# Documento Técnico: Arquitectura del Sistema PIGEM

## 1. Introducción y Visión General

La **Plataforma Integral de Gestión de Evacuaciones Multiamenaza (PIGEM)** constituye la materialización final del proyecto. No se concibe como un software monolítico, sino como un **ecosistema tecnológico modular e interoperable**, diseñado para transformar la capacidad de planificación, simulación y ejecución de evacuaciones masivas en Chile.

### 1.1 Principios Arquitectónicos Fundamentales

**Tabla 1: Principios de Diseño Arquitectónico de PIGEM**

| **Principio** | **Definición** | **Implementación** | **Beneficio** | **Métrica de Éxito** |
| --- | --- | --- | --- | --- |
| **Modularidad** | Separación clara de responsabilidades | Microservicios independientes | Mantenibilidad, escalabilidad | Acoplamiento <20%, cohesión >80% |
| **Interoperabilidad** | Comunicación estándar entre sistemas | APIs REST, protocolos OGC | Integración con sistemas existentes | >95% compatibilidad con estándares |
| **Escalabilidad** | Capacidad de crecimiento horizontal | Contenedores, orquestación | Manejo de cargas variables | Escalado automático 0-100K usuarios |
| **Disponibilidad** | Operación continua 24/7 | Redundancia, failover | Confiabilidad operacional | Uptime >99.9% |
| **Seguridad** | Protección de datos y accesos | Autenticación, cifrado, auditoría | Confidencialidad e integridad | 0 brechas críticas |
| **Usabilidad** | Interfaces adaptativas y accesibles | UX centrado en usuario | Adopción efectiva | SUS >80, tiempo aprendizaje <30min |

La filosofía de diseño se aleja de las “islas tecnológicas” que caracterizan la gestión de información para desastres en el país. En su lugar, PIGEM propone una arquitectura de servicios integrada, donde cada componente cumple una función específica pero se comunica con los demás a través de interfaces y estándares abiertos, garantizando la escalabilidad, mantenibilidad y, fundamentalmente, la interoperabilidad entre instituciones.

### 1.2 Contexto y Problemática Nacional

**Tabla 2: Análisis del Panorama Tecnológico Nacional en Gestión de Emergencias**

| **Institución** | **Sistemas Actuales** | **Fortalezas** | **Limitaciones** | **Integración con PIGEM** |
| --- | --- | --- | --- | --- |
| **SENAPRED** | SAE, SIAC | Cobertura nacional, alertas masivas | Datos fragmentados, sin simulación | API para alertas, datos históricos |
| **SHOA** | SNAM, cartas tsunami | Expertise técnico, datos precisos | Formato propietario, acceso limitado | Servicios OGC, metadatos estándar |
| **SERNAGEOMIN** | RNVV, SIGMIN | Datos geológicos detallados | Sistemas aislados, formatos diversos | Ingesta automática, normalización |
| **INE** | Censo, cartografía | Datos demográficos oficiales | Actualización decenal, granularidad limitada | APIs de consulta, interpolación |
| **Municipalidades** | Sistemas locales | Conocimiento territorial específico | Heterogeneidad tecnológica | Conectores estándar, capacitación |

El objetivo de esta arquitectura es proporcionar una herramienta que soporte el ciclo completo de la gestión de emergencias, desde la planificación estratégica a largo plazo (análisis de escenarios “what-if”) hasta la coordinación táctica en tiempo real durante una crisis.

### 1.3 Arquitectura de Referencia

┌─────────────────────────────────────────────────────────────────────────────┐  
│ ECOSISTEMA PIGEM │  
├─────────────────────────────────────────────────────────────────────────────┤  
│ │  
│ ┌─────────────────┐ ┌─────────────────┐ ┌─────────────────────────────┐ │  
│ │ USUARIOS │ │ INTERFACES │ │ APLICACIONES │ │  
│ │ │ │ │ │ │ │  
│ │ • Autoridades │◀─│ • Web Apps │◀─│ • Dashboard Ejecutivo │ │  
│ │ • Técnicos │ │ • Mobile Apps │ │ • Simulador Avanzado │ │  
│ │ • Ciudadanos │ │ • APIs Públicas │ │ • Centro de Control │ │  
│ │ • Investigadores│ │ • Widgets │ │ • Portal Ciudadano │ │  
│ └─────────────────┘ └─────────────────┘ └─────────────────────────────┘ │  
│ ▲ │  
│ │ │  
├────────────────────────────────┼───────────────────────────────────────────┤  
│ CAPA DE SERVICIOS │  
│ │ │  
│ ┌─────────────────┐ ┌─────────────────┐ ┌─────────────────────────────┐ │  
│ │ AUTENTICACIÓN │ │ AUTORIZACIÓN │ │ NOTIFICACIONES │ │  
│ │ & SESIONES │ │ & PERFILES │ │ & ALERTAS │ │  
│ └─────────────────┘ └─────────────────┘ └─────────────────────────────┘ │  
│ │ │  
├────────────────────────────────┼───────────────────────────────────────────┤  
│ NÚCLEO FUNCIONAL │  
│ │ │  
│ ┌─────────────────┐ ┌─────────────────┐ ┌─────────────────────────────┐ │  
│ │ DATOS GEOESP. │ │ SIMULACIÓN │ │ VISUALIZACIÓN │ │  
│ │ (MCU) │ │ HÍBRIDA │ │ ADAPTATIVA │ │  
│ │ │ │ (ABM-DES) │ │ │ │  
│ │ • Ingesta │ │ • Agentes │ │ • Mapas interactivos │ │  
│ │ • Validación │ │ • Recursos │ │ • Dashboards adaptativos │ │  
│ │ • Servicios OGC │ │ • HPC │ │ • Reportes automáticos │ │  
│ └─────────────────┘ └─────────────────┘ └─────────────────────────────┘ │  
│ │ │  
├────────────────────────────────┼───────────────────────────────────────────┤  
│ CAPA DE INFRAESTRUCTURA │  
│ │ │  
│ ┌─────────────────┐ ┌─────────────────┐ ┌─────────────────────────────┐ │  
│ │ ALMACENAMIENTO │ │ PROCESAMIENTO │ │ COMUNICACIONES │ │  
│ │ │ │ │ │ │ │  
│ │ • PostgreSQL │ │ • Kubernetes │ │ • Load Balancers │ │  
│ │ • PostGIS │ │ • Docker │ │ • Message Queues │ │  
│ │ • Redis Cache │ │ • GPU Clusters │ │ • WebSockets │ │  
│ │ • File Storage │ │ • Auto-scaling │ │ • CDN │ │  
│ └─────────────────┘ └─────────────────┘ └─────────────────────────────┘ │  
└─────────────────────────────────────────────────────────────────────────────┘

*Figura 1: Arquitectura de referencia del ecosistema PIGEM*

## 1.4 Fundamentación Científica y Componente de Investigación

### 1.4.1 Descripción del Problema Arquitectónico

La gestión de desastres en Chile enfrenta una paradoja tecnológica fundamental: mientras que las instituciones nacionales poseen capacidades técnicas avanzadas individualmente, la fragmentación arquitectónica de sus sistemas impide una respuesta coordinada y eficiente ante emergencias multiamenaza. Esta problemática se manifiesta en tres dimensiones críticas:

**Dimensión Técnica:** La heterogeneidad de arquitecturas de software (monolíticas vs. distribuidas), lenguajes de programación (Java, Python, C#, PHP) y paradigmas de datos (relacionales, NoSQL, archivos planos) genera una complejidad integrativa que requiere soluciones ad-hoc costosas y frágiles. La ausencia de estándares arquitectónicos comunes resulta en interfaces propietarias que impiden la interoperabilidad automática.

**Dimensión Operacional:** Los sistemas actuales operan en silos funcionales, donde cada institución optimiza localmente sus procesos sin considerar las dependencias sistémicas. Esta fragmentación operacional genera latencias críticas durante emergencias, donde la coordinación entre SENAPRED, SHOA, SERNAGEOMIN y municipalidades requiere comunicación manual y propensa a errores.

**Dimensión Escalabilidad:** Las arquitecturas monolíticas predominantes carecen de capacidades de escalamiento horizontal, limitando severamente la capacidad de respuesta durante eventos de alta demanda. La falta de patrones de diseño para alta disponibilidad y tolerancia a fallos compromete la continuidad operacional en momentos críticos.

### 1.4.2 Hipótesis de Investigación Arquitectónica

**Hipótesis Principal:** La implementación de una arquitectura de microservicios basada en principios de Domain-Driven Design (DDD), que integre estándares de interoperabilidad (OGC, ISO) con patrones de comunicación asíncronos y estrategias de escalamiento horizontal, mejorará significativamente la capacidad de respuesta, escalabilidad y mantenibilidad del sistema de gestión de evacuaciones multiamenaza, comparado con arquitecturas monolíticas tradicionales.

**Hipótesis Secundarias:**

1. **Hipótesis de Interoperabilidad:** La adopción de estándares abiertos (OGC, ISO 19115) en la capa de servicios reducirá en >60% el tiempo de integración con sistemas externos existentes.
2. **Hipótesis de Escalabilidad:** La arquitectura de microservicios con orquestación basada en contenedores permitirá escalamiento automático que mantenga latencias <100ms bajo cargas de hasta 10,000 usuarios concurrentes.
3. **Hipótesis de Mantenibilidad:** La modularización basada en dominios de negocio reducirá en >40% el tiempo de desarrollo y despliegue de nuevas funcionalidades.

### 1.4.3 Pregunta de Investigación Arquitectónica

**Pregunta Principal:** ¿Cómo debe diseñarse una arquitectura de software distribuida que integre efectivamente sistemas heterogéneos de gestión de desastres, garantizando interoperabilidad, escalabilidad y alta disponibilidad, mientras mantiene la complejidad operacional en niveles manejables para organizaciones con recursos técnicos limitados?

**Preguntas Secundarias:**

1. **Interoperabilidad:** ¿Qué patrones arquitectónicos y estándares técnicos son más efectivos para garantizar la comunicación fluida entre sistemas legacy y nuevos componentes?
2. **Escalabilidad:** ¿Cuáles son las estrategias óptimas de particionamiento de datos y distribución de carga que permiten mantener performance consistente durante picos de demanda?
3. **Usabilidad:** ¿Cómo puede la arquitectura del sistema facilitar interfaces adaptativas que se ajusten automáticamente a diferentes perfiles de usuario y contextos operacionales?
4. **Mantenibilidad:** ¿Qué estrategias de modularización y versionado permiten la evolución continua del sistema sin interrumpir servicios críticos?

### 1.4.4 Metodología de Validación Arquitectónica

La validación de las hipótesis arquitectónicas se realizará mediante:

**Prototipado Incremental:** Desarrollo de prototipos funcionales que implementen progresivamente cada patrón arquitectónico propuesto, permitiendo validación empírica de métricas de rendimiento, escalabilidad y mantenibilidad.

**Benchmarking Comparativo:** Evaluación sistemática de la arquitectura propuesta versus arquitecturas monolíticas tradicionales, utilizando métricas objetivas como latencia, throughput, tiempo de desarrollo y costos operacionales.

**Casos de Estudio Reales:** Implementación piloto en escenarios reales con municipalidades seleccionadas, permitiendo validación de usabilidad, adopción y efectividad operacional.

**Análisis de Complejidad:** Evaluación de métricas de complejidad arquitectónica (coupling, cohesión, complejidad ciclomática) para validar la hipótesis de mantenibilidad.

## 2. Arquitectura Modular por Componentes

PIGEM se estructura en cuatro componentes principales que funcionan de manera coordinada, cada uno implementando patrones de diseño específicos para maximizar la cohesión interna y minimizar el acoplamiento externo.

### 2.1 Componente 1: Plataforma de Datos Geoespaciales (El Fundamento)

Este es el pilar sobre el que se construye todo el sistema. Su núcleo es el **Metamodelo Cartográfico Unificado (MCU)**.

#### 2.1.1 Arquitectura Interna del MCU

**Tabla 3: Microservicios del Componente de Datos Geoespaciales**

| **Microservicio** | **Responsabilidad** | **Tecnología** | **API** | **Dependencias** | **SLA** |
| --- | --- | --- | --- | --- | --- |
| **Data Ingestion** | Ingesta automática de datos | Python + Celery | REST | File Storage, Validation | 99.5% |
| **Metadata Validator** | Validación ISO 19115 | Python + Pydantic | REST | None | 99.9% |
| **Spatial Processor** | Transformaciones geoespaciales | GDAL + PostGIS | REST | Database | 99.5% |
| **OGC Services** | Servicios web estándar | GeoServer + Java | WMS/WFS/WCS | Database, Cache | 99.9% |
| **Catalog Service** | Catálogo de metadatos | PyCSW + Python | CSW | Database | 99.5% |
| **Quality Assurance** | Control de calidad automático | Python + ML | REST | Validation, ML Models | 99.0% |

#### 2.1.2 Patrones de Diseño Implementados

**Tabla 4: Patrones de Diseño en el MCU**

| **Patrón** | **Aplicación** | **Beneficio** | **Implementación** |
| --- | --- | --- | --- |
| **Repository** | Acceso a datos geoespaciales | Abstracción de persistencia | Interface común para múltiples fuentes |
| **Factory** | Creación de procesadores | Flexibilidad en tipos de datos | Factory para parsers por formato |
| **Strategy** | Algoritmos de validación | Intercambiabilidad | Estrategias por tipo de geometría |
| **Observer** | Notificaciones de cambios | Desacoplamiento | Eventos de actualización de datos |
| **Chain of Responsibility** | Pipeline de procesamiento | Extensibilidad | Cadena de transformaciones |

* **Función Principal**: Actuar como un “lenguaje común” para toda la información geoespacial. Centraliza, estandariza y armoniza datos heterogéneos de fuentes diversas (SENAPRED, SHOA, INE, Gobiernos Regionales, sensores IoT, etc.).

#### 2.1.3 Flujo de Datos y Procesamiento

┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐  
│ FUENTES DATOS │ │ INGESTA Y │ │ VALIDACIÓN Y │  
│ EXTERNAS │ │ DETECCIÓN │ │ NORMALIZACIÓN │  
│ │ │ │ │ │  
│ • SHOA (FTP) │───▶│ • File Watcher │───▶│ • Schema Check │  
│ • INE (API) │ │ • API Polling │ │ • Geometry Fix │  
│ • SERNAGEOMIN │ │ • Format Detect │ │ • CRS Transform │  
│ • Municipios │ │ • Metadata Ext. │ │ • Encoding Fix │  
└─────────────────┘ └─────────────────┘ └─────────────────┘  
 │  
 ▼  
┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐  
│ PUBLICACIÓN │ │ ALMACENAMIENTO │ │ ENRIQUECIMIENTO │  
│ SERVICIOS │ │ PERSISTENTE │ │ Y INDEXACIÓN │  
│ │ │ │ │ │  
│ • WMS/WFS/WCS │◀───│ • PostGIS DB │◀───│ • Spatial Index │  
│ • CSW Catalog │ │ • File Storage │ │ • Metadata Gen. │  
│ • REST APIs │ │ • Redis Cache │ │ • Quality Score │  
│ • GraphQL │ │ • Backup │ │ • Relationships │  
└─────────────────┘ └─────────────────┘ └─────────────────┘

*Figura 2: Flujo de datos en el componente MCU*

* **Estándares Clave**:
  + **ISO 19115/19139**: Para la estructura de los metadatos.
  + **Estándares OGC**: Publica los datos estandarizados a través de servicios web interoperables.

### 2.2 Componente 2: Núcleo de Simulación Avanzada (El Cerebro)

Este es el motor computacional del sistema, donde reside la principal innovación del proyecto: el **modelo de simulación híbrido (ABM-DES)**.

#### 2.2.1 Arquitectura del Motor de Simulación

**Tabla 5: Módulos del Núcleo de Simulación**

| **Módulo** | **Función** | **Tecnología** | **Paralelización** | **Rendimiento** | **Dependencias** |
| --- | --- | --- | --- | --- | --- |
| **ABM Engine** | Simulación de agentes | C++ + OpenMP | CPU Multi-core | 100K agentes/min | Spatial Index |
| **DES Engine** | Eventos discretos | Python + SimPy | Limitada | 10K eventos/sec | ABM Engine |
| **SFM Calculator** | Fuerzas sociales | CUDA + C++ | GPU Masiva | 1M cálculos/sec | ABM Engine |
| **Pathfinding** | Cálculo de rutas | A\* + Dijkstra | CPU Multi-core | 1K rutas/sec | Graph DB |
| **Scenario Manager** | Gestión de escenarios | Python + FastAPI | Horizontal | 100 escenarios | Database |
| **Results Processor** | Análisis de resultados | Python + Pandas | CPU Multi-core | 1GB datos/min | File Storage |

#### 2.2.2 Arquitectura de Cómputo Híbrida

**Tabla 6: Distribución de Carga Computacional**

| **Operación** | **Hardware** | **Justificación** | **Paralelización** | **Escalabilidad** |
| --- | --- | --- | --- | --- |
| **Cálculo SFM** | GPU (CUDA) | Operaciones paralelas masivas | SIMT | Lineal con GPUs |
| **Lógica de Agentes** | CPU (OpenMP) | Lógica compleja, ramificaciones | Threads | Lineal con cores |
| **Gestión DES** | CPU (Single-thread) | Sincronización de eventos | Limitada | Vertical |
| **I/O de Datos** | SSD + RAM | Acceso rápido a datos | Async I/O | Con memoria |
| **Visualización** | GPU (WebGL) | Renderizado en tiempo real | Shaders | Con resolución |

* **Función Principal**: Simular evacuaciones masivas de la manera más realista posible, considerando tanto el comportamiento humano como las limitaciones físicas del entorno.

#### 2.2.3 Pipeline de Simulación

┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐  
│ CONFIGURACIÓN │ │ INICIALIZACIÓN │ │ EJECUCIÓN │  
│ ESCENARIO │ │ SISTEMA │ │ SIMULACIÓN │  
│ │ │ │ │ │  
│ • Parámetros │───▶│ • Crear Agentes │───▶│ • Loop Principal│  
│ • Población │ │ • Cargar Mapa │ │ • ABM Step │  
│ • Amenazas │ │ • Init DES │ │ • DES Step │  
│ • Recursos │ │ • Setup GPU │ │ • Sync & Update │  
└─────────────────┘ └─────────────────┘ └─────────────────┘  
 │  
 ▼  
┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐  
│ REPORTES Y │ │ POST-PROCESO │ │ MONITOREO │  
│ VISUALIZACIÓN │ │ RESULTADOS │ │ TIEMPO REAL │  
│ │ │ │ │ │  
│ • Dashboards │◀───│ • Agregación │◀───│ • Métricas Live │  
│ • Exportación │ │ • Estadísticas │ │ • Progress Bar │  
│ • Comparación │ │ • Validación │ │ • Performance │  
│ • Recomendac. │ │ • Archivado │ │ • Error Handle │  
└─────────────────┘ └─────────────────┘ └─────────────────┘

*Figura 3: Pipeline de ejecución de simulaciones*

### 2.3 Componente 3: Interfaces de Usuario Multimodales (La Cara Visible)

Este componente se encarga de que los usuarios puedan interactuar con la plataforma de manera intuitiva y efectiva.

#### 2.3.1 Arquitectura Frontend

**Tabla 7: Stack Tecnológico del Frontend**

| **Capa** | **Tecnología** | **Versión** | **Propósito** | **Alternativas Consideradas** |
| --- | --- | --- | --- | --- |
| **Framework UI** | React | 18.2+ | SPA moderna, ecosistema | Vue.js, Angular, Svelte |
| **Estado Global** | Redux Toolkit | 1.9+ | Gestión de estado predecible | Zustand, Recoil, Context API |
| **Mapas** | Mapbox GL JS | 2.15+ | Renderizado WebGL eficiente | Leaflet, OpenLayers, Cesium |
| **Gráficos** | D3.js + Recharts | 7.8+ | Visualizaciones interactivas | Chart.js, Plotly, Observable |
| **3D/WebGL** | Deck.gl | 8.9+ | Visualizaciones geoespaciales | Three.js, Babylon.js |
| **Estilos** | Tailwind CSS | 3.3+ | Utility-first, consistencia | Material-UI, Chakra, Styled |
| **Build** | Vite | 4.4+ | Bundling rápido, HMR | Webpack, Parcel, Rollup |
| **Testing** | Vitest + RTL | 0.34+ | Testing moderno | Jest, Cypress, Playwright |

#### 2.3.2 Patrones de Arquitectura Frontend

**Tabla 8: Patrones Implementados en el Frontend**

| **Patrón** | **Aplicación** | **Beneficio** | **Ejemplo de Uso** |
| --- | --- | --- | --- |
| **Component Composition** | Construcción de UI | Reutilización, mantenibilidad | MapContainer + LayerManager |
| **Render Props** | Lógica compartida | Flexibilidad | DataProvider para múltiples vistas |
| **Higher-Order Components** | Funcionalidad transversal | DRY principle | withAuthentication, withLoading |
| **Custom Hooks** | Lógica de estado | Separación de concerns | useSimulation, useMapData |
| **State Machines** | Flujos complejos | Predictibilidad | Wizard de configuración |
| **Error Boundaries** | Manejo de errores | Robustez | Isolación de fallos por componente |

* **Función Principal**: Traducir los datos complejos (tanto de entrada como de salida) en visualizaciones y herramientas accionables para diferentes perfiles de usuario.

#### 2.3.3 Arquitectura de Componentes Adaptativos

┌─────────────────────────────────────────────────────────────┐  
│ CAPA DE PRESENTACIÓN │  
├─────────────────┬─────────────────┬─────────────────────────┤  
│ EXECUTIVE VIEWS │ TECHNICAL VIEWS │ OPERATIONAL VIEWS │  
│ │ │ │  
│ • KPI Dashboard │ • Data Explorer │ • Control Center │  
│ • Alert Summary │ • Parameter Tuning│ • Real-time Monitor │  
│ • Decision Support│ • Result Analysis│ • Resource Manager │  
└─────────────────┴─────────────────┴─────────────────────────┘  
 │  
 ▼  
┌─────────────────────────────────────────────────────────────┐  
│ CAPA DE ADAPTACIÓN │  
├─────────────────┬─────────────────┬─────────────────────────┤  
│ USER PROFILING │ CONTEXT AWARE │ RESPONSIVE DESIGN │  
│ │ │ │  
│ • Role Detection│ • Emergency Mode│ • Device Adaptation │  
│ • Preference Mgmt│ • Time Sensitivity│ • Viewport Scaling │  
│ • Learning System│ • Cognitive Load│ • Touch Optimization │  
└─────────────────┴─────────────────┴─────────────────────────┘  
 │  
 ▼  
┌─────────────────────────────────────────────────────────────┐  
│ CAPA DE SERVICIOS │  
├─────────────────┬─────────────────┬─────────────────────────┤  
│ DATA SERVICES │ COMMUNICATION │ UTILITY SERVICES │  
│ │ │ │  
│ • API Clients │ • WebSockets │ • Authentication │  
│ • State Management│ • Push Notifications│ • Localization │  
│ • Caching │ • Real-time Sync│ • Error Handling │  
└─────────────────┴─────────────────┴─────────────────────────┘

*Figura 4: Arquitectura de componentes adaptativos del frontend*

### 2.4 Componente 4: Sistema de Alerta y Respuesta Automática (La Voz)

Este es el módulo de salida que conecta la inteligencia generada por PIGEM con el mundo exterior.

#### 2.4.1 Arquitectura del Sistema de Notificaciones

**Tabla 9: Canales de Comunicación Implementados**

| **Canal** | **Tecnología** | **Audiencia** | **Latencia** | **Capacidad** | **Confiabilidad** |
| --- | --- | --- | --- | --- | --- |
| **SAE Integration** | API REST + WebHooks | Población general | <30 segundos | Nacional | 99.9% |
| **SMS Masivo** | Twilio/AWS SNS | Autoridades, equipos | <10 segundos | 10K/min | 99.5% |
| **Push Notifications** | Firebase/APNS | Apps móviles | <5 segundos | 100K/min | 99.0% |
| **Email Alerts** | SendGrid/SES | Técnicos, autoridades | <60 segundos | 50K/hora | 99.8% |
| **WebSockets** | Socket.io | Aplicaciones web | <1 segundo | 10K conexiones | 99.0% |
| **Social Media** | APIs Twitter/Facebook | Comunicación pública | <120 segundos | Variable | 95.0% |

#### 2.4.2 Motor de Reglas de Notificación

**Tabla 10: Tipos de Alertas y Criterios de Activación**

| **Tipo de Alerta** | **Criterio de Activación** | **Audiencia** | **Prioridad** | **Canales** |
| --- | --- | --- | --- | --- |
| **Emergencia Crítica** | Tsunami confirmado, <30min ETA | Toda la población | Crítica | SAE + SMS + Push + Sirenas |
| **Alerta Temprana** | Sismo >7.0, zona costera | Población en riesgo | Alta | SAE + Push + Social Media |
| **Congestión Severa** | >90% capacidad, >20min espera | Coordinadores | Media | WebSocket + SMS |
| **Recurso Agotado** | Albergue 100% ocupado | Equipos de terreno | Media | WebSocket + Push |
| **Sistema Down** | Servicio crítico offline | Administradores | Alta | SMS + Email + Slack |
| **Simulación Completa** | Escenario finalizado | Analistas | Baja | Email + WebSocket |

* **Función Principal**: Comunicar los resultados y las alertas de manera efectiva y multicanal.

#### 2.4.3 Pipeline de Procesamiento de Alertas

┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐  
│ DETECCIÓN │ │ EVALUACIÓN │ │ ENRUTAMIENTO │  
│ EVENTOS │ │ Y PRIORIZACIÓN │ │ INTELIGENTE │  
│ │ │ │ │ │  
│ • Simulación │───▶│ • Reglas Engine │───▶│ • Channel Select│  
│ • Sensores IoT │ │ • ML Scoring │ │ • Audience Filter│  
│ • Sistemas Ext. │ │ • Context Aware │ │ • Rate Limiting │  
│ • Manual Trigger│ │ • Deduplication │ │ • Personalization│  
└─────────────────┘ └─────────────────┘ └─────────────────┘  
 │  
 ▼  
┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐  
│ SEGUIMIENTO │ │ ENTREGA │ │ CONFIRMACIÓN │  
│ Y ANALYTICS │ │ MULTICANAL │ │ Y FEEDBACK │  
│ │ │ │ │ │  
│ • Delivery Track│◀───│ • Parallel Send │───▶│ • Read Receipts │  
│ • Engagement │ │ • Retry Logic │ │ • User Actions │  
│ • Effectiveness │ │ • Failover │ │ • Response Time │  
│ • A/B Testing │ │ • Load Balance │ │ • Error Handling│  
└─────────────────┘ └─────────────────┘ └─────────────────┘

*Figura 5: Pipeline de procesamiento y entrega de alertas*

## 3. Patrones de Integración y Comunicación

### 3.1 Arquitectura de Microservicios

**Tabla 11: Catálogo de Microservicios de PIGEM**

| **Dominio** | **Microservicio** | **Responsabilidad** | **Tecnología** | **Base de Datos** | **Protocolo** |
| --- | --- | --- | --- | --- | --- |
| **Identity** | auth-service | Autenticación JWT | Node.js + Express | PostgreSQL | REST + OAuth2 |
| **User Management** | user-service | Perfiles y preferencias | Python + FastAPI | PostgreSQL | REST |
| **Geospatial** | data-service | Gestión datos MCU | Python + Django | PostGIS | REST + OGC |
| **Simulation** | sim-service | Ejecución simulaciones | C++ + Python | Redis + Files | gRPC |
| **Visualization** | viz-service | Generación visualizaciones | Node.js + D3 | MongoDB | REST |
| **Notification** | alert-service | Gestión de alertas | Go + Gin | Redis + PostgreSQL | REST + WebSocket |
| **Analytics** | metrics-service | Métricas y logs | Python + Flask | InfluxDB + Elasticsearch | REST |
| **Gateway** | api-gateway | Routing y rate limiting | Kong + Lua | Redis | HTTP Proxy |

### 3.2 Patrones de Comunicación

**Tabla 12: Patrones de Comunicación Entre Servicios**

| **Patrón** | **Uso** | **Tecnología** | **Ventajas** | **Desventajas** |
| --- | --- | --- | --- | --- |
| **Synchronous REST** | Operaciones CRUD | HTTP + JSON | Simple, estándar | Acoplamiento temporal |
| **Asynchronous Messaging** | Eventos de dominio | RabbitMQ + AMQP | Desacoplamiento | Complejidad eventual |
| **Event Sourcing** | Auditoría, replay | EventStore + JSON | Trazabilidad completa | Overhead de storage |
| **gRPC** | Comunicación interna | Protocol Buffers | Performance, tipado | Menos tooling |
| **WebSockets** | Tiempo real | Socket.io | Bidireccional | Gestión de conexiones |
| **GraphQL** | APIs públicas | Apollo Server | Flexibilidad cliente | Complejidad caching |

### 3.3 Gestión de Datos Distribuidos

**Tabla 13: Estrategias de Consistencia de Datos**

| **Tipo de Dato** | **Patrón** | **Consistencia** | **Disponibilidad** | **Justificación** |
| --- | --- | --- | --- | --- |
| **Datos Geoespaciales** | Master-Slave | Eventual | Alta | Lectura intensiva, updates poco frecuentes |
| **Sesiones de Usuario** | Distributed Cache | Fuerte | Media | Experiencia de usuario coherente |
| **Resultados Simulación** | Event Sourcing | Eventual | Alta | Inmutabilidad, auditabilía |
| **Configuración Sistema** | Consensus | Fuerte | Media | Criticidad para operación |
| **Métricas y Logs** | Time Series | Eventual | Muy Alta | Volumen alto, tolerancia a pérdidas |

## 4. Seguridad y Conformidad

### 4.1 Modelo de Seguridad por Capas

**Tabla 14: Controles de Seguridad por Capa**

| **Capa** | **Amenazas** | **Controles** | **Estándares** | **Herramientas** |
| --- | --- | --- | --- | --- |
| **Aplicación** | Injection, XSS, CSRF | Input validation, Output encoding | OWASP Top 10 | SonarQube, SAST |
| **API** | Broken authentication, Excessive data | OAuth2, Rate limiting, RBAC | OWASP API Security | Kong, OAuth2 Proxy |
| **Servicio** | Insecure dependencies | Vulnerability scanning, Updates | NIST Framework | Snyk, Dependabot |
| **Red** | Man-in-the-middle, DDoS | TLS 1.3, WAF, Load balancing | ISO 27001 | Cloudflare, NGINX |
| **Infraestructura** | Privilege escalation | Container security, Secrets mgmt | CIS Benchmarks | Vault, Falco |
| **Datos** | Data breach, Unauthorized access | Encryption at rest/transit, Backup | GDPR, LOPD | PostgreSQL TDE, Backup encryption |

### 4.2 Gestión de Identidad y Acceso

**Tabla 15: Matriz de Roles y Permisos**

| **Rol** | **Datos Geoespaciales** | **Simulaciones** | **Alertas** | **Administración** | **APIs Públicas** |
| --- | --- | --- | --- | --- | --- |
| **Super Admin** | CRUD | CRUD | CRUD | CRUD | CRUD |
| **System Admin** | Read | Read | CRUD | CRU | Read |
| **Emergency Coordinator** | Read | CRU | CRUD | - | Read |
| **Technical Analyst** | Read | CRUD | Read | - | Read |
| **Municipal Authority** | Read | Read | Read | - | Read |
| **Researcher** | Read | Read | - | - | Read |
| **Citizen** | Read (public) | - | - | - | Read (limited) |

## 5. Despliegue y Operaciones

### 5.1 Arquitectura de Despliegue

**Tabla 16: Estrategia de Despliegue por Ambiente**

| **Ambiente** | **Propósito** | **Infraestructura** | **Datos** | **Monitoreo** | **SLA** |
| --- | --- | --- | --- | --- | --- |
| **Desarrollo** | Desarrollo activo | Local Docker | Datos sintéticos | Logs locales | N/A |
| **Testing** | Pruebas automatizadas | CI/CD Pipeline | Datos de prueba | Test reports | N/A |
| **Staging** | Pre-producción | Kubernetes cluster | Subset producción | Full monitoring | 95% |
| **Producción** | Operación real | Multi-AZ Kubernetes | Datos reales | 24/7 monitoring | 99.9% |
| **DR** | Recuperación desastres | Región secundaria | Backup/Replica | Monitoring básico | 99.5% |

### 5.2 Métricas de Rendimiento y SLAs

**Tabla 17: Service Level Agreements (SLAs)**

| **Servicio** | **Disponibilidad** | **Latencia P95** | **Throughput** | **MTTR** | **RPO** | **RTO** |
| --- | --- | --- | --- | --- | --- | --- |
| **API Gateway** | 99.95% | <100ms | 10K req/sec | <15min | N/A | <5min |
| **Data Services** | 99.9% | <500ms | 1K req/sec | <30min | <1hour | <30min |
| **Simulation Engine** | 99.5% | <30sec | 100 sim/hour | <1hour | <4hours | <2hours |
| **Notification System** | 99.9% | <10sec | 50K msg/min | <15min | <15min | <10min |
| **Database** | 99.95% | <50ms | 5K qps | <30min | <15min | <1hour |

### 5.3 Estrategia de Monitoreo y Observabilidad

**Tabla 18: Stack de Observabilidad**

| **Aspecto** | **Herramienta** | **Métricas Clave** | **Alertas** | **Retención** |
| --- | --- | --- | --- | --- |
| **Métricas** | Prometheus + Grafana | CPU, RAM, Latencia, Throughput | SLO violations | 90 días |
| **Logs** | ELK Stack | Error rate, Request logs | Error spikes | 30 días |
| **Trazas** | Jaeger | Request tracing, Dependencies | Slow transactions | 7 días |
| **APM** | New Relic | Application performance | Performance degradation | 30 días |
| **Uptime** | Pingdom | Service availability | Downtime | 1 año |
| **Seguridad** | SIEM (Splunk) | Security events | Threats detected | 1 año |

## 6. Casos de Uso y Validación

### 6.1 Escenarios de Validación

**Tabla 19: Casos de Uso para Validación del Sistema**

| **Escenario** | **Descripción** | **Actores** | **Datos Requeridos** | **Métricas de Éxito** |
| --- | --- | --- | --- | --- |
| **Simulacro Nacional** | Ejercicio masivo coordinado | SENAPRED, Municipios, Ciudadanos | Población real, infraestructura actual | >95% participación, <2min latencia |
| **Emergencia Real** | Tsunami en Valparaíso | Autoridades, Equipos respuesta | Datos tiempo real, alertas SAE | <30seg alerta, 0 falsos negativos |
| **Planificación Urbana** | Nuevo plan evacuación | Planificadores, Consultores | Censos, proyecciones demográficas | Plan optimizado, reducción 20% tiempo |
| **Investigación Académica** | Estudio comportamiento | Investigadores, Estudiantes | Datasets históricos, simulaciones | Publicaciones, validación científica |
| **Capacitación** | Entrenamiento operadores | Instructores, Aprendices | Escenarios sintéticos | >80% aprobación, <30min aprendizaje |

### 6.2 Métricas de Adopción y Impacto

**Tabla 20: KPIs del Sistema PIGEM**

| **Categoría** | **Métrica** | **Valor Objetivo** | **Frecuencia** | **Responsable** |
| --- | --- | --- | --- | --- |
| **Adopción** | Organizaciones usuarias | >50 instituciones | Mensual | Product Owner |
| **Uso** | Simulaciones ejecutadas | >1000/mes | Mensual | Technical Lead |
| **Performance** | Tiempo promedio simulación | <15 minutos | Continua | DevOps Team |
| **Calidad** | Precisión predicciones | >90% accuracy | Trimestral | Data Science Team |
| **Satisfacción** | Net Promoter Score | >50 | Semestral | UX Team |
| **Impacto** | Reducción tiempo evacuación | >20% mejora | Anual | Research Team |

## 7. Bibliografía Actualizada

### 7.1 Referencias en Arquitectura de Software

* Bass, L., Clements, P., & Kazman, R. (2021). *Software Architecture in Practice* (4th ed.). Addison-Wesley Professional.
* Fowler, M. (2018). *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional.
* Newman, S. (2021). *Building Microservices: Designing Fine-Grained Systems* (2nd ed.). O’Reilly Media.
* Richardson, C. (2018). *Microservices Patterns: With Examples in Java*. Manning Publications.

### 7.2 Referencias en Sistemas Distribuidos

* Kleppmann, M. (2017). *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. O’Reilly Media.
* Tanenbaum, A. S., & Van Steen, M. (2016). *Distributed Systems: Principles and Paradigms* (3rd ed.). Pearson.
* Van Steen, M., & Tanenbaum, A. S. (2023). *Distributed Systems* (4th ed.). Distributed Systems Pearson.

### 7.3 Referencias en Gestión de Emergencias

* Aydinoglu, A. C., & Bilgin, M. S. (2015). Developing an open geographic data model and analysis tools for disaster management: landslide case. *Natural Hazards and Earth System Sciences, 15*(2), 335–347. https://doi.org/10.5194/nhess-15-335-2015
* Bizid, I., Faiz, S., Boursier, P., & Yusuf, J. C. M. (2014). Integration of Heterogeneous Spatial Databases for Disaster Management. In J. Parsons & D. Chiu (Eds.), *Advances in Conceptual Modeling* (pp. 77–86). Springer International Publishing. https://doi.org/10.1007/978-3-319-14139-8\_10
* Mazimwe, A., Hammouda, I., & Gidudu, A. (2019). An Empirical Evaluation of Data Interoperability—A Case of the Disaster Management Sector in Uganda. *ISPRS International Journal of Geo-Information, 8*(11), 484. https://doi.org/10.3390/ijgi8110484
* Sopha, B. M., Triasari, A. I., & Cheah, L. (2021). Sustainable Humanitarian Operations: Multi-Method Simulation for Large-Scale Evacuation. *Sustainability, 13*(13), 7488. https://doi.org/10.3390/su13137488

### 7.4 Referencias en Estándares y Frameworks

* International Organization for Standardization. (2018). *ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. ISO.
* Open Geospatial Consortium. (2023). *OGC API - Features - Part 1: Core*. OGC Document 17-069r4.
* OWASP Foundation. (2021). *OWASP Top Ten 2021*. https://owasp.org/Top10/
* The Twelve-Factor App. (2017). *The Twelve-Factor App methodology*. https://12factor.net/

**Pablo Antonio Jordán González**  
Estudiante Doctorado Ciencias de la Ingeniería Mención Informática  
Universidad de Santiago de Chile  
Director de I+D HealthPixel Spa.