATE(created\_at) as evaluation\_date,

COUNT(\*) as total\_evaluations,

COUNT(CASE WHEN risk\_level = 'HIGH' THEN 1 END) as high\_risk\_cases,

AVG(risk\_score) as avg\_risk\_score

FROM evaluations

GROUP BY DATE(created\_at)

ORDER BY evaluation\_date DESC;

-- Refresh automático cada hora

CREATE OR REPLACE FUNCTION refresh\_daily\_stats()

RETURNS void AS $

BEGIN

REFRESH MATERIALIZED VIEW CONCURRENTLY mv\_daily\_stats;

END;

$ LANGUAGE plpgsql;

**3. Políticas de Retención de Datos:**

-- Política de eliminación de datos antiguos

CREATE OR REPLACE FUNCTION cleanup\_old\_data()

RETURNS void AS $

BEGIN

-- Eliminar evaluaciones mayores a 7 años

DELETE FROM evaluations

WHERE created\_at < CURRENT\_DATE - INTERVAL '7 years';

-- Archivar alertas resueltas mayores a 2 años

INSERT INTO alerts\_archive

SELECT \* FROM alerts

WHERE resolved\_at IS NOT NULL

AND resolved\_at < CURRENT\_DATE - INTERVAL '2 years';

DELETE FROM alerts

WHERE resolved\_at IS NOT NULL

AND resolved\_at < CURRENT\_DATE - INTERVAL '2 years';

END;

$ LANGUAGE plpgsql;

**Modelo de Datos NoSQL (Redis Cache)**

{

"user\_session:{user\_id}": {

"user\_id": 12345,

"session\_token": "jwt\_token\_here",

"last\_activity": "2025-01-15T10:30:00Z",

"permissions": ["read\_own\_data", "create\_evaluation"],

"ttl": 3600

},

"evaluation\_cache:{evaluation\_id}": {

"risk\_score": 0.75,

"risk\_level": "MEDIUM",

"recommendations": ["Monitorear síntomas", "Consulta en 48h"],

"processing\_time\_ms": 1250,

"ttl": 1800

},

"nlp\_model\_cache:{model\_version}": {

"model\_metadata": {

"accuracy": 0.92,

"last\_trained": "2025-01-01",

"feature\_count": 1024

},

"ttl": 86400

}

}

**11. Calidad**

**11.1 Escenario de Seguridad**

Escenario: Protección de Datos Médicos Sensibles

Estímulo: Un atacante intenta acceder a información médica de pacientes almacenada en el sistema

Fuente del Estímulo: Atacante externo con conocimientos técnicos avanzados

Ambiente: Sistema en producción con usuarios activos procesando datos médicos

Artefacto: Base de datos, APIs, y canales de comunicación

**Respuesta:**

* **Prevención:**

Cifrado AES-256 para datos en reposo

TLS 1.3 para datos en tránsito

Autenticación multi-factor obligatoria para administradores

Tokenización de datos sensibles

* **Detección:**

Monitoring de intentos de acceso no autorizado

Alertas automáticas por patrones sospechosos

Logging completo de todas las operaciones sensibles

* **Respuesta:**

Bloqueo automático de IP sospechosas

Notificación inmediata al equipo de seguridad

Revocación automática de tokens comprometidos

Activación de protocolo de respuesta a incidentes

**Medida de Respuesta:**

* Tiempo de detección: < 5 minutos
* Tiempo de bloqueo automático: < 30 segundos
* Zero data breach tolerance

**Controles de Seguridad Implementados:**

# Configuración de Seguridad

security\_controls:

encryption:

data\_at\_rest: "AES-256-GCM"

data\_in\_transit: "TLS 1.3"

key\_management: "AWS KMS"

authentication:

method: "JWT + OAuth 2.0"

mfa\_required: true

session\_timeout: "30 minutes"

password\_policy: "12+ chars, mixed case, numbers, symbols"

authorization:

rbac: true

principle: "least\_privilege"

resource\_isolation: true

monitoring:

siem: "AWS CloudTrail + GuardDuty"

log\_retention: "7 years"

anomaly\_detection: true

**11.2 Escenario de Usabilidad**

Escenario: Evaluación Rápida de Síntomas por Usuario con Baja Alfabetización Digital

Estímulo: Un paciente de 55 años con educación básica necesita evaluar síntomas respiratorios urgentemente usando la aplicación móvil

Fuente del Estímulo: Usuario final con experiencia tecnológica limitada en situación de estrés

Ambiente: Aplicación móvil en smartphone Android básico con conectividad 3G intermitente

Artefacto: Interfaz de usuario móvil, formulario de ingreso de síntomas, sistema de PLN

Respuesta Esperada:

**Simplicidad de Interfaz:**

* Proceso de evaluación completable en máximo 3 pasos
* Botones grandes (mínimo 44px) con iconos descriptivos
* Texto en lenguaje simple, sin términos técnicos
* Funcionalidad de voz-a-texto para describir síntomas

**Guía Contextual:**

* Tutorial interactivo de 60 segundos en primera apertura
* Sugerencias automáticas mientras el usuario escribe síntomas
* Validación en tiempo real con mensajes de ayuda amigables
* Ejemplos visuales de cómo describir síntomas comunes

**Tolerancia a Errores:**

* Capacidad de corrección sin pérdida de información previamente ingresada
* Función "deshacer" para acciones accidentales
* Guardado automático del progreso cada 10 segundos
* Modo offline básico para áreas con conectividad limitada

**Medidas de Usabilidad:**

* Tiempo promedio de completar evaluación: < 3 minutos
* Tasa de abandono del proceso: < 15%
* Satisfacción del usuario (escala 1-10): > 8.0
* Número de intentos para completar tarea: ≤ 1.5

**Métricas de Seguimiento:**

// Analytics de Usabilidad

const usabilityMetrics = {

task\_completion\_time: "< 180 seconds",

error\_rate: "< 5%",

user\_satisfaction\_score: "> 8.0/10",

help\_usage\_rate: "< 20%",

return\_user\_rate: "> 70%"

};

**11.3 Escenario de Adaptabilidad**

Escenario: Incorporación de Nuevo Modelo de PLN para Detección de COVID-19

Estímulo: Aparición de nueva variante respiratoria que requiere actualización del modelo de análisis PLN

Fuente del Estímulo: Equipo médico y científico que identifica patrones sintomáticos específicos

Ambiente: Sistema en producción procesando 500+ evaluaciones diarias con modelo PLN v2.1

Artefacto: Servicio de PLN, base de datos de entrenamiento, pipeline de procesamiento

Respuesta Requerida:

**Actualización Sin Interrupción:**

* Despliegue de modelo PLN v2.2 usando estrategia Blue-Green
* Procesamiento paralelo con ambos modelos durante período de transición
* Comparación automática de resultados entre versiones
* Rollback automático si precisión < 85%

**Reentrenamiento Dinámico:**

* Ingesta de nuevos datos sintomáticos validados por médicos
* Pipeline de reentrenamiento automatizado con MLOps
* Validación cruzada con dataset de reserva (20% datos)
* A/B testing con porcentaje controlado de usuarios (10%)

**Configuración Flexible:**

* Parámetros del modelo ajustables vía variables de entorno
* Pesos de características modificables sin redespliegue
* Umbrales de riesgo configurables por región geográfica
* Reglas de negocio externalizadas en archivo de configuración

**Arquitectura de Adaptabilidad:**

# Configuración de Modelo PLN

nlp\_model\_config:

version: "2.2.0"

deployment\_strategy: "blue\_green"

rollback\_threshold: 0.85

models:

primary:

path: "/models/respiratory\_v2.2"

accuracy\_threshold: 0.90

confidence\_threshold: 0.75

fallback:

path: "/models/respiratory\_v2.1"

accuracy\_threshold: 0.85

features:

symptom\_weights:

fever: 0.3

cough: 0.4

breathing\_difficulty: 0.8

chest\_pain: 0.6

risk\_thresholds:

low: 0.3

medium: 0.6

high: 0.8

retraining:

schedule: "weekly"

min\_samples: 1000

validation\_split: 0.2

auto\_deploy: false

**Medidas de Adaptabilidad:**

* Tiempo de despliegue de nuevo modelo: < 15 minutos
* Tiempo de rollback: < 5 minutos
* Downtime durante actualización: 0 segundos
* Tiempo de reentrenamiento: < 4 horas

**11.4 Escenario de Disponibilidad**

Escenario: Fallo Completo del Data Center Principal Durante Pico de Demanda

Estímulo: Interrupción total del servicio en la región AWS primary (us-east-1) debido a desastre natural

Fuente del Estímulo: Falla de infraestructura externa (corte eléctrico masivo, desastre natural)

Ambiente: Viernes por la tarde con 2,000 usuarios activos y 150 evaluaciones siendo procesadas simultáneamente

Artefacto: Toda la infraestructura de producción en región primaria (compute, storage, database)

Respuesta del Sistema:

**Detección Automática:**

* Health checks fallan en múltiples servicios simultáneamente
* Monitoring detecta falla regional en < 2 minutos
* Sistema de alertas notifica equipo de operaciones automáticamente

**Failover Automático:**

* Route 53 redirige tráfico a región secundaria (sa-east-1)
* Load balancers activan instancias standby en región backup
* Base de datos activa réplica de lectura como master temporal
* Cache Redis se reconstruye desde backup más reciente

**Recuperación de Datos:**

* Recovery Point Objective (RPO): < 15 minutos de pérdida de datos
* Recovery Time Objective (RTO): < 30 minutos para restauración completa
* Sincronización automática una vez restaurada región primaria

**Arquitectura de Alta Disponibilidad:**

# Configuración Multi-Región

disaster\_recovery:

primary\_region: "us-east-1"

backup\_region: "sa-east-1"

database:

replication: "cross\_region\_async"

backup\_frequency: "every\_15\_minutes"

point\_in\_time\_recovery: true

storage:

replication: "cross\_region"

versioning: enabled

backup\_retention: "30\_days"

compute:

auto\_scaling:

min\_instances: 2

max\_instances: 20

target\_cpu: 70%

health\_checks:

interval: "30\_seconds"

timeout: "5\_seconds"

unhealthy\_threshold: 2

monitoring:

uptime\_target: "99.9%"

max\_downtime\_monthly: "43\_minutes"

alert\_thresholds:

response\_time: "> 3\_seconds"

error\_rate: "> 1%"

availability: "< 99%"

**Métricas de Disponibilidad Garantizadas:**

* Uptime SLA: 99.9% (máximo 8.76 horas downtime/año)
* RTO (Recovery Time Objective): < 30 minutos
* RPO (Recovery Point Objective): < 15 minutos
* MTTR (Mean Time To Recovery): < 45 minutos
* Tiempo de detección de fallas: < 2 minutos

**11.5 Escenario de Rendimiento bajo Carga Extrema**

Escenario: Pico de Tráfico Durante Emergencia Sanitaria Regional

Estímulo: Brote epidemiológico en Tacna genera 10x el tráfico normal con 50,000 evaluaciones simultáneas

Fuente del Estímulo: Emergencia de salud pública que impulsa uso masivo del sistema por parte de la población

Ambiente: Sistema experimentando carga 1000% superior al promedio normal durante 6 horas continuas

Artefacto: APIs, base de datos, servicios de PLN, infrastructure cloud

Respuesta del Sistema:

**Escalamiento Automático:**

* Auto Scaling Groups activan instancias adicionales automáticamente
* Load balancers distribuyen carga entre 50+ instancias de aplicación
* Database read réplicas se escalan de 2 a 8 instancias
* PLN service se escala horizontalmente con contenedores adicionales

**Optimización de Recursos:**

* Cache hit ratio optimizado al 95% para reducir carga en BD
* Compression activado para reducir bandwidth en 60%
* CDN cache TTL extendido para contenido estático
* Connection pooling optimizado para manejar 10,000+ conexiones concurrentes

**Degradación Gradual:**

* Funcionalidades no críticas desactivadas temporalmente
* Reportes complejos diferidos a horarios de menor carga
* Notificaciones agrupadas para reducir overhead
* Timeout ajustados dinámicamente para prevenir cascading failures

**Configuración de Rendimiento:**

# Configuración para Carga Extrema

performance\_config:

auto\_scaling:

target\_cpu\_utilization: 60%

scale\_out\_cooldown: 300 # seconds

scale\_in\_cooldown: 600 # seconds

min\_instances: 5

max\_instances: 100

database:

connection\_pool:

min\_connections: 10

max\_connections: 200

connection\_timeout: 30

read\_replicas:

min\_replicas: 2

max\_replicas: 10

auto\_scaling: true

cache:

redis\_cluster:

nodes: 6

memory\_per\_node: "8GB"

eviction\_policy: "allkeys-lru"

ttl\_default: 3600

rate\_limiting:

requests\_per\_minute: 100

burst\_capacity: 200

grace\_period: 300 # seconds during high load

circuit\_breaker:

failure\_threshold: 10

timeout: 60 # seconds

recovery\_timeout: 30 # seconds

**Métricas de Rendimiento Objetivo:**

* Tiempo de respuesta promedio: < 2 segundos (incluso bajo carga extrema)
* Throughput máximo: 1,000 evaluaciones PLN por minuto
* Concurrent users soportados: 50,000+
* Error rate durante picos: < 0.5%
* Tiempo de escalamiento: < 5 minutos para duplicar capacidad

**Plan de Contingencia:**

Nivel 1 - Carga Alta (5x normal):

* Escalamiento automático estándar
* Monitoreo intensivo activado

Nivel 2 - Carga Extrema (10x normal):

* Escalamiento agresivo activado
* Funcionalidades no esenciales deshabilitadas
* Equipo de operaciones alertado

Nivel 3 - Saturación (>15x normal):

* Modo de emergencia activado
* Queue system implementado para evaluaciones
* Comunicación pública sobre demoras temporales

**Monitoreo y Alertas:**

# Sistema de Monitoreo

monitoring:

metrics:

- response\_time\_percentile\_95: "> 3 seconds"

- error\_rate: "> 1%"

- cpu\_utilization: "> 80%"

- memory\_utilization: "> 85%"

- database\_connections: "> 150"

- queue\_depth: "> 1000"

alerts:

critical:

- service\_unavailable

- database\_down

- high\_error\_rate

warning:

- high\_response\_time

- scaling\_event

- resource\_exhaustion

dashboards:

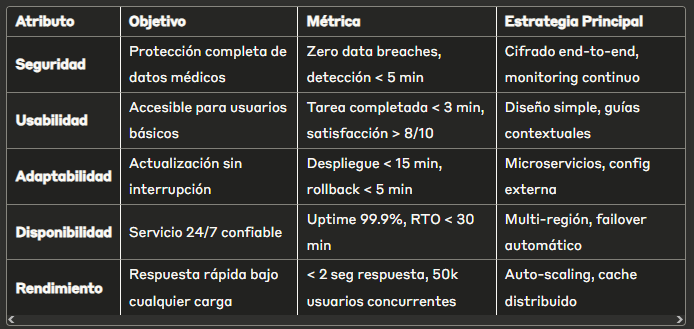
- real\_time\_performance

- infrastructure\_health

- business\_metrics

- user\_experience

**Resumen de Atributos de Calidad:**



Esta arquitectura de calidad asegura que el sistema RespiCare-Tacna pueda cumplir con los más altos estándares de servicio médico digital, garantizando seguridad, confiabilidad y usabilidad para todos los usuarios en la región de Tacna.