

PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR
SEDE MANABÍ

CARRERA DE SOFTWARE

TRABAJO DE TITULACIÓN

DESARROLLO DE UNA APLICACIÓN WEB CON INTELIGENCIA ARTIFICIAL PARA LA GESTIÓN AUTOMATIZADA DE REPORTES DE INCIDENTES EN EL CAMPUS UNIVERSITARIO, DIRIGIDA A ESTUDIANTES DE LA PUCE MANABÍ

LÍNEA DE INVESTIGACIÓN
TECNOLOGÍA DE LA INFORMACIÓN Y LA COMUNICACIÓN

SUBLÍNEA DE INVESTIGACIÓN
INGENIERÍA DE SOFTWARE, INNOVACIÓN Y EMPRENDIMIENTO EN TIC

PREVIO AL TÍTULO DE
INGENIERO EN SOFTWARE

AUTOR
LUIS ALBERTO VELASCO ANCHUNDIA

TUTOR
ING. JOSSELYN TATIANA GÓMEZ BRAVO, M.Sc.

PORTOVIEJO, ABRIL 2026

Certificación del Tutor

En mi calidad de tutor del trabajo de integración curricular, certifico haber revisado el presente manuscrito de investigación, el mismo que se ajusta a las normas vigentes de la Pontificia Universidad Católica del Ecuador Sede Manabí, cumpliendo la Normativa del Trabajo de Integración Curricular; en consecuencia, es apto para su presentación y sustentación.
En la ciudad de Portoviejo a los veinticinco días del mes de marzo de dos mil veintiséis.

ING. JOSSELYN TATIANA GÓMEZ BRAVO, M.Sc.

CI: 1315046498

Tutora

2026

Acta de aprobación del Tribunal

El jurado examinador aprueba el presente trabajo de integración curricular en nombre de la Pontificia Universidad Católica del Ecuador, Sede Manabí.

En la ciudad de Portoviejo a los veinticinco días del mes de marzo de dos mil veintiséis.

ING. JOSÉ NARANJO, M. Eng
CI: 1003528674
Lector Revisor 1

ING. VICTOR ALONZO, Mgtr.
CI: 1307757441
Lector Revisor 2

ING. JOSSELYN GÓMEZ, M.Sc.
CI: 1315046498
Tutora

2026

Declaración de Originalidad

Este manuscrito no contiene ningún tipo de material que ha sido aceptado para la obtención de un título universitario en otra institución, excepto en forma de información de soporte que ha sido debidamente citada en mi trabajo. Este trabajo es de total responsabilidad del autor, quien declara bajo juramento que ninguna sección de este trabajo de integración curricular infringe los derechos de autor de nadie.

En la ciudad de Portoviejo a los veinticinco días del mes de marzo de dos mil veintiséis.

LUIS ALBERTO VELASCO ANCHUNDIA

CI: 1312172818

Autor

2026

Declaración de Derecho del Autor

Autorizo a la Pontificia Universidad Católica del Ecuador a distribuir este manuscrito de investigación en medios físicos y electrónicos con el fin de promover la divulgación de mis resultados a la comunidad científica y a la sociedad en general. Adicionalmente autorizo el uso de los contenidos de esta investigación como bibliografía para fines académicos, por cualquier medio o procedimiento, citando como fuente de información al autor de este trabajo. En la ciudad de Portoviejo a los veinticinco días del mes de marzo de dos mil veintiséis.

LUIS ALBERTO VELASCO ANCHUNDIA

CI: 1312172818

Autor

2026

Dedicatoria

Dedico este trabajo, ante todo, a mis padres, los pilares sobre los que he construido este sueño y la razón de cada uno de mis logros. A mi madre, por su presencia constante, su amor incondicional y su apoyo inquebrantable en cada desafío; gracias por ser mi fortaleza en los momentos más difíciles. A mi padre, por brindarme las oportunidades y el entorno necesario para crecer, por su confianza y por enseñarme, con su ejemplo, el valor de la perseverancia. Este logro es, en gran medida, reflejo de todo lo que ustedes han hecho por mí.

Y con profunda emoción, a mi abuela María, quien siempre creyó en mí y con quien compartí tantas conversaciones imaginando este día. Aunque hoy no pueda entregarte este título en vida, sé que me acompañas con orgullo. Este logro también es tuyo, como testimonio de tu amor y de la huella imborrable que dejaste en mi vida.

Agradecimientos

Expreso mi gratitud a mis profesores, quienes con su exigencia, paciencia y vocación docente no solo me transmitieron conocimientos técnicos, sino que contribuyeron a la formación de mi criterio profesional. Gracias por ser mentores y por compartir conmigo las herramientas que hoy me permiten alcanzar este hito. De manera especial, agradezco a mi tutora, por su guía acertada y su constante disposición para orientar esta investigación hacia buen puerto.

A mis padres, por ser el pilar fundamental de mi vida. Gracias por sus sacrificios silenciosos, sus enseñanzas diarias y por ese respaldo incondicional que nunca faltó. Sin su esfuerzo y acompañamiento permanente, este logro no habría sido posible; son ustedes los principales artífices de mi formación.

A mi hermano, por su compañía y su cariño infaltable a lo largo de este camino. Deseo que este logro pueda servirle como motivación para seguir construyendo su propio camino y alcanzar sus sueños.

Finalmente, a mis amigos y compañeros, quienes hicieron más ligero el camino. A todos los que, directa o indirectamente, aportaron con un consejo, una palabra de aliento o simplemente con su compañía en los momentos de mayor exigencia. Gracias por formar parte de mi historia.

Resumen

La gestión ineficiente de servicios en entornos universitarios, caracterizada a menudo por la pérdida de información en canales informales, motivó la presente investigación en la PUCE Manabí. El objetivo principal fue desarrollar y validar técnicamente un prototipo de aplicación web integrado con un módulo de Inteligencia Artificial para automatizar la clasificación y asignación de reportes de incidentes estudiantiles. La metodología aplicada abarcó el ciclo completo de ingeniería de software, desde el análisis de requisitos y la definición de una taxonomía basada en datos recientes, hasta el diseño de una arquitectura *serverless* utilizando modelos de lenguaje (LLM). Los resultados evidencian que el sistema alcanza una precisión del 90 % en la clasificación automática y una latencia promedio de 2.4 segundos, logrando estandarizar el 100 % de las entradas en el entorno de pruebas. Adicionalmente, se valida un mecanismo de seguridad que identifica y deriva casos ambiguos a revisión manual. Se concluye que la solución tecnológica propuesta optimiza la trazabilidad operativa y reduce significativamente la carga administrativa, demostrando la viabilidad técnica de implementar estrategias de *smart campus* mediante el uso de IA generativa en el contexto local.

Palabras clave: automatización, incidentes, inteligencia artificial, modelos de lenguaje de gran tamaño

Abstract

The inefficient management of services in university environments, often characterized by information loss across informal communication channels, motivated the present study conducted at PUCE Manabí. The primary objective was to develop and technically validate a web-based application prototype integrated with an Artificial Intelligence module to automate the classification and assignment of student incident reports. The methodology encompassed the full software engineering lifecycle, including requirements analysis and the definition of a data-driven taxonomy based on recent datasets, as well as the design of a serverless architecture leveraging large language models (LLMs). The results demonstrate that the system achieves a 90 % accuracy rate in automated classification and an average latency of 2.4 seconds, successfully standardizing 100 % of input data within the testing environment. Furthermore, a security mechanism was validated to identify ambiguous cases and route them for manual review. It is concluded that the proposed technological solution enhances operational traceability and significantly reduces administrative workload, thereby demonstrating the technical feasibility of implementing smart campus strategies through the use of generative AI within the local context.

Keywords: automation, incident management, artificial intelligence, large language models

Abreviaturas, siglas y términos

A continuación, se definen las siglas técnicas, acrónimos y términos específicos utilizados en el desarrollo de este trabajo de investigación:

- **API**: *Application Programming Interface* (Interfaz de Programación de Aplicaciones)
- **BaaS**: *backend as a service* (Backend como Servicio, referente a la plataforma Supabase)
- **Clasificación**: Proceso de categorizar un incidente y determinar el departamento responsable mediante el uso de algoritmos.
- **Confianza**: Valor numérico (escala 0-1) que indica el nivel de certeza o probabilidad que el modelo de IA asigna a su predicción.
- **Incidente**: Reporte de un problema, avería o situación anómala dentro del campus que requiere atención operativa.
- **JWT**: *JSON Web Token* (Estándar abierto para la creación de tokens de acceso seguros)
- **LLM**: *Large Language Model* (Modelo de Lenguaje Grande utilizado para la clasificación automática de texto)
- **mobile-first**: Enfoque de diseño web que prioriza la optimización de la experiencia de usuario en dispositivos móviles antes que en escritorio.
- **Onboarding**: Proceso de introducción guiada para familiarizar a los nuevos usuarios con las funcionalidades del sistema.
- **RBAC**: *Role-Based Access Control* (Control de Acceso Basado en Roles de usuario)
- **REST**: *Representational State Transfer* (Estilo arquitectónico de software para sistemas hipermedia distribuidos)

- **RLS**: *Row Level Security* (Seguridad a Nivel de Fila implementada en PostgreSQL)
- **SLA**: *Service Level Agreement* (Acuerdo de Nivel de Servicio que define los tiempos máximos de respuesta esperados)
- **SSR**: *Server-Side Rendering* (Renderizado del lado del servidor para mejorar el rendimiento de carga inicial)

Tabla de Contenido

1 Introducción **1**

Contexto y justificación 1

Justificación práctica e institucional 2

Justificación social 3

Justificación tecnológica 3

Planteamiento del problema 3

Preguntas de investigación 5

Pregunta general 5

Preguntas específicas 5

Objetivos de la investigación 6

Objetivo general 6

Objetivos específicos 6

Alcance 7

Antecedentes 8

Marco Conceptual 10

Smart Campus (Campus Inteligente) 10

Modelos de lenguaje de gran escala (LLMs) 10

Procesamiento de Lenguaje Natural (NLP) 11

Interfaz de Programación de Aplicaciones (API) 11

Seguridad a Nivel de Fila (RLS) 11

Renderizado del Lado del Servidor (SSR) 12

JSON Web Token (JWT) 12

2 Metodología **13**

Enfoque metodológico 13

Componente cualitativo 13

Componente cuantitativo 14

Diseño de investigación 16

Fase 1: identificación del problema y definición de objetivos 17

Fase 2: diseño y arquitectura de la solución 18

Fase 3: construcción y desarrollo (prototipado) 22

Fase 4: validación y evaluación técnica 22

Población y muestra 23

Población y muestra de informantes (fase de análisis) 23

Población y muestra de datos (fase de validación) 24

Técnicas e instrumentos 25

Técnicas de análisis y diseño 25

Técnicas de desarrollo (construcción del artefacto) 26

Técnicas de validación 26

Procedimientos 27

Fase 1: formalización de requisitos (estándar de ingeniería de requisitos) 27

Fase 2: estrategia de desarrollo ágil e incremental 28

Fase 3: selección del modelo de inteligencia artificial 32

Fase 4: ingeniería de prompts e iteración 34

Fase 5: validación técnica 34

Consideraciones éticas 35

Privacidad y protección de datos personales 35

Transparencia y responsabilidad algorítmica 35

Manejo de datos en la fase de validación 36

3 Resultados **37**

Resultados del análisis de requisitos y definición de taxonomía 37
Evaluación del desempeño técnico de la arquitectura 41
Resultados del desarrollo del prototipo funcional 42
Validación del cumplimiento de requisitos y modelo IA 44
Matriz de confusión y precisión del modelo 44
Caso de prueba documentado de forma integral 45
Estadística de cumplimiento de requisitos funcionales 46
Síntesis de hallazgos 48

4 Discusión **49**
Interpretación de los resultados 49
Comparación con estudios previos 50
Fortalezas y limitaciones 51
Implicaciones y recomendaciones 53

5 Conclusiones **54**

Referencias bibliográficas **60**

Lista de Figuras

1 Taxonomía jerárquica de clasificación de incidentes 15

2 Arquitectura del sistema web por capas 19

3 Flujo del proceso de gestión de incidentes 20

4 Ejemplo técnico del flujo de clasificación IA 21

5 Diagrama Entidad-Relación del sistema 30

6 Comparativa de rendimiento entre modelos LLM 33

7 Distribución estimada de incidentes (Pareto) 38

8 Comparativa de dispersión de canales 40

9 Latencia de procesamiento por transacción 41

10 Interfaces finales del prototipo móvil 43

11 Matriz de confusión del corpus de validación 44

Lista de Tablas

1 Taxonomía de clasificación definida a partir de la frecuencia de incidentes 39

2 Documentación completa del caso de prueba CP-04 45

3 Resumen estadístico de conformidad de requisitos por módulo 47

4 Requisitos funcionales del sistema - Módulo de autenticación 67

5 Requisitos funcionales del sistema - Módulo de incidentes 68

6 Requisitos funcionales del sistema - Módulo de clasificación IA 69

7 Requisitos funcionales del sistema - Módulo de gestión departamental 70

8 Requisitos funcionales del sistema - Módulo de flujo del proceso 71

9 Requisitos funcionales del sistema - Módulo de comentarios 71

10 Requisitos funcionales del sistema - Módulo de administración 72

11 Requisitos no funcionales del sistema - Seguridad 72

12 Requisitos no funcionales del sistema - Usabilidad 73

13 Requisitos no funcionales del sistema - Mantenibilidad y escalabilidad 73

14 Requisitos no funcionales del sistema - Disponibilidad e integrabilidad 73

15 Resultados detallados de la clasificación automática (corpus de validación) 81

16 Ejecución de la lista de verificación (*checklist*) 83

17 Criterios de evaluación para la selección del modelo de IA 85

18 Indicadores clave de rendimiento de Gemini 3 Flash 86

19 Estimaciones de frecuencia de incidentes proporcionadas por el informante clave . 89

20 Cálculo de frecuencia acumulada para el análisis de Pareto 90

Lista de Apéndices

Apéndice A: Guía de entrevista semi-estructurada 62

Apéndice B: Especificación de requisitos del sistema (SRS) 64

Apéndice C: Descripción del flujo de procesos del sistema 74

Apéndice D: Desglose de la arquitectura técnica del sistema 77

Apéndice E: Matriz de ejecución de pruebas del modelo de IA 80

Apéndice F: Lista de verificación de requisitos (inspección funcional) 82

Apéndice G: Análisis comparativo y criterios de selección del modelo de IA 84

Apéndice H: Fundamentación de la distribución estimada de incidentes 88

Introducción

Contexto y justificación

La transformación digital en la educación superior ha dejado de ser una opción para convertirse en una necesidad estratégica. En el contexto actual, las instituciones académicas buscan evolucionar hacia modelos de gestión conocidos como *Smart Campus* o Campus Inteligente (Min-Allah & Alrashed, 2020). Esta tendencia no se limita únicamente a la implementación de hardware o conectividad, sino que busca fundamentalmente optimizar la eficiencia operativa y la gestión de recursos institucionales mediante la aplicación de tecnología avanzada e investigación, redefiniendo la manera en que se administran los servicios universitarios. En este marco, la administración de la infraestructura física y los servicios de soporte se convierte en un área crítica para la digitalización, alineándose con los esfuerzos regionales por modernizar los servicios públicos y privados para garantizar transparencia y eficiencia (Cristia & Vlaicu, 2022).

Sin embargo, la implementación de plataformas web para la gestión de servicios presenta desafíos particulares. Investigaciones recientes sugieren que el éxito de estas herramientas no depende exclusivamente de la disponibilidad tecnológica, sino de la integración efectiva entre la calidad técnica del sistema y la facilidad de uso percibida (Sri Wahyuningsih, Herli Sumerli et al., 2024). En este sentido, la literatura actual enfatiza que la calidad en el diseño de la interfaz y los mecanismos de captura de información son determinantes para la satisfacción y adopción estudiantil de los portales universitarios (Adeyemi & Issa, 2020). Estudios previos sobre transparencia y gestión digital, como los coordinados por Belloso de Núñez y Primera Mendoza (2015), indican que la eficacia de estos sistemas está intrínsecamente ligada a criterios de usabilidad y objetividad; si la información no se captura correctamente desde el inicio, el proceso

de gestión se vuelve ineficiente.

Paralelamente, el avance acelerado de la Inteligencia Artificial (IA), y específicamente de los Modelos de Lenguaje (LLMs) (Zhao et al., 2024), ha abierto nuevas posibilidades para el procesamiento de información no estructurada. Investigaciones recientes han demostrado la viabilidad técnica de utilizar IA generativa y Procesamiento de Lenguaje Natural (NLP) (Hirschberg & Manning, 2015) para automatizar la clasificación de reportes textuales. Un ejemplo relevante es el estudio de Hölzing et al., 2024, el cual evidenció que el rendimiento de modelos de IA en la identificación y análisis de incidentes críticos es estadísticamente comparable, e incluso complementario, al de expertos humanos, validando el uso de esta tecnología para gestionar reportes de texto libre con alta precisión.

En el contexto nacional ecuatoriano, las Instituciones de Educación Superior (IES) han comenzado a desarrollar soluciones tecnológicas internas como respuesta a los retos de la era digital (Murillo Quimiz et al., 2024). Existen antecedentes metodológicos validados internacionalmente que documentan el desarrollo de módulos web propios para la gestión de procesos académicos y administrativos, permitiendo una mejor adaptación a los requisitos institucionales específicos (Saltos-Rivas et al., 2021). No obstante, la revisión de la literatura actual no evidencia una integración documentada de estas tecnologías diseñada específicamente para la taxonomía de incidentes y mantenimiento en el contexto de un campus universitario local.

Bajo estas premisas, la presente investigación se justifica en tres dimensiones fundamentales: práctica, social y tecnológica.

Justificación práctica e institucional

Para la Pontificia Universidad Católica del Ecuador Sede Manabí (PUCE Manabí), el desarrollo del prototipo propuesto resuelve una ineficiencia operativa crítica. Actualmente, la gestión de incidentes se realiza mediante canales informales que carecen de estandarización. La implementación de este sistema permitirá automatizar la clasificación y el enrutamiento de

reportes, otorgando a la institución un control sistemático y trazabilidad sobre las solicitudes. Esto facilitará la transición de un modelo de gestión reactivo y manual a uno basado en datos, optimizando la asignación de recursos a los departamentos de mantenimiento, seguridad y limpieza.

Justificación social

El proyecto impacta directamente en la experiencia de la comunidad estudiantil. Al proveer un canal de reporte único, accesible y transparente, se elimina la incertidumbre y frustración asociadas a los procesos actuales. El sistema democratiza el acceso a los servicios del campus, permitiendo que los estudiantes no solo reporten incidencias de manera sencilla, sino que también puedan visualizar el estado y la resolución de sus solicitudes, fomentando una cultura de participación y cuidado de los espacios universitarios.

Justificación tecnológica

Este trabajo contribuye a la estrategia de modernización del campus mediante la aplicación validada de tecnologías de vanguardia. La integración de un modelo de Inteligencia Artificial dentro de una arquitectura web moderna no solo automatiza una tarea cognitiva (la clasificación de incidentes), sino que sirve como un caso de estudio replicable sobre cómo los LLMs pueden ser integrados en flujos de trabajo administrativos reales. El proyecto demuestra cómo la tecnología *Smart Campus* puede ser adaptada y aplicada efectivamente en el contexto local para resolver problemas cotidianos de gestión.

Planteamiento del problema

A nivel global, la gestión de servicios institucionales se orienta hacia la digitalización y la automatización, impulsada por la necesidad imperativa de optimizar la eficiencia operativa y

garantizar la transparencia. La literatura reciente valida la adopción de la IA y los LLMs como herramientas efectivas para el análisis crítico y la clasificación de grandes volúmenes de datos no estructurados, superando las limitaciones del procesamiento manual (Hölzing et al., 2024). Este enfoque tecnológico se alinea con el concepto de *smart campus*, un modelo estratégico que las instituciones de educación superior buscan implementar para mejorar la calidad de sus servicios mediante la estandarización (Min-Allah & Alrashed, 2020).

A pesar de esta clara directriz global hacia la modernización, el contexto local en la PUCE Manabí presenta una brecha tecnológica significativa. Actualmente, el proceso de gestión de incidentes cotidianos en el campus, se caracteriza por la ausencia de un canal formal y centralizado. La comunicación se realiza a través de medios informales y desestructurados, principalmente mediante aplicaciones de mensajería instantánea como WhatsApp o correos electrónicos extensos y no estandarizados.

Esta informalidad constituye el núcleo de la problemática. La dependencia de canales no oficiales genera un flujo de información caótico, donde los reportes carecen de trazabilidad, se duplican las solicitudes y la clasificación de los problemas depende exclusivamente de la intervención humana. Este enfoque manual crea un cuello de botella operativo: el personal encargado debe leer, interpretar y derivar cada mensaje individualmente, lo que es propenso a errores y demoras significativas.

La ineficiencia del modelo actual de gestión de incidencias impacta negativamente en la comunidad universitaria. En el contexto actual de la institución, se observa que los procesos para el reporte y seguimiento de incidencias presentan demoras, falta de trazabilidad y una experiencia poco práctica para los estudiantes, lo que genera inconformidad y reduce la efectividad de la atención. Desde la perspectiva institucional, la inexistencia de una base de datos estructurada limita el control de la información y la generación de indicadores de desempeño, dificultando la toma de decisiones orientadas a la mejora continua.

Esta problemática es consistente con lo reportado en estudios recientes sobre transformación

digital en instituciones de educación superior, donde se evidencia que la gestión de procesos mediante sistemas no centralizados o con bajo nivel de digitalización provoca ineficiencias operativas, pérdida de información y limita la toma de decisiones basada en datos (Alenezi, 2023; Díaz-García, Montero-Navarro et al., 2023).

En consecuencia, el problema central no se limita únicamente a la ausencia de una plataforma web, sino a la falta de un mecanismo inteligente capaz de interpretar la ambigüedad del lenguaje natural con el que los estudiantes describen las incidencias. La carencia de un sistema que integre la captura estructurada de datos con procesos de clasificación automatizada perpetúa un modelo de gestión reactivo, poco escalable e ineficiente.

Preguntas de investigación

Pregunta general

¿Cómo se puede diseñar, desarrollar y validar técnicamente un prototipo de aplicación web, integrada con Inteligencia Artificial, para automatizar la clasificación de reportes de incidentes en la PUCE Manabí, basándose en la taxonomía institucional y los requisitos del personal de gestión?

Preguntas específicas

1. ¿Cuáles son los requisitos funcionales y no funcionales que debe cumplir el prototipo, derivados del diagnóstico del proceso actual de gestión de incidentes y de la taxonomía de clasificación institucional de la PUCE Manabí?
2. ¿Cómo se puede diseñar la arquitectura que integre la aplicación web y ejecute la clasificación automática de los reportes de incidentes mediante Inteligencia Artificial?
3. ¿De qué manera se puede desarrollar el prototipo funcional de la aplicación web para

implementar el flujo de clasificación automatizada de incidentes, desde la captura del reporte hasta su asignación departamental?

4. ¿Cuál es el nivel de exactitud del modelo de Inteligencia Artificial en la clasificación automática de incidentes, evaluado mediante un corpus de prueba controlado?
5. ¿Cuál es el grado de cumplimiento de los requisitos funcionales del prototipo desarrollado, verificado mediante una lista de inspección funcional?

Objetivos de la investigación

Objetivo general

Desarrollar un prototipo funcional de una aplicación web, integrada con un módulo de Inteligencia Artificial, para la gestión automatizada de la clasificación de reportes de incidentes generados por los estudiantes de la PUCE Manabí.

Objetivos específicos

- Analizar el proceso actual de gestión de incidentes y definir los requisitos funcionales, no funcionales y la taxonomía de clasificación de la PUCE Manabí, mediante entrevistas con el personal clave.
- Diseñar la arquitectura que integre la aplicación web y ejecute el módulo de clasificación automática de IA.
- Desarrollar el prototipo funcional de la aplicación web y el módulo de IA, basado en la arquitectura y los requisitos definidos.
- Evaluar la exactitud del modelo de Inteligencia Artificial en la clasificación automática de incidentes mediante un corpus de prueba controlado y estratificado.

- Verificar el grado de cumplimiento de los requisitos funcionales del prototipo mediante una lista de inspección funcional por módulos del sistema.

Alcance

El alcance de la presente investigación se circunscribe al diseño, desarrollo y validación técnica de un prototipo funcional (*Proof of Concept* - PoC). Según Kendig (2016), la investigación basada en pruebas de concepto no se limita a la construcción de un artefacto, sino que constituye una metodología fundamental para validar la viabilidad epistémica y práctica de una nueva tecnología en un contexto inexplorado. Bajo esta premisa, el proyecto abarca el ciclo completo de ingeniería de software, desde la conceptualización basada en el levantamiento de requisitos hasta la verificación experimental de la precisión del componente de Inteligencia Artificial.

Alcance funcional y tecnológico: el desarrollo del artefacto de software contempla la implementación de una arquitectura *full-stack* moderna y escalable. Específicamente, el alcance técnico incluye:

- Plataforma web: construcción de una aplicación web progresiva utilizando el *framework* Next.js, que integra tanto la interfaz de usuario (Frontend) como la lógica de negocio (Backend). El sistema permite la gestión de roles diferenciados (estudiante, trabajador, administrador) y asegura la integridad de los datos mediante validación de correos institucionales.
- Módulo de inteligencia artificial: integración del modelo LLM vía API para la clasificación automática de reportes. El alcance de la clasificación cubre una taxonomía específica definida para el campus, compuesta por seis categorías fijas: mantenimiento, infraestructura, seguridad, limpieza, tecnología y Otros. La categoría Otros es una cola de control reservada para incidentes que el modelo no puede clasificar con confianza suficiente ($C < 0{,}75$), los cuales son derivados automáticamente a revisión manual.

- Infraestructura de datos: implementación de una base de datos relacional PostgreSQL gestionada a través de Supabase, incluyendo políticas de seguridad a nivel de fila (RLS) para garantizar la privacidad de la información y servicios de almacenamiento para la evidencia fotográfica de los incidentes.

Alcance de la validación: la fase de validación se limita estrictamente a la evaluación técnica del prototipo. Se medirá el desempeño del modelo de IA mediante métricas de precisión (*accuracy*), comparando la clasificación automática frente a un corpus de datos de prueba etiquetado manualmente. Asimismo, se verificará el cumplimiento de los requisitos funcionales.

Limitaciones y exclusiones: es fundamental establecer que el entregable final es un prototipo validado en un entorno de pruebas controlado (entorno de desarrollo/staging). Por lo tanto, el alcance del proyecto excluye explícitamente:

1. La implementación, despliegue o puesta en producción del sistema en los servidores o infraestructura física real de la PUCE Manabí.
2. La ejecución de pruebas masivas con usuarios finales reales (estudiantes) o evaluaciones de usabilidad (UX) en campo.
3. La medición de impacto organizacional, tiempos de respuesta operativa o pruebas de estrés y carga del servidor.

Antecedentes

La revisión de la literatura permite identificar tres líneas de investigación que constituyen los antecedentes directos de este trabajo: la clasificación automática de incidentes mediante aprendizaje automático, la aplicación de procesamiento de lenguaje natural para el soporte técnico, y el desarrollo de sistemas de gestión de incidentes en contextos universitarios.

En el ámbito de la clasificación automática, Miliano et al. (2020) desarrollaron un sistema de

categorización de ticketsa de helpdesk basado en aprendizaje automático, obteniendo una exactitud del 82 % con el algoritmo Random Forest. Su trabajo evidencia que la categorización manual de incidentes por parte de operadores humanos es propensa a errores de enrutamiento y que la automatización del proceso reduce significativamente los tiempos de atención. Este estudio sienta las bases técnicas para la asignación automática de incidentes a sus unidades de atención correspondientes mediante algoritmos supervisados.

Complementando esta línea, Nicholls et al. (2022) propusieron un sistema de clasificación de correos de helpdesk universitario mediante redes neuronales, entrenado sobre un corpus real de texto no estructurado, con el objetivo de generar respuestas automáticas contextualizadas al tipo de problema detectado. Su trabajo valida que el procesamiento de lenguaje natural puede interpretar con precisión las descripciones informales con las que los usuarios reportan incidentes técnicos, superando las limitaciones de los enfoques basados en palabras clave rígidas.

Desde la perspectiva institucional, Alobuela Carrillo (2022) documentó las deficiencias en la gestión de incidentes de TI en la Pontificia Universidad Católica del Ecuador (PUCE) y propuso una estrategia de mejora basada en el marco de referencia ITIL 4. Su diagnóstico evidencia que la ausencia de un proceso formal y estandarizado para la gestión de solicitudes constituye un problema reconocido a nivel institucional en la misma red universitaria en la que se desarrolla la presente investigación. No obstante, dicha propuesta se limitó al ámbito de TI interno y no contempló la automatización mediante inteligencia artificial ni una interfaz digital de captura para usuarios finales del campus.

Estos antecedentes demuestran, por separado, la viabilidad técnica de la clasificación automática de incidentes con ML/NLP y la necesidad institucional de formalizar la gestión de reportes. Sin embargo, la revisión de la literatura no identificó trabajos que integren conjuntamente una interfaz web de captura multimodal, una taxonomía operativa definida a partir del contexto específico del campus, y un modelo de lenguaje de gran escala (LLM) para la clasificación y enrutamiento automático en una única solución. Esta brecha constituye el espacio de contribución

que ocupa la presente investigación.

Marco Conceptual

Para la correcta interpretación del alcance y la solución técnica propuesta, es fundamental establecer las definiciones teóricas de los conceptos clave que sustentan esta investigación.

Smart Campus (Campus Inteligente)

Se define como un entorno educativo que integra tecnologías avanzadas de información y comunicación (TIC) con la infraestructura física para mejorar la toma de decisiones, la eficiencia operativa y la calidad de vida de la comunidad universitaria. Según Min-Allah y Alrashed (2020), un Smart Campus no se limita a la conectividad, sino que evoluciona hacia un ecosistema sensible al contexto que utiliza datos para automatizar servicios y responder proactivamente a las necesidades de estudiantes y personal administrativo. En el presente proyecto, este concepto se materializa mediante la digitalización del servicio de mantenimiento.

Modelos de lenguaje de gran escala (LLMs)

Los LLMs son algoritmos de inteligencia artificial basados en aprendizaje profundo, entrenados con vastos conjuntos de datos textuales para entender, resumir, generar y predecir nuevo contenido. Zhao et al. (2024) los describe como modelos capaces de capturar la sintaxis y semántica del lenguaje humano con una complejidad sin precedentes, permitiendo realizar tareas de razonamiento generalizado. En esta investigación, el LLM actúa como el motor cognitivo encargado de interpretar la ambigüedad en los reportes de incidentes.

Procesamiento de Lenguaje Natural (NLP)

Es una rama de la inteligencia artificial que se ocupa de la interacción entre las computadoras y el lenguaje humano. Chowdhary (2020) define al NLP como la disciplina que habilita a las máquinas para leer y comprender el lenguaje en su contexto natural, extrayendo significado e intención. A diferencia de los sistemas basados en palabras clave rígidas, las técnicas modernas de NLP permiten analizar descripciones informales (ej. jerga estudiantil) y estructurarlas en categorías técnicas definidas.

Interfaz de Programación de Aplicaciones (API)

Una API (*Application Programming Interface*) es un conjunto de definiciones y protocolos que permite la comunicación e integración entre componentes de software heterogéneos. En su tesis doctoral, Fielding (2000) estableció los principios arquitectónicos REST (*Representational State Transfer*), los cuales rigen la mayoría de las APIs web modernas. En el contexto de este prototipo, las APIs funcionan como los puentes que orquestan el flujo de datos entre la interfaz de usuario, la base de datos y el motor de inteligencia artificial externo.

Seguridad a Nivel de Fila (RLS)

La Seguridad a Nivel de Fila (*Row Level Security* - RLS) es una característica de los sistemas de gestión de bases de datos relacionales que permite controlar el acceso a filas específicas de una tabla basándose en las características del usuario que ejecuta la consulta. A diferencia de la seguridad tradicional basada en la lógica de la aplicación, RLS delega la autorización directamente al motor de base de datos. Según la documentación técnica de The PostgreSQL Global Development Group (2023), esto asegura que las políticas de privacidad se apliquen de manera consistente, independientemente de la interfaz desde la que se acceda a los datos. En este proyecto, RLS es el mecanismo que garantiza que cada estudiante visualice únicamente sus propios reportes, aislando la información sensible a nivel de infraestructura.

Renderizado del Lado del Servidor (SSR)

El Renderizado del Lado del Servidor (*server-side rendering* - SSR) es una técnica de desarrollo web donde el código HTML de una página se genera en el servidor para cada petición, enviando al navegador una estructura lista para ser visualizada. Vavera et al. (2017) explica que, a diferencia del renderizado en el cliente (CSR) típico de las aplicaciones de una sola página (SPA), el SSR mejora significativamente los tiempos de carga inicial y el rendimiento en dispositivos con recursos limitados. Para este prototipo, el uso de SSR (implementado a través de Next.js) es fundamental para cumplir con el requisito de accesibilidad móvil (*mobile-first*), asegurando una respuesta rápida incluso en conexiones inestables dentro del campus.

JSON Web Token (JWT)

El estándar Jones et al. (2015) define al *JSON web token (JWT)* como un método compacto y autónomo para transmitir información de forma segura entre partes como un objeto JSON. Estos tokens permiten la autenticación sin estado (*stateless*), ya que contienen toda la información necesaria sobre la identidad y los privilegios del usuario dentro de su carga útil, firmada digitalmente para evitar manipulaciones. En la arquitectura propuesta, los JWT son el vehículo que transporta la identidad del estudiante desde el cliente web hasta la base de datos, permitiendo que las políticas de seguridad (RLS) identifiquen al usuario sin necesidad de consultas de sesión redundantes en el servidor.

Metodología

La presente investigación adopta un enfoque sistemático y estructurado para el desarrollo del prototipo de software. A continuación, se detalla el enfoque metodológico utilizado para el análisis de la información y el diseño de la investigación que guía la construcción del artefacto tecnológico.

Enfoque metodológico

De acuerdo con la naturaleza del problema y los objetivos planteados, la presente investigación se define como de tipo aplicada. Este enfoque se justifica dado que el estudio no busca únicamente la generación de conocimiento teórico, sino la aplicación práctica de tecnologías existentes, específicamente LLMs para la resolución de un problema operativo específico: la gestión de incidentes en el campus de la PUCE Manabí.

El estudio se rige bajo un paradigma mixto (cualitativo - cuantitativo), integrando ambos enfoques para lograr una comprensión holística del fenómeno y una validación técnica robusta de la solución propuesta.

Componente cualitativo

En la fase inicial de análisis, se aplicó un enfoque cualitativo para alinear la lógica del sistema con la realidad operativa del campus. A través de una entrevista semi-estructurada realizada a los gestores de infraestructura (cuyos detalles se encuentran en el Apéndice A), se definieron las reglas de negocio para la clasificación automática.

Los hallazgos de este instrumento fueron determinantes para estructurar la base de datos.

Específicamente, en respuesta a la validación de categorías (pregunta 2), se confirmó la necesidad de segregar los problemas tecnológicos de los estructurales. Asimismo, ante la interrogante sobre el manejo de reportes ambiguos (pregunta 3), el gestor enfatizó la importancia de un mecanismo de derivación para revisión humana, lo que justificó la creación de la categoría "Otros".

Basado en estos requisitos, se estableció una taxonomía jerárquica compuesta por seis macro-categorías que permiten segmentar la naturaleza del incidente. Estas áreas se definieron de la siguiente manera:

- *Mantenimiento*: agrupa reparaciones operativas cotidianas (fontanería, electricidad, mobiliario), asignadas a cuadrillas de mantenimiento general.
- *Infraestructura*: comprende daños estructurales mayores a la edificación que requieren intervención técnica especializada.
- *Seguridad*: categoría crítica para la protección física, que prioriza amenazas directas, control de accesos y reportes de vandalismo.
- *Limpieza*: enfocada en la higiene de espacios comunes y la gestión de residuos.
- *Tecnología*: agrupa exclusivamente los fallos de conectividad y hardware institucional.
- *Otros*: una categoría de control diseñada para capturar la ambigüedad. Si la confianza del modelo es baja, el incidente se deriva aquí para revisión manual.

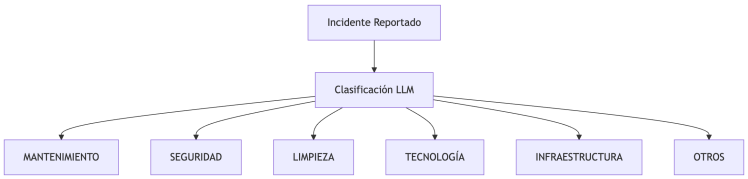
El desglose completo de los sub-tipos técnicos (*labels*) utilizados por el modelo para el enrutamiento se detalla visualmente en la Figura 1.

Componente cuantitativo

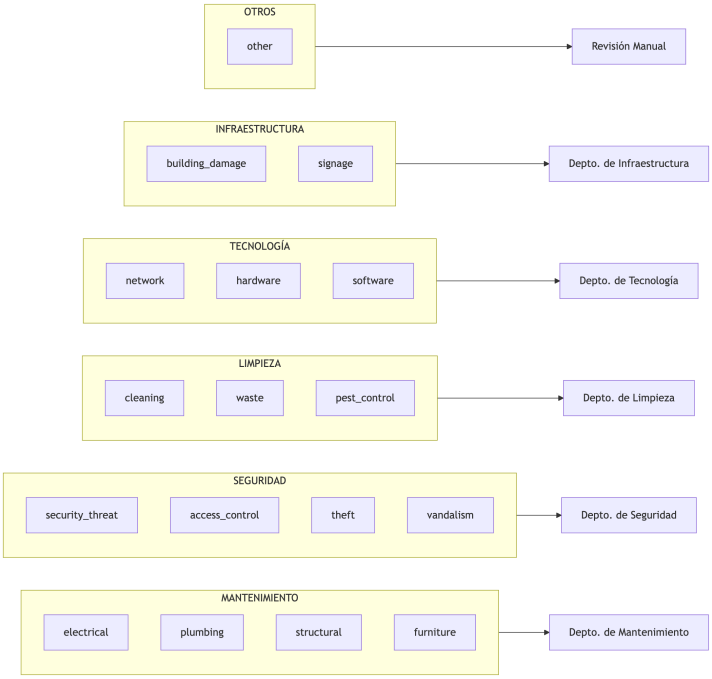
El enfoque cuantitativo se aplica estrictamente en la fase de validación del artefacto tecnológico para medir objetivamente su desempeño y conformidad técnica. Dado el alcance del proyecto como PoC, este componente se divide en dos dimensiones de análisis:

Figura 1

Taxonomía jerárquica para la clasificación de incidentes y asignación departamental.



(a) *Flujo de clasificación mediante LLM hacia categorías principales.*



(b) *Subcategorías de cada categoría principal y departamento responsable.*

Nota. Elaboración propia basada en entrevista a experto. La Figura 1a presenta el flujo de clasificación del LLM hacia las categorías principales, mientras que la Figura 1b detalla las subcategorías de cada área y su departamento responsable asignado.

Medición de precisión del modelo IA

Se evalúa el rendimiento del algoritmo de clasificación mediante la métrica de exactitud (*accuracy*). Se utiliza un corpus de prueba controlado para verificar si el modelo alcanza o supera el umbral de confianza establecido (≥ 0.75) en la asignación automática de departamentos.

Verificación de cumplimiento técnico

Se cuantifica el nivel de cumplimiento funcional del prototipo mediante una lista de verificación (*checklist*) basada en la inspección funcional y la ejecución controlada del sistema.

En lugar de validar exhaustivamente la totalidad de los requisitos teóricos (RF y RNF), se adoptó una estrategia de validación por agrupación funcional. Se seleccionaron exclusivamente los requisitos nucleares (*core requirements*) bajo los siguientes criterios de inclusión:

1. Ser indispensables para demostrar el flujo del sistema de extremo a extremo (*end-to-end*).
2. Ser validables técnicamente sin requerir pruebas masivas con usuarios finales.
3. Cuyo incumplimiento implicaría una falla crítica del prototipo.

Un requisito se consideró cumplido ("1") o no cumplido ("0") basándose en la observabilidad de su funcionamiento dentro del entorno de pruebas controlado.

Diseño de investigación

La estructura de la investigación se fundamenta en el marco metodológico de *design science research (DSR)*. Este paradigma es el idóneo ya que dirige la investigación hacia la creación y evaluación de un artefacto tecnológico innovador en este caso, diseñado para resolver un problema organizacional identificado (Hevner et al., 2004).

La adopción de este paradigma se alinea con las tendencias actuales en ingeniería de software

empírica, donde la Ciencia del Diseño permite evaluar y comunicar contribuciones de investigación orientadas a la solución de problemas prácticos en contextos reales (Engström, Storey et al., 2020). De esta manera, se garantiza el rigor en la construcción del conocimiento prescriptivo necesario para el desarrollo del prototipo.

A diferencia de los diseños experimentales tradicionales que buscan únicamente observar un fenómeno, el DSR iterativo permite construir, evaluar y refinar la solución tecnológica para garantizar su utilidad práctica. El proceso de investigación se estructuró en un ciclo de cuatro fases secuenciales e interconectadas:

Fase 1: identificación del problema y definición de objetivos

En esta fase inicial, se realizó un diagnóstico situacional del proceso de gestión de incidentes en la PUCE Manabí. Se identificaron las ineficiencias críticas derivadas del uso de canales informales (whatsApp, correo) y se definió la necesidad de automatizar la clasificación de reportes.

Las actividades clave de esta fase incluyeron:

1. Levantamiento de información: entrevistas con actores clave para comprender el flujo actual.
2. Definición de requisitos: formalización de las necesidades de los usuarios en una Especificación de Requisitos de Software (SRS) bajo el estándar del *Institute of Electrical and Electronics Engineers* (IEEE) 830-1998 (ver Apéndice B).
3. Estructuración de la taxonomía: definición de las categorías y reglas de negocio para el modelo de IA.

Fase 2: diseño y arquitectura de la solución

En esta etapa, se tradujeron los requisitos funcionales en especificaciones técnicas concretas. Se diseñó una arquitectura orientada a servicios que combina la robustez de una *API REST* para la lógica de negocio crítica con la agilidad de un *backend as a service* (BaaS) para la recuperación de datos en tiempo real.

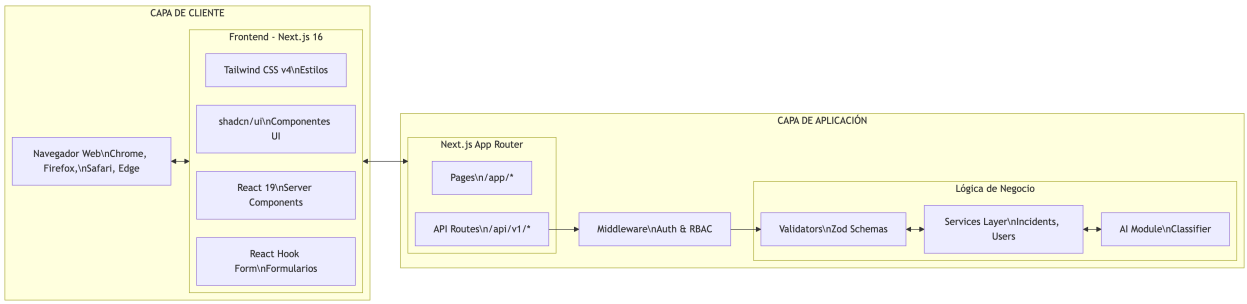
El diseño arquitectónico, visualizado en la Figura 2, se fundamenta en cuatro capas lógicas:

- *Capa de cliente*: interfaz reactiva construida sobre Next.js 16, optimizada para dispositivos móviles (*mobile-first*) y enriquecida con componentes de UI modernos.
- *Capa de aplicación*: núcleo lógico que gestiona las transacciones complejas y la integración con la IA a través de rutas de API protegidas, validando entradas mediante esquemas estrictos (Zod).
- *Capa de datos*: persistencia relacional en PostgreSQL gestionada a través de Supabase. Esta capa implementa un modelo de seguridad dual: validación en servidor y protección a nivel de fila (*row level security* - RLS) para consultas directas.
- *Servicios externos*: integración con el motor cognitivo Google Gemini para la clasificación inteligente.

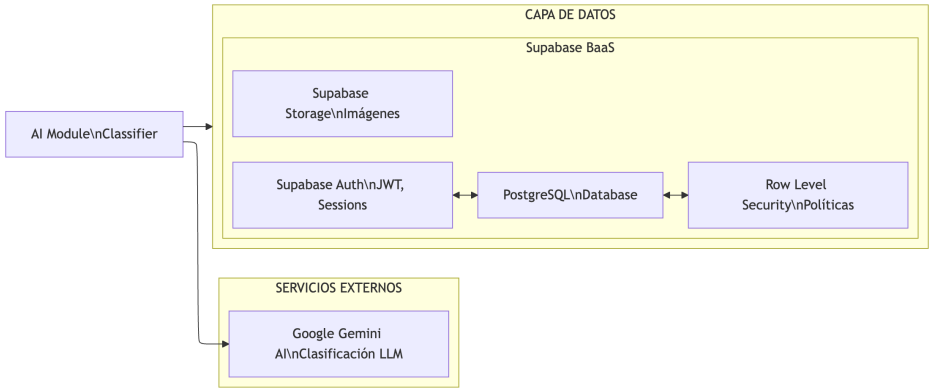
Dada la complejidad de los componentes y sus interacciones, el detalle ampliado del flujo y de la arquitectura se documenta en los Apéndices C y D.

Figura 2

Arquitectura del sistema web organizada por capas lógicas.



(a) *Capa de cliente y capa de aplicación.*



(b) *Capa de datos y servicios externos.*

Nota. Elaboración propia. La Figura 2a presenta las capas de cliente y aplicación del sistema, mientras que la Figura 2b detalla la capa de datos (Supabase BaaS) y los servicios externos integrados.

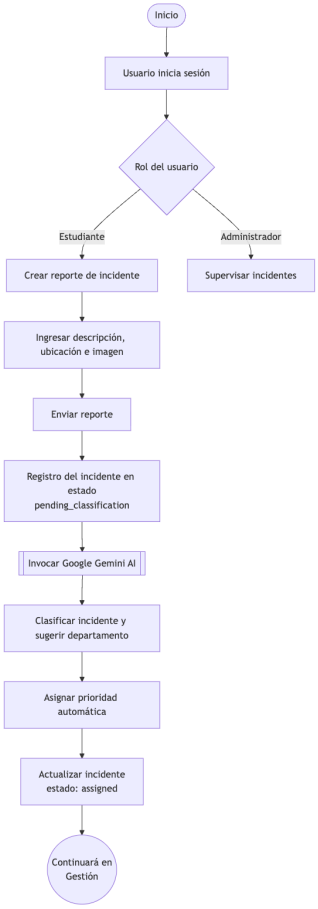
Diseño del flujo operativo

Adicionalmente a la arquitectura técnica, se modeló la lógica transaccional del sistema dividiéndola en dos etapas críticas: la captura inteligente y la gestión operativa.

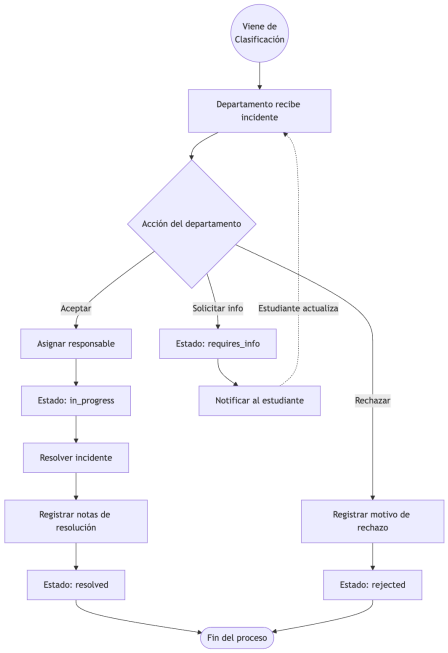
La Figura 3 ilustra este procedimiento. En la primera fase (3a), se detalla cómo el sistema utiliza la IA para transformar el reporte en datos estructurados. En la segunda fase (3b), se muestra el ciclo de vida de resolución por parte del personal humano.

Figura 3

Diagrama de flujo del proceso de gestión de incidentes (vista completa).



(a) *Fase de captura y clasificación.*



(b) *Fase de gestión y resolución.*

Nota. Elaboración propia.

El diseño detallado de cada nodo de decisión y las reglas de transición de estados se encuentran documentados en el Apéndice C.

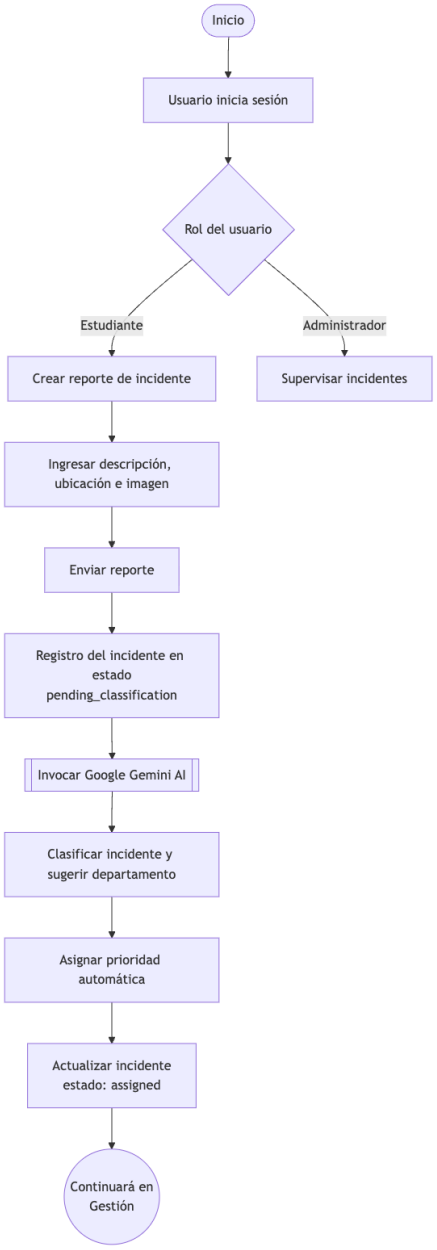
Ejemplo técnico representativo del flujo de IA

Con el fin de mantener en el cuerpo principal una evidencia técnica directa del funcionamiento interno, la Figura 4 presenta el tramo crítico de captura y clasificación automática asistida por IA. Este ejemplo sintetiza la secuencia mínima de procesamiento desde la recepción del reporte hasta

la decisión de asignación, mientras que la descripción exhaustiva de nodos, excepciones y variantes operativas se conserva en los Apéndices C y D.

Figura 4

Ejemplo técnico representativo del flujo de clasificación automática asistida por IA.



Nota. Elaboración propia. El resto de diagramas técnicos y su descripción integral se presentan en los apéndices correspondientes.

Fase 3: construcción y desarrollo (prototipado)

Esta fase comprendió la implementación técnica del artefacto. Se utilizó una metodología de desarrollo ágil para construir el prototipo funcional, utilizando un stack tecnológico híbrido:

- Frontend moderno: implementación de *Server Components* con React 19 y estilos atómicos mediante Tailwind CSS v4.
- Gestión de datos híbrida: se implementó una estrategia mixta de comunicación:
    - API Routes (/api/v1/*): para operaciones transaccionales complejas, escritura segura y comunicación con el motor de IA.
    - Cliente Supabase: para lectura de datos de alta frecuencia y suscripciones en tiempo real directamente desde el cliente, protegidas mediante políticas RLS.
- Inteligencia artificial: configuración del modelo `gemini-3-flash-preview` con parámetros de temperatura baja ($T = 0{,}1$) para garantizar respuestas determinísticas.

Fase 4: validación y evaluación técnica

La fase final del ciclo DSR tiene como objetivo demostrar la viabilidad técnica del artefacto desarrollado. Considerando el alcance académico y exploratorio del proyecto, la validación se realizó a través de una verificación técnica basada en inspección funcional y ejecución controlada.

Para este propósito, se adoptó una estrategia de validación por agrupación funcional. En lugar de validar exhaustivamente la totalidad de los requisitos, se seleccionaron exclusivamente los requisitos críticos (core requirements) indispensables para demostrar el funcionamiento del prototipo.

Los criterios de inclusión para la validación fueron:

- Requisitos indispensables para el flujo de extremo a extremo (*End-to-End*).

- Funcionalidades validables técnicamente sin intervención masiva de usuarios.
- Módulos cuyo fallo bloquearía la operación del sistema.

Bajo estos criterios, la evaluación se centró en verificar:

1. La seguridad del control de acceso (autenticación y RLS).
2. La integridad del ciclo de vida del incidente (creación → gestión).
3. La precisión de la clasificación automática de la IA frente a escenarios de prueba controlados.

Un requisito se consideró "cumplido" cuando su funcionalidad fue observable, verificable y reproducible dentro del entorno de pruebas controlado.

Población y muestra

Dada la naturaleza tecnológica y mixta de la investigación, la definición de la población y muestra se aborda desde dos perspectivas: una orientada a los actores informantes para el levantamiento de requisitos (cualitativa) y otra orientada a los datos de prueba para la validación del algoritmo de IA (cuantitativa).

Población y muestra de informantes (fase de análisis)

La población de estudio para la definición de requisitos estuvo constituida por los actores clave involucrados en la gestión operativa y tecnológica del campus.

- Muestra: se seleccionó a un (1) informante clave: un representante del área de gestión de infraestructura/servicios.

- Criterio de inclusión: personal con conocimiento directo de los flujos de trabajo actuales de reporte de incidentes y con capacidad de toma de decisiones sobre la taxonomía institucional.

Esta decisión metodológica se fundamenta en la técnica de muestreo propositivo (o muestreo por criterio), reconocida en la literatura de investigación cualitativa aplicada como una estrategia válida cuando el objetivo no es la representatividad estadística, sino el acceso al conocimiento experto sobre un proceso específico (Hernández Sampieri & Mendoza Torres, 2018). En el contexto de la ingeniería de software, el levantamiento de requisitos con un informante de alto conocimiento del dominio permite capturar la lógica de negocio con mayor fidelidad que una encuesta masiva a actores sin dominio técnico del proceso. El objetivo de esta fase fue definir la taxonomía de clasificación, tarea para la cual la profundidad del conocimiento del informante supera a la amplitud de la muestra.

Población y muestra de datos (fase de validación)

Para la validación técnica del modelo de inteligencia artificial y la verificación funcional del prototipo, la unidad de análisis no fueron personas, sino casos de prueba simulados que representan incidentes reales.

- Población de datos: el universo de posibles incidentes reportables en las categorías de mantenimiento, seguridad, limpieza y tecnología dentro del campus universitario.
- Muestra (corpus de prueba): se construyó un *dataset* controlado de 10 reportes de incidentes sintéticos. Esta muestra fue diseñada estratificadamente para cubrir la totalidad de la taxonomía definida (mantenimiento, seguridad, TI, limpieza), incluyendo casos con descripciones ambiguas para desafiar la precisión del modelo.

El tamaño del corpus se justifica en concordancia con el alcance de la investigación como PoC. En la literatura de validación de modelos de clasificación de texto, la estratificación de la muestra

es un criterio de calidad más determinante que el volumen absoluto de instancias cuando el objetivo es verificar el comportamiento del clasificador ante los distintos tipos de entrada definidos (Miliano et al., 2020; Nicholls et al., 2022). Este enfoque permite evaluar sistemáticamente la capacidad de generalización del modelo sobre los escenarios operativos previstos, dentro de los límites éticos y prácticos de un entorno de desarrollo controlado sin datos reales de usuarios.

Técnicas e instrumentos

Para la ejecución de las fases del ciclo DSR, se emplearon técnicas diferenciadas según el objetivo de cada etapa. A continuación, se detallan las técnicas y los instrumentos asociados:

Técnicas de análisis y diseño

Entrevista semi-estructurada

Esta técnica cualitativa se aplicó en la fase 1 para dialogar con los gestores del campus. Su propósito principal fue comprender la lógica de negocio y definir la estructura jerárquica de los incidentes.

- Instrumento: guía de entrevista, la cual se encuentra en el Apéndice A. Un guión de preguntas abiertas diseñado para identificar los tipos de problemas más frecuentes y los criterios de derivación, lo cual permitió construir la taxonomía de clasificación utilizada por el LLM.

Toma de requerimientos

Técnica utilizada para traducir las necesidades en especificaciones técnicas formales. A través de este proceso, se definieron las funcionalidades del sistema y las restricciones operativas.

- Instrumento: especificación de requisitos de software (SRS). Documento técnico estandarizado que sistematiza la totalidad de los requisitos funcionales (RF) y no funcionales (RNF). El documento completo se encuentra adjunto en el Apéndice B.

Técnicas de desarrollo (construcción del artefacto)

Para la fase 3, se emplearon técnicas de ingeniería de software modernas para la materialización del prototipo.

- Instrumentos de desarrollo (stack tecnológico):
    - IDE: Visual Studio Code como entorno de desarrollo integrado.
    - *Frameworks*: Next.js 16 (React) para la lógica y Tailwind CSS para el diseño de interfaz.
    - Infraestructura: Supabase como plataforma *backend as a service* (Base de datos y Auth).
    - IA: Google AI Studio y SDK de Gemini API para la configuración y ajuste (*tuning*) de los *prompts* del sistema.

Técnicas de validación

Ejecución controlada de casos de prueba

Técnica empleada en la fase 4 para medir la eficacia del modelo de IA. Consistió en ingresar los reportes del "corpus de prueba" en el sistema y registrar la respuesta automática generada.

- Instrumento: *matriz de pruebas de clasificación*. Una tabla estructurada que compara, para cada caso de prueba, la categoría esperada (definida por el humano) vs. la categoría obtenida (predicha por la IA), permitiendo calcular la precisión.

Inspección funcional

Técnica utilizada para verificar el cumplimiento de los requisitos críticos del software.

- Instrumento: *lista de verificación (checklist) de requisitos core*. Un formato binario (cumple/no cumple) diseñado para validar la operatividad de los módulos de autenticación, gestión de incidentes y el flujo de clasificacion de incidentes, descartando funcionalidades no críticas para el alcance del PoC.

Procedimientos

La ejecución de la investigación se llevó a cabo siguiendo una secuencia lógica de actividades alineadas con las fases del ciclo DSR. A continuación, se detallan los procedimientos técnicos y metodológicos aplicados, fundamentados en estándares de ingeniería de software y literatura reciente sobre inteligencia artificial generativa.

Fase 1: formalización de requisitos (estándar de ingeniería de requisitos)

Aunque el desarrollo siguió una filosofía ágil, se estableció una base documental sólida para garantizar la calidad del artefacto. Para ello, se adoptó el estándar internacional ISO/IEC/IEEE (ISO/IEC/IEEE, 2018), normativa vigente que sustituye al anterior y que proporciona el marco de referencia actual para la ingeniería de requisitos en el ciclo de vida del software.

1. Se documentaron los requisitos funcionales nucleares siguiendo la estructura de especificación recomendada por el estándar (Ver Apéndice B).
2. Se validaron las restricciones de seguridad y privacidad antes de iniciar la codificación.

Fase 2: estrategia de desarrollo ágil e incremental

Dada la naturaleza exploratoria del proyecto y la necesidad de integrar tecnologías emergentes, se adoptó una metodología de desarrollo ágil. Según Pressman y Maxim (2020), los modelos de proceso prescriptivos tradicionales pueden resultar ineficaces en entornos donde los requisitos cambian rápidamente; en contraste, el enfoque ágil prioriza la adaptabilidad y la entrega continua de incrementos funcionales de software. Bajo esta premisa, el ciclo de vida se estructuró en iteraciones cortas (*sprints*) de dos semanas.

Antes de iniciar los *sprints*, se definió la arquitectura tecnológica basándose en los principios de *Desarrollo Web Moderno*. Se seleccionó Next.js 16 debido a su capacidad de renderizado híbrido y su arquitectura de *Server Actions*, la cual permite ejecutar lógica de backend directamente desde los componentes de interfaz, reduciendo la latencia de red y simplificando la gestión de estados (*Next.js Documentation: App Router Architecture*, 2024).

El procedimiento de construcción técnica se estructuró en tres iteraciones funcionales:

Iteración 1: arquitectura base y seguridad (core)

El primer ciclo se centró en establecer una infraestructura segura y escalable, priorizando el control de acceso y la segregación de entornos.

1. Orquestación del entorno híbrido: se configuró el proyecto bajo la arquitectura *App Router* de Next.js. Se implementó el patrón de clientes de Supabase (`@supabase/ssr`) para gestionar sesiones de forma isomórfica (tanto en el servidor como en el navegador).
2. Middleware de protección: se desarrolló un mecanismo de intercepción (`middleware.ts`) que actúa como barrera de seguridad antes de que se renderice cualquier página. Este script valida la integridad del *token JWT* en las cookies; si el token es inválido o ha expirado, el middleware redirige al usuario al login antes de consumir recursos del servidor.
3. Restricción de dominio (*auth hook*): para garantizar que el sistema sea exclusivo para la

comunidad universitaria, se configuró una regla de validación en el proveedor de identidad. Esta regla rechaza cualquier intento de autenticación OAuth cuyo metadato `email` no contenga el sufijo `@pucesm.edu.ec`.

Iteración 2: lógica de negocio y persistencia

En esta etapa se construyó el núcleo transaccional. Para garantizar la calidad del software, se implementó un enfoque de "validación de esquema primero"(*schema-first validation*).

1. Modelado de datos y seguridad (RLS): se diseñó el esquema en PostgreSQL (Ver Figura 5). Para cumplir con los requisitos de privacidad, se implementó RLS. A diferencia de la seguridad por software, RLS aplica las reglas en el motor de base de datos.

   El Segmento de código 2.1 ilustra el procedimiento de implementación de la política de seguridad a nivel de fila:

```
CREATE POLICY "Acceso a incidentes propios"
ON public.incidents
FOR ALL
USING (auth.uid() = user_id);

```

Segmento de código 2.1: Política RLS para acceso a incidentes propios

2. *Server actions* y validación con Zod: se prescindió de una *API REST* tradicional para reducir la sobrecarga HTTP. Se utilizaron *server actions* protegidas. Para mitigar errores de entrada y ataques de inyección, se integró la librería Zod, definiendo esquemas estrictos para cada operación (McDonnell, 2024).

   El Segmento de código 2.2 presenta el esquema de validación implementado:

```
const CreateIncidentSchema = z.object({
  title: z.string().min(5).max(100),
  description: z.string().min(10),
```

```
      category: z.enum(CategoryList),
      image_url: z.string().url()
    });

```

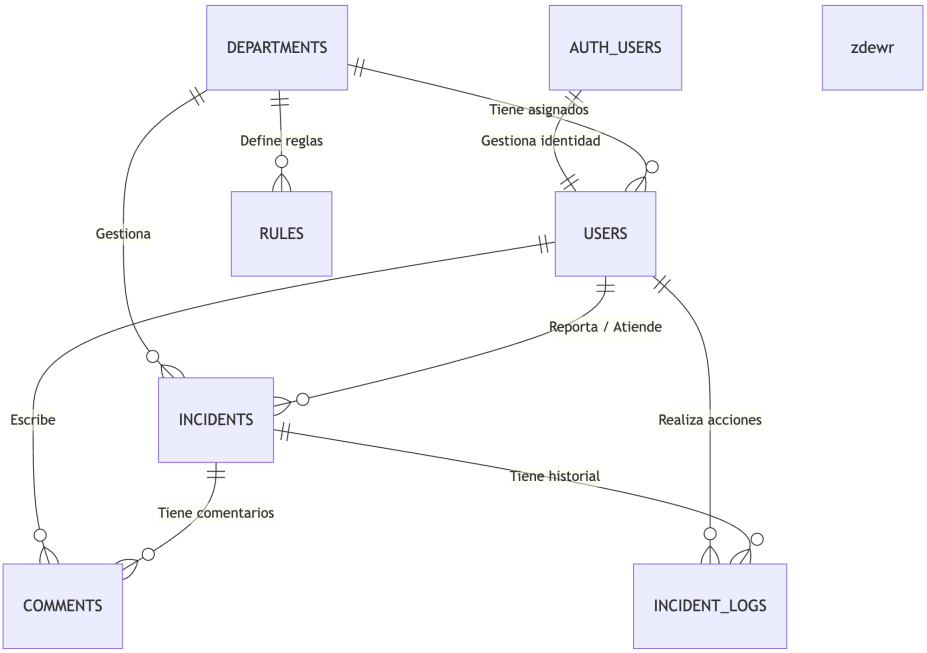
Segmento de código 2.2: Esquema de validación Zod para la creación de incidentes

Este mecanismo asegura que ningún dato malformado llegue a la capa de persistencia.

3. Gestión de evidencias: se configuró el almacenamiento de imágenes mediante *buckets* privados en Supabase Storage, implementando la generación de identificadores únicos (UUID v4) para evitar colisiones de nombres en los archivos adjuntos.

Figura 5

Diagrama Entidad-Relación (Nivel Conceptual) del sistema. Se visualiza la arquitectura de información y las interacciones lógicas entre los módulos principales.



Nota. Elaboración propia.

Iteración 3: integración del motor cognitivo

La fase final abordó la integración del modelo Gemini 3 Flash. El desafío técnico principal fue transformar los datos no estructurados (imagen + texto) en una estructura JSON determinística.

1. Ingeniería del *payload* multimodal: se implementó una utilidad de conversión para transformar las imágenes a formato Base64. El sistema construye un *payload* compuesto por el *system prompt* (reglas de negocio), el texto del reporte y la evidencia visual codificada.

2. Configuración de salida estructurada: para garantizar que el sistema pueda procesar la respuesta de la IA sin errores de análisis (*parsing*), se configuró el modelo en modo JSON, como se muestra en el Segmento de código 2.3.

```
    const model = genAI.getGenerativeModel({
      model: "gemini-3-flash-preview",
      generationConfig: {
        responseMimeType: "application/json",
        temperature: 0.1
      }
    });

```

Segmento de código 2.3: Configuración del modelo Gemini con salida JSON estructurada

3. Algoritmo de decisión (*thresholding*): se programó la lógica post-inferencia que evalúa el nivel de confianza devuelto por la IA.

   El nivel de confianza $C \in [0{,}0,\ 1{,}0]$ es generado directamente por el modelo Gemini como parte de su respuesta JSON estructurada; no es un valor calculado externamente, sino que el modelo lo produce en función de la claridad y especificidad de la descripción del reporte, siguiendo instrucciones explícitas del *system prompt*. Para garantizar que el valor se

encuentre en el rango válido, el sistema aplica una normalización:
$C = \text{máx}(0,\ \text{mín}(1,\ C_{\text{raw}}))$.

El umbral de $C \geq 0{,}75$ fue determinado empíricamente durante la fase de ingeniería de *prompts*. Con la temperatura configurada en $T = 0{,}1$, parámetro deliberadamente bajo para maximizar el determinismo de las respuestas, se observó que clasificaciones con $C < 0{,}75$ presentaban asignación al departamento `unknown` o inestabilidad entre categorías, lo que estableció este valor como punto de corte operativo.

La lógica de asignación automática requiere que se cumplan dos condiciones simultáneamente:

$$\text{shouldAutoAssign} = (C \geq 0{,}75) \ \wedge \ (\text{suggestedDepartment} \neq \texttt{unknown})$$

Si ambas condiciones se satisfacen, el sistema ejecuta la asignación automática del departamento correspondiente; caso contrario, marca el incidente con la categoría Otros y lo deriva a la cola de revisión manual.

Fase 3: selección del modelo de inteligencia artificial

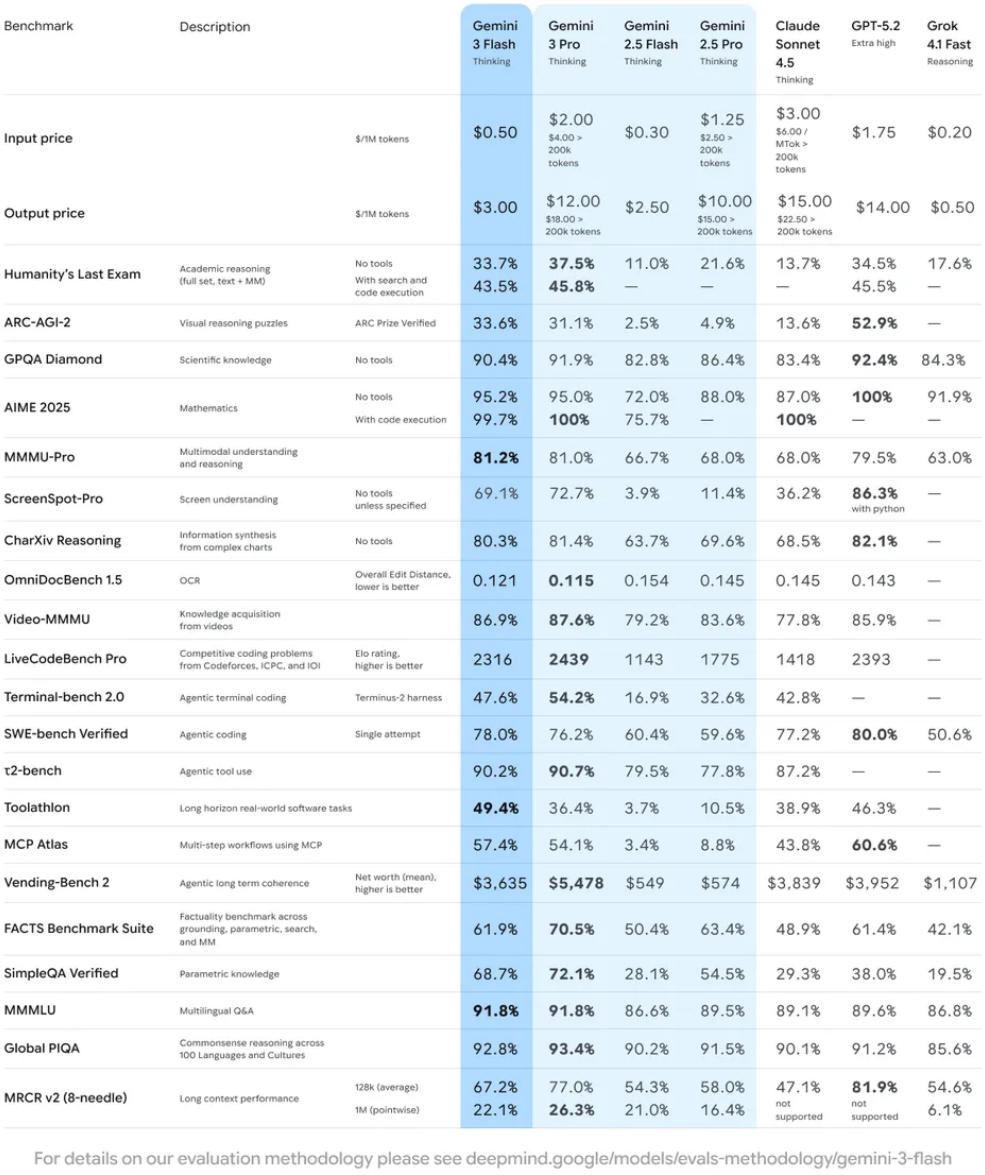
La elección del motor de inferencia fue una decisión crítica basada en criterios de eficiencia operativa (latencia) y sostenibilidad económica (costo por token). Se realizó un análisis comparativo entre las opciones disponibles en el mercado, seleccionando la familia de modelos Gemini 3 Flash.

Tal como indica Doshi (2025), la arquitectura “Flash“ está diseñada específicamente para ofrecer inteligencia de vanguardia optimizada para la velocidad y el volumen. Este modelo permite procesar grandes cantidades de reportes con una latencia mínima y a una fracción del costo de los modelos “Pro“ o “Ultra“, lo cual es indispensable para la viabilidad de un sistema institucional que debe escalar sin generar costos prohibitivos.

Como se muestra en la Figura 6, la serie Flash presenta una ventaja significativa en términos de latencia y costo frente a otros modelos LLM. El desglose de los criterios de evaluación y la matriz de decisión que sustentan esta selección se documentan en el Apéndice G.

Figura 6

Comparativa de rendimiento y costo entre modelos LLM.



Benchmark	Description		Gemini 3 Flash Thinking	Gemini 3 Pro Thinking	Gemini 2.5 Flash Thinking	Gemini 2.5 Pro Thinking	Claude Sonnet 4.5 Thinking	GPT-5.2 Extra high	Grok 4.1 Fast Reasoning
Input price		$/1M tokens	$0.50	$2.00 $4.00 > 200k tokens	$0.30	$1.25 $2.50 > 200k tokens	$3.00 $6.00 / MTok > 200k tokens	$1.75	$0.20
Output price		$/1M tokens	$3.00	$12.00 $18.00 > 200k tokens	$2.50	$10.00 $15.00 > 200k tokens	$15.00 $22.50 > 200k tokens	$14.00	$0.50
Humanity's Last Exam	Academic reasoning (full set, text + MM)	No tools	33.7%	**37.5%**	11.0%	21.6%	13.7%	34.5%	17.6%
		With search and code execution	43.5%	**45.8%**	—	—	—	45.5%	—
ARC-AGI-2	Visual reasoning puzzles	ARC Prize Verified	33.6%	31.1%	2.5%	4.9%	13.6%	**52.9%**	—
GPQA Diamond	Scientific knowledge	No tools	90.4%	91.9%	82.8%	86.4%	83.4%	**92.4%**	84.3%
AIME 2025	Mathematics	No tools	95.2%	95.0%	72.0%	88.0%	87.0%	**100%**	91.9%
		With code execution	99.7%	**100%**	75.7%	—	**100%**	—	—
MMMU-Pro	Multimodal understanding and reasoning		**81.2%**	81.0%	66.7%	68.0%	68.0%	79.5%	63.0%
ScreenSpot-Pro	Screen understanding	No tools unless specified	69.1%	72.7%	3.9%	11.4%	36.2%	**86.3%** with python	—
CharXiv Reasoning	Information synthesis from complex charts	No tools	80.3%	81.4%	63.7%	69.6%	68.5%	**82.1%**	—
OmniDocBench 1.5	OCR	Overall Edit Distance, lower is better	0.121	**0.115**	0.154	0.145	0.145	0.143	—
Video-MMMU	Knowledge acquisition from videos		86.9%	**87.6%**	79.2%	83.6%	77.8%	85.9%	—
LiveCodeBench Pro	Competitive coding problems from Codeforces, ICPC, and IOI	Elo rating, higher is better	2316	**2439**	1143	1775	1418	2393	—
Terminal-bench 2.0	Agentic terminal coding	Terminus-2 harness	47.6%	**54.2%**	16.9%	32.6%	42.8%	—	—
SWE-bench Verified	Agentic coding	Single attempt	78.0%	76.2%	60.4%	59.6%	77.2%	**80.0%**	50.6%
τ2-bench	Agentic tool use		90.2%	**90.7%**	79.5%	77.8%	87.2%	—	—
Toolathlon	Long horizon real-world software tasks		**49.4%**	36.4%	3.7%	10.5%	38.9%	46.3%	—
MCP Atlas	Multi-step workflows using MCP		57.4%	54.1%	3.4%	8.8%	43.8%	**60.6%**	—
Vending-Bench 2	Agentic long term coherence	Net worth (mean), higher is better	$3,635	**$5,478**	$549	$574	$3,839	$3,952	$1,107
FACTS Benchmark Suite	Factuality benchmark across grounding, parametric, search, and MM		61.9%	**70.5%**	50.4%	63.4%	48.9%	61.4%	42.1%
SimpleQA Verified	Parametric knowledge		68.7%	**72.1%**	28.1%	54.5%	29.3%	38.0%	19.5%
MMMLU	Multilingual Q&A		**91.8%**	**91.8%**	86.6%	89.5%	89.1%	89.6%	86.8%
Global PIQA	Commonsense reasoning across 100 Languages and Cultures		92.8%	**93.4%**	90.2%	91.5%	90.1%	91.2%	85.6%
MRCR v2 (8-needle)	Long context performance	128k (average)	67.2%	77.0%	54.3%	58.0%	47.1% not supported	**81.9%** not supported	54.6%
		1M (pointwise)	22.1%	**26.3%**	21.0%	16.4%			6.1%

For details on our evaluation methodology please see deepmind.google/models/evals-methodology/gemini-3-flash

Nota. Tomado de Doshi, 2025. Ver Apéndice G para el análisis detallado.

Fase 4: ingeniería de prompts e iteración

La integración de la IA no se limitó a una consulta directa, sino que requirió un proceso de ingeniería de *prompts* avanzado para mitigar alucinaciones y sesgos.

Prompting iterativo para la veracidad

Para garantizar que la clasificación de la IA fuera correcta y minimizar las alucinaciones, se aplicó una estrategia de *Iterative Refinement*. Estudios validados como el de Madaan et al. (2023) demuestran que los modelos de lenguaje mejoran significativamente su rendimiento y veracidad cuando no generan una respuesta final de inmediato, sino que pasan por un ciclo de retroalimentación y ajuste de instrucciones. Siguiendo este principio, se optimizó el *system prompt* en múltiples ciclos de prueba hasta estabilizar la salida JSON y asegurar la consistencia de la taxonomía.

Few-Shot prompting (aprendizaje con ejemplos)

Para mejorar la precisión en casos ambiguos, se implementó la técnica de *Few-Shot Prompting*, que consiste en incluir ejemplos de "entrada-salida" dentro del contexto del prompt. Según Ma et al. (2023), proporcionar ejemplos guiados reduce la inestabilidad del aprendizaje en contexto (In-Context Learning) y ayuda al modelo a encontrar la clasificación óptima basándose en patrones preestablecidos en lugar de suposiciones abiertas.

Fase 5: validación técnica

Finalmente, se ejecutó el procedimiento de validación mediante la matriz de pruebas detallada en la sección de Instrumentos, verificando que la latencia y precisión del modelo seleccionado cumplieran con los parámetros operativos definidos.

Consideraciones éticas

El desarrollo de la presente investigación y la construcción del artefacto tecnológico se rigieron por principios éticos fundamentales, alineados con el Código de Ética y Conducta Profesional de la Association for Computing Machinery (ACM) y el IEEE (Association for Computing Machinery (ACM), 2024), así como con las normativas de investigación de la PUCE Manabí.

A continuación, se detallan las medidas adoptadas para garantizar la integridad, privacidad y responsabilidad en el manejo de la información:

Privacidad y protección de datos personales

Dado que el sistema está diseñado para gestionar información sensible (correos institucionales, geolocalización e imágenes), se implementaron mecanismos de privacidad por diseño:

- Segregación de datos: se implementaron políticas de seguridad a nivel de fila (RLS) en la base de datos para garantizar que ningún estudiante pueda acceder a reportes o datos personales de otros usuarios, cumpliendo con el principio de confidencialidad.
- Minimización de datos: el sistema solicita estrictamente la información necesaria para la gestión del incidente, evitando la recolección excesiva de datos personales no relevantes para el objetivo del estudio.
- Tratamiento de la imagen: las fotografías subidas por los usuarios se procesan exclusivamente con fines de clasificación técnica y mantenimiento, asegurando que no se utilicen para vigilancia o identificación biométrica no autorizada.

Transparencia y responsabilidad algorítmica

El uso de Inteligencia Artificial Generativa (Google Gemini) implica desafíos éticos relacionados con el sesgo y la toma de decisiones automatizada. Para mitigar estos riesgos:

- *Human-in-the-loop* (supervisión humana): el sistema fue diseñado bajo el principio de que la IA actúa como una herramienta de asistencia y no como un juez final. Los incidentes con baja confianza ($< 75\,\%$) son derivados obligatoriamente a revisión humana, y los administradores conservan la potestad de anular o reclasificar cualquier decisión algorítmica.
- Transparencia del modelo: se documentó explícitamente en la interfaz que la clasificación es realizada por un asistente virtual, informando al usuario sobre la naturaleza automatizada del proceso para mantener la confianza en el sistema.

Manejo de datos en la fase de validación

Para la fase de pruebas y validación del prototipo descrita en este capítulo, se respetó rigurosamente el anonimato de la comunidad universitaria:

- Uso de datos sintéticos: no se utilizaron datos reales de estudiantes ni reportes históricos privados para el entrenamiento o prueba del modelo. El corpus de prueba fue construido íntegramente con datos ficticios (sintéticos) generados por el investigador para simular escenarios de uso.
- Entorno controlado: las pruebas se realizaron en un entorno de desarrollo aislado, garantizando que ninguna notificación o correo de prueba impactara a usuarios reales o departamentos operativos de la universidad durante la investigación.

Resultados

El presente capítulo detalla los hallazgos obtenidos tras la ejecución de la metodología propuesta y la implementación del prototipo. Los resultados se presentan organizados en correspondencia con los objetivos específicos de la investigación, abarcando desde el análisis inicial de requisitos hasta la validación cuantitativa del modelo de Inteligencia Artificial y la conformidad funcional del software desarrollado.

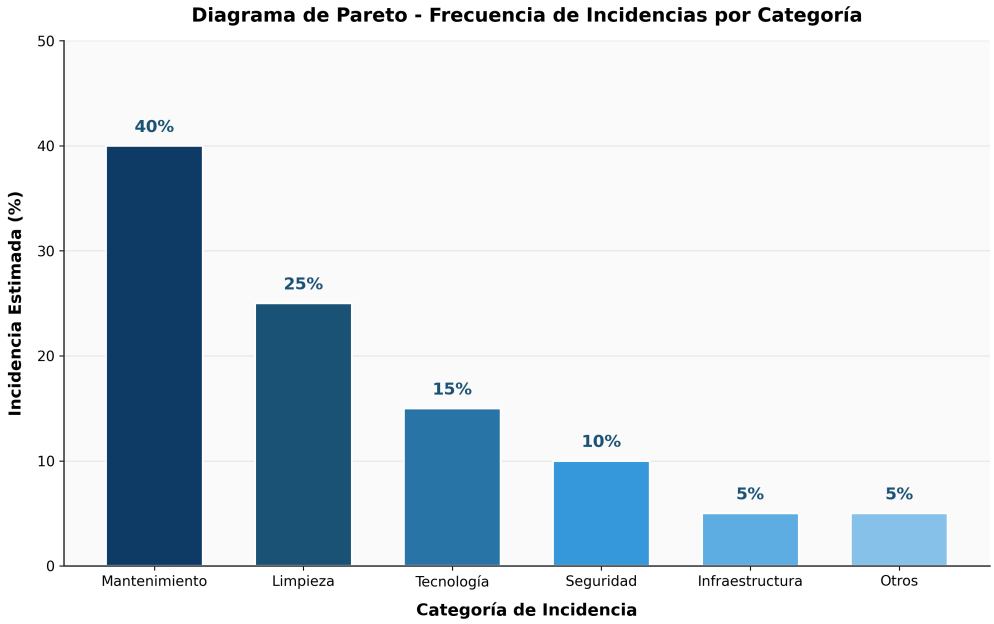
Resultados del análisis de requisitos y definición de taxonomía

En cumplimiento del primer objetivo específico, se procesó la información recabada en la entrevista con el personal gestor (cuyo instrumento completo se detalla en el Apéndice A). Para definir la taxonomía de clasificación, se aplicó un análisis de frecuencia sobre las tipologías de incidentes históricos reportados verbalmente por el experto.

El análisis de los datos cualitativos permitió identificar la distribución de la demanda de soporte. Como se observa en la Figura 7, el experto informante estimó, con base en su experiencia operativa, que el 80 % de la carga de atención se concentra en tres áreas críticas: mantenimiento (40 %), limpieza (25 %) y tecnología (15 %). Estas cifras constituyen estimaciones cualitativas derivadas de la entrevista (Apéndice A) y no registros contables formales, dado que la institución no disponía de un sistema de registro histórico estructurado previo a esta investigación. El procedimiento de obtención de estos datos y su análisis de frecuencia acumulada se documentan en el Apéndice H.

Figura 7

Distribución estimada de la tipología de incidentes históricos (Análisis de Pareto).



Nota. Elaboración propia basada en datos de la entrevista (Apéndice A). Ver Apéndice H para la fundamentación detallada.

Estos hallazgos estadísticos fundamentaron la estructura de la taxonomía. Se descartó el uso de una lista plana en favor de una estructura jerárquica que prioriza las categorías de alta frecuencia para optimizar el entrenamiento del modelo de IA. La Tabla 1 presenta la taxonomía resultante derivada de este análisis de datos.

Tabla 1

Taxonomía de clasificación definida a partir de la frecuencia de incidentes

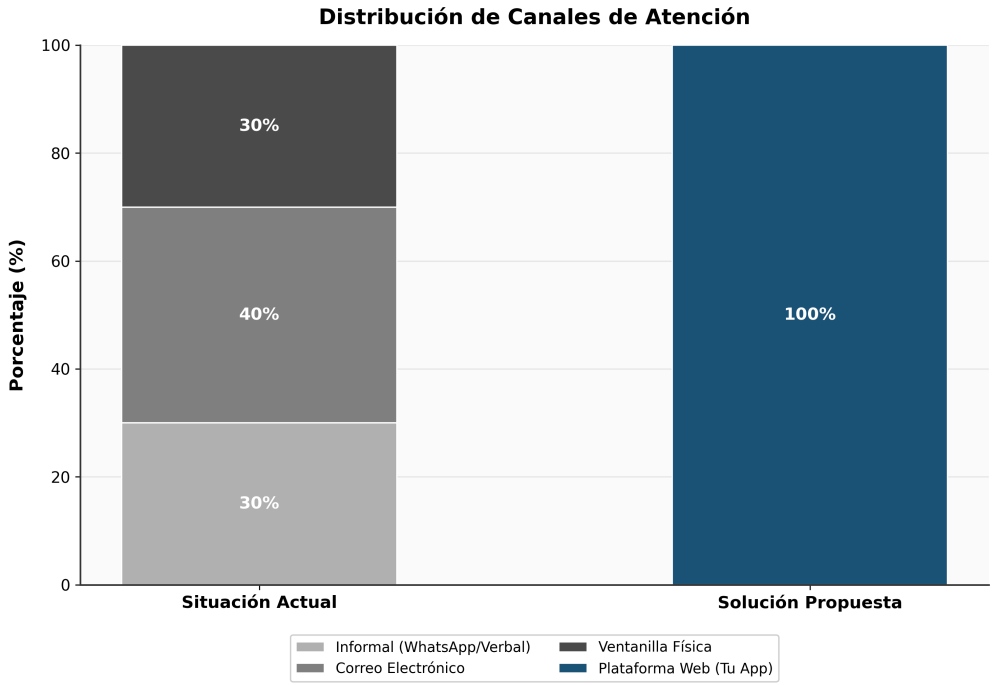
Macro-Categoría	Alcance Definido	Incidencia Est.
Mantenimiento	Reparaciones eléctricas, hidrosanitarias y mobiliario.	Alta (40 %)
Limpieza	Higiene de espacios comunes y gestión de residuos.	Media (25 %)
Tecnología	Conectividad, redes y equipos institucionales.	Media (15 %)
Seguridad	Control de accesos y vigilancia.	Baja (10 %)
Infraestructura	Daños estructurales a la edificación.	Baja (5 %)
Otros	Casos atípicos o ambiguos (Cola de revisión).	Baja (5 %)

Adicionalmente, se caracterizó el estado de los canales de reporte. El experto informante estimó, con base en su experiencia dentro del departamento, que aproximadamente el 30 % de las solicitudes llegaban por canales informales como la atención presencial directa o el uso de WhatsApp, en lugar de canales formales estandarizados, lo que generaba una pérdida de trazabilidad y datos no estructurados.

La implementación de la taxonomía unificada y el canal digital centralizado permitió corregir esta dispersión. Como se ilustra en la Figura 8, en el entorno de pruebas controlado, el 100 % de los reportes procesados a través del prototipo digital siguieron el esquema estructurado, demostrando que el sistema unifica en un único canal digital el punto de entrada para todas las categorías de incidentes, eliminando la dispersión entre departamentos y medios informales observada en la línea base. Esta medición corresponde exclusivamente al comportamiento del sistema en el entorno de validación técnica.

Figura 8

Comparativa de dispersión de canales: Situación inicial vs. Solución implementada.



Nota. Elaboración propia basada en diagnóstico operativo.

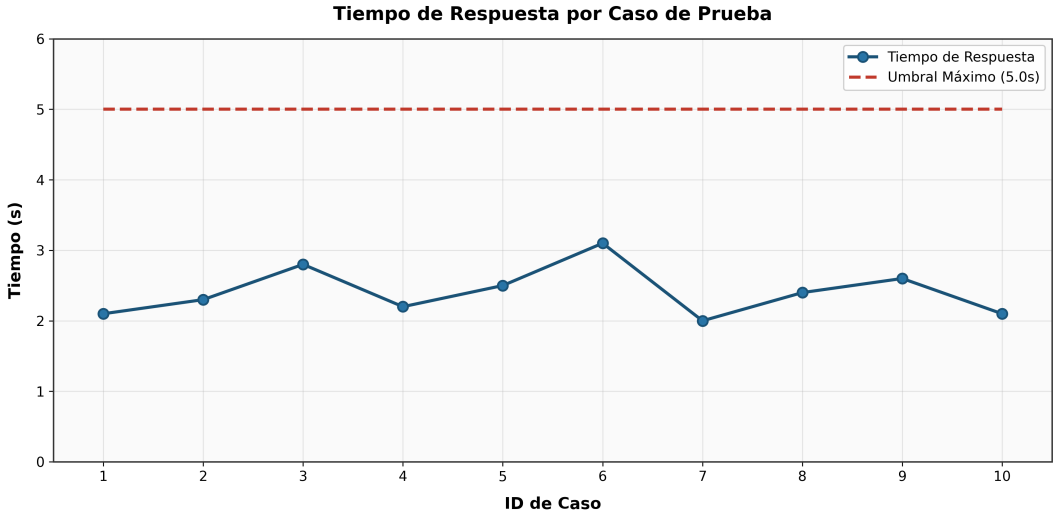
Evaluación del desempeño técnico de la arquitectura

La evaluación del desempeño técnico constituye el indicador de validación del diseño arquitectónico propuesto en el segundo objetivo: demostrar que la integración entre la aplicación web, la API de IA y la base de datos bajo una arquitectura *serverless* es viable y cumple con parámetros de rendimiento aceptables para un sistema de gestión institucional. Para ello, se evaluó la latencia del sistema en condiciones de operación normal. Se registraron los tiempos de respuesta del flujo completo (end-to-end), abarcando la recepción de la imagen, la inferencia en la API de Google Gemini y la persistencia en la base de datos.

La Figura 9 presenta los resultados de las mediciones. El tiempo promedio de procesamiento se situó en 2.4 segundos ($\sigma = 0{,}4s$).

Figura 9

Latencia de procesamiento por transacción (Inferencia IA + Persistencia DB).



Nota. Logs de ejecución del sistema.

Los valores obtenidos se mantuvieron por debajo del umbral de referencia de 5 segundos, valor consistente con los parámetros de tolerancia de espera en sistemas de información *web* establecidos por Nah (2004), quien identificó 5 segundos como el límite superior de tolerancia para respuestas en aplicaciones *web* interactivas. Este resultado valida técnicamente la

integración de los servicios *serverless* dentro de un presupuesto de rendimiento aceptable.

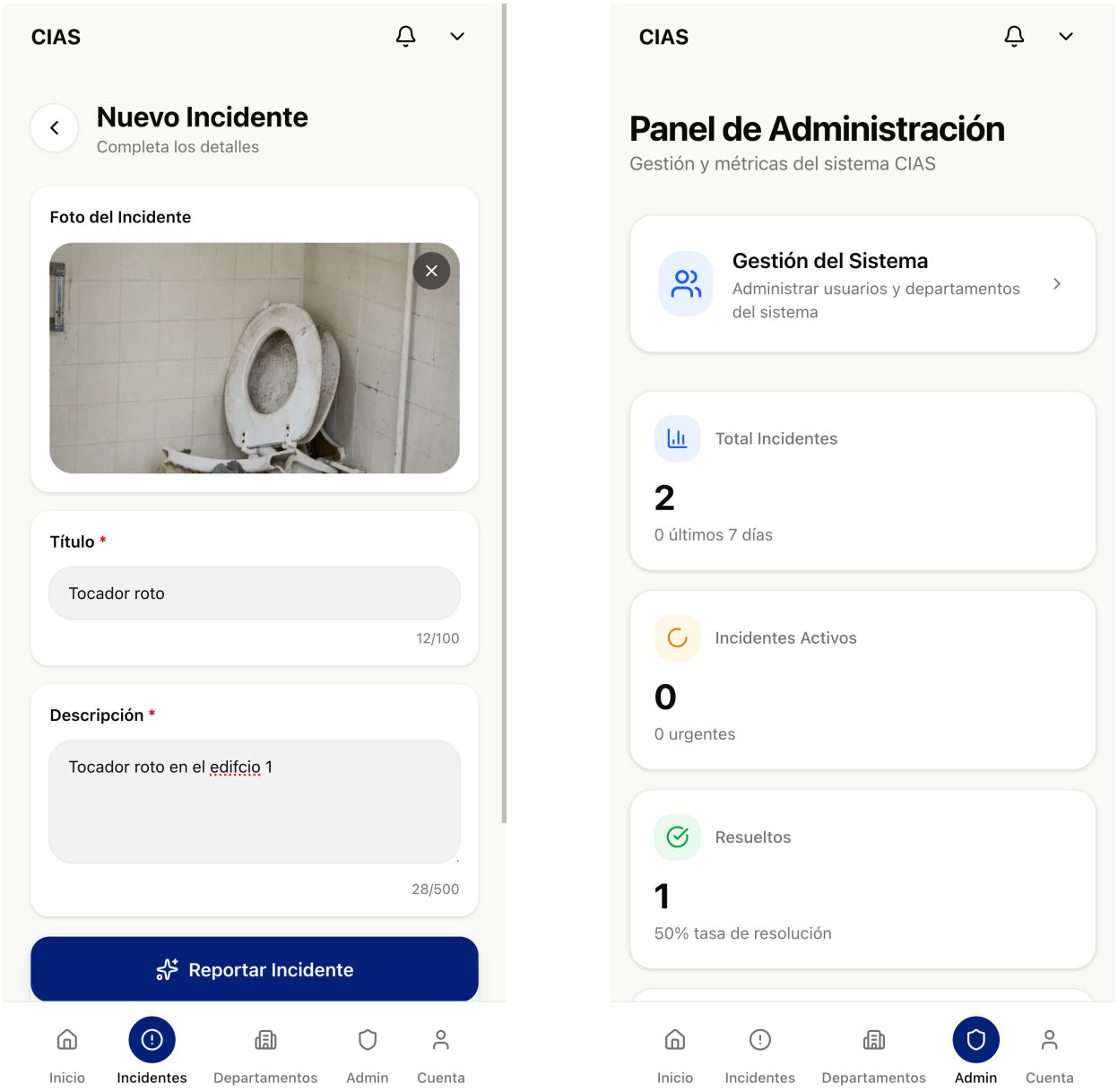
Resultados del desarrollo del prototipo funcional

En cumplimiento del tercer objetivo, se finalizó la construcción del aplicativo web. El desarrollo integró los módulos de cliente y administración según las especificaciones de diseño.

La Figura 10 evidencia la interfaz gráfica implementada, la cual habilitó la captura multimodal (texto, imagen y geolocalización) y la gestión de estados del incidente.

Figura 10

Interfaces finales del prototipo desplegado en dispositivo móvil.



(a) *Módulo de captura (Estudiante)*

(b) *Módulo de gestión (Admin)*

Nota. Elaboración propia.

El código fuente resultante, incluyendo la configuración del cliente y la lógica de negocio, se encuentra versionado en el repositorio del proyecto:

https://github.com/LuisRoft/pucem-reports

Validación del cumplimiento de requisitos y modelo IA

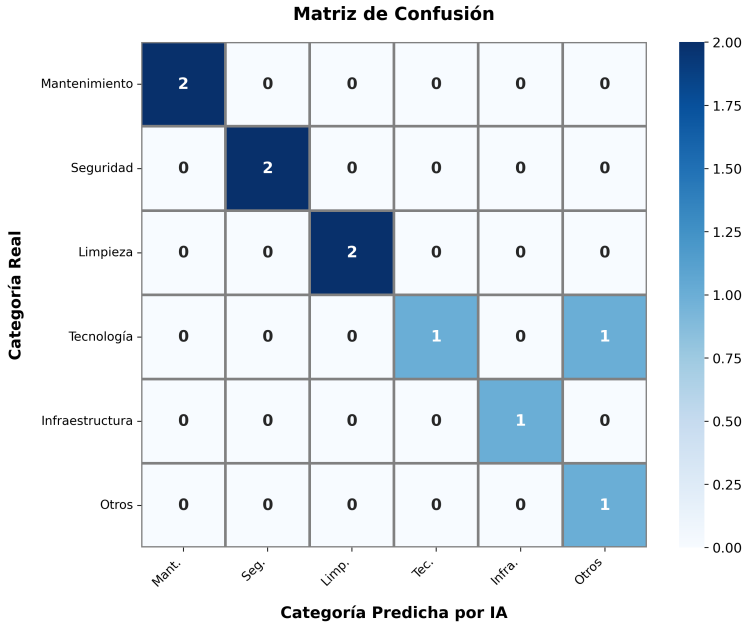
Para dar respuesta al cuarto y quinto objetivo específico, se ejecutaron las pruebas de validación en dos frentes: la precisión del modelo cognitivo y la conformidad funcional del software.

Matriz de confusión y precisión del modelo

Se evaluó el desempeño del clasificador sobre un corpus controlado de 10 casos de prueba. La Matriz de Confusión (Figura 11) resume las coincidencias entre la categoría esperada y la inferida. El modelo alcanzó una exactitud global (*accuracy*) del 90 %.

Se registró una desviación en el caso de prueba #4 (detallado en el Apéndice E), correspondiente a la categoría *tecnología*. En dicha instancia, el sistema asignó la etiqueta Otros.

Figura 11
Matriz de confusión obtenida del corpus de validación.



Nota. Elaboración propia.

El análisis de los logs del sistema evidenció que, en el caso no coincidente, el nivel de confianza reportado por la API fue de 0.62, inferior al promedio del resto de casos ($> 0{,}85$). Este comportamiento activó la regla de negocio de asignación por defecto a la cola de revisión manual.

Caso de prueba documentado de forma integral

Con el propósito de fortalecer la trazabilidad del origen de los datos estadísticos, se documenta de manera completa el caso de prueba CP-04, correspondiente a la única desviación controlada del corpus. Los demás casos del corpus se mantienen consolidados en el Apéndice E.

Tabla 2
Documentación completa del caso de prueba CP-04

Componente	**Descripción documentada**
Escenario de prueba	Incidente académico con impacto tecnológico: "El proyector del aula 201 no enciende y tenemos presentación mañana". Categoría esperada por criterio humano: *Tecnología*.
Procedimiento de ejecución	(1) Ingreso autenticado como estudiante en entorno de *staging*. (2) Registro del incidente con descripción textual y evidencia fotográfica. (3) Envío del formulario y monitoreo simultáneo de panel *Network*, consola del navegador y panel de Supabase.
Resultado observado	La API devolvió `category = Otros`, `confidence = 0.62` y `suggestedDepartment = unknown`. En aplicación de la regla de umbral ($C < 0{,}75$), el sistema no realizó autoasignación y derivó el incidente a revisión manual.
Evidencia de verificación	(a) Registro en matriz del corpus: Caso #4 del Apéndice E. (b) Traza de respuesta JSON observada en consola (`category`, `confidence`, `suggestedDepartment`). (c) Persistencia del estado en base de datos verificada en Supabase conforme al protocolo de inspección funcional.

Nota. Elaboración propia a partir de la ejecución controlada del corpus de validación.

Estadística de cumplimiento de requisitos funcionales

La validación funcional se llevó a cabo mediante inspección directa, ejecutando acciones de usuario sobre el sistema desplegado en el entorno de *staging* en Vercel, con base de datos activa en Supabase. Las pruebas se realizaron en el navegador Arc (basado en Chromium) en equipo de escritorio, y se verificó la respuesta adaptativa (*responsive*) en un dispositivo móvil iPhone 11. Para determinar el cumplimiento de cada requisito se emplearon tres instrumentos de observación simultánea:

1. Panel de red (*Network*) de las DevTools: para confirmar los códigos de respuesta HTTP, el *payload* de las peticiones y la latencia de cada operación.
2. Consola del navegador: para inspeccionar los logs de respuesta de la API de IA, verificando específicamente los campos `category`, `confidence` y `suggestedDepartment` del JSON devuelto por Gemini, así como la ausencia de errores de ejecución en cliente.
3. Panel de gestión de Supabase: para confirmar la persistencia correcta en la base de datos, incluyendo las coordenadas GPS registradas en la tabla de incidentes, las URL de imágenes almacenadas en Storage y la aplicación efectiva de las políticas de seguridad a nivel de fila (RLS).

Un requisito se consideró cumplido (✓) cuando el comportamiento observado coincidía exactamente con el comportamiento esperado definido en la especificación de requisitos (Apéndice B). En lugar de una verificación unitaria aislada, se midió el índice de cumplimiento agrupado por módulos críticos del sistema. El protocolo completo, con la acción ejecutada y la observación registrada para cada caso, se detalla en el Apéndice F.

La Tabla 3 presenta el resumen estadístico de la ejecución de pruebas. Se validó el 100 % de los requisitos funcionales prioritarios, sin que se registraran defectos bloqueantes en el despliegue final.

Tabla 3

Resumen estadístico de conformidad de requisitos por módulo

Módulo del Sistema	Req. Totales	Cumplidos	% Conformidad	Defectos
Autenticación y Seguridad	4	4	100 %	0
Captura y Geolocalización	6	6	100 %	0
Procesamiento IA	5	5	100 %	0
Gestión Administrativa	8	8	100 %	0
TOTAL GENERAL	**23**	**23**	**100 %**	**0**

Nota. Elaboración propia basada en la ejecución del *checklist* (Apéndice F).

El detalle granular de cada prueba ejecutada, incluyendo las condiciones de entrada y los resultados observados, se encuentra documentado en el Apéndice F.

Síntesis de hallazgos

Los resultados obtenidos demostraron la viabilidad técnica y operativa de la solución propuesta. Se evidenció una reducción en la dispersión de la información mediante la centralización del canal de reporte, tiempos de procesamiento de IA inferiores a 3 segundos y una precisión del 90 % en la clasificación automática, con un mecanismo de control para los casos de baja confianza. Asimismo, las pruebas funcionales certificaron la conformidad total del prototipo frente a los requisitos establecidos.

Discusión

El presente capítulo ofrece una interpretación crítica de los hallazgos expuestos anteriormente, analizando su significado y contrastándolos con la literatura científica existente. A través de este análisis, se busca determinar el impacto real del prototipo desarrollado en la optimización de la gestión de incidentes, así como establecer los límites de su alcance técnico y operativo dentro del contexto institucional.

Interpretación de los resultados

El análisis de los datos obtenidos confirma la viabilidad técnica y operativa de integrar Inteligencia Artificial generativa en los flujos de trabajo administrativos, validando la hipótesis implícita de que la automatización puede resolver la fragmentación en la gestión de servicios universitarios.

El diagnóstico inicial reveló una deficiencia estructural crítica: la gestión de un 30 % de los incidentes a través de canales informales. Esta dispersión no solo generaba ineficiencia operativa, sino que impedía la toma de decisiones basada en datos. La implementación de la taxonomía jerárquica, fundamentada en el análisis de Pareto que identificó a mantenimiento y limpieza como las áreas de mayor carga, demostró ser el factor determinante para eliminar la ambigüedad. Al lograr la estandarización total de las entradas en el entorno de pruebas, se evidencia que la estructuración del dato desde su captura es la clave para la trazabilidad, superando las limitaciones del modelo manual anterior.

Sobre esta base de datos estructurada, la arquitectura técnica desplegada demostró su eficiencia operativa. Los tiempos de latencia promedio de 2.4 segundos sugieren que el enfoque *Serverless*

es idóneo para este tipo de aplicaciones, pues logra soportar la carga computacional del procesamiento de lenguaje natural (NLP) y la gestión multimedia sin comprometer la experiencia de usuario. Este equilibrio entre complejidad técnica y velocidad de respuesta indica que la delegación de tareas cognitivas a la IA no afecta negativamente la usabilidad del sistema.

Más allá del rendimiento transaccional, la validación del componente cognitivo arrojó una precisión del 90 % utilizando el modelo Gemini 3.0 Flash. Sin embargo, el hallazgo más relevante desde la perspectiva de la ingeniería de software no es el alto porcentaje de aciertos, sino el comportamiento del sistema ante la incertidumbre. La derivación automática del caso ambiguo hacia la categoría "Otros", con un nivel de confianza de 0.62, denota un funcionamiento conservador y seguro. Este mecanismo garantiza que el prototipo no fuerce clasificaciones erróneas ante información insuficiente, priorizando la revisión humana y evitando el despliegue innecesario de recursos técnicos a incidentes mal tipificados.

Comparación con estudios previos

Los resultados de esta investigación confirman tres consistencias clave con la literatura: (a) la viabilidad técnica de los LLMs para la clasificación de reportes no estructurados, corroborada por Hölzing et al. (2024); (b) la dependencia de la trazabilidad respecto a la calidad de captura de información, alineada con Belloso de Núñez y Primera Mendoza (2015); y (c) la materialización del concepto *Smart Campus* en la capa de software sin requerir infraestructura de sensores físicos, conforme a Min-Allah y Alrashed (2020). A continuación, se detalla cada una de estas convergencias.

La capacidad del modelo para clasificar incidentes con alta fiabilidad corrobora lo expuesto por Hölzing et al. (2024), quienes sostienen que el rendimiento de los modelos de IA en tareas de identificación de riesgos es estadísticamente comparable al de expertos humanos. Mientras dicha investigación abordaba la viabilidad teórica, el presente estudio demuestra su aplicabilidad práctica en un entorno controlado, confirmando que los LLMs modernos pueden asumir tareas de

triaje técnico reduciendo la carga administrativa.

Asimismo, la mejora sustancial en la trazabilidad de los reportes se alinea con los postulados de Belloso de Núñez y Primera Mendoza (2015), quienes enfatizan que la eficacia de los sistemas de gestión depende intrínsecamente de la objetividad en la captura de información. Los hallazgos de este proyecto expanden esta idea, sugiriendo que la digitalización por sí sola es insuficiente si no se acompaña de mecanismos inteligentes que estandaricen la entrada de datos. Esto contrasta con los desafíos de adopción mencionados por Sri Wahyuningsih, Herli Sumerli et al. (2024), evidenciando que la "facilidad de uso" no reside únicamente en la interfaz visual, sino en la capacidad del sistema para interpretar el lenguaje natural del usuario, minimizando la fricción cognitiva al momento de generar un reporte.

En un sentido más amplio, este trabajo materializa el concepto de *Smart Campus* analizado por Min-Allah y Alrashed (2020). A diferencia de enfoques centrados exclusivamente en la implementación de sensores físicos (*hardware*), los resultados obtenidos validan que la inteligencia institucional también puede residir en la capa de software (*software-defined operations*). Al automatizar la gestión de recursos mediante algoritmos, se responde a la necesidad planteada por Cristia y Vlaicu (2022) de modernizar los servicios para garantizar eficiencia, adaptando esta tendencia global a la realidad y recursos de una universidad ecuatoriana, tal como sugiere Murillo Quimiz et al. (2024).

Fortalezas y limitaciones

La principal fortaleza del estudio radica en su enfoque híbrido, el cual combina una interfaz moderna con un motor cognitivo avanzado, resolviendo simultáneamente los problemas de accesibilidad para el estudiante y de gestión para el personal administrativo. La arquitectura modular adoptada asegura que el sistema sea escalable y mantenible a largo plazo.

No obstante, es necesario reconocer ciertas limitaciones técnicas y operativas inherentes a la

naturaleza de un prototipo:

- Dependencia de conectividad: la arquitectura actual requiere una conexión a internet activa y estable para la comunicación con las APIs de Google y la base de datos. La ausencia de un modo *offline* limita la capacidad de reporte en zonas del campus con baja cobertura, lo cual representa una vulnerabilidad operativa que debe ser abordada en futuras iteraciones.
- Privacidad y geolocalización: aunque el módulo de GPS es funcional, su efectividad está supeditada a los permisos del navegador del usuario. Se observó que la denegación de acceso a la ubicación resulta en reportes sin referencia geográfica precisa, lo que podría dificultar la atención del incidente.
- Sostenibilidad de costos: si bien el modelo actual es viable bajo el esquema de precios vigente, la dependencia de proveedores externos introduce un riesgo financiero asociado a la fluctuación de costos por uso de APIs y tokens de procesamiento en escenarios de alta demanda.
- Alcance de la validación técnica: al tratarse de un prototipo, no se abordaron requisitos no funcionales complejos como pruebas de estrés concurrentes o la integración con sistemas legados de la universidad, pasos necesarios para un eventual despliegue en producción.
- Ausencia de validación con usuarios finales: la validación del prototipo se realizó exclusivamente mediante inspección funcional y un corpus de prueba sintético, sin incluir a estudiantes reales de la PUCE Manabí como participantes. Esta decisión responde al alcance declarado de la investigación como Prueba de Concepto, cuyos límites excluyen explícitamente la ejecución de pruebas con usuarios finales en campo. En consecuencia, los resultados de usabilidad, facilidad de uso percibida y aceptación tecnológica del sistema no han sido medidos, lo que constituye una limitación relevante para la generalización de los hallazgos más allá del entorno técnico de pruebas.

Implicaciones y recomendaciones

Desde una perspectiva práctica, los resultados implican un cambio de paradigma en la gestión universitaria, facilitando la transición de un modelo reactivo a uno predictivo y basado en datos. La capacidad de almacenar incidentes estructurados permitirá a la institución generar mapas de calor y optimizar los presupuestos de mantenimiento preventivo. Teóricamente, el estudio contribuye al cuerpo de conocimiento sobre la aplicación de modelos de lenguaje ligeros en tareas de clasificación logística, demostrando su eficiencia sin incurrir en altos costos computacionales.

Con base en lo expuesto, se plantean las siguientes recomendaciones para la evolución del proyecto:

1. Implementación de modo *offline-first*: desarrollar capacidades de almacenamiento local para permitir la redacción y guardado de reportes sin conexión, con sincronización automática al restablecerse la red.

2. Estrategias híbridas de localización: incorporar selectores manuales de ubicación (mapas interactivos o escaneo de códigos QR en edificios) como alternativa robusta ante fallos o restricciones del GPS.

3. Validación con usuarios reales: como continuación directa de este trabajo, se recomienda ejecutar una prueba piloto con una muestra representativa de estudiantes de la PUCE Manabí durante un semestre académico. Esta validación permitiría medir la aceptación tecnológica del sistema mediante el Modelo de Aceptación Tecnológica (TAM) (Sri Wahyuningsih, Herli Sumerli et al., 2024), obteniendo retroalimentación auténtica sobre la usabilidad, funcionalidad y pertinencia de la aplicación en un contexto de uso real. Los resultados de dicha evaluación fortalecerían la validez interna y externa del estudio, aportando evidencia empírica que trasciende la verificación técnica alcanzada en este prototipo.

Conclusiones

El desarrollo de esta investigación permitió materializar una solución tecnológica que responde directamente a la necesidad de modernización en la gestión de servicios de la PUCE Manabí. Tras completar las fases de análisis, diseño, construcción y validación, se presentan las siguientes conclusiones en correspondencia con los objetivos planteados:

En primera instancia, se determinó que la problemática de la gestión de incidentes no era únicamente tecnológica, sino estructural. El análisis del proceso evidenció que el uso de canales informales provocaba la pérdida de trazabilidad en el 30 % de los casos. La definición de una taxonomía de clasificación, fundamentada en la frecuencia histórica de reportes (mantenimiento, limpieza y tecnología), fue el paso decisivo para estandarizar la entrada de datos. Esto confirma que, antes de automatizar un proceso con Inteligencia Artificial, es indispensable ordenar la lógica del negocio.

Con respecto al diseño arquitectónico, se concluye que la integración de servicios *Serverless* con LLMs es técnicamente viable y eficiente para el contexto universitario. La arquitectura implementada logró orquestar la comunicación entre el cliente web, la base de datos y la API de Google Gemini con una latencia promedio de 2.4 segundos. Este tiempo de respuesta valida que es posible incorporar procesamiento cognitivo avanzado en aplicaciones móviles sin sacrificar la fluidez de la experiencia de usuario.

En cuanto al desarrollo del prototipo, se logró construir un aplicativo funcional que centraliza exitosamente el flujo de reporte. La interfaz *mobile-first* eliminó las barreras de entrada para los estudiantes, permitiendo la captura estructurada de incidentes con evidencia fotográfica y geolocalización en un solo paso. Esta herramienta transforma lo que antes era un mensaje de texto ambiguo en un ticket de soporte accionable, dotando a la institución de un mecanismo

formal de control y seguimiento.

En lo que respecta a la evaluación del modelo de IA, la validación técnica demostró la robustez del componente cognitivo. El modelo alcanzó una exactitud del 90 % en la clasificación automática, demostrando ser una herramienta eficaz para reducir la carga administrativa de triaje. Más importante aún, el sistema probó ser seguro: la implementación de umbrales de confianza permitió identificar correctamente los casos ambiguos y derivarlos a revisión manual, garantizando que la automatización no induzca a errores operativos.

Finalmente, la verificación del cumplimiento funcional certificó que el 100 % de los requisitos prioritarios del prototipo fueron satisfechos sin defectos bloqueantes en ninguno de los cuatro módulos evaluados (autenticación, captura, procesamiento IA y gestión administrativa). Este resultado confirma que la arquitectura modular adoptada y el enfoque de validación por agrupación funcional permitieron garantizar la operatividad integral del sistema dentro del alcance definido para la prueba de concepto.

En síntesis, este proyecto trasciende el ejercicio académico para convertirse en una base sólida para la transformación digital del campus. Se ha demostrado que la tecnología *Smart Campus* no requiere infraestructuras millonarias, sino soluciones inteligentes, escalables y bien diseñadas que resuelvan problemas reales de la comunidad universitaria.

Referencias Bibliográficas

Adeyemi, I. O., & Issa, A. O. (2020). Integrating Information System Success Model (ISSM) and Technology Acceptance Model (TAM): Proposing Students' Satisfaction with University Web Portal Model. *Record and Library Journal*, *6*(1), 69-79. https://doi.org/10.20473/rlj.V6-I1.2020.69-79

Alenezi, M. (2023). Digital Transformation in Higher Education: A Critical Review. *Education Sciences*, *13*(2), 205. https://doi.org/10.3390/educsci13020205

Alobuela Carrillo, J. A. (2022). *Propuesta de gestión de incidentes y gestión de solicitudes de servicio, basándose en la metodología ITIL 4 en la dirección de informática de la Pontificia Universidad Católica del Ecuador* [Tesis de maestría, Pontificia Universidad Católica del Ecuador]. https://repositorio.puce.edu.ec/items/4afb96bd-fc1e-46a8-985e-bfb796d38cd8

Association for Computing Machinery (ACM). (2024). *The Software Engineering Code of Ethics and Professional Practice* [Código de ética profesional]. https://www.acm.org/code-of-ethics/software-engineering-code

Belloso de Núñez, N., & Primera Mendoza, N. (2015). Usabilidad de sitios web para la transparencia de la gestión en alcaldías. *TELOS. Revista de Estudios Interdisciplinarios en Ciencias Sociales*, *17*(3), 418-437. https://www.redalyc.org/articulo.oa?id=99342598003

Chowdhary, K. (2020). *Natural Language Processing*. Springer. https://doi.org/10.1007/978-81-322-3975-9

Cristia, J., & Vlaicu, R. (2022). *Digitalizar los servicios públicos: Oportunidades para América Latina y el Caribe*. Banco Interamericano de Desarrollo. https://doi.org/10.18235/0004424

Díaz-García, V., Montero-Navarro, A., et al. (2023). Managing Digital Transformation: A Case Study in a Higher Education Institution. *Electronics*, *12*(11), 2522. https://doi.org/10.3390/electronics12112522

Doshi, T. (2025, diciembre). *Gemini 3 Flash: Inteligencia de vanguardia diseñada para la velocidad* [Anuncio oficial en el Blog de Google. Accedido: 2026-01-20]. Google. https://blog.google/intl/es-419/actualizaciones-de-producto/gemini-3-flash/

Engström, E., Storey, M.-A., et al. (2020). How software engineering research aligns with design science: a review. *Empirical Software Engineering*, *25*, 2630-2660. https://doi.org/10.1007/s10664-020-09836-9

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* [Tesis doctoral, University of California, Irvine] [Tesis Doctoral].

Hernández Sampieri, R., & Mendoza Torres, C. P. (2018). *Metodología de la investigación: Las rutas cuantitativa, cualitativa y mixta* [Edición impresa]. McGraw-Hill Interamericana Editores.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, *28*(1), 75-105. https://doi.org/10.2307/25148625

Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. *Science*, *349*(6245), 261-266. https://doi.org/10.1126/science.aaa8685

Hölzing, C. R., et al. (2024). The Potential of Using Generative AI/NLP to Identify and Analyse Critical Incidents in a Critical Incident Reporting System (CIRS). *Healthcare*, *12*(19), 1964. https://doi.org/10.3390/healthcare12191964

ISO/IEC/IEEE. (2018). *ISO/IEC/IEEE 29148:2018 Systems and software engineering – Life cycle processes – Requirements engineering*. International Organization for Standardization. https://www.iso.org/standard/72089.html

Jones, M., Bradley, J., & Sakimura, N. (2015). *JSON Web Token (JWT)* (RFC N.º 7519). Internet Engineering Task Force (IETF). https://doi.org/10.17487/RFC7519

Kendig, C. E. (2016). What is Proof of Concept Research and how does it Generate Epistemic and Ethical Categories for Future Scientific Practice? *Science and Engineering Ethics*, *22*(3), 735-753. https://doi.org/10.1007/s11948-015-9654-0

Ma, H., et al. (2023). Fairness-guided few-shot prompting for large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, *36*, 43136-43155. https://proceedings.neurips.cc/paper_files/paper/2023/file/87349926-Metadata.json

Madaan, A., Tandon, N., Gupta, P., et al. (2023). Self-Refine: Iterative Refinement with Self-Feedback. *Advances in Neural Information Processing Systems (NeurIPS)*, *36*, 46534-46554. https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf

McDonnell, C. (2024). *Zod: TypeScript-first schema validation with static type inference* [Librería de software v3.22]. https://zod.dev/

Miliano, A., Steven, I., Kosim, K. P., Jayadi, R., & Mauritsius, T. (2020). Machine Learning-based Automated Problem Categorization in a Helpdesk Ticketing Application. *2020 8th International Conference on Orange Technology (ICOT)*, 1-6. https://doi.org/10.1109/ICOT51877.2020.9468789

Min-Allah, N., & Alrashed, S. (2020). Smart campus—A sketch. *Sustainable Cities and Society*, *59*, 102231. https://doi.org/10.1016/j.scs.2020.102231

Murillo Quimiz, L. R., Figueroa Morán, G. L., Merchán Carreño, E. J., & Mero Suárez, K. V. (2024). *Transformación Digital en la Educación Superior en Ecuador: Retos y Oportunidades*. Editorial Internacional Runaiki. https://doi.org/10.55867/runaiki.2024.01

Nah, F. F.-H. (2004). A study on tolerable waiting time: how long are Web users willing to wait? *Behaviour & Information Technology*, *23*(3), 153-163. https://doi.org/10.1080/01449290410001669914

Next.js Documentation: App Router Architecture [Documentación oficial del framework]. (2024). Vercel Inc. Consultado el 20 de enero de 2026, desde https://nextjs.org/docs

Nicholls, R., Fellows, R., Battle, S., & Ihshaish, H. (2022). Problem Classification for Tailored Help Desk Auto Replies. *Artificial Neural Networks and Machine Learning – ICANN 2022*, *13532*, 445-454. https://doi.org/10.1007/978-3-031-15937-4_37

Pressman, R. S., & Maxim, B. R. (2020). *Software Engineering: A Practitioner's Approach* (9.ª ed.) [Referencia estándar en ingeniería de software]. McGraw-Hill Education.

Saltos-Rivas, R., Pincay-Ponce, J., Nuñez-Campanario, L., & Tovar-Viera, R. (2021). Web Information System for the Management of Linkage Projects in a Higher Education Institution [Estudio de caso validado sobre desarrollo de software universitario en Ecuador]. *Advances in Intelligent Systems and Computing*, *1327*, 250-260. https://doi.org/10.1007/978-3-030-68080-0_19

Sri Wahyuningsih, S., Herli Sumerli, A. C., et al. (2024). Application of Technology Acceptance Model and DeLone and McLean IS Success Model to Measure Information System Design for Academic Activities. *Jurnal Informasi dan Teknologi*, *6*(1), 43-49. https://doi.org/10.60083/jidt.v6i1.470

The PostgreSQL Global Development Group. (2023). *PostgreSQL 16.1 Documentation: Row Security Policies*. https://www.postgresql.org/docs/current/ddl-rowsecurity.html

Vavera, M., et al. (2017). Comparison of server-side and client-side rendering in web applications. *Journal of Systems Integration*, *8*(3), 31-45. https://doi.org/10.20470/jsi.v8i3.313

Zhao, W. X., Zhou, K., Li, J., et al. (2024). A Survey of Large Language Models [Publicación oficial]. *IEEE Transactions on Knowledge and Data Engineering*. https://doi.org/10.1109/TKDE.2024.3377708

Apéndices

Apéndice A: Guía de entrevista semi-estructurada

Datos generales

- Objetivo: validar la taxonomía de clasificación de incidentes y definir los requisitos técnicos y operativos del sistema.
- Dirigido a: gestor de Infraestructura y Servicios de la PUCEM.
- Instrumento: guion de preguntas abiertas para validación cualitativa.

Cuestionario

A continuación, se presentan las interrogantes planteadas durante la fase de levantamiento de información:

1. Diagnóstico actual: ¿Cuál es el proceso actual para reportar incidentes en el campus y cuáles son las principales dificultades o "cuellos de botella" que enfrentan hoy (ej. reportes perdidos, lentitud, falta de seguimiento)?
2. Validación de categorías (taxonomía): basado en su experiencia operativa, ¿está de acuerdo en agrupar los problemas frecuentes en estas 4 áreas macro para la clasificación automática?
    - Mantenimiento (electricidad, fontanería, mobiliario)
    - Seguridad (accesos, robos)
    - Limpieza (desechos, plagas)

- TI (internet, equipos)

¿Considera necesario añadir alguna otra categoría principal?

3. Manejo de ambigüedad: cuando llega un reporte confuso que no encaja claramente en una categoría técnica (ej. “algo huele mal“ o “ruido extraño“), ¿a qué departamento se debería derivar por defecto para su primera revisión humana?

4. Requisitos de entrada (datos): para poder atender un reporte sin perder tiempo en inspecciones previas, ¿es indispensable que el sistema exija al estudiante una fotografía y su ubicación exacta (geolocalización)?

5. Roles y privacidad: ¿Quiénes deben tener acceso al sistema para gestionar estos reportes? ¿El acceso debe restringirse solo a jefes de área o también al personal operativo?

6. Métricas de gestión: ¿Qué indicadores o estadísticas necesitaría ver en un panel de control (*dashboard*) para evaluar si el servicio de mantenimiento está mejorando?

Apéndice B: Especificación de requisitos del sistema (SRS)

Introducción y alcance

Este documento define los requisitos técnicos para el prototipo de gestión de incidentes de la PUCESM. Su estructura se fundamenta en el estándar ISO/IEC/IEEE 29148:2018 (ISO/IEC/IEEE, 2018), sirviendo como base contractual para el diseño, desarrollo y validación del software.

Propósito del sistema

El sistema tiene como objetivo automatizar el flujo de reporte de incidentes dentro del campus mediante una aplicación web progresiva (*mobile-first*). La solución integra un modelo de Inteligencia Artificial (LLM) para clasificar y enrutar automáticamente las solicitudes, resolviendo la falta de trazabilidad y centralización del proceso actual.

Límites del sistema (alcance)

Funcionalidades incluidas:

- Autenticación federada (correo institucional) y gestión de roles (RBAC).
- Captura multimodal de incidentes (texto, imagen, geolocalización).
- Motor de inferencia IA para clasificación automática y detección de prioridad.
- Flujo de estados (pendiente $\rightarrow$ asignado $\rightarrow$ en progreso $\rightarrow$ resuelto).
- *Dashboard* administrativo para métricas y auditoría.

Exclusiones explícitas:

- Desarrollo de aplicaciones nativas (iOS/Android).
- Notificaciones *push* en tiempo real (se limita a correo/interfaz).
- Integración con sistemas académicos legados (SGA/ERP).
- Módulos financieros o de facturación.

Descripción general

Perspectiva del producto

El sistema es una solución *SaaS* (Software as a Service) desplegada sobre arquitectura *serverless*. Utiliza un enfoque de renderizado híbrido (SSR) para garantizar el rendimiento en dispositivos móviles.

Características de los usuarios

- Estudiante: actor principal. Genera reportes y consulta historial. Requiere validación de dominio @pucesm.edu.ec.
- Personal operativo (*worker*): atiende y resuelve incidentes asignados a su departamento.
- Supervisor: gestiona la cola de incidentes del departamento y reasigna tareas.
- Administrador: gestión global de usuarios, departamentos y configuración del sistema.
- Sistema IA (actor no humano): procesa el lenguaje natural para devolver metadatos de clasificación.

Suposiciones y dependencias

El funcionamiento del sistema está condicionado a las siguientes premisas operativas y técnicas:

1. Infraestructura: disponibilidad continua de los servicios en la nube (Vercel, Supabase) y conectividad a internet en el campus.
2. Servicios externos: operatividad de la API de Google Gemini para la clasificación.
3. Hardware: los dispositivos de los usuarios deben contar con navegador moderno y GPS funcional.
4. Restricciones: el almacenamiento de evidencia multimedia está limitado a 10 MB por archivo y el sistema opera exclusivamente en idioma español.

Requisitos funcionales

Tabla 4

Requisitos funcionales del sistema - Módulo de autenticación

Código	Nombre	Descripción	Prioridad
RF-01	Registro de estudiantes	El sistema debe permitir el registro de estudiantes utilizando exclusivamente correo institucional (@pucesm.edu.ec)	Alta
RF-02	Validación de correo	El sistema debe validar que el correo ingresado pertenezca al dominio @pucesm.edu.ec durante el registro	Alta
RF-03	Verificación de email	El sistema debe enviar un correo de verificación después del registro y activar la cuenta solo tras la confirmación	Alta
RF-04	Inicio de sesión	El sistema debe permitir el inicio de sesión mediante email y contraseña	Alta
RF-05	Validación de contraseña	El sistema debe requerir contraseñas con un mínimo de 8 caracteres	Alta
RF-06	Redirección por rol	El sistema debe redirigir automáticamente a los usuarios a su página correspondiente según su rol después del login	Alta
RF-07	Creación de usuarios	El sistema debe permitir a los administradores crear usuarios con roles worker, supervisor o admin	Alta
RF-09	Gestión de sesiones	El sistema debe mantener sesiones activas con renovación automática de tokens JWT	Alta
RF-10	Protección de rutas	El sistema debe proteger las rutas según el rol del usuario autenticado, denegando acceso a recursos no autorizados	Alta

Tabla 5
Requisitos funcionales del sistema - Módulo de incidentes

Código	Nombre	Descripción	Prioridad
RF-11	Creación de incidentes	El sistema debe permitir a los estudiantes crear incidentes con título, descripción, imagen opcional y ubicación geográfica	Alta
RF-12	Validación de título	El título del incidente debe tener entre 5 y 100 caracteres	Alta
RF-13	Validación de descripción	La descripción del incidente debe tener entre 10 y 500 caracteres	Alta
RF-14	Subida de imágenes	El sistema debe permitir adjuntar imágenes en formato JPEG, PNG o WebP con tamaño máximo de 10 MB	Alta
RF-15	Captura de geolocalización	El sistema debe capturar automáticamente la ubicación geográfica del dispositivo del usuario al crear un incidente	Media
RF-16	Selección de prioridad	El sistema debe permitir seleccionar la prioridad del incidente (baja, media, alta, urgente) o dejar en modo automático	Media
RF-17	Historial de incidentes	El sistema debe mostrar al estudiante el historial de todos sus incidentes reportados	Alta
RF-18	Detalle de incidente	El sistema debe mostrar la información completa de cada incidente incluyendo estado, clasificación y comentarios	Alta
RF-19	Filtrado de incidentes	El sistema debe permitir filtrar incidentes por estado, prioridad y rango de fechas	Media
RF-20	Paginación de listados	El sistema debe paginar los listados de incidentes mostrando un número configurable de elementos por página	Media

Tabla 6
Requisitos funcionales del sistema - Módulo de clasificación IA

Código	Nombre	Descripción	Prioridad
RF-21	Clasificación automática	El sistema debe clasificar automáticamente cada incidente nuevo utilizando el modelo Google Gemini AI	Alta
RF-22	Resultado de clasificación	La clasificación debe retornar: categoría, departamento sugerido, prioridad sugerida, nivel de confianza (0-1) y razonamiento	Alta
RF-23	Auto-asignación	El sistema debe asignar automáticamente el incidente al departamento sugerido si la confianza es mayor o igual al 75 %	Alta
RF-24	Clasificación pendiente	Si la confianza de clasificación es menor al 75 %, el incidente debe quedar en estado pendiente para revisión manual	Alta
RF-25	Categorías soportadas	El sistema debe soportar categorías como: eléctrico, plomería, estructural, mobiliario, seguridad, limpieza, TI, etc.	Alta
RF-26	Mapeo de departamentos	El sistema debe mapear las categorías a los departamentos: mantenimiento, seguridad, limpieza, TI, infraestructura	Alta
RF-27	Rzonamiento IA	El sistema debe almacenar el razonamiento del modelo de IA para fines de auditoría y transparencia	Media

Tabla 7

Requisitos funcionales del sistema - Módulo de gestión departamental

Código	Nombre	Descripción	Prioridad
RF-28	Listado de deptos.	El sistema debe mostrar la lista de todos los departamentos activos del campus	Alta
RF-29	Estadísticas	Cada departamento debe mostrar estadísticas de incidentes: pendientes, en progreso y resueltos	Alta
RF-30	Acceso a incidentes	El sistema debe permitir al personal acceder a los incidentes asignados a su departamento	Alta
RF-31	Visibilidad	Los trabajadores y supervisores solo deben ver incidentes asignados a su propio departamento	Alta
RF-32	Deptos. predefinidos	El sistema debe incluir: Mantenimiento, Seguridad, Limpieza, Tecnología/TI e Infraestructura	Alta
RF-33	Atributos	Cada departamento debe tener: nombre único, descripción, color distintivo e icono representativo	Media

Tabla 8

Requisitos funcionales del sistema - Módulo de flujo del proceso

Código	Nombre	Descripción	Prioridad
RF-34	Estados del incidente	El sistema debe soportar los estados: pendiente, asignado, en progreso, requiere información, resuelto y rechazado	Alta
RF-35	Estado inicial	Todo incidente nuevo debe iniciar en estado “pendiente de clasificación”	Alta
RF-36	Transición automática	Si la IA clasifica con alta confianza, el estado debe cambiar automáticamente a “asignado”	Alta
RF-37	Cambio manual	Los trabajadores deben poder cambiar el estado a: en progreso, requiere información, resuelto o rechazado	Alta
RF-38	Motivo de rechazo	Al marcar un incidente como rechazado, el sistema debe requerir un motivo de rechazo	Media
RF-39	Notas de resolución	Al marcar un incidente como resuelto, el sistema debe permitir agregar notas de resolución	Media
RF-40	Fecha de resolución	El sistema debe registrar automáticamente la fecha y hora cuando un incidente se marca como resuelto	Media

Tabla 9

Requisitos funcionales del sistema - Módulo de comentarios

Código	Nombre	Descripción	Prioridad
RF-41	Agregar comentarios	El sistema debe permitir agregar comentarios a los incidentes por parte del personal autorizado	Alta
RF-42	Límite	Los comentarios deben tener un máximo de 1000 caracteres	Media
RF-44	Metadatos	El sistema debe mostrar el autor y fecha de creación de cada comentario	Alta

Tabla 10
Requisitos funcionales del sistema - Módulo de administración

Código	Nombre	Descripción	Prioridad
RF-48	Acceso restringido	El panel de administración debe ser accesible únicamente para usuarios con rol administrador	Alta
RF-49	Dashboard	El administrador debe poder visualizar métricas: total incidentes, tiempo promedio, tasa resolución e incidentes activos	Alta
RF-50	Gestión de usuarios	El administrador debe poder crear, visualizar y gestionar usuarios del sistema	Alta
RF-51	Asignación	El administrador debe poder asignar usuarios a departamentos específicos	Alta
RF-52	Búsqueda	El administrador debe poder buscar usuarios por nombre, email o cédula	Media
RF-53	Gestión deptos.	El administrador debe poder acceder a la configuración y gestión de departamentos	Alta

Requisitos no funcionales

Tabla 11
Requisitos no funcionales del sistema - Seguridad

Código	Categoría	Descripción	Métrica
RNF-06	Seguridad	El sistema debe implementar control de acceso basado en roles (RBAC)	4 roles definidos
RNF-07	Seguridad	Las contraseñas deben almacenarse utilizando algoritmos de hash seguros	bcrypt/Argon2
RNF-08	Seguridad	La base de datos debe implementar Row Level Security (RLS) para proteger datos	Políticas RLS
RNF-09	Seguridad	El sistema debe validar y sanitizar todas las entradas para prevenir inyecciones	Validación Zod
RNF-10	Seguridad	Los tokens de sesión deben renovarse automáticamente antes de expirar	Refresh auto

Tabla 12
Requisitos no funcionales del sistema - Usabilidad

Código	Categoría	Descripción	Métrica
RNF-11	Usabilidad	La interfaz debe seguir un enfoque mobile-first con diseño responsive	Breakpoints
RNF-12	Usabilidad	El sistema debe ser compatible con navegadores web modernos	Chrome, Firefox, Safari
RNF-13	Usabilidad	Los formularios deben mostrar validación en tiempo real al usuario	Feedback
RNF-14	Usabilidad	Los mensajes de error deben ser claros, descriptivos y en idioma español	Textos español
RNF-16	Usabilidad	El sistema debe mostrar indicadores visuales durante operaciones asíncronas	Spinners/Loaders

Tabla 13
Requisitos no funcionales del sistema - Mantenibilidad y escalabilidad

Código	Categoría	Descripción	Métrica
RNF-17	Mantenibilidad	El código debe utilizar tipado estático para prevenir errores en tiempo de compilación	TypeScript strict
RNF-18	Mantenibilidad	La arquitectura debe seguir el principio de separación de responsabilidades	Capas
RNF-19	Mantenibilidad	Los componentes de UI deben ser reutilizables y modulares	Comp. React
RNF-20	Mantenibilidad	El código debe seguir estándares de calidad verificables mediante linting	ESLint config
RNF-22	Escalabilidad	El almacenamiento de archivos debe ser escalable horizontalmente	Supabase Storage
RNF-23	Escalabilidad	La aplicación debe ser desplegable en plataformas *serverless*	Vercel

Tabla 14
Requisitos no funcionales del sistema - Disponibilidad e integrabilidad

Código	Categoría	Descripción	Métrica
RNF-26	Integrabilidad	El sistema debe exponer una API RESTful con formato de respuesta consistente	Endpoints /api
RNF-27	Integrabilidad	Las respuestas de API deben seguir una estructura estandarizada	ApiResponse

Apéndice C: Descripción del flujo de procesos

Introducción

Este apéndice detalla la lógica secuencial representada en el diagrama de flujo general. El proceso describe la interacción entre los actores humanos (Estudiante, Administrador, Departamento) y los componentes automatizados (IA, Backend) a lo largo del ciclo de vida de un incidente.

El flujo se estructura en cuatro fases críticas:

Fase 1: Autenticación y enrutamiento

El proceso inicia con el ingreso del usuario al sistema.

1. Inicio de sesión: el sistema valida las credenciales y el dominio institucional.
2. Evaluación de rol: el sistema consulta los permisos del usuario y bifurca el flujo:
    - Si el rol es administrador, se dirige al panel de supervisión global.
    - Si el rol es estudiante, se habilita la interfaz de reporte de incidentes.

Fase 2: Generación y procesamiento inteligente (estudiante)

Esta fase comprende la captura de datos y la intervención del modelo de IA.

1. Creación del reporte: el estudiante ingresa los datos obligatorios: descripción textual del problema, ubicación exacta y evidencia fotográfica.

2. Envío y estado inicial: al enviar el formulario, el sistema registra el incidente en la base de datos con el estado inicial `pending_classification`.

3. Invocación de IA: de forma asíncrona, el sistema envía el *payload* (texto + imagen) a la API de Google Gemini AI.

4. Clasificación automática: el modelo procesa la entrada y retorna:

    - La categoría técnica del incidente.
    - El departamento sugerido para su atención.
    - Una asignación de prioridad automática basada en la gravedad detectada.

5. Transición de estado: una vez clasificado, el sistema actualiza el estado del incidente a `assigned` y notifica al departamento correspondiente.

Fase 3: gestión operativa (departamento)

El departamento receptor evalúa el incidente clasificado y toma una decisión tripartita:

Opción A: solicitud de información

Si el reporte es ambiguo o incompleto:

- El encargado solicita más detalles al estudiante.
- El incidente pasa al estado `requires_info`.
- El estudiante complementa el reporte.

Opción B: rechazo del incidente

Si el reporte no es válido, es duplicado o no procede:

- El encargado registra el motivo del rechazo (justificación obligatoria).
- El incidente pasa al estado `rejected`.
- El proceso finaliza.

Opción C: aceptación y resolución

Si el reporte es válido y proceden a repararlo:

1. Asignación: se designa un responsable técnico.
2. Ejecución: el incidente pasa al estado `in_progress` mientras se realizan las labores de mantenimiento.
3. Cierre: una vez solucionado, se registran las notas de resolución y evidencia del trabajo.
4. Finalización: el incidente pasa al estado final `resolved`.

Fase 4: supervisión (administrador)

Paralelamente a la gestión operativa, el flujo del Administrador permite:

- Supervisar el listado global de incidentes en todos sus estados.
- Consultar métricas de rendimiento y reportes estadísticos.

El proceso concluye cuando el incidente alcanza un estado terminal (resuelto o rechazado) o cuando el usuario finaliza su sesión de supervisión.

Apéndice D: Desglose de la arquitectura técnica

Introducción

Este anexo describe los componentes técnicos que conforman la solución. La arquitectura sigue un patrón monolítico modular basado en el *framework* Next.js 16, implementando un patrón de acceso a datos híbrido que maximiza el rendimiento sin comprometer la seguridad.

A continuación, se detallan las cuatro capas estructurales ilustradas en el diseño arquitectónico:

Capa de cliente

Es el punto de entrada para los usuarios finales. Se priorizó el rendimiento y la experiencia de usuario (UX).

- Navegador web: el sistema es compatible con motores modernos (Chromium, Gecko, WebKit), garantizando accesibilidad universal.
- React 19 y *server components*: se utiliza la última versión de la librería React. Los componentes de servidor (RSC) pre-renderizan contenido estático, mientras que los componentes de cliente gestionan la interactividad.
- Tailwind CSS v4 y shadcn/ui: el sistema de diseño utiliza clases utilitarias para una estilización rápida y componentes accesibles pre-construidos.
- Cliente Supabase (SDK): el *frontend* instancia un cliente ligero que permite realizar consultas de lectura (*SELECT*) directamente a la base de datos bajo el amparo de las

políticas de seguridad, reduciendo la latencia al evitar saltos innecesarios por la API para datos públicos o propios del usuario.

Capa de aplicación

Constituye el cerebro del sistema, donde residen las reglas de negocio críticas y las operaciones de escritura complejas.

- Next.js *app router*: gestiona el enrutamiento y orquesta la estructura del proyecto.
- API Routes (/api/v1/*): se diseñaron *endpoints* RESTful para manejar lógica sensible que no debe ejecutarse en el cliente, tal como:
    - La orquestación de la llamada a la IA (para no exponer las API Keys).
    - Transacciones de escritura que requieren validaciones de múltiples tablas.
    - Sanitización avanzada de datos de entrada.
- *Middleware* (autenticación): intercepta las peticiones hacia las rutas protegidas para verificar la validez del Token JWT y los permisos del rol de usuario.
- Validadores (Zod): librería de definición de esquemas utilizada en el servidor para asegurar que los datos recibidos en la API cumplan estrictamente con los formatos esperados.

Capa de datos

Responsable de la persistencia y la integridad de la información. Se utiliza Supabase como plataforma integral (BaaS).

- PostgreSQL: motor de base de datos relacional robusto.

- Row Level Security (RLS): este es el componente crítico que permite la arquitectura híbrida. Las políticas de seguridad se aplican directamente en el motor de base de datos. Esto garantiza que, ya sea que la petición venga de la API interna o directamente del cliente Supabase en el *frontend*, el usuario solo pueda acceder a los datos que le corresponden.
- Supabase auth: gestiona la identidad de los usuarios y emite los tokens que consumen tanto el *middleware* como las políticas RLS.
- Storage: sistema de almacenamiento de objetos para guardar las evidencias fotográficas de los reportes.

Servicios externos

- Google Gemini AI: motor de inferencia externo. El sistema se conecta a este servicio exclusivamente desde la Capa de Aplicación (API Routes) para procesar el texto y las imágenes, retornando la clasificación estructurada en JSON.

Apéndice E: Matriz de pruebas del modelo de inteligencia artificial

Registro de ejecución

La presente matriz documenta los datos brutos obtenidos tras la ejecución del corpus de validación sobre el módulo de clasificación automática.

Detalles de la prueba:

- Fecha: 23 de enero de 2026.
- Modelo: Google Gemini 3.0 Flash (`gemini-3.0-flash`).
- Configuración: temperatura 0.1 (determinística).

Tabla 15

Resultados detallados de la clasificación automática (corpus de validación)

ID	Descripción del Incidente	Esperado	Obtenido	Conf.	Estado
1	Las luces del pasillo del edificio B se fundieron y está completamente oscuro.	Mantenimiento	Mantenimiento	0.98	✓
2	Hay una persona sospechosa rondando el estacionamiento desde hace una hora.	Seguridad	Seguridad	0.99	✓
3	Los baños del tercer piso están muy sucios y huelen mal, necesitan limpieza urgente.	Limpieza	Limpieza	0.97	✓
4	El proyector del aula 201 no enciende y tenemos presentación mañana.	Tecnología	Otros	0.62	△
5	Se cayó un letrero de señalización en la entrada principal del campus.	Infraestructura	Infraestructura	0.92	✓
6	El inodoro del baño de hombres del edificio A está tapado y desbordando agua.	Mantenimiento	Mantenimiento	0.96	✓
7	No hay conexión WiFi en toda la biblioteca desde esta mañana.	Tecnología	Tecnología	0.94	✓
8	Encontré una ventana rota en el laboratorio de química, parece vandalismo.	Seguridad	Seguridad	0.89	✓
9	Hay basura acumulada en el patio central, los contenedores están llenos.	Limpieza	Limpieza	0.98	✓
10	Hay un problema general en el edificio D, algo no está bien pero no sé qué es exactamente.	Otros	Otros	0.75	✓

Nota 1: Conf. = Nivel de Confianza (0-1).

Nota 2: △ = Desviación controlada por baja confianza.

Apéndice F: Lista de verificación (checklist)

Protocolo de inspección

La siguiente tabla detalla la ejecución de la inspección funcional realizada al prototipo. Se evaluó cada requisito crítico mediante una prueba de aceptación de usuario (UAT) ejecutada directamente sobre el entorno de *staging*.

- Evaluador: Luis Alberto Velasco Anchundia
- Versión del software: v1.0.0-beta
- Entorno: *staging* desplegado en Vercel, base de datos en Supabase
- Dispositivos: navegador Arc (basado en Chromium) en escritorio; iPhone 11 para verificación *responsive*
- Herramientas de observación: panel *Network* y consola de DevTools; panel de gestión de Supabase
- Criterio de aceptación: el comportamiento observado debe coincidir exactamente con el comportamiento esperado definido en la SRS (Apéndice B).

Tabla 16

Ejecución de la lista de verificación (checklist)

ID Req	Prueba Realizada (Acción)	Comportamiento Esperado	Observación	Estado
RF-01	Intentar registro con correo personal (@gmail.com) y luego con institucional.	Sistema debe bloquear @gmail y permitir @pucesm.edu.ec	El sistema mostró error "Debes usar tu correo institucional..." con Gmail, luego con el institucional se registró con éxito.	✓
RF-11	Completar formulario adjuntando foto y activando GPS.	El sistema debe capturar coordenadas y subir imagen.	La imagen se guardó en Storage y las coordenadas $(-1{,}05, -80{,}45)$ se registraron en BD.	✓
RF-21	Enviar reporte y verificar log de IA.	La IA debe devolver un JSON con categoría.	El sistema recibió respuesta de Gemini con la categoría correcta.	✓
RF-23	Enviar incidente claro de "Fuga de agua".	Sistema debe asignar automáticamente a "Mantenimiento".	El estado cambió a "Asignado" y el departamento "Mantenimiento" recibió el incidente.	✓
RF-34	Trabajador cambia estado de "Asignado" a "En Progreso".	El cambio debe reflejarse en la vista del estudiante.	El estudiante visualizó la actualización de estado en tiempo real.	✓
RF-49	Acceder al Dashboard como Administrador.	Se deben ver gráficos de barras por categoría.	Los contadores coincidieron con el número de incidentes creados en la prueba.	✓

Nota: Estado ✓ = Cumple (Pass).

Apéndice G: Análisis comparativo y criterios de selección del modelo de IA

Introducción

El presente apéndice documenta el análisis técnico-económico que sustenta la selección de Gemini 3 Flash como motor de inferencia del prototipo, cuya síntesis visual se presenta en la Figura 6 del cuerpo principal. Los datos de rendimiento y costo utilizados en este análisis provienen de la publicación técnica oficial de Google sobre la familia de modelos Gemini 3 (Doshi, 2025).

Criterios de evaluación

La selección del modelo de IA no se realizó de forma arbitraria, sino mediante la definición de criterios alineados con los requisitos técnicos y la viabilidad operativa del prototipo. La Tabla 17 presenta los seis criterios establecidos y su justificación.

Tabla 17
Criterios de evaluación para la selección del modelo de IA

ID	Criterio	Justificación para el proyecto	Peso
C1	Costo por token	Viabilidad económica para uso institucional sostenido. Un PoC universitario requiere costos mínimos de operación.	Alto
C2	Latencia de inferencia	Tiempo de respuesta inferior a 5 segundos para cumplir con los umbrales de tolerancia en sistemas web interactivos.	Alto
C3	Soporte multimodal	Capacidad de procesar texto e imagen simultáneamente (evidencia fotográfica del incidente).	Alto
C4	Salida JSON nativa	Generación determinística en formato JSON para integración directa con el *backend* sin *parsing* adicional.	Alto
C5	Idioma español	Comprensión precisa de descripciones coloquiales en español, incluyendo jerga estudiantil.	Medio
C6	API y documentación	SDK oficial disponible, documentación técnica completa y estabilidad del servicio.	Medio

Indicadores de rendimiento del modelo seleccionado

Según los *benchmarks* publicados por Doshi (2025), Gemini 3 Flash alcanza rendimiento de frontera en múltiples indicadores de referencia mientras mantiene una ventaja significativa en velocidad y costo. La Tabla 18 sintetiza los datos clave y su relevancia para el proyecto.

Tabla 18
Indicadores clave de rendimiento de Gemini 3 Flash

Indicador	Valor	Relevancia para el proyecto
GPQA Diamond (razonamiento académico)	90.4 %	Capacidad de razonamiento complejo para clasificar reportes ambiguos.
MMMU Pro (comprensión multimodal)	81.2 %	Comprensión integrada de texto e imagen para analizar evidencia fotográfica.
SWE-bench Verified (codificación agéntica)	78.0 %	Generación confiable de salida JSON estructurada y uso de herramientas.
Costo de entrada	\$0.50 / 1M tokens	Operación económicamente viable para un entorno institucional.
Costo de salida	\$3.00 / 1M tokens	Respuestas cortas en formato JSON minimizan el gasto por transacción.
Velocidad vs. Gemini 2.5 Pro	3× más rápido	Cumple holgadamente con el umbral de latencia objetivo ($<$ 5s).
Eficiencia de tokens	30 % menos tokens	Reduce el costo operativo por transacción frente a modelos de generación anterior.

Nota. Datos extraídos de Doshi, 2025. Precios vigentes a diciembre de 2025.

Justificación de la decisión

La Figura 6 presenta la comparativa completa publicada por Doshi (2025), donde Gemini 3 Flash se evalúa frente a modelos como Gemini 3 Pro, GPT-5.2 Extra High, Claude Sonnet 4.5 y Grok 4.1 Fast. Del análisis de estos datos contrastados con los criterios definidos, se derivan las siguientes conclusiones:

1. Frontera de Pareto costo-rendimiento: Gemini 3 Flash se posiciona en la frontera de Pareto entre calidad, costo y velocidad, ofreciendo rendimiento comparable al de modelos de categoría Pro a una fracción del precio. Esto lo convierte en la opción óptima para un prototipo institucional sin presupuesto dedicado a servicios de IA.

2. Soporte multimodal nativo: el modelo integra nativamente la comprensión visual y textual,

lo cual es un requisito indispensable del sistema dado que los reportes incluyen evidencia fotográfica junto con la descripción textual del incidente.

3. Modo JSON nativo: la configuración `responseMimeType: .ªpplication/json"` garantiza salida determinística sin necesidad de *prompts* adicionales de formato, eliminando errores de *parsing* en la integración con el *backend*.

4. Modelos descartados: los modelos de la categoría Pro (Gemini 3 Pro, GPT-5.2) fueron descartados por su costo significativamente superior ($5\times$ a $10\times$ mayor), el cual resulta incompatible con la sostenibilidad de un sistema institucional. Modelos de código abierto (Llama, Mistral) fueron descartados por requerir infraestructura de cómputo dedicada (*self-hosting*), lo cual excede el alcance del PoC.

Apéndice H: Fundamentación de la distribución estimada de incidentes

Introducción

Este apéndice documenta el procedimiento de obtención y estructuración de los datos cualitativos que sustentan la distribución presentada en la Figura 7 del cuerpo principal (Diagrama de Pareto). Su propósito es proporcionar trazabilidad entre la fuente primaria de información y la representación estadística utilizada para fundamentar la taxonomía de clasificación del sistema.

Procedimiento de obtención de datos

Durante la ejecución de la entrevista semi-estructurada (Apéndice A), específicamente en respuesta a la pregunta 2 (validación de categorías), el informante clave —representante del área de gestión de infraestructura y servicios de la PUCE Manabí— proporcionó estimaciones porcentuales sobre la distribución de incidentes reportados históricamente. Estas estimaciones fueron elaboradas por el informante en conjunto con el equipo técnico de su departamento, basándose en su experiencia operativa acumulada.

Es importante precisar que estas cifras constituyen **estimaciones cualitativas** derivadas del conocimiento experto, y no registros contables formales. La institución no disponía de un sistema de registro histórico estructurado previo a esta investigación, por lo cual las frecuencias reportadas representan la percepción operativa del equipo de gestión fundamentada en su experiencia diaria de atención de incidentes.

Datos brutos del informante

La Tabla 19 presenta las estimaciones proporcionadas por el experto, organizadas por categoría de incidente y acompañadas de la descripción operativa que el informante asoció a cada una.

Tabla 19
Estimaciones de frecuencia de incidentes proporcionadas por el informante clave

Categoría	Descripción operativa del informante	Frecuencia Est.
Mantenimiento	Reparaciones eléctricas, hidrosanitarias y de mobiliario. Constituye la mayor carga operativa del departamento.	40 %
Limpieza	Higiene de espacios comunes y gestión de residuos. Segunda área con mayor volumen de solicitudes.	25 %
Tecnología	Fallas de conectividad WiFi, equipos de proyección y hardware institucional.	15 %
Seguridad	Control de accesos, reportes de vandalismo y situaciones que requieren vigilancia.	10 %
Infraestructura	Daños estructurales mayores a la edificación. Ocurrencia esporádica.	5 %
Otros / Ambiguos	Reportes que no encajan claramente en ninguna categoría técnica definida.	5 %
Total		**100 %**

Nota. Datos proporcionados por el informante clave durante la entrevista semi-estructurada (Apéndice A).

Derivación del análisis de Pareto

A partir de los datos de la tabla anterior, se aplicó el principio de Pareto (regla 80/20) para identificar las categorías de mayor impacto operativo. La Tabla 20 presenta el cálculo de frecuencias acumuladas que fundamenta la construcción de la Figura 7.

Tabla 20

Cálculo de frecuencia acumulada para el análisis de Pareto

Orden	Categoría	Frecuencia (%)	Acumulado (%)
1	Mantenimiento	40	40
2	Limpieza	25	65
3	Tecnología	15	80
4	Seguridad	10	90
5	Infraestructura	5	95
6	Otros	5	100

Nota. Elaboración propia. El acumulado confirma que tres categorías (mantenimiento, limpieza y tecnología) concentran el 80 % de la carga operativa estimada, validando el principio de Pareto.

Implicaciones para el diseño del sistema

El análisis de Pareto reveló que tres categorías concentran el 80 % de la carga de atención estimada. Este hallazgo tuvo dos implicaciones directas en el diseño del prototipo:

1. **Priorización en la ingeniería de prompts:** se incluyeron ejemplos representativos (*few-shot*) de las tres categorías prioritarias (mantenimiento, limpieza y tecnología) en el *system prompt* del modelo de IA, con el objetivo de maximizar la precisión de clasificación en los escenarios de mayor frecuencia operativa.

2. **Estructura de la taxonomía:** se confirmó la necesidad de mantener las seis categorías como estructura jerárquica completa. Aunque las tres categorías menores representan solo el 20 % del volumen, incluyen incidentes de alta criticidad (seguridad) y casos estructurales (infraestructura) que requieren derivación especializada, así como una categoría de control (otros) para gestionar la ambigüedad.