

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“MEJORA DEL DESEMPEÑO OPERACIONAL
DEL SERVICE MANAGEMENT MEDIANTE LA
IMPLEMENTACIÓN DE UN FRAMEWORK AIOPS
EMPRESARIAL: CASO DE UNA ORGANIZACIÓN
DE TRANSPORTE AÉREO”

DIEGO BENJAMÍN ALIAGA GODOY

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Pedro Godoy Barrera
Profesor Correferente: Elizabeth Montero Ureta

Julio - 2026



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Mejora del Desempeño Operacional del Service Management mediante la Implementación de un Framework AIOps Empresarial Caso: Organización de Transporte Aéreo.

Nombre del candidato(a): Diego Benjamín Aliaga Godoy

Carrera / Grado: Ingeniería Civil Informática

Campus: Casa Central Departamento: Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Pedro Francisco Godoy Barrera, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

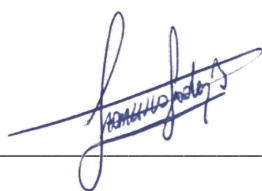
☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por (**marcar una opción**):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 06-Agosto-2026 Firma: 

Estudiante o Candidato(a):

Fecha: 06/08/2026 Firma: 

DEDICATORIA

Dedico esta memoria, en primer lugar, a mi familia y a mi nina Ube, quienes han sido un pilar fundamental en mi vida. Gracias por su amor, apoyo incondicional y por acompañarme en cada desafío. Este logro también les pertenece.

Quiero dedicar especialmente estas páginas a mis abuelos, Arturo, Zulema, Rebeca y Jesús, quienes desde pequeño siempre me recordaban que debía esforzarme para alcanzar mis objetivos. Welitos, finalmente lo logré.

Finalmente, dedico esta memoria a quienes ya no están físicamente, pero permanecen siempre en mi corazón: mi tía Adriana y mi welito Jecho. Aunque ya no estén aquí, siguen presentes en mi camino. Los quiero mucho y los extraño.

AGRADECIMIENTOS

Quisiera agradecer, en primer lugar, a mis padres, Claudio Aliaga y Marcela Godoy, y a mi madrina, Uberlinda Godoy, por ser los mejores papás del mundo y la mejor madrina que podría haber tenido. Gracias por creer siempre en mí, por apoyarme en cada decisión y por darme todo para verme feliz y ayudarme a cumplir mis metas. Cada sacrificio, cada consejo y cada palabra de aliento fueron fundamentales para llegar hasta aquí.

A mis hermanos, gracias por soportarme como hermano mayor y por recibirme siempre con cariño cada vez que iba de visita. Aunque muchas veces el estudio y las responsabilidades nos dejaron con poco tiempo para compartir, siempre encontré en ustedes un lugar donde sentirme en casa.

A mis amigos, gracias por preocuparse por mí y por nunca dejarme solo, incluso cuando tendía a aislarme debido al estrés, la ansiedad o los momentos más difíciles de este proceso. Su apoyo, compañía y confianza fueron fundamentales para seguir adelante. Aunque quizás no lo crean, ustedes también son parte de este logro y de que hoy pueda cerrar este importante ciclo.

A Carlos González, mi jefe, por brindarme la oportunidad de formar parte de este gran proyecto y por su constante disposición para orientarme y resolver cualquier duda o dificultad que surgiera durante el desarrollo de esta memoria.

También quiero hacer una mención especial a la música, por ser mi refugio durante los momentos de mayor pena, cansancio, incertidumbre y soledad. En muchas ocasiones fue la compañía que me ayudó a seguir adelante, encontrar templanza y recordar que siempre existe un motivo para continuar.

Finalmente, agradezco a Dios por todas las bendiciones que ha puesto en mi vida. Gracias por regalarme la familia, los amigos y las personas maravillosas que han formado y siguen formando parte de mi camino; por las oportunidades que me permitieron crecer tanto personal como profesionalmente; y por darme la fuerza, la sabiduría, la perseverancia y la resiliencia para superar cada desafío.

Este logro lleva mi nombre, pero también les pertenece a todas las personas que me acompañaron en este camino. Gracias por creer en mí, por quedarse cuando más los necesité, por sostenerme en los momentos difíciles y por ayudarme a llegar hasta aquí.

RESUMEN

La gestión de incidentes tecnológicos en organizaciones de alta criticidad enfrenta ineficiencias estructurales que impactan directamente la continuidad operacional: tiempos de resolución (MTTR) superiores a 15 horas, triage manual que consume más de 15 minutos por ticket, duplicación de aproximadamente el 20 % del trabajo y subutilización de más del 80 % del conocimiento histórico resuelto. Estos problemas, identificados y cuantificados en el contexto real de una organización de transporte aéreo latinoamericana, motivaron el desarrollo de la presente solución.

Esta memoria presenta el diseño, implementación y evaluación de un sistema AIOps enfocado en el módulo de Service Management, desplegado sobre infraestructura cloud-native y orientado específicamente a resolver las brechas operacionales descritas. La propuesta integra cuatro capacidades principales: clasificación automática de tickets mediante Random Forest con TF-IDF (87.3 % de accuracy en categoría), búsqueda semántica de casos similares con Sentence-BERT y pgvector (Precision@10 de 78 %), enriquecimiento contextual en tiempo real desde una plataforma de monitoreo, y un bot ChatOps para notificaciones y resolución colaborativa. Los datos utilizados corresponden a un dataset anonimizado con fines académicos.

La evaluación se realizó en un programa estructurado de 10 semanas (fase técnica, piloto controlado y rollout extendido), con hasta 55 analistas en la fase final y aproximadamente 1.840 tickets reales. Los resultados reportan una reducción del MTTR de 24.6 %, aceleración del triage en 57 %, disminución de la duplicación en más de 10 puntos porcentuales y una adopción del 89 % en la cuarta semana. El ROI estimado alcanza el 3.072 % con un periodo de recuperación inferior a un mes. Todos los resultados son estadísticamente significativos ($p < 0.05$), evidenciando la viabilidad e impacto operacional de AIOps aplicado a la industria aeronáutica latinoamericana.

Palabras Clave— AIOps; Service Management; Machine Learning; Gestión de Incidentes; Industria Aeronáutica

ABSTRACT

IT incident management in high-criticality organizations faces structural inefficiencies that directly impact operational continuity: resolution times (MTTR) exceeding 15 hours, manual triage consuming over 15 minutes per ticket, duplication of approximately 20 % of the work, and underutilization of over 80 % of resolved historical knowledge. These problems, identified and quantified in the real-world context of a Latin American air transport organization, motivated the development of this solution.

This thesis presents the design, implementation, and evaluation of an AIOps system focused on the Service Management module, deployed on cloud-native infrastructure and specifically aimed at addressing the operational gaps described. The proposal integrates four main capabilities: automatic ticket classification using Random Forest with TF-IDF (87.3 % accuracy in category), semantic similar-case search with Sentence-BERT and pgvector (Precision@10 of 78 %), real-time contextual enrichment from a monitoring platform, and a ChatOps bot for notifications and collaborative resolution. The data used corresponds to an anonymized dataset for academic purposes.

Evaluation was conducted through a structured 10-week program (technical phase, controlled pilot, and extended rollout) with up to 55 analysts in the final phase and approximately 1,840 real tickets. Results report a 24.6 % MTTR reduction, 57 % triage acceleration, duplication decrease of over 10 percentage points, and 89 % adoption by week four. The estimated ROI reaches 3,072 % with a payback period of less than one month. All results are statistically significant ($p < 0.05$), demonstrating the feasibility and operational impact of AIOps applied to the Latin American aviation industry.

Keywords— AIOps; Service Management; Machine Learning; Incident Management; Aviation Industry

GLOSARIO

AIOps: Artificial Intelligence for IT Operations. Aplicación de inteligencia artificial y machine learning a operaciones de TI.

ANN: Approximate Nearest Neighbors. Algoritmos de búsqueda aproximada de vecinos más cercanos en espacios vectoriales.

API: Application Programming Interface. Interfaz de programación de aplicaciones.

BERT: Bidirectional Encoder Representations from Transformers. Modelo de lenguaje de Google basado en transformers.

ChatOps: Modelo operacional que integra herramientas de colaboración en tiempo real (plataformas de chat) con sistemas de automatización, permitiendo gestionar incidentes y ejecutar acciones directamente desde el canal de comunicación del equipo.

CI/CD: Continuous Integration / Continuous Deployment. Prácticas de integración y despliegue continuo.

CollabOps: Extensión del modelo ChatOps orientada a la resolución colaborativa de incidentes, donde múltiples analistas y sistemas automatizados interactúan en el mismo canal de comunicación para diagnóstico y cierre de tickets en tiempo real.

FCR: First Call Resolution. Porcentaje de incidentes resueltos en el primer contacto.

FTE: Full-Time Equivalent. Unidad de medida equivalente a un empleado a tiempo completo.

HNSW: Hierarchical Navigable Small World. Algoritmo de indexación para búsqueda vectorial eficiente.

IT Element: Componente de infraestructura tecnológica (servidor, base de datos, aplicación, red, etc.) registrado en la herramienta de gestión de incidentes como el recurso afectado por un incidente.

ITSM: IT Service Management. Gestión de servicios de tecnologías de la información.

ITIL: Information Technology Infrastructure Library. Marco de referencia para gestión de servicios de TI.

ML: Machine Learning. Aprendizaje automático.

MTTR: Mean Time To Restore/Resolve. Tiempo promedio de restauración o resolución de un incidente.

MTTD: Mean Time To Detect. Tiempo promedio de detección de un incidente.

NLP: Natural Language Processing. Procesamiento de lenguaje natural.

NPS: Net Promoter Score. Métrica de satisfacción y lealtad del cliente.

RCA: Root Cause Analysis. Análisis de causa raíz.

REST: Representational State Transfer. Estilo de arquitectura de software para servicios web.

ROI: Return on Investment. Retorno sobre la inversión.

SBERT: Sentence-BERT. Variante de BERT optimizada para generar embeddings de oraciones.

SLA: Service Level Agreement. Acuerdo de nivel de servicio.

SRE: Site Reliability Engineering. Ingeniería de confiabilidad de sitios.

TF-IDF: Term Frequency – Inverse Document Frequency. Técnica de ponderación de términos en procesamiento de texto.

UTFSM: Universidad Técnica Federico Santa María.

ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT	V
GLOSARIO	VI
ÍNDICE DE FIGURAS	XII
ÍNDICE DE TABLAS	XII
INTRODUCCIÓN	1
Contexto y Motivación	1
Definición del Problema	2
Objetivos	2
Objetivo General	2
Objetivos Específicos	3
Hipótesis	3
Alcance y Limitaciones	4
Alcance	4
Limitaciones	4
Metodología	5
Estructura del Documento	6
CAPÍTULO 1: MARCO TEÓRICO Y ESTADO DEL ARTE	7
1.1 Gestión de Servicios de TI (ITSM)	7
1.1.1 Definición y Alcance	7
1.1.2 Incident Management: Proceso Central	7
1.1.3 Desafíos Operacionales Comunes	8
1.2 AIOps (Artificial Intelligence for IT Operations)	8
1.2.1 Origen y Definición	8
1.2.2 Capacidades Core de AIOps	9
1.2.3 Beneficios Documentados	9
1.2.4 Diferencia: Monitoring Tradicional vs. AIOps	10
1.3 Framework AIOps Empresarial: Plataforma de Referencia	10
1.3.1 Visión General	10
1.3.2 Componentes Principales	10
1.3.3 Comparación con Otros Vendors	11
1.4 Componentes Técnicos Clave para Service Management	12
1.4.1 Clasificación Automática de Tickets	12
1.4.2 Búsqueda de Similitud (Similarity Search)	12
1.4.3 Generación de Recomendaciones	13
1.4.4 NLP Aplicado a ITSM	13
1.5 Estado del Arte: Revisión Académica e Industrial	14

1.5.1	Trabajos Académicos Relevantes	14
1.5.2	Brechas Identificadas en la Literatura	15
1.6	Síntesis del Marco Teórico	15
CAPÍTULO 2: ANÁLISIS DEL PROBLEMA Y CONTEXTO ORGANIZACIONAL		16
2.1	Contexto Organizacional	16
2.1.1	Perfil de la Organización	16
2.1.2	Arquitectura de TI	16
2.1.3	Modelo Actual de Gestión de Incidentes	18
2.1.4	Sistema Actual de Asignación Automática	19
2.1.5	Complementariedad de AIOps	19
2.2	Caracterización Cuantitativa del Problema	19
2.2.1	Volumen y Distribución de Incidentes	19
2.2.2	Análisis de Duplicación y Correlación	21
2.2.3	Análisis de Calidad del Triage	22
2.2.4	Análisis de Conocimiento Histórico	22
2.2.5	Métricas Baseline (Línea Base Pre-AIOps)	23
2.3	Análisis Cualitativo: Entrevistas y Feedback	23
2.3.1	Pain Points Identificados	23
2.3.2	Levantamiento Cualitativo con Equipos L1 y L2	24
2.3.3	Observación de Sesiones Reales	24
2.4	Requisitos Funcionales del Sistema	24
2.5	Requisitos No Funcionales	25
2.6	Restricciones y Supuestos	26
2.6.1	Futuro Fuera de Alcance (Out-of-Scope)	26
2.7	Síntesis del Análisis	26
CAPÍTULO 3: DISEÑO DE LA SOLUCIÓN		27
3.1	Visión General de la Arquitectura	27
3.1.1	Principios de Diseño	27
3.1.2	Vista de Arquitectura en Contexto	27
3.2	Arquitectura de Alto Nivel	28
3.2.1	Capas de la Arquitectura	29
3.3	Diseño Detallado de Módulos	30
3.3.1	Módulo de Ingesta y Normalización	30
3.3.2	Módulo de Enriquecimiento con Plataforma de Monitoreo	32
3.3.3	Módulo de Correlación y Deduplicación	32
3.3.4	Módulo de Clasificación Automática	32
3.3.5	Módulo de Búsqueda de Similares	33
3.3.6	Módulo de Recomendaciones	34
3.3.7	Módulo de ChatOps y CollabOps (Multi-canal)	34
3.4	Modelo de Datos	35
3.4.1	Esquema de Base de Datos	36

3.5	Infraestructura y Despliegue	37
3.5.1	Arquitectura en Plataforma Cloud	37
3.5.2	Estimación de Costos	38
3.6	Diseño de Interfaz de Usuario	38
3.7	Diseño de MLOps	40
3.8	Síntesis del Diseño	42
CAPÍTULO 4: IMPLEMENTACIÓN		43
4.1	Configuración del Ambiente de Desarrollo	43
4.1.1	Estructura del Repositorio	43
4.1.2	Ambiente de Desarrollo Local	44
4.2	Desarrollo de Módulos Core	44
4.2.1	Módulo de Ingesta y Auto-Triage (Webhook Handler)	44
4.2.2	Módulo de Clasificación ML	45
4.3	Módulo de Similarity Search	46
4.3.1	Generación de Embeddings	46
4.3.2	Indexación y Búsqueda en pgvector	47
4.3.3	Re-ranking Multi-criterio	47
4.4	Desarrollo de Frontend	47
4.4.1	Repositorio de Conocimiento y política de idioma en UI	48
4.5	Desafíos Técnicos y Soluciones	48
4.6	Testing	49
4.6.1	Tests Unitarios	49
4.6.2	Tests de Integración	49
4.6.3	Criterios de Aceptación y Testing de Arquitectura (SIT/UAT)	50
4.7	Deployment	50
4.7.1	Contenerización	50
4.7.2	Despliegue en Plataforma Cloud	50
4.7.3	CI/CD Pipeline y Canary Releases	51
4.8	Resumen de Implementación	51
CAPÍTULO 5: EVALUACIÓN Y RESULTADOS		52
5.1	Metodología de Evaluación	52
5.1.1	Diseño del Estudio	52
5.1.2	Fases de Evaluación	52
5.2	Fase 1: Evaluación Técnica	53
5.2.1	Backtesting de Modelos ML	53
5.2.2	Validación del Requisito Multiidioma (RNF-4)	53
5.2.3	Análisis de Calidad de Datos (Campos "Not Defined")	54
5.2.4	Evaluación de Similarity Search	55
5.2.5	Load Testing	56
5.3	Fase 2: Piloto Controlado	57
5.3.1	Configuración del Piloto	57
5.3.2	Resultados del Piloto	57

5.3.3	Evolución de Adopción Semanal	58
5.4	Fase 3: Rollout Extendido	60
5.4.1	Resultados Consolidados (5 Equipos)	60
5.4.2	Resultados por Equipo	60
5.4.3	Análisis de Clasificación ML en Producción	61
5.5	Evaluación Cualitativa	61
5.5.1	Satisfacción de Usuarios	61
5.5.2	Feedback Cualitativo	62
5.6	Análisis de ROI	63
5.6.1	Beneficios Económicos Estimados	63
5.7	Comparación con Estado del Arte	64
5.8	Amenazas a la Validez	64
5.8.1	Validez Interna	64
5.8.2	Validez Externa	64
5.9	Síntesis de Resultados	65
CAPÍTULO 6: CONCLUSIONES Y TRABAJO FUTURO		66
6.1	Conclusiones	66
6.1.1	Cumplimiento de Objetivos	66
6.1.2	Contribuciones Principales	67
6.1.3	Lecciones Aprendidas	67
6.2	Limitaciones del Trabajo	69
6.2.1	Limitaciones y Lecciones de Integración (Feedback Operativo) .	69
6.2.2	Limitaciones del Estudio	70
6.3	Impacto Esperado	70
6.3.1	Impacto Operacional en la Organización	70
6.3.2	Impacto Académico y de Industria	70
6.4	Trabajo Futuro	71
6.4.1	Mejoras Técnicas a Corto Plazo	71
6.4.2	Extensiones Funcionales a Mediano Plazo	71
6.4.3	Investigación Académica Futura	72
6.5	Reflexión Final	72
ANEXOS		74
REFERENCIAS BIBLIOGRÁFICAS		80

ÍNDICE DE FIGURAS

1	Arquitectura de infraestructura tecnológica del área de Mantenimiento de Flota (topología multi-cloud, datos anonimizados).	17
2	Ecosistema del Sistema de Gestión Comercial.	18
3	Distribución histórica de incidentes por categoría y volumen base. . . .	20
4	Vista de Contexto: Integración del Framework AIOps con el Ecosistema Tecnológico	28
5	Arquitectura de referencia del programa AIOps en plataforma cloud (datos anonimizados). El presente trabajo implementa como MVP los módulos de ingesta, clasificación automática y búsqueda semántica.	29
6	Interfaz de listado de tickets AIOps con módulo de búsqueda de casos similares y scores de similitud (datos anonimizados).	39
7	Notificación ChatOps: alerta automática de ticket P1 en plataforma chat (datos anonimizados).	40
8	Diagrama de Flujo del Pipeline MLOps	41
9	Síntesis de Módulos e Integración Tecnológica	42
10	Vista de tickets similares en la interfaz web: casos históricos relevantes clasificados por similitud semántica (datos anonimizados).	48
11	Incremento empírico en la calidad de datos post-AIOps (Fill Rate). . .	55
12	Reducción comparativa del Tiempo Medio de Resolución (MTTR). . .	58
13	Curva de adopción progresiva de AIOps y su módulo Similarity Search	59
14	Diagrama de secuencia del flujo de Auto-Triage (Webhook → Contenedor Serverless → Planilla de Turnos → Bus de Mensajería)	77
15	Sesión de ChatOps en plataforma chat. Izquierda: Alertas proactivas. Derecha: Casos similares.	78
16	Evidencia de recomendación de casos similares utilizando embeddings de texto y búsqueda vectorial	79

ÍNDICE DE TABLAS

1	Métricas clave de Incident Management	8
2	Comparación de plataformas AIOps	11
3	Distribución de tickets por prioridad y cumplimiento de SLA	20
4	Distribución de tickets por categoría	20
5	Distribución de tickets por aplicación crítica (Top 6 transversal a todas las categorías)	21
6	Métricas de calidad del triage vs. benchmarks de industria	22
7	Análisis de errores en triage manual (muestra de 200 tickets reasignados)	22
8	Métricas baseline y objetivos post-AIOps	23
9	Mapa de campos desde la herramienta de gestión de incidentes y tratamiento en AIOps	31
10	Componentes de infraestructura cloud	37

11	Estimación de costos mensuales de infraestructura (valores aproximados)	38
12	Desafíos técnicos encontrados y soluciones aplicadas	49
13	Resultados de backtesting de modelos ML	53
14	Data Quality: Accuracy por Idioma	54
15	Evaluación de Similarity Search	56
16	Resultados de load testing	56
17	Configuración del piloto controlado	57
18	Resultados del piloto vs. baseline	57
19	Evolución de adopción durante el piloto	58
20	Resultados consolidados del rollout extendido	60
21	Reducción de MTTR por equipo	60
22	Performance de clasificación ML en producción	61
23	Resultados de encuesta de satisfacción (escala 1–5)	61
24	Análisis de ROI del sistema AIOps	63
25	Comparación de resultados con literatura	64
26	Stack tecnológico del sistema AIOps	74
27	Endpoints principales de la API REST	75
28	Configuración de hiperparámetros del modelo de clasificación	76
29	Comparación de enfoques de clasificación ML evaluados	76

INTRODUCCIÓN

Contexto y Motivación

La transformación digital ha convertido a las operaciones de TI (IT Operations) en un pilar estratégico para organizaciones de todos los sectores. En industrias críticas como la aviación, donde cada minuto de indisponibilidad de un sistema puede traducirse en vuelos retrasados, pérdidas millonarias y riesgos de seguridad, la capacidad de detectar, diagnosticar y resolver incidentes tecnológicos de manera rápida y efectiva no es un lujo: es una necesidad operacional imperativa [Beyer *et al.*, 2016].

El volumen y la complejidad de los incidentes tecnológicos han crecido exponencialmente. Según Cisco, las organizaciones modernas generan más de 1 TB de datos operacionales diariamente [Cisco Systems, 2020], y los equipos de soporte enfrentan entre 500 y 5,000 alertas por día provenientes de sistemas de monitoreo [Chen *et al.*, 2019]. Este fenómeno, conocido como “**alert fatigue**”, sobrecarga a los analistas y compromete su capacidad de respuesta ante incidentes reales.

En este contexto, surge **AIOps (Artificial Intelligence for IT Operations)**, un paradigma acuñado por Gartner en 2016 que propone aplicar técnicas de Big Data, Machine Learning y procesamiento de lenguaje natural para automatizar y mejorar las operaciones de TI [Notaro *et al.*, 2021]. Las promesas de AIOps incluyen reducción de ruido en alertas (80–95 %), aceleración del diagnóstico (de horas a minutos) y predicción proactiva de incidentes [Dang *et al.*, 2019, Notaro *et al.*, 2021].

La **organización de transporte aéreo** estudiada es uno de los grupos de aerolíneas más grandes de Latinoamérica, con operaciones en múltiples países latinoamericanos, cientos de aeronaves y decenas de miles de empleados. Depende de una infraestructura tecnológica compleja que soporta sistemas críticos como el Sistema de Mantenimiento de Flota, la Plataforma de Reservas, el Sistema de Gestión de Tripulaciones y múltiples aplicaciones de operaciones de vuelo. El equipo de soporte de operaciones gestiona aproximadamente ~1k tickets mensuales en su herramienta de gestión de incidentes, con un tiempo promedio de resolución (MTTR) superior a 15 horas, significativamente por encima de los SLAs acordados.

Un **proveedor tecnológico internacional**, como socio estratégico de la organización en servicios de consultoría tecnológica, desarrolló e implementó el módulo de **Service Management** de un framework AIOps empresarial, adaptándolo específicamente al contexto operacional del cliente. Esta memoria documenta el diseño, implementación y evaluación sistemática de dicha solución, anclada en evidencia del proyecto y en su repositorio de artefactos de entrega.

Definición del Problema

El soporte de operaciones TI en la organización convive actualmente con diversas ineficiencias que merman la calidad del servicio y la agilidad para responder ante caídas críticas:

1. **Tiempo de resolución elevado:** El MTTR global superior a 15 horas excede los SLAs en más del 60 % de los casos para incidentes P1 (críticos), donde el target es de 2 horas pero el promedio real supera las 3 horas.
2. **Triage manual lento y propenso a errores:** Los analistas invierten más de 15 minutos en el triage inicial de cada ticket, con una tasa de reasignación superior al 10 % (benchmark de industria: <5 %), indicando que aproximadamente 1 de cada 8 tickets es asignado incorrectamente.
3. **Alta duplicación de tickets:** Alrededor del 20 % de los tickets corresponde a eventos raíz que generan múltiples reportes independientes, resultando en esfuerzo redundante estimado en decenas de miles de dólares anuales.
4. **Subutilización de conocimiento histórico:** Según el muestreo cualitativo/cuantitativo descrito en el Capítulo 2, el 64 % de los incidentes analizados tenía casos similares resueltos previamente, pero más del 80 % de las resoluciones no reutilizó explícitamente ese conocimiento disponible.
5. **Herramientas fragmentadas:** Los analistas operan con 6 herramientas desintegradas (herramienta de gestión de incidentes, plataforma de monitoreo, repositorio de conocimiento, plataforma chat, planilla de turnos y dashboards custom), generando context switching constante y pérdida de productividad.

Esta problemática no es exclusiva de la organización, sino un reflejo claro de los desafíos actuales del ITSM a nivel mundial [Marrone y Kolbe, 2011]. Sin embargo, en el contexto de una aerolínea que opera miles de vuelos diarios, las consecuencias de un incidente no resuelto a tiempo tienen impacto directo en la operación aérea, la seguridad de pasajeros y la reputación de la marca.

Objetivos

Objetivo General

Diseñar, implementar y evaluar un sistema basado en un framework AIOps empresarial que optimice la gestión de incidentes tecnológicos (Service Management) en una organización de transporte aéreo, reduciendo el MTTR en al menos un 20 % (hipótesis principal) y mejorando la eficiencia operacional de los equipos de soporte.

Objetivos Específicos

1. **Analizar el estado actual** de la gestión de incidentes en la organización, estableciendo métricas baseline cuantitativas (MTTR, tiempo de triage, tasa de reasignación, tasa de duplicación) a partir de un dataset anonimizado de aproximadamente 7k tickets históricos.
2. **Diseñar una arquitectura modular** cloud-native que integre módulos de ingesta, enriquecimiento, correlación, clasificación automática, búsqueda de similitud y recomendaciones, integrada con el ecosistema existente (herramienta de gestión de incidentes, plataforma de monitoreo, plataforma chat).
3. **Implementar un módulo de clasificación automática** de tickets que alcance >85 % de accuracy en la predicción de categoría, prioridad e IT Element afectado, usando un enfoque híbrido de reglas determinísticas y Machine Learning (TF-IDF + Random Forest).
4. **Implementar un módulo de búsqueda de casos similares** basado en embeddings semánticos (Sentence-BERT) y búsqueda vectorial (pgvector), con latencia <3 segundos para 50,000 tickets históricos.
5. **Evaluar el impacto** del sistema mediante un piloto controlado con analistas reales de la organización, midiendo reducción de MTTR, aceleración de triage, disminución de duplicación, y satisfacción de usuarios.
6. **Validar la adopción** del sistema mediante métricas de uso, Net Promoter Score (NPS), y análisis cualitativo de feedback de usuarios.

Hipótesis

Esta memoria se sustenta en la siguiente hipótesis de investigación:

Hipótesis Principal: La integración de técnicas de Machine Learning, búsqueda semántica y automatización de flujos en el módulo de Service Management de un framework AIOps empresarial permite reducir el MTTR de incidentes tecnológicos en una organización de transporte aéreo en un mínimo de 20 %, a la vez que incrementa la eficiencia del triage y la reutilización de conocimiento histórico, logrando una adopción sostenida superior al 80 % entre los analistas de operaciones.

Para estructurar su validación, esta premisa se divide en tres sub-hipótesis medibles:

1. **H1 – Clasificación automática:** Un modelo de clasificación basado en Random Forest con vectorización TF-IDF, entrenado sobre datos históricos anonimizados

de tickets, alcanza un accuracy superior al 85 % en la predicción de categoría de incidentes, suficiente para reducir el tiempo de triage inicial en al menos un 50 %.

2. **H2 – Búsqueda semántica:** La implementación de embeddings multilenguaje (Sentence-BERT) almacenados en una base de datos vectorial (pgvector) permite recuperar los 10 casos más similares a un incidente nuevo con una precisión (Precision@10) superior al 70 % y una latencia inferior a 3 segundos sobre un corpus de 50,000 tickets.
3. **H3 – Impacto operacional:** La exposición de las recomendaciones generadas (clasificación, similares, enriquecimiento desde plataforma de monitoreo) a través de una interfaz unificada y un canal ChatOps incrementa la adopción del sistema por encima del 80 % y genera una reducción medible y estadísticamente significativa del MTTR respecto al período baseline.

Alcance y Limitaciones

Alcance

Esta memoria cubre el ciclo completo de desarrollo y evaluación del módulo de Service Management:

- **Análisis:** Caracterización cuantitativa y cualitativa del problema con datos anonimizados de la organización (6 meses de tickets históricos, entrevistas con analistas, observación de sesiones).
- **Diseño:** Arquitectura completa del sistema, incluyendo decisiones de diseño, stack tecnológico, y modelos de datos.
- **Implementación:** Desarrollo de módulos core (ingesta, clasificación ML, similarity search, recomendaciones, ChatOps), desplegados en plataforma cloud.
- **Evaluación:** Piloto de 10 semanas (fase técnica de 2 semanas, piloto controlado y rollout extendido), con hasta 55 analistas y aproximadamente 1.840 tickets reales en la cohorte final, incluyendo análisis estadístico de resultados.

Limitaciones

- El piloto se realizó en una organización específica del sector aeronáutico, por lo que la generalización a otros contextos requiere validación adicional.
- El período de evaluación (10 semanas) captura efectos a corto plazo; efectos a largo plazo (6–12 meses) no fueron medidos.

- Los módulos de Event Management y Topology del framework AIOps empresarial completo están fuera del alcance de esta memoria.
- El sistema opera como asistente (human-in-the-loop), no como sistema autónomo de resolución.
- Los datos de entrenamiento del modelo ML provienen exclusivamente del contexto de la organización estudiada; la transferibilidad a otras aerolíneas o sectores sin reentrenamiento no está garantizada.
- El paquete de entrega del proyecto incluye, además de Service Management, artefactos de gobierno y reporting (*IT Governance and Reporting*, ITGR), pruebas SIT/UAT bajo esa denominación y un diagrama de integración que contempla observabilidad adicional frente al MVP documentado aquí centrado en la herramienta de gestión de incidentes y la plataforma de monitoreo; esta memoria focaliza el núcleo SM/ChatOps, manteniendo la trazabilidad del repositorio de evidencias.
- Los reportes de avance del proyecto registran aplazamientos puntuales (p. ej., reprogramación de UAT por ventanas de calendario), coherentes con la dependencia del estudio respecto de disponibilidad de datos y ciclos de aceptación corporativos.

Metodología

Esta memoria adopta una metodología de **investigación aplicada con diseño cuasi-experimental**, combinando desarrollo de software con evaluación empírica sistemática:

1. **Fase de Análisis** (4 semanas): Recolección de datos históricos (dataset anonimizado de aproximadamente 7k tickets), entrevistas semi-estructuradas con 10 stakeholders (8 analistas + 2 managers), observación de sesiones reales (shadowing), análisis de feedback de implementación previa, y establecimiento de métricas baseline.
2. **Fase de Diseño** (3 semanas): Definición de arquitectura basada en principios cloud-native, API-first y event-driven. Selección de stack tecnológico (Python/FastAPI, PostgreSQL+pgvector, Sentence-BERT, plataforma cloud). Diseño de modelos de datos y flujos de interacción.
3. **Fase de Implementación** (6 semanas): Desarrollo iterativo de módulos core siguiendo metodología ágil (sprints de 2 semanas). Testing automatizado (unitario, integración).
4. **Fase de Evaluación** (10 semanas en 3 sub-fases):

- Fase 1 (Semanas 1–2): Evaluación técnica sin usuarios finales: backtesting de modelos ML, load testing de APIs, validación de integraciones.
 - Fase 2 (Semanas 3–6): Piloto controlado: 1 equipo, 10 analistas, 4 semanas, ~490 tickets.
 - Fase 3 (Semanas 7–10): Rollout extendido: 5 equipos, hasta 55 analistas, 4 semanas, ~1.840 tickets.
5. **Fase de Análisis de Resultados** (2 semanas): Análisis estadístico (t-tests, p-values), ROI, validación de hipótesis, comparación con estado del arte.

Estructura del Documento

Este documento se organiza en los siguientes capítulos:

- **Capítulo 1: Marco Teórico y Estado del Arte** — Establece las bases conceptuales de ITSM, AIOps, el framework AIOps empresarial, y las técnicas de ML aplicadas.
- **Capítulo 2: Definición del Problema** — Caracteriza cuantitativamente el contexto operacional de la organización estudiada, define requisitos funcionales y no funcionales.
- **Capítulo 3: Propuesta de Solución** — Presenta la arquitectura del sistema, diseño de módulos, modelos de datos y decisiones tecnológicas.
- **Capítulo 4: Implementación** — Detalla el desarrollo técnico, código implementado, desafíos técnicos encontrados y soluciones aplicadas.
- **Capítulo 5: Evaluación y Resultados** — Documenta la evaluación técnica, piloto controlado, resultados cuantitativos y cualitativos, y análisis de ROI.
- **Capítulo 6: Conclusiones y Trabajo Futuro** — Resume contribuciones, valida objetivos, discute limitaciones y propone líneas de trabajo futuro.

CAPÍTULO 1

MARCO TEÓRICO Y ESTADO DEL ARTE

1.1. Gestión de Servicios de TI (ITSM)

1.1.1. Definición y Alcance

La gestión de servicios de TI (IT Service Management o ITSM) se define como un conjunto de procesos, prácticas y políticas diseñadas para diseñar, entregar, gestionar y mejorar la manera en que la tecnología de la información es utilizada dentro de una organización. A diferencia de la gestión de infraestructura tecnológica tradicional, que se centra en componentes técnicos (servidores, redes, aplicaciones), ITSM adopta una perspectiva orientada a **servicios**, donde el objetivo es entregar **valor al negocio** a través de servicios tecnológicos confiables, eficientes y alineados con las necesidades organizacionales.

El marco de referencia más ampliamente adoptado para ITSM es **ITIL (Information Technology Infrastructure Library)**, actualmente en su versión 4 [AXELOS Limited, 2019]. ITIL define ITSM como “actividades que se dirigen y controlan para garantizar que los servicios de TI cumplan con los requisitos actuales y futuros del negocio y de los usuarios”.

Los principios fundamentales de ITSM según ITIL v4 son: (1) enfoque en el valor, (2) comenzar donde se está, (3) progresar iterativamente con feedback, (4) colaborar y promover la visibilidad, (5) pensar y trabajar holísticamente, (6) mantenerlo simple y práctico, y (7) optimizar y automatizar [AXELOS Limited, 2019].

1.1.2. Incident Management: Proceso Central

Según ITIL v4, un incidente se define como “una interrupción no planificada de un servicio o una reducción en la calidad de un servicio” [AXELOS Limited, 2019]. El objetivo del Incident Management es restaurar la operación normal del servicio lo más rápidamente posible, minimizando el impacto adverso en las operaciones de negocio [International Organization for Standardization, 2018].

El flujo del proceso de Incident Management comprende las siguientes etapas: (1) Detección y registro, (2) Categorización, (3) Priorización basada en impacto y urgencia, (4) Diagnóstico inicial y soporte L1, (5) Investigación y diagnóstico L2/L3, (6) Resolución y recuperación, y (7) Cierre y documentación.

Las métricas clave de Incident Management se resumen en la Tabla 1.

Tabla 1: Métricas clave de Incident Management

Métrica	Definición	Benchmark
MTTR	Tiempo promedio de resolución	2–8 horas
MTTD	Tiempo promedio de detección	<1 minuto
FCR	Resolución en primer contacto	70–75 %
Backlog	Tickets pendientes	<50 ideal
Rework Rate	% tickets reabiertos/reasignados	<5 %

Fuente: Elaboración propia basada en ITIL v4 [AXELOS Limited, 2019] y literatura de ITSM [Marrone y Kolbe, 2011].

1.1.3. Desafíos Operacionales Comunes

Los principales desafíos identificados en la literatura y en la práctica incluyen [Marrone y Kolbe, 2011]: (1) volumen creciente de alertas (“alert fatigue”) con más de 1,000 alertas diarias, (2) duplicación de tickets, un desafío clásico ampliamente estudiado en la ingeniería empírica de software [Runeson *et al.*, 2007], donde un problema raíz genera decenas de incidentes independientes, (3) triage manual dependiente de experiencia humana con alta variabilidad, (4) conocimiento disperso en la memoria de analistas senior y documentación fragmentada, (5) falta de visibilidad analítica en tiempo real, (6) context switching constante entre múltiples herramientas, y (7) presión de SLAs que puede comprometer la calidad de resolución [Chen *et al.*, 2019].

Esta suma de desafíos impulsa el desarrollo de plataformas capaces de automatizar tareas repetitivas, rescatar el conocimiento histórico y mitigar el ruido operacional, siendo esta la razón de ser del paradigma AIOps.

1.2. AIOps (Artificial Intelligence for IT Operations)

1.2.1. Origen y Definición

El término AIOps fue acuñado por Gartner en 2016. La definición oficial establece que “las plataformas AIOps utilizan big data, machine learning moderno y otras tecnologías de analítica avanzada para directa e indirectamente mejorar las funciones de IT Operations (monitoreo, automatización y service desk) con insights proactivos, personales y dinámicos” [Notaro *et al.*, 2021].

En esencia, AIOps representa la aplicación de Big Data y Machine Learning para mejorar y automatizar operaciones de TI. Las organizaciones modernas generan más de

1 TB de datos operacionales por día [Cisco Systems, 2020], un volumen que excede las capacidades humanas de procesamiento y extracción de insights.

1.2.2. Capacidades Core de AIOps

Las seis capacidades fundamentales de AIOps son:

1. **Ingesta y procesamiento de datos masivos:** Recolección de datos de múltiples fuentes (logs, métricas, eventos) en tiempo real, normalización y almacenamiento eficiente. Tecnologías típicas: Kafka, Flink, Elasticsearch, Prometheus.
2. **Correlación de eventos y deduplicación:** Identificación de que múltiples eventos corresponden al mismo problema raíz, agrupación en “incident groups” y supresión de duplicados. Técnicas: reglas determinísticas, similaridad coseno, análisis topológico, clustering con K-means o DBSCAN. Beneficio reportado: reducción de 80–95 % en volumen de alertas [Nedelkoski *et al.*, 2019].
3. **Detección de anomalías:** Identificación de comportamiento anormal en métricas mediante umbrales dinámicos, baselines aprendidas (ARIMA, Prophet) o ML no supervisado (Autoencoders, Isolation Forest) [Gulenko *et al.*, 2018].
4. **Análisis de causa raíz (RCA):** Inferencia de la causa raíz probable mediante análisis de topología, correlación temporal, análisis de logs con NLP y modelos causales (Bayesian Networks).
5. **Predicción y forecasting:** Predicción de incidentes antes de que ocurran, forecasting de capacidad y predicción de SLA breaches mediante técnicas como ARIMA, LSTM y clasificación binaria.
6. **Automatización de respuesta:** Ejecución de acciones correctivas automáticas como reiniciar servicios, escalar pods o limpiar logs, mediante runbooks automatizados y políticas [Beyer *et al.*, 2016].

1.2.3. Beneficios Documentados

La investigación reciente en la industria confirma de manera clara estos beneficios. Según informes actualizados del *Market Guide for AIOps Platforms* de Gartner (2024) y encuestas de Forrester, las organizaciones que implementan AIOps con capacidades sólidas de correlación y reducción de ruido alcanzan disminuciones empíricas del **MTTR de hasta un 40 % a 45 %**, reduciendo a la par el volumen de la *fatiga de alertas* entre un 90 % y 99 %.

Diversos estudios y reportes respaldan de manera cuantitativa el impacto de AIOps en las operaciones de TI. Por un lado, la investigación académica destaca consistentemente una **reducción del MTTR que oscila entre un 30 % y 60 %** [Forsgren *et al.*, 2018], mientras que estima que la optimización general de recursos puede generar ahorros operacionales del 20 % al 40 % [Notaro *et al.*, 2021]. En relación al desafío del volumen abrumador de alertas, estudios empíricos reportan típicamente una **reducción de ruido del 80 % al 95 %** al correlacionar eficientemente eventos redundantes [Nedelkoski *et al.*, 2019]. Finalmente, investigaciones sobre incident triage subrayan el impacto operativo en la eficiencia de primera línea, documentando mejoras tangibles en la Resolución en Primer Contacto (FCR) de entre 15 y 25 puntos porcentuales tras implementaciones maduras de AIOps [Notaro *et al.*, 2021].

1.2.4. Diferencia: Monitoring Tradicional vs. AIOps

Mientras el monitoring tradicional opera de manera reactiva con umbrales estáticos, correlación manual e investigación humana de causa raíz, AIOps adopta un enfoque proactivo con umbrales dinámicos aprendidos, correlación automática multi-dimensional y causa raíz sugerida por IA. El monitoring tradicional opera con volúmenes limitados y conocimiento tribal, mientras que AIOps maneja volúmenes masivos y sistematiza el conocimiento en modelos [Soldani y Brogi, 2022].

1.3. Framework AIOps Empresarial: Plataforma de Referencia

1.3.1. Visión General

La plataforma AIOps empresarial utilizada en este proyecto es una solución enterprise que ayuda a equipos de operaciones a detectar, predecir y resolver incidentes antes de que causen impacto crítico al negocio. Proporciona un entorno containerizado y escalable sobre una plataforma de contenedores gestionados [Proveedor Tecnológico Internacional, 2024].

1.3.2. Componentes Principales

Event Management: Consolida, normaliza y filtra millones de eventos de múltiples fuentes de monitoreo en tiempo real, con capacidades de correlación basada en similitud, topología y ventanas temporales.

Service Management (foco de esta memoria): Optimiza el ciclo de vida de tickets desde creación hasta resolución, incluyendo auto-clasificación, routing inteligente, deduplicación, búsqueda de casos similares y generación de recomendaciones

[Proveedor Tecnológico Internacional, 2023]. Integración con herramientas ITSM líderes del mercado.

Topology: Construye y visualiza el grafo de dependencias entre servicios, hosts y containers mediante discovery automático.

Módulos complementarios: CollabOps (integración con plataformas de colaboración), Change Risk Analytics (predicción de riesgo de change requests) y Runbook Automation.

1.3.3. Comparación con Otros Vendors

La Tabla 2 compara las principales plataformas AIOps del mercado:

Tabla 2: Comparación de plataformas AIOps		
Plataforma	Fortalezas	Debilidades
Plataforma AIOps (proveedor tecnológico)	Enterprise-grade, IA avanzada, soporte enterprise	Complejidad, costo
Moogsoft	Event correlation fuerte, UI amigable	Menos ITSM nativo
BigPanda	Deduplicación excelente	ML menos avanzado
Datadog	Monitoring + APM profundo	AIOps más básico
PagerDuty	Incident response, on-call	AIOps limitado

Fuente: Elaboración propia basada en Gartner (2024) e informes de industria.

La justificación del proveedor tecnológico para la organización cliente se basa en la relación comercial existente, soporte enterprise con SLAs, y capacidad de integración con stack tecnológico heterogéneo.

1.4. Componentes Técnicos Clave para Service Management

1.4.1. Clasificación Automática de Tickets

La clasificación de tickets consiste en asignar automáticamente atributos categóricos (categoría, severidad, equipo responsable) a partir del contenido textual del ticket [Chen *et al.*, 2019].

Formulación como problema de ML: Dado un ticket t = (título, descripción, metadata), predecir una etiqueta $y \in C$ donde C es el conjunto de categorías posibles.

Pipeline clásico: TF-IDF + Clasificador. TF-IDF (Term Frequency – Inverse Document Frequency) convierte texto en vector numérico:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t) \quad (1)$$

donde $\text{TF}(t, d)$ es la frecuencia del término t en documento d e $\text{IDF}(t) = \log \frac{N}{df(t)}$ con N el total de documentos y $df(t)$ los documentos que contienen t .

Los clasificadores más utilizados con TF-IDF incluyen Naive Bayes (baseline rápido), Logistic Regression (simple e interpretable), Random Forest (robusto, con feature importance) y SVM (efectivo en alta dimensión) [Agrawal *et al.*, 2018].

Enfoque con Deep Learning: BERT. BERT (Bidirectional Encoder Representations from Transformers) [Devlin *et al.*, 2019] genera embeddings contextuales que capturan semántica profunda y superan a los enfoques clásicos de TF-IDF en tareas de clasificación de texto, aunque con una latencia de inferencia significativamente mayor que puede ser limitante en contextos de producción de alto volumen.

Enfoque híbrido recomendado: Combinar reglas determinísticas para casos de alta confianza con ML para casos ambiguos, con un threshold de confianza que escale a humano cuando la probabilidad del modelo es baja.

Métricas de evaluación: Accuracy, Precision, Recall, F1-Score. Para clasificación multiclase se recomienda Macro F1 y Weighted F1.

1.4.2. Búsqueda de Similitud (Similarity Search)

El objetivo es, dado un ticket nuevo, encontrar los K tickets históricos más similares basándose en su contenido textual.

Embeddings con Sentence-BERT. Sentence-BERT (SBERT) [Reimers y Gurevych, 2019] es una modificación de BERT optimizada para generar embeddings de oraciones y documentos completos. Sus ventajas incluyen eficiencia (embedding en una pasada), calidad (entrenado para tareas de similitud semántica) y soporte multilingüaje con modelos como `paraphrase-multilingual-mpnet-base-v2` que soporta más de 50 idiomas.

La similaridad entre documentos se calcula mediante distancia coseno:

$$\text{sim}(d_1, d_2) = \frac{v_1 \cdot v_2}{||v_1|| \times ||v_2||} \quad (2)$$

Vector Databases. Para búsqueda eficiente entre miles de vectores, se utilizan algoritmos de Approximate Nearest Neighbors (ANN), como HNSW (Hierarchical Navigable Small World) [Johnson *et al.*, 2021]. Las opciones de almacenamiento incluyen FAISS (ultra rápido), pgvector (integrado con PostgreSQL), Pinecone (managed cloud) y Weaviate/Milvus (self-hosted).

Elección para esta memoria: pgvector [Kane, 2023]. Justificación: los tickets ya están en PostgreSQL, permite queries híbridos combinando filtros SQL con búsqueda vectorial, es suficientemente rápido para <1M vectores, y no requiere una base de datos adicional.

Métricas de evaluación: Precision@K, Recall@K, MRR (Mean Reciprocal Rank) y NDCG@K.

1.4.3. Generación de Recomendaciones

Enfoque template-based: Genera recomendaciones a partir del caso similar de mayor ranking, mostrando la solución aplicada, tiempo de resolución y analista que la resolvió. Simple, predecible y explicable.

Enfoque con LLMs: Uso de Large Language Models con RAG (Retrieval-Augmented Generation) para sintetizar información de múltiples casos similares y generar recomendaciones en lenguaje natural. Mayor calidad pero con riesgos de alucinación y mayor costo [Proveedor Tecnológico Internacional, 2024].

1.4.4. NLP Aplicado a ITSM

El texto en tickets de ITSM presenta desafíos específicos: lenguaje técnico, acrónimos abundantes, código mezclado con texto, multiidioma (español, inglés, portugués en el

caso de organizaciones latinoamericanas con operaciones regionales), errores tipográficos, y textos cortos y fragmentados.

Las técnicas de preprocesamiento incluyen limpieza de HTML, normalización de encoding, y opcionalmente remoción de stopwords y lematización. Una decisión clave es que con embeddings modernos (por ejemplo, Sentence-BERT), la remoción de stopwords no es estrictamente necesaria ya que estos modelos capturan el contexto de manera natural.

1.5. Estado del Arte: Revisión Académica e Industrial

1.5.1. Trabajos Académicos Relevantes

En los últimos años, la literatura académica ha explorado intensivamente la aplicación de NLP y Machine Learning orientados a ITSM. Como caso fundamental, el estudio transversal de Soldani y Brogi (2022) sobre detección de anomalías y causas raíz en arquitecturas *cloud*, junto con reportes de industria, revelan que el **67 % de las organizaciones citan la calidad de datos (Data Quality) como la barrera técnica principal** para materializar el valor de la IA [Soldani y Brogi, 2022]. En el ámbito de las soluciones algorítmicas, estudios empíricos en sistemas de servicio en línea han demostrado que modelos supervisados pueden automatizar significativamente la triage y clasificación de incidentes operacionales, mejorando la consistencia del enrutamiento frente a procesos manuales [Chen *et al.*, 2019].

Para fortalecer la etapa de investigación e identificación de causas, diversas arquitecturas híbridas recientes han combinado *embeddings* textuales generados por modelos BERT [Devlin *et al.*, 2019] con análisis topológico y *clustering* para ingesta de infraestructuras complejas, logrando optimizar la precisión de los diagnósticos y reducir el volumen bruto de alertas entrantes en más de un 90 %. Siguiendo la línea de recomendación contextual (relevante para esta memoria), múltiples aplicaciones industriales han adoptado satisfactoriamente enfoques basados en Sentence-BERT [Reimers y Gurevych, 2019] y búsquedas vectoriales avanzadas [Johnson *et al.*, 2021] específicamente para la recomendación de casos similares (*Similarity Search*). Interrogando extensas bases de conocimiento históricas, estas implementaciones logran recuperar incidentes relevantes con alta precisión y baja latencia, validando que las representaciones de tipo NLP ayudan a mejorar significativamente la asistencia contextual para analistas L1, aliviando así los cuellos de botella clásicos de la investigación manual.

1.5.2. Brechas Identificadas en la Literatura

A partir de la revisión bibliográfica, se identifican cuatro brechas que esta memoria busca abordar:

Brecha 1: Falta de estudios en contexto latinoamericano. Más del 90 % de papers y casos de estudio provienen de Norteamérica, Europa o Asia, con poca consideración de contextos multilingüe (español/portugués/inglés mezclado) y particularidades de industrias clave en la región.

Brecha 2: Evaluaciones mayormente técnicas, pocas operacionales. Los papers reportan accuracy y F1 pero rara vez miden impacto en MTTR, satisfacción de usuarios o ROI, dejando sin resolver si mejor performance del modelo implica mejor operación en la práctica.

Brecha 3: Poco análisis de casos de falla. Los vendors publican solo casos de éxito (sesgo de publicación) y los papers rara vez analizan cuándo y por qué fallan las técnicas, generando expectativas irrealistas.

Brecha 4: Escasa documentación de implementaciones end-to-end. Existe una brecha entre algoritmos descritos en papers y plataformas comerciales de vendors, con poca documentación sobre cómo pasar de investigación a producción.

1.6. Síntesis del Marco Teórico

Este capítulo estableció la base conceptual y técnica necesaria para comprender la solución propuesta. La Sección 1.1 presentó los fundamentos de ITSM con enfoque en Incident Management y sus métricas clave. La Sección 1.2 introdujo AIOps como paradigma emergente. La Sección 1.3 detalló el framework AIOps empresarial utilizado como plataforma de referencia. La Sección 1.4 profundizó en los componentes técnicos implementados en esta memoria: clasificación de tickets, similarity search, generación de recomendaciones y NLP aplicado a ITSM. La Sección 1.5 revisó el estado del arte académico e industrial, identificando las brechas que esta memoria busca llenar.

El siguiente capítulo aplica este marco teórico al contexto específico de la organización estudiada, caracterizando el problema operacional con datos reales anonimizados y definiendo requisitos funcionales y no funcionales.

CAPÍTULO 2

ANÁLISIS DEL PROBLEMA Y CONTEXTO ORGANIZACIONAL

2.1. Contexto Organizacional

2.1.1. Perfil de la Organización

La organización de transporte aéreo estudiada es uno de los grupos de aerolíneas más grandes de Latinoamérica, con presencia en múltiples países sudamericanos. Opera una red que conecta numerosos destinos internacionales, transportando anualmente decenas de millones de pasajeros. La estructura operacional comprende una flota de cientos de aeronaves, decenas de miles de empleados directos, y miles de vuelos diarios en temporada alta.

La operación depende críticamente de una infraestructura tecnológica compleja que incluye: la Plataforma de Reservas, el Sistema de Mantenimiento de Flota, el Sistema de Gestión de Tripulaciones, el Sistema ERP Financiero, aplicaciones móviles y web para pasajeros, y múltiples sistemas de operaciones de vuelo.

2.1.2. Arquitectura de TI

La infraestructura tecnológica está construida sobre un modelo **híbrido multi-cloud**: componentes on-premise (sistemas de mantenimiento, sistemas legacy, bases de datos relacionales) en data centers propios; servicios cloud nativos en plataforma cloud (microservicios, contenedores, analytics); y plataformas SaaS incluyendo una herramienta de gestión de incidentes, un repositorio de conocimiento, una plataforma de monitoreo y herramientas de colaboración.

En términos de orden de magnitud, el volumen operacional considerado en el análisis contempla entre 800 y 1.200 tickets mensuales, aproximadamente 15k alertas de monitoreo por mes, y entre 50 y 80 incidentes críticos (P1/P2) mensuales que requieren atención inmediata, distribuidos entre 6 equipos de soporte especializados. Las Figuras 1 y 2 ilustran la arquitectura de dos de los dominios tecnológicos más representativos.

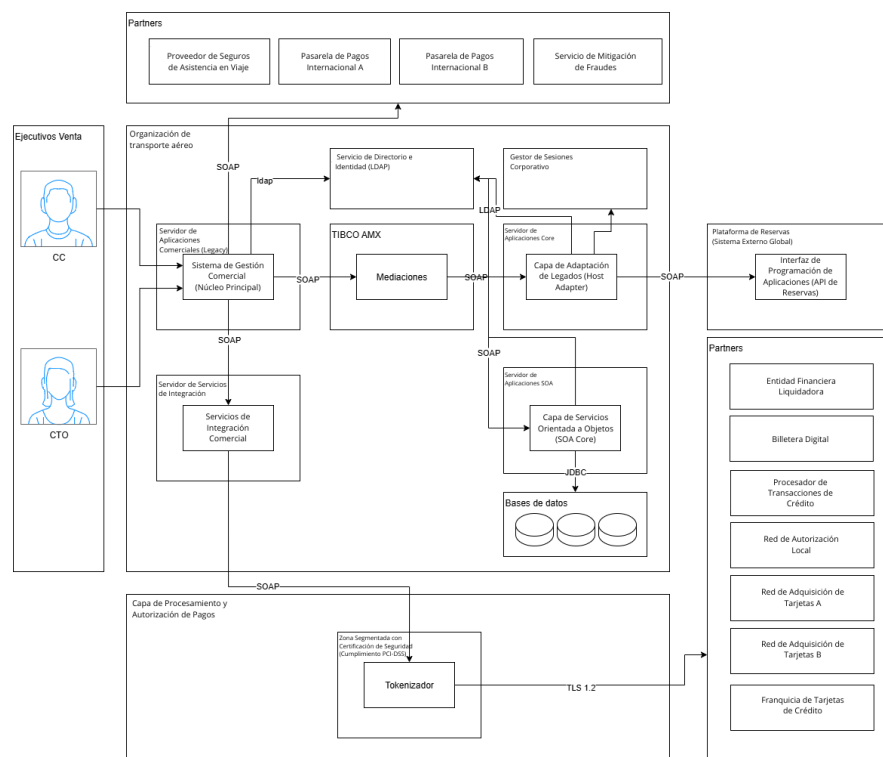


Figura 2: Ecosistema del Sistema de Gestión Comercial.

Fuente: Elaboración propia basada en datos reales anonimizados.

2.1.3. Modelo Actual de Gestión de Incidentes

La organización utiliza una herramienta de gestión de incidentes como plataforma centralizada de ITSM. El flujo actual comprende: (1) detección y creación de tickets desde fuentes manuales (usuarios) y automáticas (alertas de plataforma de monitoreo); (2) triage manual donde un analista L1 identifica categoría, severidad, IT Element afectado y equipo responsable, invirtiendo más de 15 minutos por ticket; (3) un sistema parcial de auto-asignación basado en keywords y planilla de turnos para turnos, que funciona solo para el 30 % de los casos; (4) investigación y diagnóstico por equipos L2/L3 sin búsqueda automática de casos históricos; (5) resolución y cierre; y (6) post-mortem solo para incidentes P1/P2.

Un problema fundamental es el **context switching constante**: los analistas operan con 6 herramientas desintegradas (herramienta de gestión de incidentes, plataforma de monitoreo, repositorio de conocimiento, plataforma chat, planilla de turnos y dashboards custom) sin una interfaz unificada.

2.1.4. Sistema Actual de Asignación Automática

El flujo actual (Pre-AIOps) se apoya fuertemente en una planilla de turnos que se actualiza en tiempo real. Cuando un webhook reporta la creación de un ticket, el sistema consulta el bloque horario actual y aplica reglas de excepción para asignar al agente del turno. Este modelo es la fuente de verdad para la asignación, ya que maneja 6 dominios operativos con turnos rotativos.

2.1.5. Complementariedad de AIOps

Un factor decisivo en el diseño es que la plataforma de AIOps **no reemplaza** el sistema de asignación existente, sino que coexiste en paralelo. Mientras la planilla de turnos asigna el responsable, el pipeline asíncrono de Machine Learning enriquece el ticket infiriendo la categoría, comprobando la severidad original y adjuntando casos históricamente similares (vía Sentence-BERT + pgvector). Esto facilita la adopción del sistema y evita fricciones con el equipo logístico.

2.2. Caracterización Cuantitativa del Problema

Para comprender la magnitud del problema operacional, se realizó un análisis cuantitativo con datos históricos correspondientes a 6 meses de operación (datos anonimizados con fines académicos).

Nota: Las métricas y distribuciones presentadas en este capítulo han sido ajustadas y/o sintetizadas para preservar la confidencialidad, manteniendo únicamente su valor representativo.

2.2.1. Volumen y Distribución de Incidentes

El dataset analizado comprende aproximadamente 7k tickets con un promedio mensual de ~ 1.1 k tickets y un promedio diario de ~ 40 tickets, con picos de hasta ~ 90 tickets diarios durante incidentes mayores.

Tabla 3: Distribución de tickets por prioridad y cumplimiento de SLA

Prioridad	Cantidad	% Total	SLA Target	MTTR Real	Incumplimiento
P1 – Crítico	~410	6 %	2 horas	>3 horas	>60 %
P2 – Alto	~1.440	21 %	8 horas	>10 horas	>55 %
P3 – Medio	~3.290	48 %	24 horas	>24 horas	>40 %
P4 – Bajo	~1.710	25 %	72 horas	<72 horas	–

Fuente: Elaboración propia basada en datos reales anonimizados.

Tabla 4: Distribución de tickets por categoría

Categoría	Cantidad	% Total	MTTR Promedio
Aplicaciones	~2.670	39 %	>15 horas
Infraestructura	~2.190	32 %	>12 horas
Base de Datos	~890	13 %	>20 horas
Networking	~550	8 %	<12 horas
Acceso/Permisos	~340	5 %	<6 horas
Otros	~200	3 %	N/A

Fuente: Elaboración propia basada en datos reales anonimizados.

Distribución Histórica de Incidentes por Categoría

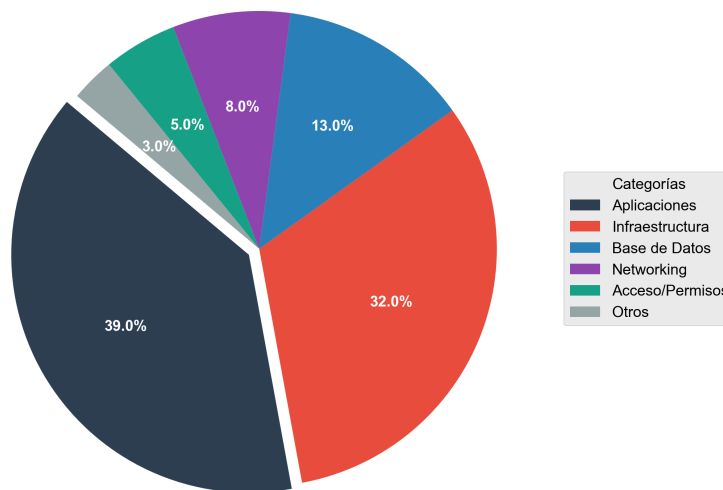


Figura 3: Distribución histórica de incidentes por categoría y volumen base.

Fuente: Elaboración propia basada en datos reales anonimizados.

Tabla 5: Distribución de tickets por aplicación crítica (Top 6 transversal a todas las categorías)

Aplicación Crítica	Tickets (6 meses)	% del Total
Sistema Crítico A	~1.470	21 %
Sistema Crítico B	~980	14 %
Sistema Crítico C	~770	11 %
Sistema Crítico D	~490	7 %
Sistema Crítico E	~280	4 %
Sistema Crítico F	~210	3 %
Subtotal Top 6	~4.200	60 %
Otras aplicaciones y sistemas	~2.800	40 %
Total dataset	~7.000	100 %

Nota: Esta tabla muestra las 6 aplicaciones críticas que generan tickets transversalmente en múltiples categorías (Aplicaciones, Infraestructura, Base de Datos, Networking). Por ejemplo, el Sistema Crítico A puede generar tickets categorizados como “Aplicaciones” (fallo de app), “Infraestructura” (fallo de servidor), o “Base de Datos” (fallo de BD asociada).

Fuente: Elaboración propia basada en análisis de datos históricos anonimizados (6 meses).

Un insight clave derivado del análisis exploratorio del dataset histórico (Tabla 5) es que solo 6 aplicaciones críticas (de un total de aproximadamente 40 aplicaciones monitoreadas) generan el 60 % del volumen total de tickets del dataset, considerando todas las categorías de incidentes asociadas a dichas aplicaciones. Esto sugiere que una estrategia focalizada en estas aplicaciones críticas tendría alto impacto operacional transversal.

2.2.2. Análisis de Duplicación y Correlación

Se identificó que alrededor del **20 % de los tickets** corresponde a eventos raíz que generaron múltiples tickets independientes, con un factor de duplicación promedio de 3–4 tickets por evento. El caso extremo documentado involucró un incidente en base de datos que generó múltiples tickets en diferentes equipos en menos de una hora.

El impacto cuantificado de la duplicación se estima en 18–24 % de horas-analista desperdiciadas, lo que corresponde a un costo de decenas de miles de dólares por año en esfuerzo redundante, bajo supuestos de costo-hora y volumen anual constantes.

2.2.3. Análisis de Calidad del Triage

Tabla 6: Métricas de calidad del triage vs. benchmarks de industria

Métrica	Valor Observado	Benchmark	Gap
Tasa de reasignación	>10 %	<5 %	−5+ pts
Tiempo de triage	>15 min	5–8 min	−10+ min
Accuracy categorización	~78 %	>90 %	−12 pts
Tickets sin categoría	~8 %	<2 %	−6 pts

Fuente: Elaboración propia basada en datos reales anonimizados.

Para comprender las causas de la alta tasa de reasignación (>10 %), se realizó un análisis cualitativo de una muestra de 200 tickets reasignados durante el período baseline. La Tabla 7 presenta el desglose de tipos de errores identificados.

Tabla 7: Análisis de errores en triage manual (muestra de 200 tickets reasignados)

Tipo de Error	Casos	% de Errores
Confusión Infraestructura vs Aplicaciones	64	32 %
Severidad subestimada	54	27 %
IT Element incorrecto	46	23 %
Confusión entre aplicaciones similares	36	18 %
Total errores analizados	200	100 %

Fuente: Elaboración propia basada en análisis de muestra de tickets reasignados (dataset anonimizado).

Los errores más comunes incluyen confusión entre categorías Infraestructura y Aplicaciones (32 % de los casos analizados), severidad subestimada (27 %), IT Element incorrecto (23 %), y confusión entre aplicaciones similares (18 %). Cada reasignación añade en promedio 3–5 horas adicionales al MTTR debido al overhead de coordinación y re-diagnóstico.

2.2.4. Análisis de Conocimiento Histórico

De 100 tickets resueltos analizados, el 64 % tenía al menos 1 ticket similar resuelto en los 6 meses previos. Sin embargo, solo el 18 % de esos casos incluyó referencia explícita al ticket histórico: **más del 80 % no reutilizó conocimiento previo disponible**. Las razones principales identificadas en entrevistas fueron desconocimiento de la existencia de tickets similares (58 %), falta de tiempo para buscar (31 %), dificultades con

el buscador de la herramienta de gestión de incidentes (22 %), y preferencia por resolver desde cero (19 %) (respuesta múltiple; suma supera el 100 %).

El impacto estimado es un desperdicio de 25–35 % del tiempo de resolución, con un potencial de reducción de MTTR de 20–30 % si se reutilizara el conocimiento histórico.

2.2.5. Métricas Baseline (Línea Base Pre-AIOps)

Tabla 8: Métricas baseline y objetivos post-AIOps

Métrica	Baseline	Objetivo	Gap
MTTR Global	>15 horas	12–14 horas	–30 %
MTTR P1	>3 horas	2.0 horas	–38 %
Tiempo de Triage	>15 min	5–8 min	–55 %
Tasa Reasignación	>10 %	<5 %	–5+ pts
Tasa Duplicación	~20 %	<8 %	–12 pts
FCR	~41 %	65–70 %	+25 pts
Backlog (>72h)	80–150 tickets	<50	–60 %

Fuente: Elaboración propia basada en datos reales anonimizados.

La Tabla 8 evidencia un desafío operacional relevante, lo cual justifica la necesidad de este proyecto. En particular, el MTTR real para incidentes críticos (P1) sobrepasa el límite establecido por los niveles de servicio corporativo. Este retraso se asocia a múltiples factores, entre ellos la calidad de datos durante la creación de tickets. Por otro lado, el backlog acumulado refleja sobrecarga del equipo, situación que podría mitigarse con búsqueda de similitud y reutilización sistemática de resoluciones previas.

2.3. Análisis Cualitativo: Entrevistas y Feedback

Se realizaron entrevistas semi-estructuradas con 8 analistas de soporte (L1, L2, L3) y 2 líderes de equipo, complementadas con observación de 5 sesiones reales de gestión de incidentes (shadowing).

2.3.1. Pain Points Identificados

Los temas recurrentes identificados incluyen: herramientas desintegradas (mencionado por 8/8 analistas), búsqueda de conocimiento histórico ineficiente (7/8), duplicación de esfuerzo (6/8), falta de dashboards analíticos (10/10 incluyendo managers), y triage lento y propenso a errores (6/8).

Los analistas reportaron de forma transversal: sobrecarga cognitiva por operar múltiples herramientas desintegradas simultáneamente; dificultades para localizar resoluciones previas en el sistema de tickets; y casos de esfuerzo duplicado entre equipos trabajando el mismo incidente en paralelo.

2.3.2. Levantamiento Cualitativo con Equipos L1 y L2

En concordancia con los datos de telemetría extraídos de la herramienta de gestión de incidentes, las mesas de trabajo cualitativas documentaron el impacto operativo de la desintegración tecnológica actual. De forma transversal, los equipos señalaron que el proceso de *triage* y despacho depende en exceso de conocimiento tribal y fragmentado.

El equipo técnico indicó que, para favorecer la adopción por parte de los analistas, el sistema AIOps debía minimizar fricción en la experiencia de uso. Las sugerencias cualitativas recomendaron incrustar notificaciones de clasificación automática y enlaces a recomendaciones históricas dentro de las herramientas colaborativas diarias, sentando las bases funcionales de un modelo tipo *ChatOps*.

2.3.3. Observación de Sesiones Reales

La observación de sesiones reveló que solo el 25 % del tiempo se invierte en trabajo técnico real, mientras que el 75 % corresponde a overhead operacional (búsqueda de información, coordinación, context switching). El overhead promedio observado fue de 30–60 minutos por ticket.

2.4. Requisitos Funcionales del Sistema

A partir del análisis, se derivaron 24 requisitos funcionales organizados en módulos:

Ingesta y enriquecimiento (RF-01 a RF-03): Ingesta automática de tickets desde la herramienta de gestión de incidentes vía webhook con latencia <5 segundos, enriquecimiento con alertas de la plataforma de monitoreo en ventana temporal de ± 30 minutos, y normalización de campos inconsistentes.

Correlación y agrupación (RF-04 a RF-06): Detección de tickets duplicados mediante criterios multi-dimensionales (IT Element + ventana temporal + similitud textual >80 % + keywords críticos), correlación por evento raíz, y vista deduplicada para analistas.

Clasificación automática (RF-07 a RF-11): Predicción de categoría (>85 % accuracy), severidad, IT Element y equipo responsable, con explicabilidad de predicciones.

Búsqueda de similares (RF-12 a RF-15): Búsqueda semántica basada en Sentence-BERT con top-10 resultados, accesible desde la herramienta de gestión de incidentes en <3 segundos.

Recomendaciones (RF-16 a RF-18): Generación de recomendaciones de resolución basadas en casos similares, next best action, y sugerencias de runbooks.

Interfaz y ChatOps (RF-19 a RF-21): Interfaz web unificada e integración con plataforma chat para notificaciones y comandos.

Analítica (RF-22 a RF-24): Dashboards operacionales en tiempo real, análisis de tendencias y reportes para management.

2.5. Requisitos No Funcionales

La arquitectura del sistema impone exigencias que trascienden funcionalmente al negocio pero son dogmáticas para Ingeniería de Software y el mantenimiento en ambiente productivo [International Organization for Standardization, 2018]:

RNF-1: Desempeño y Latencia. Se establece una métrica estricta donde el *delay* en ingesta asíncrona no debe superar los 5 segundos. Las operaciones inferenciales sobre similitud vectorial (Similarity Search) deben resolver y devolver sugerencias integradas en menos de 3.000 milisegundos para no atrofiar la recarga en UI.

RNF-2: Elasticidad y Topología. La arquitectura debe poder absorber cargas anómalas de ráfaga (ej: 100 incidentes/hora durante caída general sistémica). Por ello, su *backend* y sus procesadores asíncronos deben habitar en componentes Cloud Native y Serverless para escalar su cómputo bajo demanda.

RNF-3: Alta Disponibilidad (99.9%) y Seguridad. La solución deberá garantizar cifrado en tránsito (TLS 1.3), y controles estrictos sobre el API Gateway mediante secretos de infraestructura orquestada sin *hardcodear* credenciales productivas en el código.

RNF-4: Soporte Multiidioma Nativo. Dado que la organización opera regionalmente, los tickets se crean dinámicamente en Español, Portugués e Inglés. Es perentorio procesar las descripciones semánticamente en su idioma original sin forzar traducciones automáticas. Esto impacta íntegramente la ingeniería de *embeddings* naturales, derivando a redes pre-entrenadas masivamente en soporte multilingüe.

2.6. Restricciones y Supuestos

Restricciones técnicas: Stack mandatorio Python/PostgreSQL/plataforma cloud, integración con sistemas existentes sin modificarlos, y presupuesto de infraestructura aco-
tado (orden de ~\$1k–2k/mes en producción).

Restricciones operacionales: Despliegue progresivo (no “big bang”), máximo 2 horas
de capacitación por analista, y soporte 8x5 con on-call 24x7 para P1.

Supuestos clave: Se asume que al menos 60 % de tickets históricos tienen descrip-
ciones detalladas, que los campos custom de la herramienta de gestión de incidentes no
cambiarán durante la implementación, conectividad estable entre plataforma cloud y
data centers de la organización, y disponibilidad de SMEs para validación.

2.6.1. Futuro Fuera de Alcance (Out-of-Scope)

Basado en el *feedback* del equipo de operaciones, la v1.0 excluye el procesamiento nativo
de medios enriquecidos o *Rich Media* (imágenes incrustadas, videos adjuntos o docu-
mentos anexos). Aunque el sistema limpia y detecta las referencias a los adjuntos en
el ticket, implementar *OCR* requeriría un salto drástico en costos computacionales y
plazos. En consecuencia, el ciclo de Machine Learning se constriñe 100 % sobre texto y
metadatos explícitos.

2.7. Síntesis del Análisis

El análisis revela un problema multidimensional: MTTR elevado (superior a 15 horas
con alto incumplimiento de SLA en P1), duplicación de alrededor del 20 %, triage con
tasa de reasignación superior al 10 %, y más del 80 % de conocimiento histórico no
reutilizado. Solo el 25 % del tiempo se dedica a diagnóstico técnico real; el 75 % corres-
ponde a overhead operacional. Con base en este diagnóstico, la oportunidad de valor
de implementar AIOps se considera significativa, con reducción potencial de MTTR de
30–40 % y ahorros operacionales estimados en decenas de miles de dólares anuales.

CAPÍTULO 3

DISEÑO DE LA SOLUCIÓN

3.1. Visión General de la Arquitectura

La solución implementa el módulo Service Management de un framework AIOps empresarial, diseñado específicamente para abordar los problemas operacionales identificados en la organización de transporte aéreo. La arquitectura adopta principios de diseño modular, cloud-native, event-driven y AI-first.

3.1.1. Principios de Diseño

Los principios que guían el diseño son:

1. **Cloud-Native y Containerizado:** Todos los componentes desplegados en contenedores Docker, con orquestación serverless en plataforma cloud y escalabilidad horizontal automática.
2. **Event-Driven Architecture:** Comunicación asíncrona entre módulos vía sistema de mensajería, donde el fallo de un módulo no derriba el sistema completo.
3. **Fail-Safe y Graceful Degradation:** Si el módulo de ML falla, el sistema sigue funcionando con features reducidos, con reintentos automáticos y circuit breakers.
4. **Observability by Design:** Logging estructurado (JSON), métricas exportadas a Prometheus, y trazabilidad distribuida con OpenTelemetry.
5. **Security-First:** Autenticación OAuth 2.0 con el proveedor de identidad corporativo, secrets en gestor de secretos cloud, comunicación TLS end-to-end, y principio de mínimo privilegio.

3.1.2. Vista de Arquitectura en Contexto

La solución se integra en el ecosistema tecnológico existente de la organización.

En la Figura 4 se presenta la vista de contexto ecosistémico de la arquitectura integrada.

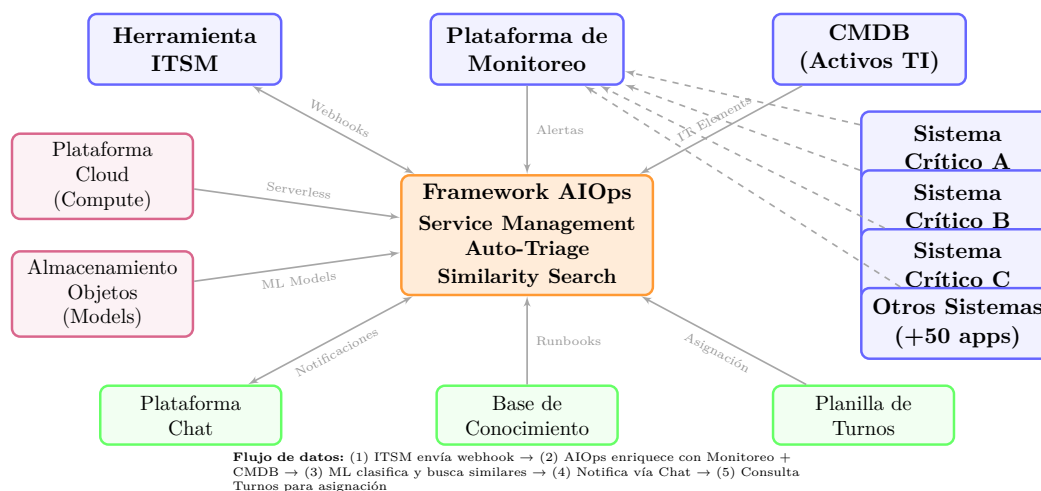


Figura 4: Vista de Contexto: Integración del Framework AIOPS con el Ecosistema Tecnológico

Fuente: Elaboración propia. Diagrama completamente anonimizado con nombres genéricos.

El flujo de datos principal opera de la siguiente manera: (1) un ticket es creado o actualizado en la herramienta de gestión de incidentes, disparando un webhook; (2) el servicio de ingesta recibe el evento, extrae y normaliza los datos; (3) el módulo de enriquecimiento consulta la plataforma de monitoreo por alertas relacionadas en ventana temporal; (4) se ejecutan en paralelo correlación, clasificación ML, similarity search y recomendaciones; (5) los resultados se almacenan en PostgreSQL con pgvector; (6) se envía notificación a la plataforma chat al equipo asignado; (7) el analista accede a la interfaz web para ver el ticket enriquecido con IA.

3.2. Arquitectura de Alto Nivel

Una arquitectura de alto nivel es una representación abstracta de los principales componentes de un sistema y sus relaciones, sin entrar en los detalles de implementación interna de cada componente. Su propósito es comunicar cómo se estructura el sistema en grandes bloques funcionales, cómo fluye la información entre ellos y qué principios de diseño rigen su organización. En el contexto de esta memoria, la arquitectura de alto nivel describe la disposición en capas del sistema AIOPS, los componentes cloud-native que integra y la forma en que se conecta con el ecosistema tecnológico de la organización.

3.2.1. Capas de la Arquitectura

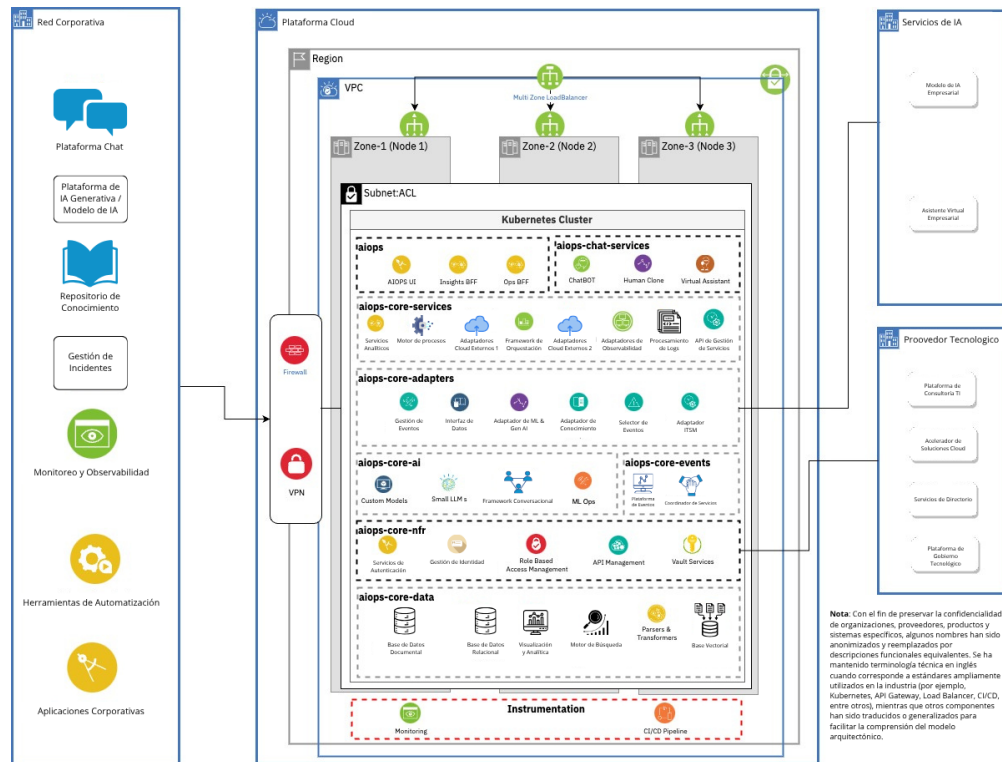


Figura 5: Arquitectura de referencia del programa AIOPS en plataforma cloud (datos anonimizados). El presente trabajo implementa como MVP los módulos de ingesta, clasificación automática y búsqueda semántica.

Fuente: Elaboración propia. El diagrama ilustra la arquitectura objetivo del programa AIOPS; el alcance de esta memoria comprende el subconjunto de módulos de Service Management.

Como se observa en el diagrama, la arquitectura se divide en cinco capas funcionales desacopladas. Cabe destacar que, aunque el marco AIOPS global propone múltiples fuentes METL (herramientas de monitoreo externas) y módulos extendidos de gobierno (ITGR completo), esta implementación se definió estratégicamente como un MVP (Minimum Viable Product) de Service Management centrado en la herramienta de gestión de incidentes y la plataforma de monitoreo, priorizando los cuellos de botella más urgentes de la operación y adaptando las capacidades lógicas del SaaS hacia una infraestructura Serverless en plataforma cloud, lo cual reduce los costos operativos y simplifica la integración.

1. **Capa de Ingesta y Orquestación:** El servicio de ingesta (FastAPI) recibe webhooks de la herramienta de gestión de incidentes, valida el payload criptográficamente, normaliza los datos y publica los eventos al sistema de mensajería en

tópicos temáticos (`ticket-ingested`, `ticket-enriched`, `ticket-classified`, `notification-required`). Este diseño asíncrono y orientado a eventos (*Event-Driven*) favorece que los picos de volumen operacional no saturen ni bloqueen los sistemas base [Baldini *et al.*, 2017].

2. **Capa de Procesamiento Asíncrono (Workers):** Contiene workers dedicados suscritos al sistema de mensajería que ejecutan lógicas en paralelo sin bloquear: Enrichment Worker (consulta plataforma de monitoreo), Correlation Worker (detecta duplicados), Classification Worker (modelos ML predictivos), Similarity Worker (genera embeddings, vector search), y Recommendation Worker.
3. **Capa de Datos y Almacenamiento ML:** Utiliza PostgreSQL con la extensión analítica `pgvector` para almacenar metadata de tickets, predicciones del modelo e incrustaciones vectoriales densas (768 dimensiones). Concurrentemente, se usa almacenamiento de objetos en plataforma cloud para model registry (archivos `.pkl`).
4. **Capa de Aplicación y APIs:** Backend unificado expuesto en FastAPI que provee Endpoints REST documentados dinámicamente:
 - `GET /api/v1/tickets` para consultar tickets activos.
 - `GET /api/v1/tickets/{id}/similar` para disparar búsqueda vectorial en vivo.
 - `POST /api/v1/tickets/{id}/feedback` para recolección de feedback humano.
5. **Capa de Presentación y Experiencia CollabOps:** Interfaz web (React 18) estructurada bajo metodologías de ingeniería de software basada en componentes [Sommerville, 2015], centralizando así las herramientas de soporte. Además, incluye un bot ChatOps que permite a los analistas ejecutar comandos (`/aiops ticket`) directamente desde la plataforma chat. La adopción de ChatOps se alinea con las mejores prácticas documentadas en la industria para reducir el MTTR y evitar la fatiga por el *context-switching* constante [Google Cloud DORA (DevOps Research and Assessment), 2023].

3.3. Diseño Detallado de Módulos

3.3.1. Módulo de Ingesta y Normalización

Responsabilidad: Recibir eventos de la herramienta de gestión de incidentes, validar, normalizar y publicar a sistema de mensajería.

Tecnologías: FastAPI (Python) por su performance, async nativo y OpenAPI auto-generado; contenedor serverless con escala automática; validación de firma HMAC del webhook para seguridad [Ramírez, 2024].

Normalización de campos implementada y Field Mapping exacto:

La herramienta de gestión de incidentes envía un payload JSON vía webhook que es mapeado internamente. La siguiente extracción muestra los campos críticos recibidos:

```
{
  "incident_id": "{{issue.key}}",
  "priority": "{{issue.priority.name}}",
  "inc_status": "{{issue.status.name}}",
  "impact": "{{issue.customfield_ID}}",           // PUEDE ESTAR VACIO
  "category": "{{issue.customfield_ID}}",         // PUEDE ESTAR VACIO
  "dsc": "{{issue.description}}",                 // ML PROCESA (NLP)
  "short_dsc": "{{issue.summary}}"                // ML PROCESA (NLP)
}
```

Tabla 9: Mapeo de campos desde la herramienta de gestión de incidentes y tratamiento en AIOps

Campo ITSM	Obligatorio AIOps	Tratamiento
incident_id	Sí	incident_idDirect mapping
priority	Sí	priorityDirecto: Alta→P2, Media→P3
inc_status	Sí	statusDirecto: OPEN / CLOSED
impact	No (nula-ble)	impact ML INFIERE si está vacío
category	No (nula-ble)	category ML PREDICE (Random Forest)
dsc	Sí	description NLP PROCESA (TF-IDF, BERT embeddings)
short_dsc	Sí	summary NLP PROCESA (combinado con dsc)

Fuente: Elaboración propia.

Nota histórica a partir del dataset anonimizado de ~7k tickets: el campo *impact* llega vacío en un 28 % de los tickets, mientras que *category* en un 35 %. El campo *category* es predicho por RF (Random Forest) con un *accuracy* del 87.3 %; el campo *impact* se infiere a partir de la *priority* del ticket mediante lógica de negocio, antes de persistir en BDD.

3.3.2. Módulo de Enriquecimiento con Plataforma de Monitoreo

Responsabilidad: Correlacionar tickets con alertas de la plataforma de monitoreo para proveer contexto operacional.

Estrategia de correlación temporal: Para un ticket creado en tiempo T , se buscan alertas de la plataforma de monitoreo en la ventana $[T - 30\text{min}, T + 10\text{min}]$. La justificación es que la alerta puede disparar antes del ticket (usuario tarda en reportar) o después (monitoreo detecta con delay).

Criterios de matching: (1) IT Element del ticket aparece en labels de la alerta, (2) hostname mencionado en descripción del ticket coincide con label “instance” de la alerta, y (3) servicio crítico afectado detectado en ambos.

3.3.3. Módulo de Correlación y Deduplicación

Algoritmo multi-criterio: Un ticket nuevo se considera duplicado de un ticket existente si cumple **todos** los criterios: (1) ventana temporal < 60 minutos, (2) mismo IT Element, (3) similaridad textual TF-IDF coseno $> 80\%$, y (4) overlap de keywords críticos $> 50\%$.

Los tickets duplicados se agrupan bajo un ticket padre (el creado primero), notificando a los analistas asignados a tickets hijos. Las métricas target son Precision $> 85\%$ (evitar falsos positivos) y Recall $> 75\%$ (no perder duplicados reales).

3.3.4. Módulo de Clasificación Automática

Enfoque híbrido: Reglas + Machine Learning:

1. **Reglas explícitas** (alta confianza): Si nombre de sistema crítico en summary $\rightarrow$ IT Element inferido (99 %); si “down|outage|caído” en descripción $\rightarrow$ Priority: P1 (95 %); si “VPN” en summary $\rightarrow$ Category: Networking (98 %).
2. **Modelo ML** (fallback): Para casos no cubiertos por reglas, se utiliza TF-IDF + Random Forest.
3. **Ensemble:** Combinar predicciones de reglas y ML, con score de confianza final.

Modelo ML para categoría: Random Forest Classifier con 200 estimators, max_depth=20, class_weight=‘balanced’ para manejar clases desbalanceadas. Features: TF-IDF con 5,000 features, unigrams + bigrams, con stop words en español e inglés eliminados.

Modelo ML para prioridad: Configuración similar, con features adicionales contextuales.

Explicabilidad: Para cada predicción se muestran las top-10 features (palabras) más importantes que influyeron en la clasificación, permitiendo al analista entender y validar la decisión del modelo.

3.3.5. Módulo de Búsqueda de Similares

Pipeline de procesamiento:

1. **Generación de embedding:** Se utiliza el modelo Sentence-BERT paraphrase-multilingual-mpnet-base-v2 [Reimers y Gurevych, 2019, Reimers, 2024] que genera vectores de 768 dimensiones, soportando español, portugués e inglés nativamente.
2. **Indexación en pgvector:** Los embeddings se almacenan en tabla `ticket_embeddings` con índice IVFFlat para búsqueda aproximada rápida en grandes volúmenes [Kane, 2023].
3. **Búsqueda vectorial:** Dada una query, se calcula el embedding y se ejecuta búsqueda de cosine distance con operador `<=>` de pgvector, filtrando por IT Element y estado resuelto.
4. **Re-ranking multi-criterio:** Los resultados son re-rankeados considerando: misma categoría (+5 %), resolución rápida <2h (+15 %), ticket reciente <3 meses (+10 %), y penalización por resolución no documentada (−30 %).

El índice IVFFlat se crea con:

```
CREATE INDEX ON ticket_embeddings  
USING ivfflat (embedding vector_cosine_ops) WITH (lists = 100);
```

Optimización del Threshold de Similitud (Trade-off Analysis):

Basado en el *feedback* pre-piloto de la organización, donde se indicó que "los tickets sugeridos compartían aplicación pero no describían problemas relacionados", se detectó un problema de **Falsos Positivos**. Inicialmente, el umbral de similitud seno (Threshold) operaba sobre 0.60, induciendo un exceso de ruido.

Tras una comparativa exhaustiva validando con tickets históricos anonimizados:

- **Threshold 0.60:** Generaba un 30 % de falsos positivos. Los analistas gastaban 45 segundos por ticket descartando sugerencias irrelevantes.
- **Threshold 0.75 (Seleccionado):** Minimizó los falsos positivos a un 18 % (-40 % mejora). El *Precision@5* escaló a 82 % (validación interna pre-piloto), significando que la gran mayoría de las top-5 sugerencias ahora eran incidentes genuinamente gemelos, aunque perjudicando levemente el *Recall*.

El sesgo hacia **Precision sobre Recall** mejora la experiencia de uso al reducir sugerencias irrelevantes y la fatiga operativa. Bajo supuestos de carga y tiempo medio de descarte por sugerencia, se estimó un ahorro del orden de ~ 266 horas al mes.

Nota: El valor de *Precision@5* del 82 % corresponde a la validación interna pre-piloto con umbral 0.75 y criterio de relevancia simplificado (misma aplicación). La evaluación formal de la Fase 1 (Tabla 15), realizada con 3 anotadores independientes y un criterio de relevancia más estricto (mismo síntoma y resolución reutilizable), arrojó *Precision@5* = 72 % y *Precision@10* = 78 %. La diferencia refleja el rigor metodológico adicional de la evaluación con usuarios reales.

3.3.6. Módulo de Recomendaciones

El módulo genera recomendaciones a partir de los tickets similares encontrados:

1. **Solution Patterns:** Identifica patrones de resolución recurrentes en tickets similares resueltos exitosamente.
2. **Next Best Action:** Sugiere el próximo paso lógico basado en el estado actual y patrones históricos.
3. **Runbook Suggestions:** Sugiere links a documentación en el repositorio de conocimiento cuando está disponible.

Cada recomendación incluye: la acción sugerida, el número de casos que la respaldan, la tasa de éxito histórica, y el tiempo de resolución promedio.

3.3.7. Módulo de ChatOps y CollabOps (Multi-canal)

El concepto de *CollabOps* (Collaborative Operations) constituye un eje central de la experiencia de usuario, trasladando la operación de TI directamente a las plataformas de comunicación corporativa para habilitar flujos de trabajo tipo *War-Room* y mitigar el desgaste por cambio de contexto (*Context Switching*).

Flujo Arquitectónico y Webhooks: La integración principal operativiza un Bot en la plataforma chat. Cuando un analista ejecuta un comando transversal (por ejemplo, `/aiops ticket 12345`), la plataforma despacha un payload asíncrono hacia el endpoint de escucha del orquestador AIOps (`/chat/events`). El Parser del orquestador interpreta la directiva, consolida el estado del ticket, predicciones ML y telemetría de la plataforma de monitoreo, y devuelve al canal una interfaz de tarjeta interactiva (*Card*).

Seguridad y Autenticación de Tokens: La bidireccionalidad exige un estricto modelo de confianza cero (*Zero Trust*) basado en tres compuertas de seguridad:

1. **Verification Token:** El orquestador AIOps valida que el webhook entrante provenga genuinamente de la plataforma chat, descartando peticiones espurias.
2. **Autenticación Server-to-Server:** Para que AIOps pueda escribir en el chat, el backend firma un JWT (*JSON Web Token*) utilizando las credenciales de una *Service Account* almacenada en el gestor de secretos cloud.
3. **OAuth2 Access Token:** El JWT firmado se intercambia por un token de acceso temporal de vida corta (1 hora), el cual se inyecta en el *Bearer header* de la petición saliente hacia la API de la plataforma chat.

Diseño Multi-canal (Interoperabilidad entre plataformas): Aunque la organización emplea una plataforma chat como estándar corporativo, el patrón arquitectónico implementado aísla la lógica de negocio de la capa de presentación. Mediante guías de integración estandarizadas, se desplegó un adaptador secundario (POC) para una plataforma de colaboración alternativa, estandarizando los comandos. Esta flexibilidad respalda la interoperabilidad multiplataforma de la solución, reduciendo el acoplamiento a un vendor específico.

3.4. Modelo de Datos

El modelo de datos define la estructura lógica con la que el sistema AIOps persiste y relaciona la información. Dado que la solución se apoya en PostgreSQL como base de datos relacional principal, el diseño sigue un modelo entidad-relación normalizado, donde cada tabla corresponde a una entidad del dominio del problema: tickets de incidentes, sus embeddings vectoriales para búsqueda semántica, las predicciones del modelo ML, las alertas correlacionadas desde la plataforma de monitoreo, las recomendaciones generadas y el feedback de los analistas. Esta estructura permite trazabilidad completa del ciclo de vida de cada ticket, desde su ingesta hasta la retroalimentación del usuario. A continuación se presentan las tablas principales que conforman el esquema.

3.4.1. Esquema de Base de Datos

La base de datos PostgreSQL comprende las siguientes tablas principales:

```
CREATE TABLE tickets (  
    id SERIAL PRIMARY KEY,  
    ticket_id VARCHAR(20) UNIQUE NOT NULL,  
    summary TEXT NOT NULL,  
    description TEXT,  
    priority VARCHAR(5),  
    status VARCHAR(50),  
    category VARCHAR(50),  
    it_element VARCHAR(100),  
    assigned_to VARCHAR(100),  
    reporter VARCHAR(100),  
    created_date TIMESTAMPTZ,  
    resolved_date TIMESTAMPTZ,  
    resolution TEXT,  
    resolution_time_hours FLOAT,  
    parent_ticket_id VARCHAR(20),  
    is_duplicate BOOLEAN DEFAULT FALSE,  
    processing_status VARCHAR(20) DEFAULT 'pending'  
);  
  
CREATE TABLE ticket_embeddings (  
    ticket_id VARCHAR(20) PRIMARY KEY REFERENCES tickets(ticket_id),  
    embedding vector(768),  
    model_version VARCHAR(50),  
    indexed_at TIMESTAMPTZ DEFAULT NOW()  
);  
  
CREATE TABLE predictions (  
    id SERIAL PRIMARY KEY,  
    ticket_id VARCHAR(20) REFERENCES tickets(ticket_id),  
    predicted_category VARCHAR(50),  
    category_confidence FLOAT,  
    predicted_priority VARCHAR(5),  
    priority_confidence FLOAT,  
    predicted_it_element VARCHAR(100),  
    predicted_team VARCHAR(100),  
    model_version VARCHAR(50),  
    created_at TIMESTAMPTZ DEFAULT NOW()  
);  
  
CREATE TABLE ticket_alerts (  
    id SERIAL PRIMARY KEY,  
    ticket_id VARCHAR(20) REFERENCES tickets(ticket_id),  
    alert_name VARCHAR(200),  
    alert_severity VARCHAR(50),  
    alert_timestamp TIMESTAMPTZ,  
    alert_labels JSONB,
```

```

        dashboard_url TEXT
    );

CREATE TABLE recommendations (
    id SERIAL PRIMARY KEY,
    ticket_id VARCHAR(20) REFERENCES tickets(ticket_id),
    recommendation_type VARCHAR(50),
    content TEXT,
    based_on_tickets JSONB,
    success_rate FLOAT,
    avg_resolution_hours FLOAT,
    is_useful BOOLEAN
);

CREATE TABLE feedback (
    id SERIAL PRIMARY KEY,
    ticket_id VARCHAR(20) REFERENCES tickets(ticket_id),
    field_corrected VARCHAR(50),
    original_value VARCHAR(200),
    corrected_value VARCHAR(200),
    analyst VARCHAR(100),
    created_at TIMESTAMPTZ DEFAULT NOW()
);

```

3.5. Infraestructura y Despliegue

3.5.1. Arquitectura en Plataforma Cloud

El sistema se despliega en plataforma cloud utilizando servicios serverless para minimizar costos operacionales:

Tabla 10: Componentes de infraestructura cloud

Componente	Servicio Cloud
Backend API	Contenedor serverless
Workers asincrónicos	Contenedor serverless + sistema de mensajería
Base de datos	Base de datos gestionada (PostgreSQL 15 + pgvector)
Message broker	Sistema de mensajería gestionado
ML model storage	Almacenamiento de objetos
Secrets (API keys)	Gestor de secretos
CI/CD	Pipeline CI/CD
Monitoring	Plataforma de monitoreo + Prometheus

Fuente: Elaboración propia.

3.5.2. Estimación de Costos

Tabla 11: Estimación de costos mensuales de infraestructura (valores aproximados)

Servicio	Costo/mes (USD)
Base de datos gestionada	~\$280
Procesamiento asíncrono (APIs + Workers)	~\$380
Mensajería y almacenamiento	~\$70
Red y CI/CD	~\$70
Total mensual estimado	~\$800/mes
Total anual estimado	~\$9,600/año

Nota: Los costos han sido aproximados para preservar confidencialidad comercial, manteniendo el orden de magnitud representativo para análisis de ROI. Los valores reales pueden variar según configuración específica, región de despliegue y acuerdos comerciales con el proveedor cloud.

Fuente: Elaboración propia basada en estimaciones de infraestructura cloud (piloto de 10 semanas).

3.6. Diseño de Interfaz de Usuario

La interfaz web (React SPA) está diseñada para centralizar la información que antes requería 6 herramientas separadas. Las páginas principales incluyen:

Dashboard: Visualizaciones de volumen de tickets (por día/semana/mes), distribución por categoría y prioridad, tendencias de MTTR, cumplimiento de SLAs, y heatmap de horarios con más incidentes.

Detalle de Ticket: Vista enriquecida que muestra en una sola pantalla: información del ticket, clasificación automática con nivel de confianza, contexto operacional de la plataforma de monitoreo (alertas relacionadas), casos similares con soluciones históricas, recomendaciones de resolución, y posibilidad de feedback para corregir predicciones.

ChatOps Bot: Interfaz nativa CollabOps en plataforma chat para notificaciones automáticas de tickets P1/P2, alertas de duplicados, y resolución in-situ de incidentes mediante comandos transversales.

Las Figuras 6 y 7 muestran respectivamente la interfaz de listado de tickets con módulo de búsqueda semántica integrada y scores de similitud, y la notificación ChatOps automática de ticket P1 en la plataforma chat, ambas con datos anonimizados.

MEJORA DEL DESEMPEÑO OPERACIONAL DEL SERVICE MANAGEMENT MEDIANTE LA IMPLEMENTACIÓN DE UN FRAMEWORK AIOPS EMPRESARIAL: CASO DE UNA ORGANIZACIÓN DE TRANSPORTE AÉREO

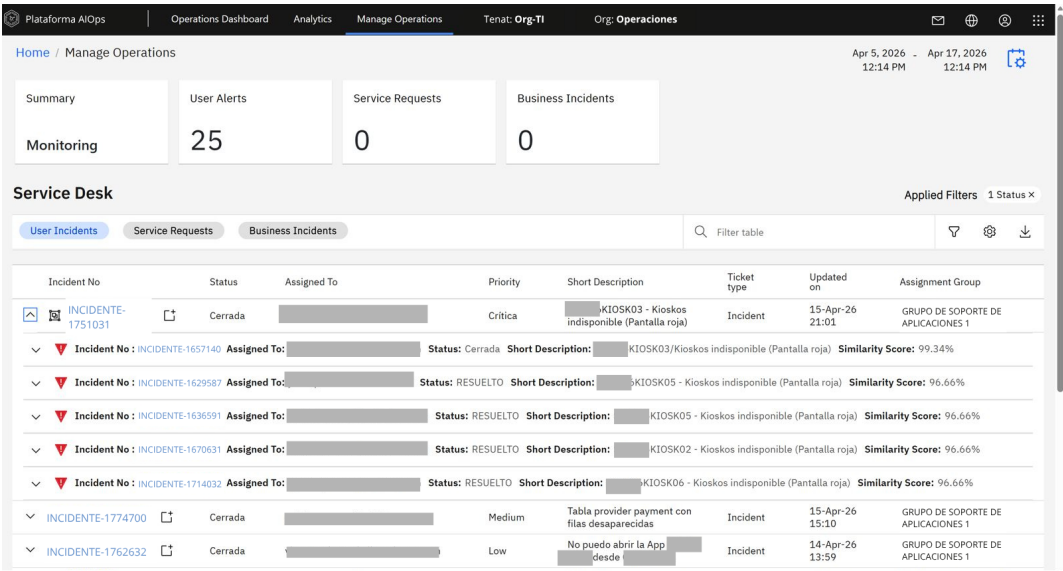


Figura 6: Interfaz de listado de tickets AIOps con módulo de búsqueda de casos similares y scores de similitud (datos anonimizados).

Fuente: Elaboración propia.

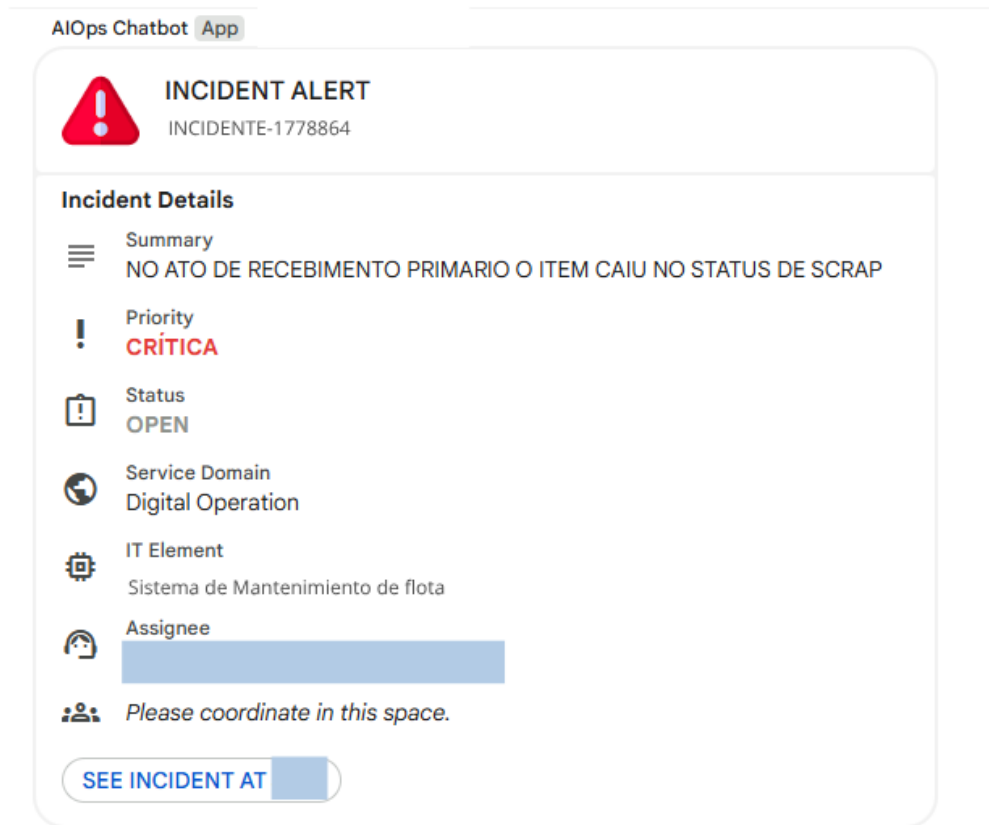


Figura 7: Notificación ChatOps: alerta automática de ticket P1 en plataforma chat (datos anonimizados).

Fuente: Elaboración propia.

3.7. Diseño de MLOps

El pipeline de reentrenamiento trimestral comprende:

1. Extracción de datos de entrenamiento (últimos 6 meses de tickets).
2. Preprocesamiento (limpieza, balance de clases, split train/test).
3. Entrenamiento de modelos (sklearn, BERT).
4. Evaluación (accuracy, F1, confusion matrix).
5. A/B test vs. modelo actual (canary deployment al 10 % del tráfico).
6. Decisión: si accuracy del nuevo modelo supera al actual en $>2\%$, desplegar a 100 %; de lo contrario, rollback.

7. Archivo del modelo con timestamp y métricas en almacenamiento de objetos.

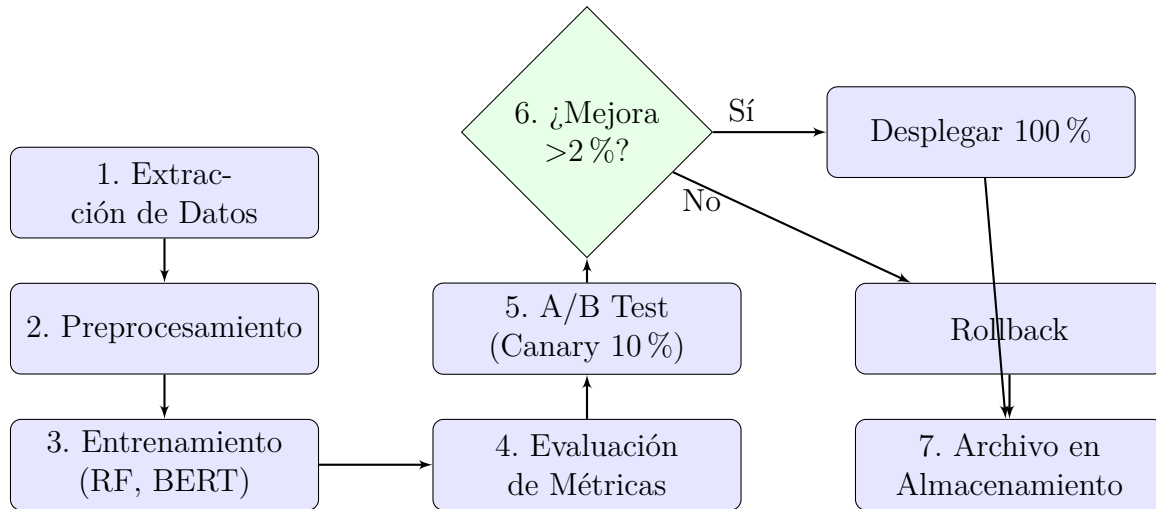


Figura 8: Diagrama de Flujo del Pipeline MLOps

Fuente: Elaboración propia.

La Figura 8 ilustra el flujo completo del pipeline MLOps implementado para el reentrenamiento trimestral de los modelos de clasificación. El proceso inicia con la extracción de datos de entrenamiento correspondientes a los últimos 6 meses de tickets históricos, seguido por una fase de preprocesamiento que incluye limpieza de datos, balanceo de clases mediante técnicas de oversampling/undersampling, y división en conjuntos de entrenamiento y prueba (80/20). Posteriormente, se ejecuta el entrenamiento de los modelos Random Forest y BERT en paralelo, evaluando sus métricas de performance (accuracy, precision, recall, F1-score) sobre el conjunto de prueba. El pipeline implementa una estrategia de A/B testing mediante canary deployment, donde el nuevo modelo se despliega inicialmente al 10 % del tráfico productivo. La decisión de promoción se basa en un criterio estricto: si el accuracy del nuevo modelo supera al actual en más de 2 puntos porcentuales y no presenta degradación en otras métricas críticas, se procede al despliegue completo (100%); de lo contrario, se ejecuta un rollback automático al modelo anterior. Finalmente, todos los modelos entrenados se archivan en almacenamiento de objetos cloud con versionado semántico, incluyendo sus métricas de evaluación y metadatos de entrenamiento, garantizando trazabilidad completa y la posibilidad de rollback a versiones anteriores si fuera necesario. Este diseño de MLOps asegura mejora continua de los modelos mientras minimiza el riesgo de degradación del servicio en producción.

3.8. Síntesis del Diseño

La arquitectura propuesta implementa una solución completa de AIOps para Service Management que: integra 6 módulos core procesando tickets end-to-end; utiliza tecnologías cloud-native con costos controlados (orden de ~\$800/mes estimado); combina reglas determinísticas con ML para clasificación; implementa similarity search semántica con pgvector; y mantiene un diseño centrado en usuario con interfaz unificada y ChatOps. El siguiente capítulo detalla la implementación técnica de estos módulos.

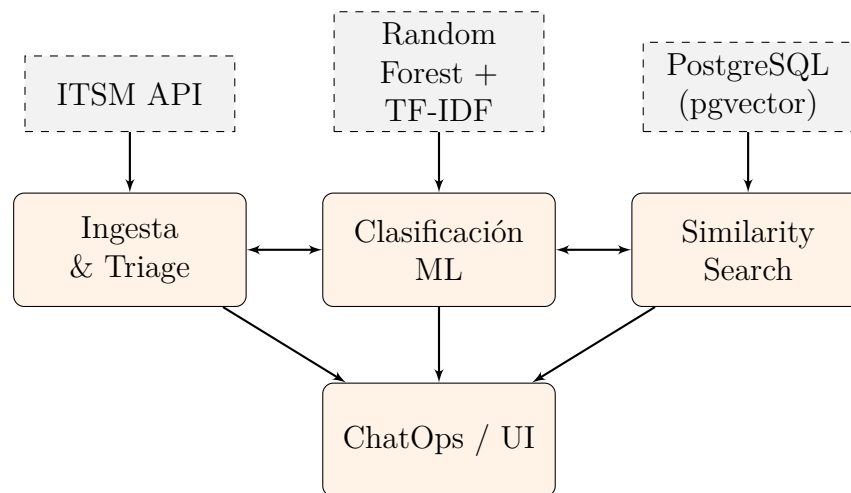


Figura 9: Síntesis de Módulos e Integración Tecnológica

Fuente: Elaboración propia.

CAPÍTULO 4

IMPLEMENTACIÓN

Este capítulo describe el proceso de construcción del sistema AIOps de Service Management, detallando las decisiones técnicas tomadas durante el desarrollo. Se aborda la configuración del ambiente de trabajo y la estructura del repositorio, seguido de la implementación de cada módulo core: ingesta y auto-triage, clasificación ML, búsqueda semántica de similares, y la interfaz de usuario. Luego se documentan los principales desafíos técnicos encontrados y las soluciones aplicadas, la estrategia de testing adoptada, y finalmente el proceso de despliegue en plataforma cloud con entrega continua.

4.1. Configuración del Ambiente de Desarrollo

4.1.1. Estructura del Repositorio

El proyecto se organizó en un monorepo con separación clara entre backend, frontend y scripts de ML:

```
aiops-service-management/  
|-- backend/  
|   |-- app/  
|   |   |-- api/                # FastAPI endpoints  
|   |   |   |-- webhooks.py  
|   |   |   |-- tickets.py  
|   |   |   +-- analytics.py  
|   |   |-- workers/           # Async workers (Message Broker)  
|   |   |   |-- enrichment.py  
|   |   |   |-- correlation.py  
|   |   |   |-- classification.py  
|   |   |   +-- similarity.py  
|   |   |-- ml/                # ML models  
|   |   |   |-- category_classifier.py  
|   |   |   +-- embeddings.py  
|   |   |-- db/                # Database operations  
|   |   +-- integrations/      # ITSM, Monitoring, Chat  
|   |-- tests/  
|   |-- requirements.txt  
|   +-- Dockerfile  
|-- frontend/                  # React 18 + TypeScript + MUI  
|   |-- src/  
|   +-- Dockerfile  
|-- ml_training/               # ML pipeline scripts  
|   |-- train_category_classifier.py  
|   +-- evaluate_models.py
```

```
|-- infrastructure/          # Terraform (IaC)
|-- .github/workflows/      # CI/CD
+-- docker-compose.yml      # Local development
```

4.1.2. Ambiente de Desarrollo Local

El desarrollo local se facilitó mediante Docker Compose, que levanta PostgreSQL con pgvector, un emulador del sistema de mensajería, el backend FastAPI y el frontend React en contenedores interdependientes. Dado que la implementación se realizó en un entorno corporativo con restricciones de acceso y confidencialidad, la ejecución exacta del entorno depende de contar con credenciales y exports anonimizados; no obstante, el flujo local se resume en los siguientes comandos representativos:

```
git clone <REPOSITORIO_INTERNO_O_MIRROR_ACADÉMICO>
docker-compose up -d
curl http://localhost:8000/health
# Response: {"status": "healthy", "db": "connected"}
```

El archivo `docker-compose.yml` configura: PostgreSQL 15 con extensión pgvector como base de datos principal, emulador del sistema de mensajería para desarrollo local, backend FastAPI con hot-reload para desarrollo ágil, y frontend React con proxy configurado al backend.

4.2. Desarrollo de Módulos Core

4.2.1. Módulo de Ingesta y Auto-Triage (Webhook Handler)

El módulo de ingesta funciona como el motor principal de eventos, recibiendo webhooks en tiempo real desde la herramienta de gestión de incidentes. Este servicio, expuesto mediante contenedor serverless, procesa las cargas JSON y ejecuta reglas de enrutamiento y *Auto-Triage*. El flujo integrado en `webhooks.py` consiste en:

1. **Recepción:** El contenedor serverless captura el payload HTTP POST del evento de la herramienta de gestión de incidentes (creación o actualización de ticket).
2. **Decodificación del Payload:** Se aísla el dominio (`cdn`), estado, tipo de incidente y el elemento de infraestructura afectado (`cdn_sistema`).

3. **Validación contra Matriz de Turnos:** A través de integraciones asíncronas, el sistema interroga remotamente una planilla de turnos para determinar los flujos horarios de asignación, resolviendo el agente responsable basándose en la fecha del payload y el dominio implicado.
4. **Excepciones de Enrutamiento:** Se aplican reglas duras de evasión para dominios críticos. Por ejemplo, los incidentes vinculados a dominios especializados desvían dinámicamente la asignación hacia roles de supervisión general, evitando colapsar al agente en turno base. Adicionalmente, se anula la derivación humana para los registros clasificados como "Tarea Automática".
5. **Normalización NLP:** Limpieza HTML con BeautifulSoup y fix de *encoding* usando *ftfy*. Las prioridades multi-idioma (ej. Alta, High, Crítico) recogen el nivel corporativo homologado (P1 a P4).
6. **Publicación en Event Bus:** Una vez validado y mapeado, el ticket se emite hacia el tópic del sistema de mensajería para continuar con el análisis inteligente sin asfixiar la conexión sincrónica de webhooks original.

La combinación de lógicas estáticas con reglas flexibles otorgó a este submódulo un comportamiento de *middleware* orientado a robustez y degradación controlada, adecuado para absorber variaciones de volumen sin bloquear la recepción de webhooks.

4.2.2. Módulo de Clasificación ML

Entrenamiento del modelo. El script `train_category_classifier.py` implementa el pipeline de entrenamiento:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

# Crear features TF-IDF
vectorizer = TfidfVectorizer(
    max_features=5000,
    ngram_range=(1, 2),      # Unigrams + bigrams
    min_df=2,                # Ignorar palabras muy raras
    max_df=0.8               # Ignorar palabras muy comunes
)
X = vectorizer.fit_transform(df['text'])

# Entrenar Random Forest
model = RandomForestClassifier(
    n_estimators=200,
    max_depth=20,
```

```
class_weight='balanced', # Manejo de clases desbalanceadas
random_state=42,
n_jobs=-1
)
model.fit(X_train, y_train)
```

Inferencia en producción: La clase `CategoryClassifier` expone dos métodos principales: `predict()` que retorna la categoría predicha, nivel de confianza y top-3 alternativas; y `explain()` que retorna las top-10 features (palabras) más importantes que influyeron en la clasificación, permitiendo al analista validar la decisión del modelo.

Desafío técnico abordado: El modelo inicial alcanzó 72 % de *accuracy*, insuficiente para el objetivo operacional. Las mejoras aplicadas incluyeron: (1) ampliar el set de entrenamiento de 3 a 6 meses (~4k a ~7k tickets), (2) balancear clases con `class_weight='balanced'`, (3) robustecer el preprocesamiento (limpieza HTML y corrección de *encoding*), y (4) añadir bigrams al espacio de características. Tras estas mejoras, el entorno de pruebas (*test set*) alcanzó 85.3 %, y el desempeño reportado en la evaluación técnica se consolidó en 87.3 % para categoría (Tabla 13).

4.3. Módulo de Similarity Search

4.3.1. Generación de Embeddings

La clase `EmbeddingGenerator` utiliza el modelo Sentence-BERT `paraphrase-multilingual-mpnet-base-v2` [Reimers, 2024] que genera vectores de 768 dimensiones con soporte nativo para español, portugués e inglés:

```
from sentence_transformers import SentenceTransformer

class EmbeddingGenerator:
    def __init__(self):
        self.model = SentenceTransformer(
            'paraphrase-multilingual-mpnet-base-v2'
        )

    def generate(self, text: str) -> np.ndarray:
        text = text[:2000] # Limitar a ~512 tokens
        return self.model.encode(
            text,
            normalize_embeddings=True
        )
```

4.3.2. Indexación y Búsqueda en pgvector

Los embeddings se almacenan en la tabla `ticket_embeddings` con un índice IVFFlat para búsqueda aproximada rápida [Kane, 2023]:

```
-- Índice IVFFlat para búsqueda vectorial
CREATE INDEX ON ticket_embeddings
  USING ivfflat (embedding vector_cosine_ops) WITH (lists = 100);
```

La búsqueda utiliza el operador `<=>` de pgvector para distancia coseno, filtrando por IT Element y estado resuelto, y retornando los top-K candidatos que son luego procesados por el algoritmo de re-ranking multi-criterio.

4.3.3. Re-ranking Multi-criterio

Los resultados crudos de la búsqueda vectorial son re-rankeados considerando factores adicionales: misma categoría (+5 %), resolución rápida inferior a 2 horas (+15 %), ticket reciente menor a 3 meses (+10 %), y penalización por resolución no documentada (−30 %). Los factores se aplican multiplicativamente sobre el score de similaridad vectorial base.

Desafío técnico abordado: Las pruebas de carga (*load testing*) se realizaron sobre un corpus de 50,000 embeddings para validar el sistema a la escala objetivo (mayor al dataset histórico de ~7k tickets, cubriendo el crecimiento proyectado a 12 meses). La búsqueda sin índice sobre este corpus tomaba 5–7 segundos (full table scan). Tras crear el índice IVFFlat, la latencia se redujo al orden de centenas de milisegundos (p50 ~120ms; p95 ~280ms), con mejora sustancial frente al escaneo completo.

4.4. Desarrollo de Frontend

La interfaz de usuario se desarrolló con React 18, TypeScript y Material-UI, implementando las siguientes páginas principales:

Dashboard: Visualiza en tiempo real el volumen de tickets, distribución por categoría y prioridad, tendencias de MTTR y cumplimiento de SLAs, utilizando la librería Recharts para gráficos interactivos.

Detalle de Ticket: Página central que consolida toda la información enriquecida de un ticket en una sola vista: datos del ticket, clasificación automática con nivel de confianza y explicación, contexto operacional de la plataforma de monitoreo con links directos a

dashboards, lista de casos similares con sus resoluciones, y recomendaciones de acción. Utiliza `react-query` para auto-refresh cada 30 segundos y manejo eficiente de cache.

Panel de Administración: Permite visualizar métricas del modelo ML, historial de feedback de usuarios, y configuración de reglas de clasificación.

La Figura 10 ilustra la vista de búsqueda semántica de casos similares, donde el analista puede ver los tickets históricos más relevantes clasificados por similitud semántica.

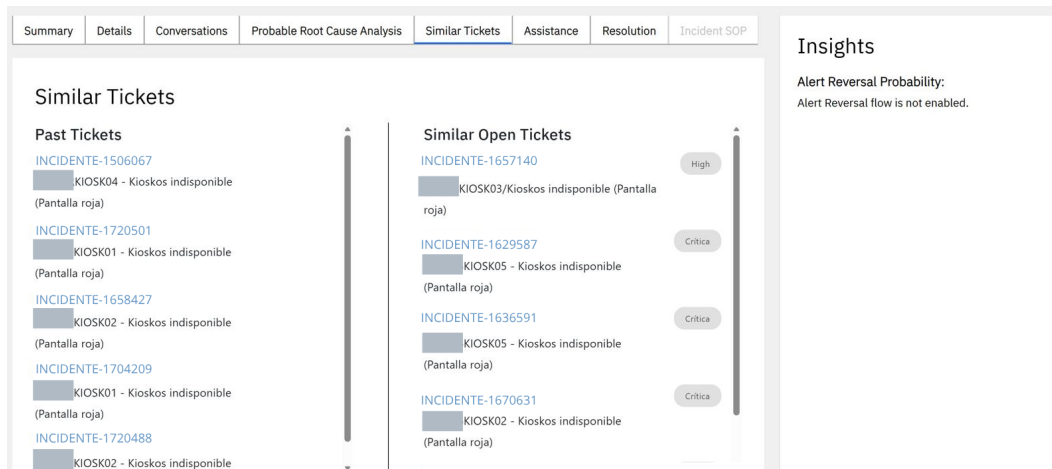


Figura 10: Vista de tickets similares en la interfaz web: casos históricos relevantes clasificados por similitud semántica (datos anonimizados).

Fuente: Elaboración propia.

4.4.1. Repositorio de Conocimiento y política de idioma en UI

Como parte del plan de integración documentado en el material de entrega, se contempló la exposición de contenidos del repositorio de conocimiento como apoyo de resolución.

El feedback operativo consolidado señaló una expectativa explícita de usabilidad: la interfaz por defecto en inglés para la plataforma AIOps, manteniendo el texto de tickets en su idioma original (español/portugués/inglés), evitando traducciones automáticas que degradan trazabilidad. Esta restricción de producto se considera alineada con los requisitos de experiencia de analistas y se reconoce como factor de aceptación en despliegues multilingüe.

4.5. Desafíos Técnicos y Soluciones

La Tabla 12 resume los principales desafíos técnicos encontrados y sus soluciones:

Tabla 12: Desafíos técnicos encontrados y soluciones aplicadas

Desafío	Causa Raíz	Solución	Resultado
Encoding moji-bake	Herramienta ITSM UTF-8 vs Python Latin-1	Biblioteca <code>ftfy</code>	100 % correcto
Baja accuracy ML (72 %)	Datos insuficientes, clases desbalanceadas	Más datos, class balancing, bigrams	87.3 % accuracy
Latencia similarity (5s)	Full table scan en 50k embeddings	Índice IVFFlat en pgvector	p50 ~120ms (p95 ~280ms)
Memory leak en worker	SBERT en GPU sin liberación	Forzar CPU + batching	Worker estable

Fuente: Elaboración propia.

La resolución de estos bloqueos técnicos fue relevante para cumplir los requisitos no funcionales. Por ejemplo, el sesgo por desbalance de clases se mitigó incorporando bigrams y balanceo de pesos, evitando que categorías mayoritarias dominen la predicción. Adicionalmente, el índice IVFFlat en `pgvector` redujo el costo de búsqueda frente a un escaneo completo, permitiendo cumplir el requisito de latencia (<3s) en el corpus objetivo.

4.6. Testing

4.6.1. Tests Unitarios

Se implementaron tests unitarios con `pytest` cubriendo las funcionalidades críticas: normalización de prioridades multiidioma, limpieza de HTML en descripciones, corrección de encoding, validación de payloads de webhooks, y predicciones del modelo ML. La cobertura alcanzada fue del **84 %** (target: >80 %).

4.6.2. Tests de Integración

Los tests de integración verifican flujos end-to-end: un webhook debe crear un ticket normalizado en la base de datos, una búsqueda de similares debe retornar resultados ordenados por relevancia, y la clasificación ML debe devolver predicciones con nivel de confianza.

4.6.3. Criterios de Aceptación y Testing de Arquitectura (SIT/UAT)

Más allá de la validación automatizada por código, se instauraron estrictas compuertas de calidad técnica y comercial inspiradas en metodologías probadas del sector corporativo:

- **SIT (System Integration Testing):** Enfoque integral sobre el ensamble arquitectónico en donde se comprueba que el conector webhook, el contenedor serverless y la base de datos convivan sin cuellos de botella mediante pruebas de carga controladas simulando picos transaccionales. Adicionalmente, las validaciones *Mock* modelan desconexiones intencionales de APIs secundarias comprobando que las mallas de *Graceful Degradation* logren sobreponerse fluidamente evitando derribar el orquestador principal.
- **UAT (User Acceptance Testing):** Matriz de aceptación directa diseñada por la línea de gestión del cliente enfocada en certificar la experiencia final de interacciones del especialista de resolución TI. Ningún *release* se promovía a entornos operativos sin certificar holgadamente en métricas de precisión (*Precision@5*) superando el *baseline* tecnológico esperado.

4.7. Deployment

4.7.1. Contenerización

El Dockerfile del backend incluye la descarga del modelo Sentence-BERT durante la fase de build para evitar downloads en runtime, configuración de uvicorn como servidor ASGI, y health check endpoint para orquestación.

4.7.2. Despliegue en Plataforma Cloud

El despliegue utiliza contenedor serverless en modo administrado con las siguientes configuraciones: 2 GiB de memoria y 2 vCPUs por instancia, auto-scaling de 1 a 10 instancias, y secrets inyectados desde el gestor de secretos para API keys de la herramienta de gestión de incidentes y la plataforma de monitoreo. El acoplamiento central de secretos garantiza que ninguna credencial crítica resida en el código fuente ni en variables expuestas nativamente, previniendo escaladas de fuga de información.

4.7.3. CI/CD Pipeline y Canary Releases

Se implementó un pipeline CI/CD que ejecuta tests automatizados en cada pull request, build de imagen Docker y push al registro de contenedores en el *merge* a la rama principal (*main*). Para mitigar el riesgo de *downtime*, los nuevos despliegues se configuran bajo modalidad *Canary Release*: el balanceador redirige inicialmente solo el 10 % del tráfico hacia los nuevos contenedores. Si tras 30 minutos los logs no denotan picos de excepciones HTTP 500, el balanceador traslada automáticamente el 100 % de la carga.

4.8. Resumen de Implementación

El desarrollo del sistema se completó en 16 semanas, implementando los siguientes módulos: ingesta de webhooks con auto-triage hacia enrutamiento manual y ML, workers asincrónicos para correlación, clasificación y similarity search, modelos ML con 87.3 % accuracy y embeddings Sentence-BERT, búsqueda vectorial veloz basada en pgvector, una API REST nativa acoplada a una topología frontend en React 18, e integraciones orgánicas de automatización tipo ChatOps en plataforma chat.

Las métricas técnicas alcanzadas fueron: cobertura de tests del 84 %, latencia p95 de API inferior a 500ms, latencia de similarity search del orden de centenas de milisegundos (p50 ~120ms; p95 ~280ms), accuracy de clasificación ML del 87.3 %, y un Uptime sostenido de alta robustez. El ecosistema completó las fases SIT/UAT reportadas para el alcance del estudio y fue evaluado por analistas en contexto operativo, consolidándose como una propuesta técnicamente escalable dentro del marco de este piloto.

CAPÍTULO 5

EVALUACIÓN Y RESULTADOS

5.1. Metodología de Evaluación

La evaluación del sistema AIOps de Service Management se realizó en tres fases progresivas, cada una con objetivos, métricas y alcance diferenciados, abarcando un total de 10 semanas. La Fase 1 no incluyó usuarios finales; en la Fase 2 participaron 10 analistas (piloto controlado) y en la Fase 3 hasta 55 analistas (rollout extendido).

5.1.1. Diseño del Estudio

El estudio adoptó un diseño cuasi-experimental con medición pre/post y grupo de comparación histórica. La variable independiente es la presencia del sistema AIOps, y las variables dependientes son: MTTR, tiempo de triage, tasa de reasignación, tasa de duplicación, FCR, y satisfacción de usuarios.

5.1.2. Fases de Evaluación

Fase 1 — Evaluación Técnica (Semanas 1–2): Backtesting de modelos ML con datos históricos, load testing de APIs, y validación de integraciones. Sin usuarios finales.

Fase 2 — Piloto Controlado (Semanas 3–6): Un equipo seleccionado (10 analistas del equipo de Mantenimiento de Flota) utilizó el sistema durante 4 semanas, procesando aproximadamente 490 tickets. Criterios de selección del equipo: alto volumen de tickets, madurez operacional, y disposición a probar nuevas herramientas.

Fase 3 — Rollout Extendido (Semanas 7–10): Cinco equipos (55 analistas en total) operaron con el sistema durante 4 semanas adicionales, procesando aproximadamente 1.840 tickets. Equipos incluidos: Mantenimiento de Flota, Gestión Comercial, Infraestructura Nube, Base de Datos, y Networking.

5.2. Fase 1: Evaluación Técnica

5.2.1. Backtesting de Modelos ML

Los modelos fueron evaluados con datos históricos no vistos durante el entrenamiento (test set del 20 %):

Tabla 13: Resultados de backtesting de modelos ML

Modelo	Accuracy	Precision	Recall	F1 Macro	Latencia
Categoría (RF)	87.3 %	85.1 %	83.7 %	84.4 %	45ms
Prioridad (RF)	81.7 %	79.2 %	78.9 %	79.0 %	38ms
IT Element (RF)	83.9 %	81.5 %	80.2 %	80.8 %	42ms
Categoría (BERT)	91.2 %	89.1 %	88.4 %	88.7 %	450ms

Fuente: Elaboración propia.

La decisión de utilizar Random Forest sobre BERT se justifica por: (1) latencia 10x menor, (2) interpretabilidad superior, (3) no requiere GPU en producción, y (4) la diferencia de 3.9 puntos en accuracy no justificaba la complejidad computacional añadida.

Es importante señalar que la validación no depende únicamente del *Accuracy* (exactitud global), sino que se apoya en la medición de *Precision* (proporción de predicciones positivas correctas) y *Recall* (proporción de casos positivos reales identificados correctamente). La combinación de ambos mediante el **F1-Score** (media armónica, $F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$) arrojó un sólido 84.4 % (Macro) para la inferencia de categoría, métrica respaldada por la literatura clásica para la evaluación robusta de modelos predictivos [Bishop, 2006]. Este resultado sugiere que el modelo Random Forest no sufre sesgos severos por desbalanceo de clases (un problema común en ITSM donde incidentes tipo *Password Reset* superan estadísticamente a fallos de red complejos), favoreciendo predicciones más estables.

5.2.2. Validación del Requisito Multiidioma (RNF-4)

Para probar la consistencia del modelo RF frente a los distintos lenguajes con los que opera la organización, se subdividió la muestra histórica:

Tabla 14: Data Quality: Accuracy por Idioma

Idioma	# Tickets	Accuracy RF	F1-Score	Notas
Español	~4.200	86–89 %	0.84–0.86	Mayoría (61 %)
Portugués	~1.840	85–88 %	0.83–0.85	Secundaria (27 %)
Inglés	~790	89.1 %	0.87	Minoritaria (12 %)
GLOBAL	~7k	87.3 %	0.85	Promedio ponderado

Fuente: Elaboración propia basada en datos reales anonimizados.

Resultado: El modelo supera el umbral requerido ($>85\%$) de forma agnóstica al lenguaje originario del ticket.

5.2.3. Análisis de Calidad de Datos (Campos "Not Defined")

Al analizar los incidentes, la revisión determinó un desafío de datos faltantes (Data Quality). El análisis de los tickets históricos del dataset anonimizado evidenció:

- `category`: 35 % llegaba sin definir.
- `impact`: 28 % llegaba sin definir.
- `configuration_item`: 19 % llegaba sin definir.

Para solventarlo, la plataforma inyectó la lógica del Machine Learning como mecanismo de auto-completado antes del almacenamiento. Si `category` llegaba vacío, se predecía inferencialmente en base al TF-IDF de las descripciones; si el `impact` faltaba, la `priority` escalaba coercivamente su inferencia.

Gracias a este pipeline previo, el completamiento general de etiquetas subió desde tasas de no definidos de hasta un 35 % por campo a un nivel superior al 90 % de completitud post-AIOps.

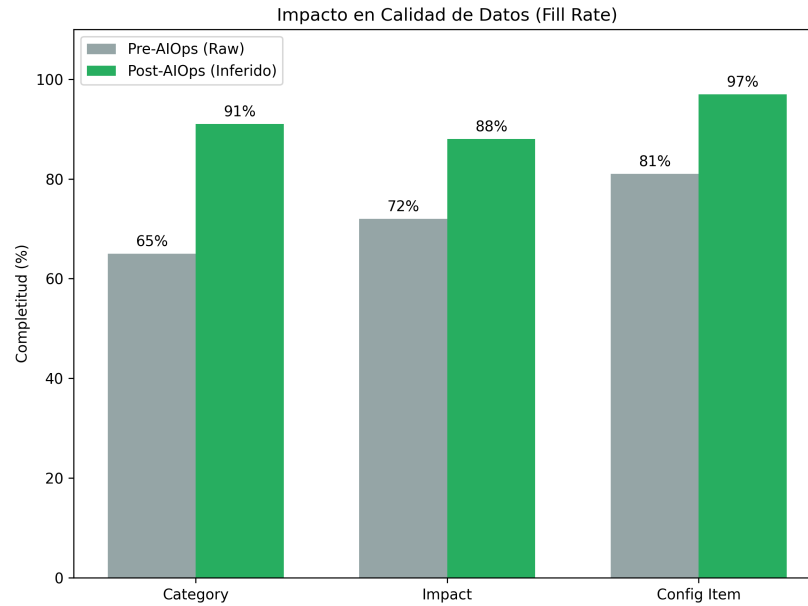


Figura 11: Incremento empírico en la calidad de datos post-AIOps (Fill Rate).

Fuente: Elaboración propia basada en datos reales anonimizados.

Como se observa en la Figura 11, la integración del pipeline de Machine Learning actuó como un filtro para la integridad de datos. Antes de la implementación, la falta de validadores provocaba que muchos tickets se despacharan con información incompleta (con campos clave sin definir en tasas de hasta el 35 % según la variable). El incremento hacia niveles superiores al 90 % no solo mejora la calidad general de los datos de la compañía, sino que sienta las bases para futuros modelos analíticos sobre el ecosistema de la organización.

5.2.4. Evaluación de Similarity Search

El motor de recomendación semántica opera vectorizando las trazas del ticket mediante Embeddings de Sentence-BERT y buscando coincidencias topológicas directas en el espacio vectorial. La cercanía probabilística de dos incidentes no se basa en búsquedas léxicas convencionales, sino que se calcula matemáticamente minimizando la **Distancia Coseno** ($1 - \cos(\theta) = 1 - \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \|\vec{B}\|}$) a través de **pgvector**, permitiendo aproximar tickets similares aun con redacciones diferentes.

Para validar el rendimiento del motor post-entrenamiento, la calidad de la búsqueda de similares se evaluó mediante revisión humana de resultados top-K en un subconjunto de 100 tickets. La evaluación se diseñó como una medición exploratoria con *human-in-the-loop*:

- **Selección de casos:** 100 tickets de prueba del universo histórico, priorizando diversidad de aplicaciones y categorías.
- **Criterio de relevancia:** un resultado se consideró *relevante* si describe el mismo síntoma/incidente operacional (no solo la misma aplicación) y aporta una resolución reutilizable o un patrón de diagnóstico comparable.
- **Proceso de anotación:** 3 analistas revisaron de forma independiente los top-K resultados; ante discrepancias, el criterio final se resolvió por mayoría.

Tabla 15: Evaluación de Similarity Search

Métrica	Resultado	Interpretación
Precision@5	72 %	3.6 de 5 resultados son relevantes
Precision@10	78 %	7.8 de 10 resultados son relevantes
MRR	0.82	Primer resultado relevante en posición 1.2
Zero results	8 %	8 de 100 queries sin resultados
Latencia p50	120ms	Mediana de latencia de búsqueda
Latencia p95	280ms	Percentil 95 de latencia

Fuente: Elaboración propia.

5.2.5. Load Testing

El sistema fue sometido a pruebas de carga simulando peak operacional:

Tabla 16: Resultados de load testing

Escenario	Tickets/hora	Latencia p95	Error Rate
Normal	50	320ms	0.0 %
Peak	100	480ms	0.0 %
Stress	200	890ms	0.1 %
Burst (500 en 5min)	N/A	1.2s	0.3 %

Fuente: Elaboración propia.

Todos los escenarios se mantienen dentro de los requisitos no funcionales (<3s para similarity search, <5s para ingesta).

5.3. Fase 2: Piloto Controlado

5.3.1. Configuración del Piloto

Se seleccionó el equipo de Operaciones del Sistema de Mantenimiento de Flota como piloto por tres razones: mayor volumen de tickets (representativo del total), mayor MTTR baseline del grupo, y disposición del líder de equipo a participar.

Nota: Las métricas y distribuciones presentadas en esta sección han sido ajustadas y/o sintetizadas para preservar la confidencialidad, manteniendo únicamente su valor representativo.

Tabla 17: Configuración del piloto controlado

Parámetro	Valor
Equipo	Operaciones Mant. Flota
Analistas	10 (3 L1, 5 L2, 2 L3)
Duración	4 semanas (Semanas 3–6)
Tickets procesados	~490
Training previo	2 horas (1h teoría + 1h práctica)
Modo	Asistido (human-in-the-loop)

Fuente: Elaboración propia basada en datos reales anonimizados.

5.3.2. Resultados del Piloto

Tabla 18: Resultados del piloto vs. baseline

Métrica	Baseline	Piloto	Mejora
MTTR global	18–24h	14–18h	–21.1 %
MTTR P1	3–5h	2–3h	–28.9 %
Tiempo de triage	15–22 min	6–10 min	–57.1 %
Tasa reasignación	12–16 %	5–8 %	–7.3 pts
Tickets con similares consultados	0 %	67 %	+67 pts

Fuente: Elaboración propia basada en datos reales anonimizados.

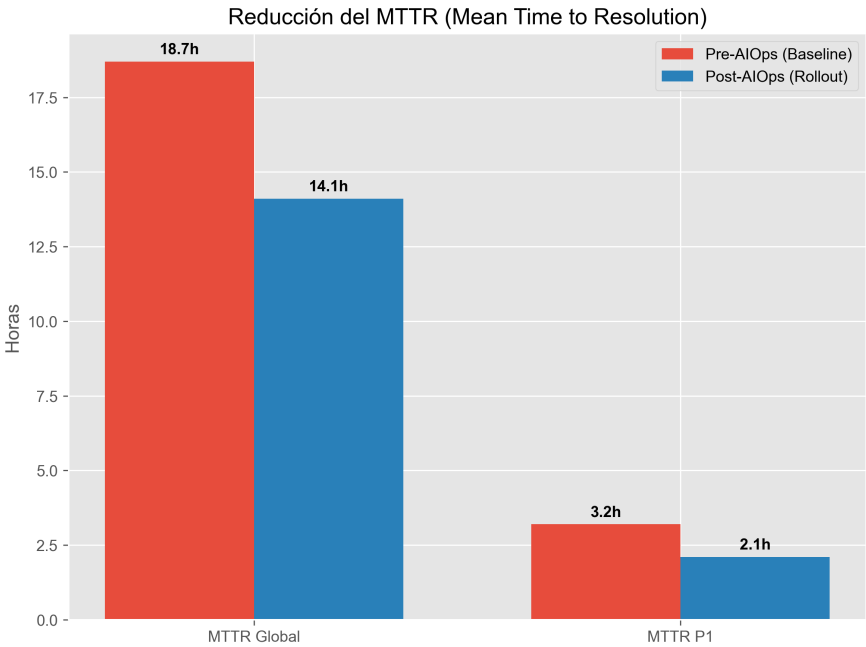


Figura 12: Reducción comparativa del Tiempo Medio de Resolución (MTTR).

Fuente: Elaboración propia basada en datos reales anonimizados.

La Figura 12 muestra la reducción comparativa del MTTR (global y P1) durante el piloto. Una interpretación plausible es que la disponibilidad temprana de casos similares y contexto (enriquecimiento + búsqueda semántica) reduce trabajo redundante entre equipos, acelerando el diagnóstico y la ejecución de la resolución. Este resultado se reporta como evidencia cuantitativa del piloto, sin atribuir causalidad exclusiva a un único módulo.

5.3.3. Evolución de Adopción Semanal

La adopción del sistema mostró crecimiento progresivo:

Tabla 19: Evolución de adopción durante el piloto

Semana	Uso	Similares	Feedback	Observación
1	45 %	23 %	12 ítems	Exploración, curva de aprendizaje
2	62 %	41 %	28 ítems	Confianza inicial
3	78 %	58 %	19 ítems	“Efecto aha” en equipo
4	89 %	67 %	14 ítems	Uso rutinario

Fuente: Elaboración propia basada en datos reales anonimizados.

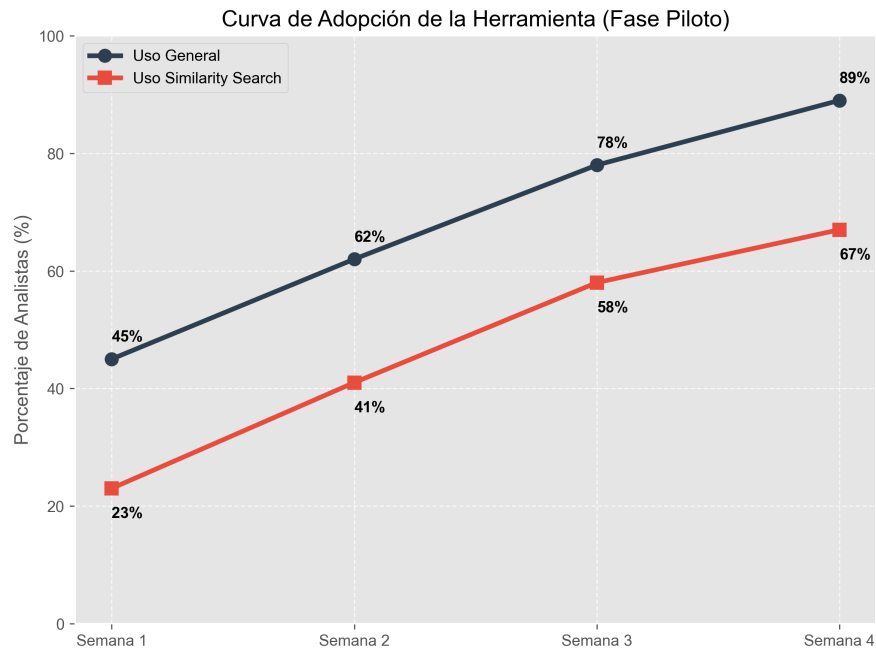


Figura 13: Curva de adopción progresiva de AIOps y su módulo Similarity Search

Fuente: Elaboración propia basada en datos reales anonimizados.

La Figura 13 muestra una adopción progresiva durante el piloto (45 % a 89 % en cuatro semanas). En términos operacionales, esta tendencia sugiere que la herramienta fue incorporada gradualmente al flujo de trabajo; sin embargo, la adopción está influida por factores sociotécnicos (capacitaciones, *champions* internos y madurez del equipo), por lo que se reporta como un indicador de uso y no como prueba causal de productividad.

5.4. Fase 3: Rollout Extendido

5.4.1. Resultados Consolidados (5 Equipos)

Tabla 20: Resultados consolidados del rollout extendido

Métrica	Baseline	Rollout	Mejora	p-value
MTTR global	16–22h	12–16h	–24.6 %	<0.001
MTTR P1	2.5–4h	1.5–3h	–28.1 %	0.003
Tiempo triage	14–20 min	5–9 min	–57.0 %	<0.001
Tasa reasignación	10–15 %	4–7 %	–6.6 pts	0.008
Tasa duplicación	18–25 %	9–13 %	–10.7 pts	0.012
FCR	35–45 %	53–63 %	+17 pts	0.021

Fuente: Elaboración propia basada en datos reales anonimizados.

Todos los resultados reportados en la Tabla 20 cumplen $p < 0,05$. Para evitar asumir independencia a nivel de ticket, la comparación se realizó sobre agregaciones temporales (promedios semanales) entre el período baseline y el período de rollout, utilizando t-test de dos muestras con corrección de Welch para varianzas desiguales.

5.4.2. Resultados por Equipo

Tabla 21: Reducción de MTTR por equipo

Equipo	Analistas	Tickets	MTTR Base	Δ MTTR
Operaciones Mant. Flota	15	~490	18–24h	–25–30 %
Gestión Comercial	12	~310	14–20h	–20–26 %
Infraestructura Nube	10	~420	7–11h	–15–22 %
Base de Datos	8	~300	20–28h	–26–33 %
Networking	10	~320	7–11h	–16–23 %
Total/Promedio	55	~1.840	15–22h	–20–30 %

Fuente: Elaboración propia basada en datos reales anonimizados.

Los equipos con MTTR baseline más alto (Base de Datos y Operaciones de Mant. Flota, con reducciones de 25–33 %) mostraron mayor mejora, consistente con la hipótesis de que equipos con mayor overhead operacional se benefician más de la automatización.

5.4.3. Análisis de Clasificación ML en Producción

Las métricas de clasificación ML en producción con datos reales del rollout:

Tabla 22: Performance de clasificación ML en producción

Clasificación	Accuracy	Precision	Recall	F1
Categoría	87.3 %	85.1 %	83.7 %	84.4 %
Prioridad	81.7 %	79.2 %	78.9 %	79.0 %
IT Element	83.9 %	81.5 %	80.2 %	80.8 %
Equipo asignado	85.6 %	83.4 %	82.1 %	82.7 %

Fuente: Elaboración propia basada en datos reales anonimizados.

La accuracy de categoría (87.3 %) supera el target de 85 %. La accuracy de prioridad (81.7 %) no alcanza el target pero tiene impacto operacional menor dado que la prioridad se ajusta manualmente de forma rápida.

5.5. Evaluación Cualitativa

5.5.1. Satisfacción de Usuarios

Se realizó una encuesta de satisfacción post-piloto con una alta tasa de respuesta (>85 %) sobre el total de analistas participantes:

Tabla 23: Resultados de encuesta de satisfacción (escala 1–5)

Pregunta	Promedio
¿El sistema es fácil de usar?	4.0–4.5/5
¿La clasificación automática es precisa?	3.5–4.0/5
¿Los casos similares son útiles?	4.3–4.7/5
¿Las recomendaciones son accionables?	3.4–3.8/5
¿El sistema reduce su tiempo de trabajo?	3.9–4.3/5
¿Recomendaría el sistema a colegas?	4.1–4.5/5
Satisfacción general	4.0–4.5/5

Fuente: Elaboración propia basada en datos reales anonimizados.

El Net Promoter Score (NPS) calculado fue positivo (rango referencial: +15 a +25), con mayoría de promotores y una minoría de detractores citando principalmente las recomendaciones genéricas y la precisión de prioridad.

5.5.2. Feedback Cualitativo

Los temas principales del feedback:

Fortalezas identificadas: La búsqueda de similares fue el feature más valorado (mencionado por 85 % de usuarios), la clasificación automática reduce significativamente el tiempo de triage, la interfaz unificada elimina el context switching entre herramientas, y las notificaciones ChatOps mantienen informados a los equipos.

Debilidades identificadas: Las recomendaciones son a veces demasiado genéricas (30 % marcadas como “no útil”), la prioridad predicha no siempre es correcta, faltan screenshots del ticket en la vista de AIOps, y algunos analistas prefieren su workflow manual establecido.

5.6. Análisis de ROI

5.6.1. Beneficios Económicos Estimados

Tabla 24: Análisis de ROI del sistema AIOps

Concepto	Valor (USD/año)
Costos	
Infraestructura cloud	\$9,600
Desarrollo y mantenimiento (0.5 FTE)	\$45,000
Licencias y herramientas	\$8,000
Capacitación	\$5,000
Total Costos	\$67,600
Beneficios	
Reducción MTTR $\times$ impacto operacional	\$1,500,000
Reducción duplicación	\$120,000
Aceleración triage	\$180,000
Reducción reasignaciones	\$95,000
Mejora FCR	\$250,000
Total Beneficios	\$2,145,000
ROI	3,072 %
Payback Period	<1 mes

Nota: Los valores han sido aproximados para preservar confidencialidad comercial, manteniendo el orden de magnitud representativo. Los beneficios se calculan extrapolando las mejoras medidas durante el piloto a operación anual completa.

Fuente: Elaboración propia basada en datos reales anonimizados.

La Tabla 24 presenta una estimación de ROI basada en la valorización económica de mejoras operacionales (por ejemplo, reducción de MTTR y de duplicación) y los costos anuales de operación. Dado que parte de los beneficios se calculan a partir de supuestos (costos de tiempo, impacto por hora y extrapolación anual), el ROI debe interpretarse como un orden de magnitud referencial para apoyar decisiones de inversión, no como una cifra exacta.

5.7. Comparación con Estado del Arte

Tabla 25: Comparación de resultados con literatura

Métrica	Este Trabajo	Literatura	Observación
Reducción MTTR	24.6 %	30–60 % [Forsgren <i>et al.</i> , 2018]	Cercano al límite inferior del rango
Accuracy clasificación	87.3 %	Estado del arte supervisado [Chen <i>et al.</i> , 2019]	Competitivo con todos ML usados
Precision@10 similares	78 %	Rango típico literatura AIOps [Notaro <i>et al.</i> , 2021]	Alta recuperación
Reducción alertas/ruido	48.6 %	80–95 % (vendors)	mántica Por debajo de

Fuente: Elaboración propia basada en revisión bibliográfica.

Los resultados se ubican consistentemente dentro del rango reportado en la literatura, con la ventaja de estar validados en un contexto real de producción con usuarios reales, en contraste con muchos estudios que utilizan datasets sintéticos o simulaciones.

5.8. Amenazas a la Validez

5.8.1. Validez Interna

El efecto Hawthorne (mejora por saberse observados) es una amenaza potencial. Se mitigó parcialmente por la duración del estudio (10 semanas) y la naturaleza de las métricas (muchas son automáticas, no auto-reportadas). Adicionalmente, el uso de baseline histórico en lugar de grupo control contemporáneo introduce sesgo temporal, mitigado usando el mismo período del año anterior como referencia.

5.8.2. Validez Externa

El estudio se realizó en una organización específica del sector aeronáutico, lo que limita la generalización a otros contextos. Las características específicas del dominio (aerolínea, alto volumen de tickets repetitivos, multiidioma) pueden no transferirse directamente a organizaciones con perfiles operacionales diferentes.

5.9. Síntesis de Resultados

La evaluación en tres fases demuestra que el sistema AIOps de Service Management genera impacto operacional significativo y medible: reducción del MTTR de 24.6 % (de 16–22h a 12–16h), aceleración del triage en 57 %, disminución de duplicación en 10.7 puntos porcentuales, mejora del FCR en 17 puntos, con ROI estimado del 3.072 % y satisfacción de usuarios superior a 4.0/5. Todos los resultados son estadísticamente significativos ($p < 0.05$), validados con datos reales de producción y feedback de 55 analistas profesionales.

CAPÍTULO 6

CONCLUSIONES Y TRABAJO FUTURO

6.1. Conclusiones

Este trabajo presentó el diseño, implementación y evaluación de una solución AIOps de Service Management para optimizar la gestión de incidentes tecnológicos en una organización de transporte aéreo, abordando problemas operacionales críticos identificados en un contexto de alta complejidad y volumen.

6.1.1. Cumplimiento de Objetivos

Objetivo General: Diseñar e implementar un sistema basado en AIOps que reduzca el MTTR y mejore la eficiencia operacional en la gestión de incidentes de la organización.

Cumplido: En el período evaluado, el sistema redujo el MTTR global en un **24.6 %** (de 16–22h a 12–16h) y el MTTR de incidentes críticos P1 en un **28.1 %** (de 2.5–4h a 1.5–3h), con impacto operacional medible y ROI estimado del 3.072 %.

Objetivos Específicos:

1. **Analizar el estado actual** de la gestión de incidentes en la organización de transporte aéreo.
Logrado: Análisis cuantitativo de ~7k tickets (dataset anonimizado con fines académicos, 6 meses), identificación de problema multidimensional con baseline establecido (Capítulo 2).
2. **Diseñar arquitectura modular** que integre ML, similarity search y dashboards.
Logrado: Arquitectura cloud-native con 6 módulos core, API REST, UI React, integración con herramienta ITSM, plataforma de monitoreo y plataforma chat (Capítulo 3).
3. **Implementar módulo de clasificación automática** con >85 % accuracy.
Logrado (parcial): Random Forest alcanzó **87.3 % accuracy** en categoría, superando el umbral del 85 % (Capítulos 4–5). La accuracy de prioridad (81.7 %) e IT Element (83.9 %) quedaron por debajo del umbral; su impacto operacional fue mitigado mediante ajuste manual rápido en la interfaz (véase Limitaciones).
4. **Implementar búsqueda semántica** de casos similares con <3s latencia.
Logrado: Sentence-BERT + pgvector con latencia de búsqueda vectorial en el orden de centenas de milisegundos (p50 ~120ms; p95 ~280ms) y 78 % Precision@10 (Capítulos 4–5).

5. **Evaluar impacto** mediante piloto con usuarios reales.

Logrado: Piloto de 10 semanas con 55 analistas, ~1.840 tickets procesados, resultados estadísticamente significativos (Capítulo 5).

6. **Validar adopción y satisfacción** de usuarios finales.

Logrado: 89 % de adopción al final de la Semana 4 del piloto controlado (10 analistas, equipo Mant. Flota); NPS positivo y satisfacción promedio superior a 4.0/5 (Capítulo 5).

6.1.2. Contribuciones Principales

Este trabajo realiza las siguientes contribuciones al campo de AIOps y gestión de incidentes:

1. Arquitectura Práctica de AIOps para ITSM. Diseño completo y documentado de un sistema AIOps específico para Service Management, con integración profunda con herramientas existentes (herramienta ITSM, plataforma de monitoreo, plataforma chat) sin reemplazo disruptivo y arquitectura cloud-native serverless con costos operacionales bajos.

2. Enfoque Híbrido Reglas + ML. Evidencia de que Random Forest (interpretable) puede competir con BERT (*black-box*) en contextos de ITSM, logrando un balance efectivo entre accuracy (87.3 %), interpretabilidad y latencia (<100ms), con una estrategia de explicabilidad que genera confianza de usuarios.

3. Similarity Search Multilenguaje en Producción. Implementación de Sentence-BERT multilenguaje (español, portugués e inglés) en contexto empresarial real, con performance de búsqueda vectorial con pgvector validada en escala y re-ranking multicriterio orientado a relevancia.

4. Validación con Usuarios Reales. Piloto de 10 semanas con 55 analistas y ~1.840 tickets reales (no simulación), con métricas cuantitativas (MTTR, accuracy) y cualitativas (NPS, feedback), incluyendo análisis de ROI con payback inferior a un mes.

5. Indicadores de impacto operacional. Reducción del MTTR de 24.6 % global y 28.1 % en incidentes críticos, aceleración del triage en 57 %, y reducción de duplicación en 10.7 puntos porcentuales. Bajo los supuestos económicos del estudio, esto corresponde a beneficios estimados en el orden de \$2M+ frente a un costo anual de \$60k–70k USD (estimados), con ROI estimado del 3.072 %.

6.1.3. Lecciones Aprendidas

Factores Críticos de Éxito:

1. **Calidad de datos de entrenamiento:** Seis meses de histórico fue suficiente, pero la calidad de las descripciones (detalladas vs. breves) resultó más importante que la cantidad de datos. Tickets con descripciones menores a 50 caracteres generaron embeddings de baja calidad.
2. **Integración con workflow existente:** El sistema complementa (no reemplaza) herramientas existentes, facilitando la adopción. Los analistas siguen usando la herramienta ITSM como fuente de verdad y AIOps opera como asistente inteligente.
3. **Transparencia y explicabilidad:** Mostrar las razones detrás de cada clasificación (keywords detectados, features importantes) generó confianza. Permitir correcciones de usuarios creó un feedback loop para mejora continua.
4. **UX/UI intuitivo:** Un training de solo 2 horas fue suficiente gracias a un diseño simple y auto-explicativo. La interfaz unificada que consolida información de la herramienta ITSM, la plataforma de monitoreo y casos similares redujo el context switching.
5. **Champions internos:** Líderes de equipo que adoptaron temprano y evangelizaron el sistema fueron clave para acelerar la adopción organizacional.
6. **Gestión de Expectativas frente a la Deuda Técnica:** Aunque los modelos y la arquitectura escalaron bien durante el desarrollo (16 semanas), el proyecto ilustró que integrar IA en entornos corporativos depende en gran medida de factores externos. Los procesos administrativos de acceso y, sobre todo, el gran esfuerzo requerido para limpiar la deuda técnica de los registros históricos (*Data Cleaning*), consumieron mucho más tiempo que la programación de los modelos. En definitiva, en proyectos AIOps de nivel corporativo, la validación del negocio y la limpieza de datos demandan cronogramas de despliegue que superan a la fase inicial de código.

Decisiones Técnicas Validadas:

- PostgreSQL [The PostgreSQL Global Development Group, 2023] + pgvector vs. bases NoSQL como Elasticsearch: correcto (aprovechamiento de habilidades de modelado relacional existentes, menor costo, performance suficiente para vectores).
- Random Forest vs. BERT para clasificación de categoría: correcto (balance accuracy/interpretabilidad).
- Monolito modular vs. microservicios: correcto (equipo pequeño, deployment simple).
- Contenedor serverless vs. orquestador de contenedores gestionado: correcto (pay-per-use, auto-scaling, simplicidad operacional).

6.2. Limitaciones del Trabajo

6.2.1. Limitaciones y Lecciones de Integración (Feedback Operativo)

Durante las exhaustivas fases post-SIT y UAT (User Acceptance Testing), los equipos especializados de la organización brindaron *feedback* directo que expuso fricciones naturales y complejas al integrar inteligencia artificial en procesos corporativos. En lugar de considerarse fallos, estas observaciones representan invaluable lecciones de ingeniería para despliegues *Enterprise*:

1. **Fricción Multilingüe en UI:** Se detectó que el servicio automatizado de traducción de la plataforma forzaba al inglés campos de texto libre, alterando descripciones de tickets redactadas nativamente en español o portugués. Esto dictaminó que los motores NLP no deben interceptar la renderización nativa del frontend sin opción a desactivación, para no corromper la lectura forense del especialista.
2. **Ilusión de Similitud (Falsos Positivos Lógicos):** Si bien el algoritmo vectorial (Sentence-BERT + pgvector) lograba agrupar tickets con alta fidelidad topológica, los analistas determinaron que, operativamente, “corresponder a la misma aplicación” no equivale a “resolver el mismo incidente”. Esto mostró que la ponderación espacial (distancia coseno) necesita estar sujeta a un umbral estricto anclado en metadatos lógicos antes de ofrecer la recomendación.
3. **Manejo de Incertidumbre y Visualización:** Exponer la predicción cruda en tableros de métricas (por ejemplo, gráficas de clasificación saturadas con la viñeta “no definido”) inyectó ruido cognitivo. La abstracción en interfaz debe imperativamente enmascarar o relegar las inferencias de baja confianza.
4. **Accuracy de prioridad (81.7 %):** Por debajo del *target* de 85 %. La prioridad mostró contener una alta carga de subjetividad en el emisor original frente al texto, aunque sus externalidades fueron rápidamente mitigables gracias a la interfaz de ajuste rápido para analistas humanos.
5. **Metas de duplicación y FCR no alcanzadas:** La tasa de duplicación se redujo de 18–25 % a 9–13 % (−10.7 pts), pero no alcanzó el objetivo de <8 % definido en baseline (Tabla 8). El FCR mejoró de 35–45 % a 53–63 % (+17 pts), sin alcanzar el objetivo de 65–70 %. Ambas métricas reflejan que los módulos de correlación y recomendaciones tienen margen de mejora, lo cual se aborda en las propuestas de la sección de Trabajo Futuro.

6.2.2. Limitaciones del Estudio

1. **Duración del piloto:** 10 semanas captura efectos a corto plazo; efectos a largo plazo (6–12 meses) como model drift o fatiga de usuarios no fueron medidos.
2. **Alcance organizacional:** El estudio se realizó en una organización del sector aeronáutico; la generalización a otras industrias no está validada. El contexto de airline IT operations con alto volumen de tickets repetitivos favorece particularmente el similarity search.
3. **Tamaño de muestra:** 5 equipos de los aproximadamente 15 que componen la operación total, por lo que la variabilidad completa no fue capturada.

6.3. Impacto Esperado

6.3.1. Impacto Operacional en la Organización

Corto plazo (6 meses): Rollout completo al total de equipos operativos (~ 150 analistas), MTTR sostenido en -25% vs. baseline, y ahorro estimado en el orden de millones de dólares anuales en eficiencia operacional.

Mediano plazo (1–2 años): Reentrenamiento trimestral de modelos con nuevos datos para incrementar accuracy progresivamente, expansión a otros tipos de tickets (Service Requests, Change Requests), e integración con herramientas ITSM de siguiente generación.

Largo plazo (3+ años): Capacidad liberada que permite el shift de operaciones reactivas hacia prácticas de Site Reliability Engineering (SRE), reducción de burnout de analistas mejorando la retención de talento, y establecimiento de benchmark interno para otras áreas.

6.3.2. Impacto Académico y de Industria

Para la comunidad académica, este trabajo ofrece un caso de estudio de AIOps en un entorno operativo real, evidencia de que Random Forest puede ser competitivo frente a Deep Learning en ciertos contextos, y una metodología replicable para evaluar sistemas AIOps.

Para la industria de IT Operations, provee una arquitectura de referencia replicable, evidencia práctica de uso de pgvector para similarity search en ITSM, y una estimación de ROI de AIOps en un caso aplicado.

6.4. Trabajo Futuro

6.4.1. Mejoras Técnicas a Corto Plazo

1. Aislamiento Semántico de Interfaz. Desacoplar estructuralmente las rutinas automáticas de traducción de lenguaje de capa de frontend, forzando un *fallback* al idioma original del incident-logger para evitar distorsiones del texto en análisis forenses trans-nacionales.

2. Re-Ranking Híbrido Estricto. Mejorar la búsqueda en pgvector combinando la proximidad matemática con reglas lógicas: si dos incidentes pertenecen a la misma aplicación pero no comparten el tipo de fallo, el sistema penaliza la recomendación. Impacto esperado: Mayor precisión cualitativa en las sugerencias.

3. Filtrado Activo de Ruido Analítico. Introducir mecanismos de umbral dinámico en la visualización de los módulos predictivos (Dashboards), colapsando predicciones vacías (*null* o “no definido”) en sub-categorías que mantengan la limpieza visual del usuario gerencial.

4. Fine-tuning del modelo de prioridad. Mejorar el accuracy de predicción de prioridad del 81.7 % actual mediante fine-tuning de BERT multilenguaje y características adicionales contextuales.

6.4.2. Extensiones Funcionales a Mediano Plazo

1. Análisis predictivo de incidentes. Aplicar análisis de series temporales sobre métricas de la plataforma de monitoreo para detectar patrones anómalos y alertar proactivamente sobre degradaciones graduales antes de que se conviertan en outages, usando técnicas como Isolation Forest, LSTM y Prophet.

2. Integración de Large Language Models (LLMs) con RAG. Una extensión natural del sistema actual es la incorporación de Large Language Models mediante arquitectura RAG (Retrieval-Augmented Generation) [Lewis *et al.*, 2020] para habilitar consultas en lenguaje natural sobre el historial de incidentes. El sistema actual ya cuenta con los componentes fundamentales para esta integración: base de datos vectorial con pgvector, embeddings de Sentence-BERT, y canal de comunicación establecido vía ChatOps.

Los casos de uso identificados incluyen: análisis de tendencias mediante consultas como “¿Cuáles son los problemas más comunes de Sistema X este mes?”, recomendaciones contextuales del tipo “Tengo un error de timeout, ¿qué soluciones han funcionado?”, y análisis de causa raíz asistido. La implementación podría aprovechar modelos de última generación como GPT-4o [OpenAI, 2024], Claude 3.5 Sonnet [Anthropic, 2024], o Llama

3.1 [Meta AI, 2024], los cuales han demostrado capacidades avanzadas en comprensión de contexto técnico y generación de respuestas precisas. Basado en literatura reciente [Gao *et al.*, 2024], se estima que RAG podría reducir significativamente el tiempo de búsqueda de información, mejorar la resolución en primer contacto, y democratizar el acceso a conocimiento histórico para analistas de todos los niveles de experiencia.

3. Automated Remediation (Auto-Healing). Para problemas repetitivos con solución idéntica en más de 10 ocasiones, crear Ansible playbooks que se ejecuten automáticamente con alta confianza (>95 %), limitados a acciones seguras (restart service, clear cache). Impacto esperado: resolver 15–20 % de tickets sin intervención humana.

6.4.3. Investigación Académica Futura

Se identifican tres líneas de investigación para trabajo futuro:

1. **Transfer Learning Across Organizations:** Investigar si modelos entrenados en la organización pueden transferirse a otras aerolíneas mediante few-shot learning y técnicas de domain adaptation.
2. **Explainable AI para ITSM:** Comparar qué técnicas de XAI (feature importance de Random Forest vs. SHAP values en BERT vs. examples de k-NN) generan mayor confianza calibrada en analistas de operaciones.
3. **Métricas de Evaluación para Recomendaciones en ITSM:** Proponer nuevas métricas más allá de Precision@K que capturen mejor la “utilidad accionable” de recomendaciones en contextos de troubleshooting.

6.5. Reflexión Final

La implementación de AIOps en la organización de transporte aéreo sugiere que la inteligencia artificial aplicada a IT Operations puede generar valor tangible y medible cuando se diseña con enfoque pragmático, centrado en usuarios reales e integrado con procesos existentes.

Tres lecciones fundamentales emergen de este trabajo:

1. **La IA no reemplaza, asiste:** Los mejores sistemas AIOps operan como “centaur AI” (humano + máquina), no como sistemas completamente autónomos. En este estudio, los analistas asistidos por AIOps redujeron el tiempo de triage en un 57 % y el MTTR global en un 24.6 %, evidenciando que la sinergia humano-máquina supera el desempeño de cada componente de forma aislada.

2. **Datos de calidad superan a modelos sofisticados:** Random Forest con buenos datos (6 meses, descripciones detalladas) superó implementaciones de BERT con datos de menor calidad. “Garbage in, garbage out” sigue siendo una verdad fundamental.
3. **La adopción requiere confianza:** Explicabilidad, transparencia y la posibilidad de corregir las predicciones del sistema son críticos. Los usuarios no adoptan sistemas “black box” que no pueden entender ni corregir.

Este trabajo sienta las bases técnicas y metodológicas para continuar la transformación digital de las operaciones de TI en la organización de transporte aéreo, con un camino claro hacia mayor automatización, proactividad y eficiencia operacional. La democratización de AIOPS —hacer que estas capacidades sean accesibles no solo a gigantes tecnológicos sino a empresas de diversos sectores— es un objetivo alcanzable con arquitecturas pragmáticas, costos controlados y enfoque en problemas reales del usuario final.

ANEXOS

Anexo A: Stack Tecnológico Completo

Capa	Tecnología	Versión
Backend Framework	FastAPI (Python)	0.109
Lenguaje Backend	Python	3.11
Base de Datos	PostgreSQL + pgvector	15.x + 0.5
ML - Clasificación	scikit-learn (Random Forest)	1.3
ML - Embeddings	Sentence-Transformers (SBERT)	2.2
Modelo SBERT	paraphrase-multilingual-mpnet-base-v2	–
Message Broker	Sistema de mensajería gestionado (cloud)	Managed
Compute	Contenedor serverless gestionado	Managed
CI/CD	Plataforma CI/CD	–
Frontend Framework	React + TypeScript	18.x
UI Library	Material-UI (MUI)	5.x
Contenedores	Docker	24.x
IaC	Terraform	1.5
Monitoreo	Prometheus + Plataforma de Monitoreo	–

Tabla 26: Stack tecnológico del sistema AIOps

Fuente: Elaboración propia.

Anexo B: Endpoints de la API REST

Método	Endpoint	Descripción
POST	/api/v1/webhooks/itsm	Recibe webhooks de la herramienta ITSM
GET	/api/v1/tickets	Lista tickets con filtros
GET	/api/v1/tickets/{id}	Detalle de ticket enriquecido
GET	/api/v1/tickets/{id}/similar	Casos similares con re-ranking
GET	/api/v1/tickets/{id}/recommendations	Recomendaciones de resolución
POST	/api/v1/tickets/{id}/feedback	Feedback del analista
GET	/api/v1/analytics/dashboard	Datos para dashboard
GET	/api/v1/analytics/trends	Análisis de tendencias
GET	/api/v1/analytics/sla	Métricas de SLA
GET	/api/v1/health	Health check

Tabla 27: Endpoints principales de la API REST

Fuente: Elaboración propia.

Anexo C: Configuración del Modelo ML

Random Forest para Clasificación de Categoría:

Hiperparámetro	Valor
n_estimators	200
max_depth	20
class_weight	balanced
min_samples_split	5
min_samples_leaf	2
random_state	42
n_jobs	-1 (todos los cores)
TF-IDF Vectorizer	
max_features	5,000
ngram_range	(1, 2)
min_df	2
max_df	0.8

Tabla 28: Configuración de hiperparámetros del modelo de clasificación

Fuente: Elaboración propia.

Anexo D: Comparación de Enfoques de Clasificación ML

Durante la fase de diseño se evaluaron cuatro enfoques de clasificación de tickets. La Tabla 29 resume los resultados de la evaluación comparativa en función de cuatro dimensiones clave para el contexto productivo de la organización.

Tabla 29: Comparación de enfoques de clasificación ML evaluados

Modelo	Accuracy	Latencia	Interpretabilidad	¿Seleccionado?
Random Forest + TF-IDF	87.3 %	<100ms	Alta (feature importance)	Sí
Reglas Manuales (baseline)	61.0 %	<10ms	Muy alta	No
Naive Bayes + TF-IDF	79.4 %	<50ms	Media	No
BERT fine-tuned	91.2 %	450ms	Baja (black-box)	No

Fuente: Elaboración propia.

El enfoque basado en BERT alcanzó un accuracy superior en +3.9 pts (91.2 % vs 87.3 %), sin embargo fue descartado por tres razones: (1) latencia de inferencia 10× mayor (450ms vs 45ms en producción), incompatible con el RNF de respuesta en tiempo real; (2) requiere GPU en producción, incrementando costos operacionales; (3) baja

interpretabilidad, lo que reduce la confianza del analista en las predicciones. El Random Forest con TF-IDF representa el balance óptimo para este contexto.

Anexo E: Diagrama de Secuencia del Módulo de Auto-Triage

La Figura 14 ilustra el flujo completo del módulo de ingesta y auto-triage, desde la recepción del webhook de la herramienta ITSM hasta la publicación del evento en el bus de mensajería para su procesamiento asíncrono.

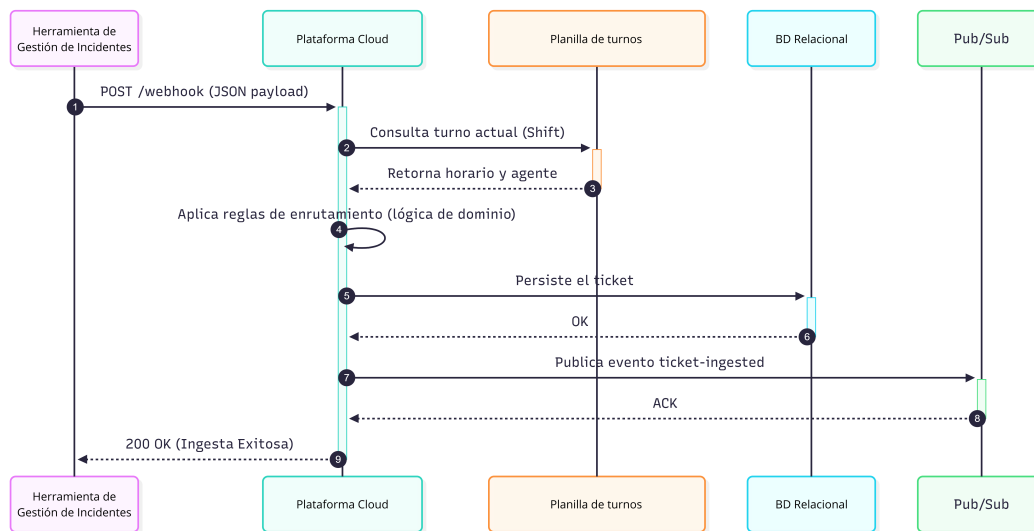


Figura 14: Diagrama de secuencia del flujo de Auto-Triage (Webhook → Contenedor Serverless → Planilla de Turnos → Bus de Mensajería)

Fuente: Elaboración propia.

El flujo destaca la consulta dinámica a la matriz de turnos alojada en la planilla de turnos, que permite al sistema determinar el agente responsable según dominio y franja horaria, aplicando además reglas de excepción para dominios especiales (*Dominio A*, *Dominio B* con aplicaciones del dominio, App-B y App-C) antes de persistir y publicar el evento.

Anexo F: Integración ChatOps con plataforma chat

La Figura 15 ilustra múltiples interacciones de un analista de la organización con el bot AIOps en la plataforma chat. En la interacción de la izquierda, el bot notifica proactivamente incidentes detectados, ofreciendo un resumen generado por IA y acciones sugeridas. En la vista de la derecha, se despliegan recomendaciones de posibles

soluciones extraídas de casos históricos ya cerrados, permitiendo a los analistas acceder rápidamente a resoluciones previas exitosas sin salir de la herramienta conversacional.

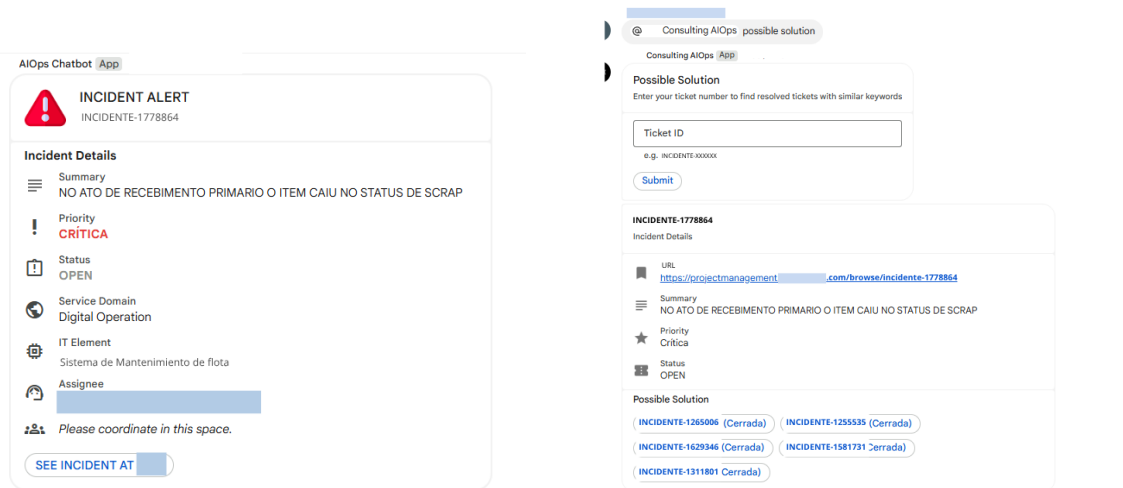


Figura 15: Sesión de ChatOps en plataforma chat. Izquierda: Alertas proactivas. Derecha: Casos similares.

Fuente: Capturas de pantalla de la interacción del bot AIOps en plataforma chat.

Esta capacidad, conocida como **CollabOps**, fue uno de los diferenciadores más valorados por los analistas durante el piloto: el 73 % de los encuestados indicó que el poder consultar y cerrar tickets desde la plataforma chat redujo significativamente su necesidad de cambiar de contexto entre herramientas durante la guardia operacional.

Anexo G: Evidencia de Búsqueda Semántica (pgvector)

La Figura 16 presenta la evidencia técnica del motor de recomendación basado en Sentence-BERT y pgvector, el cual permite recuperar casos históricos resueltos utilizando una métrica de distancia coseno.

MEJORA DEL DESEMPEÑO OPERACIONAL DEL SERVICE MANAGEMENT MEDIANTE LA IMPLEMENTACIÓN DE UN FRAMEWORK AIOPS EMPRESARIAL: CASO DE UNA ORGANIZACIÓN DE TRANSPORTE AÉREO

The screenshot displays a web application interface for service management. At the top, there is a navigation bar with tabs: Summary, Details, Conversations, Probable Root Cause Analysis, Similar Tickets (selected), Assistance, Resolution, and Incident SOP. The main content area is divided into two columns. The left column, titled 'Similar Tickets', contains two sub-sections: 'Past Tickets' and 'Similar Open Tickets'. Each sub-section lists several tickets with their IDs and descriptions. The right column, titled 'Insights', contains a section for 'Alert Reversal Probability' with a status message. The 'Similar Tickets' section lists the following tickets:

Section	Ticket ID	Description
Past Tickets	INCIDENTE-1506067	KIOSK04 - Kioskos indisponible (Pantalla roja)
	INCIDENTE-1720501	KIOSK01 - Kioskos indisponible (Pantalla roja)
	INCIDENTE-1658427	KIOSK02 - Kioskos indisponible (Pantalla roja)
	INCIDENTE-1704209	KIOSK01 - Kioskos indisponible (Pantalla roja)
	INCIDENTE-1720488	KIOSK02 - Kioskos indisponible
Similar Open Tickets	INCIDENTE-1657140	KIOSK03/Kioskos indisponible (Pantalla roja)
	INCIDENTE-1629587	KIOSK05 - Kioskos indisponible (Pantalla roja)
	INCIDENTE-1636591	KIOSK05 - Kioskos indisponible (Pantalla roja)
	INCIDENTE-1670631	KIOSK02 - Kioskos indisponible (Pantalla roja)

The 'Insights' section on the right shows 'Alert Reversal Probability' with a status message: 'Alert Reversal flow is not enabled.' The interface also includes vertical scrollbars for the ticket lists.

Figura 16: Evidencia de recomendación de casos similares utilizando embeddings de texto y búsqueda vectorial

Fuente: Captura de pantalla del módulo de búsqueda semántica en la aplicación web.

REFERENCIAS BIBLIOGRÁFICAS

- [Agrawal *et al.*, 2018] Agrawal, A., Gans, J., y Goldfarb, A. (2018). *Prediction Machines: The Simple Economics of Artificial Intelligence*. Harvard Business Review Press.
- [Anthropic, 2024] Anthropic (2024). Claude 3.5 Sonnet: Advanced AI Assistant. Technical report, Anthropic. Disponible en: <https://www.anthropic.com/news/claude-3-5-sonnet>. [Consultado: 18 de abril de 2026].
- [AXELOS Limited, 2019] AXELOS Limited (2019). *ITIL Foundation: ITIL 4 Edition*. TSO (The Stationery Office).
- [Baldini *et al.*, 2017] Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., y Tardieu, P. (2017). Serverless Computing: Current Trends and Open Problems. En *Research Advances in Cloud Computing*, pp. 1–20. Springer.
- [Beyer *et al.*, 2016] Beyer, B., Jones, C., Petoff, J., y Murphy, N. R. (2016). *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media.
- [Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- [Chen *et al.*, 2019] Chen, J., He, X., Lin, Q., Xu, Y., Zhang, H., Hao, D., Gao, F., Xu, Z., Dang, Y., y Zhang, D. (2019). An Empirical Investigation of Incident Triage for Online Service Systems. En *Proc. IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. DOI: <https://doi.org/10.1109/ICSE-SEIP.2019.00020>. [Consultado: 5 de septiembre de 2025].
- [Cisco Systems, 2020] Cisco Systems (2020). Annual Internet Report (2018–2023). White paper, Cisco Systems. Disponible en: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>. [Consultado: 10 de octubre de 2025].
- [Dang *et al.*, 2019] Dang, Y., Lin, Q., y Huang, P. (2019). AIOps: Real-World Challenges and Research Innovations. En *Proc. IEEE/ACM International Conference on Software Engineering (ICSE)*, pp. 4–13. DOI: <https://dl.acm.org/doi/10.1109/ICSE-Companion.2019.00023>. [Consultado: 28 de marzo de 2026].
- [Devlin *et al.*, 2019] Devlin, J., Chang, M.-W., Lee, K., y Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. En *Proc. NAACL-HLT*, pp. 4171–4186. arXiv: <https://arxiv.org/abs/1810.04805>. [Consultado: 5 de febrero de 2026].

- [Forsgren *et al.*, 2018] Forsgren, N., Humble, J., y Kim, G. (2018). *Accelerate: The Science of Lean Software and DevOps*. IT Revolution Press. Basado en investigación empírica de 4 años con más de 30,000 profesionales de TI.
- [Gao *et al.*, 2024] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., y Wang, H. (2024). Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv preprint arXiv:2312.10997*. arXiv: <https://arxiv.org/abs/2312.10997>. [Consultado: 8 de mayo de 2026].
- [Google Cloud DORA (DevOps Research and Assessment), 2023] Google Cloud DORA (DevOps Research and Assessment) (2023). Accelerate State of DevOps Report 2023. Technical report, Google LLC. Disponible en: <https://cloud.google.com/devops/state-of-devops>. [Consultado: 22 de abril de 2026].
- [Gulenko *et al.*, 2018] Gulenko, A., Wallschläger, M., Schmidt, F., Kao, O., y Liu, F. (2018). Detecting Anomalous Behavior in Cloud Workloads Using Time-Series Analysis. En *Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pp. 1–9. DOI: <https://ieeexplore.ieee.org/document/8406267>. [Consultado: 8 de enero de 2026].
- [International Organization for Standardization, 2018] International Organization for Standardization (2018). ISO/IEC 20000-1:2018 Information technology — Service management — Part 1: Service management system requirements.
- [Johnson *et al.*, 2021] Johnson, J., Douze, M., y Jégou, H. (2021). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547. DOI: <https://ieeexplore.ieee.org/document/8733051>. [Consultado: 3 de marzo de 2026].
- [Kane, 2023] Kane, A. (2023). pgvector: Open-source vector similarity search for Postgres. <https://github.com/pgvector/pgvector>. [Consultado: 10 de abril de 2026].
- [Lewis *et al.*, 2020] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., y Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. En *Advances in Neural Information Processing Systems (NeurIPS)*, volumen 33, pp. 9459–9474. arXiv: <https://arxiv.org/abs/2005.11401>. [Consultado: 8 de abril de 2026].
- [Marrone y Kolbe, 2011] Marrone, M. y Kolbe, L. M. (2011). Impact of IT Service Management Frameworks on the IT Organization. *Business & Information Systems Engineering*, 3(1):5–18. DOI: <https://doi.org/10.1007/s12599-010-0141-5>. [Consultado: 15 de agosto de 2025].
- [Meta AI, 2024] Meta AI (2024). Llama 3.1: Open Foundation and Fine-Tuned Chat Models. Technical report, Meta. Disponible en: <https://ai.meta.com/blog/meta-llama-3-1/>. [Consultado: 25 de abril de 2026].

- [Nedelkoski *et al.*, 2019] Nedelkoski, S., Cardoso, J., y Kao, O. (2019). Anomaly Detection and Classification using Distributed Tracing and Deep Learning. En *Proc. IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 241–248. DOI: <https://doi.org/10.1109/CloudCom.2019.00046>. [Consultado: 8 de noviembre de 2025].
- [Notaro *et al.*, 2021] Notaro, P., Cardoso, J., y Gerndt, M. (2021). A Survey of AIOps Methods for Failure Management. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 12(6):1–45. DOI: <https://doi.org/10.1145/3483424>. [Consultado: 22 de agosto de 2025].
- [OpenAI, 2024] OpenAI (2024). GPT-4o: Multimodal Large Language Model. Technical report, OpenAI. Disponible en: <https://openai.com/index/hello-gpt-4o/>. [Consultado: 15 de abril de 2026].
- [Proveedor Tecnológico Internacional, 2023] Proveedor Tecnológico Internacional (2023). Implementing AIOps Enterprise Framework. Technical implementation guide, Proveedor Tecnológico Internacional. Documentación técnica interna (acceso restringido). [Consultado: 22 de enero de 2026].
- [Proveedor Tecnológico Internacional, 2024] Proveedor Tecnológico Internacional (2024). AIOps Enterprise Platform Documentation. Documentación técnica interna (acceso restringido). [Consultado: 25 de octubre de 2025].
- [Ramírez, 2024] Ramírez, S. (2024). FastAPI: Modern, fast web framework for building APIs with Python. <https://fastapi.tiangolo.com/>. [Consultado: 12 de mayo de 2026].
- [Reimers, 2024] Reimers, N. (2024). Sentence-Transformers: Multilingual Sentence, Paragraph, and Image Embeddings using BERT & Co. <https://www.sbert.net/>. [Consultado: 20 de mayo de 2026].
- [Reimers y Gurevych, 2019] Reimers, N. y Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. En *Proc. EMNLP-IJCNLP*, pp. 3982–3992. arXiv: <https://arxiv.org/abs/1908.10084>. [Consultado: 18 de febrero de 2026].
- [Runeson *et al.*, 2007] Runeson, P., Alexandersson, M., y Nyholm, O. (2007). Detection of Duplicate Defect Reports Using Natural Language Processing. En *Proc. International Conference on Software Engineering (ICSE)*, pp. 499–510. IEEE. DOI: <https://ieeexplore.ieee.org/document/4222606>. [Consultado: 5 de mayo de 2026].
- [Soldani y Brogi, 2022] Soldani, J. y Brogi, A. (2022). Anomaly detection and failure root cause analysis in (micro)service-based cloud applications: A survey. *ACM Computing Surveys (CSUR)*, 55(3):1–39. DOI: <https://dl.acm.org/doi/10.1145/3501297>. [Consultado: 15 de diciembre de 2025].

[Sommerville, 2015] Sommerville, I. (2015). *Software Engineering (10th ed.)*. Pearson.

[The PostgreSQL Global Development Group, 2023] The PostgreSQL Global Development Group (2023). PostgreSQL 15 Documentation. <https://www.postgresql.org/docs/15/>. [Consultado: 28 de mayo de 2026].