Prototipo de sistema distribuido basado en IA para la gestión de logs que apoye procesos de DRP – ProtypeLog

Robin Beltrán Pinzón

Javier Alejandro Leon Mendoza

Diego Andrés Oviedo Cortes

Facultad de Ingeniería de Sistemas, Universidad EAN

Unidad de Estudio: Proyecto de Grado

Docente Marie Jose Chery Leal

Bogotá, 21 de noviembre de 2025

Prototipo de sistema distribuido basado en IA para la gestión de logs que apoye procesos de DRP – ProtypeLog

Robin Beltrán Pinzón

Javier Alejandro Leon Mendoza

Diego Andrés Oviedo Cortes

Trabajo de grado presentado como requisito para optar al título de:

Ingeniería de Sistemas

Facultad de Ingeniería de Sistemas, Universidad EAN

Unidad de Estudio: Proyecto de Grado

Docente Marie Jose Chery Leal

Bogotá, 21 de noviembre de 2025

Resumen

La integración de la computación en la nube y las tecnologías de redes inteligentes ha aumentado significativamente la complejidad de los sistemas empresariales, generando un crecimiento acelerado en el número de registros producidos por equipos y aplicaciones. En este contexto, el proceso de *logging (*registro de eventos) adquiere un papel estratégico, ya que el acceso oportuno y confiable a la información contenida en dichos registros es esencial para garantizar la continuidad operativa y la recuperación eficiente ante desastres. Sin embargo, los métodos tradicionales de gestión de *logs* presentan limitaciones para procesar grandes volúmenes de datos y ofrecer información útil en un marco temporal adecuado.

Este proyecto propone desarrollar un prototipo de solución tecnológica basado en un agente de inteligencia artificial, orientado hacia la automatización del proceso de *logging* en entornos distribuidos y que facilite el proceso de mantenimiento de manera que ofrezca soporte y/o apoyo a procesos de recuperación de desastres (DRP).

La validación del prototipo demostrará su potencial para mejorar la accesibilidad a los registros, optimizar la recuperación ante desastres y ofrecer una base escalable para la gestión de información en aplicaciones empresariales.

Palabras clave: registro de sistema, agentes de IA, DRP, *logging.*

Abstract

The integration of cloud computing and innovative networking technologies has significantly increased the complexity of business systems, accelerating the growth of records produced by equipment and applications. In this context, event logging takes on a strategic role because timely and reliable access to the information in these logs is essential for ensuring operational continuity and efficient disaster recovery. However, traditional log management methods are limited in their ability to process large volumes of data and provide useful information in a timely manner.

This project proposes the development of a prototype technological solution based on an artificial intelligence agent, designed to automate the logging process in distributed environments and facilitate maintenance, thereby providing support for disaster recovery processes (DRP).

Validating the prototype will demonstrate its potential to improve log accessibility, optimize disaster recovery, and provide a scalable foundation for managing information in enterprise applications.

Keywords: System log, AI agents, DRP, *logging.*

Contenido

Pág.

Lista de Tablas

Pág.

Tabla de Figuras

Introducción

En la actualidad, la alta disponibilidad de los sistemas de información en la industria colombiana es un factor clave, especialmente en sectores críticos como el gubernamental, bancario y de salud. Debido a la gran demanda de usuarios, estos sistemas deben ser resilientes a fallos; es decir, capaces de recuperarse rápidamente ante cualquier interrupción crítica, con el fin de minimizar pérdidas económicas y preservar la confianza de los usuarios.

Con la creciente necesidad de digitalizar y automatizar procesos, el desarrollo de soluciones bajo el modelo SaaS (*Software as a Service)* ha cobrado gran relevancia. Para responder a esta demanda, los sistemas suelen desarrollarse de forma ágil, lo que, sin un control de calidad adecuado, puede dar lugar a errores causados por factores humanos, problemas de rendimiento o configuraciones incorrectas.

En muchos casos, la recuperación del servicio ante una falla implica un proceso manual de diagnóstico para identificar las causas del problema y determinar las acciones correctivas necesarias. Este proyecto propone la implementación de un prototipo capaz de automatizar dicha tarea de diagnóstico y notificar al equipo encargado de las posibles soluciones, con el objetivo de reducir significativamente el tiempo de recuperación del sistema.

Para lograr este propósito, se plantea el desarrollo de un sistema distribuido orientado al procesamiento de eventos de error generados por aplicaciones web y móviles. El enfoque del proyecto se estructura en torno al marco de trabajo Kanban e incluye de manera general las etapas de diseño arquitectónico, selección del stack tecnológico, implementación y despliegue de la solución.

Objetivos

Objetivo general

Desarrollar un prototipo de solución tecnológica, basado en un agente de inteligencia artificial, para la gestión y el análisis de logs, con el fin de generar una base de conocimiento que apoye los procesos de planificación y recuperación ante desastres (DRP).

Objetivos específicos

- Definir los requerimientos funcionales y no funcionales del sistema, a través de la identificación, priorizándolos y organizándolos, basándose en una metodología ágil, con el propósito de estructurar el desarrollo del prototipo de manera clara y progresiva.
- Diseñar la arquitectura de software, empleando diagramas de modelado y principios de diseño orientados a servicios, con el fin de disponer de una base técnica que guíe la implementación del prototipo.
- Demostrar el prototipo funcional para la gestión y análisis de logs, utilizando un flujo de trabajo iterativo controlado con una metodología ágil y aplicando técnicas de inteligencia artificial para el agente, con el propósito de materializar la solución tecnológica propuesta.

Definición del problema

En el contexto actual de transformación digital, muchas empresas de consultoría y desarrollo de software ofrecen soluciones tecnológicas orientadas a automatizar procesos y satisfacer las necesidades específicas de sus clientes. No obstante, en un esfuerzo por entregar valor de forma continua y ágil, estas soluciones suelen desarrollarse bajo metodologías como Scrum o Kanban, lo cual, si no se acompaña de mecanismos y estrategias adecuadas para asegurar la calidad del software, puede comprometer la confiabilidad de este.

Aunque los enfoques ágiles permiten reducir los tiempos de entrega y mejorar la capacidad de respuesta ante los cambios del cliente, también pueden originar diversas dificultades técnicas y operativas. Entre los problemas más frecuentes destacan: vulnerabilidades y brechas de seguridad en los sistemas de información, problemas de rendimiento, caídas de servicio e indisponibilidad de los sistemas y reprocesos y errores en los flujos del negocio implementado, lo cual afecta la confianza de los usuarios finales.

La ausencia de controles de calidad robustos, sumada a la falta de mecanismos eficientes de diagnóstico y recuperación ante fallos, contribuye a pérdidas económicas, pérdida de datos sensibles y deterioro de la imagen corporativa (Pressman & Maxim, 2020). En la mayoría de los casos, la detección y solución de incidentes dependen de procesos manuales que retrasan la continuidad del servicio, especialmente en aplicaciones críticas para sectores como salud, banca o administración pública.

Justificación

Garantizar la disponibilidad, seguridad y rendimiento de los sistemas de información es fundamental para mantener la continuidad operativa de las organizaciones y preservar la confianza de los clientes. En entornos donde las aplicaciones son críticas (como en entidades gubernamentales, bancarias o de salud), cualquier falla no diagnosticada a tiempo puede generar pérdidas económicas significativas, interrupciones en el servicio y un deterioro en la reputación de la empresa.

Por ejemplo, en Latinoamérica, el costo promedio de filtraciones de datos alcanzó USD 2,76 millones en 2024, siendo aún más alto para sectores como el industrial (USD 3,54 millones) y el financiero (USD 3,22 millones) (IBM, 2024). Un estudio realizado en el país reveló que el 43 % de las empresas pierden entre 1 y 5 millones de dólares al año debido a fallos de software y mantenimiento deficiente (ACIS, 2024). También, según Atlassian, los minutos de inactividad pueden representar pérdidas de miles de dólares por minuto en empresas medianas o grandes (Atlassian, s.f.).

La implementación de un prototipo que automatice la tarea de diagnóstico y sugiera posibles soluciones a los incidentes permite reducir el tiempo de recuperación de los servicios, minimizando el impacto negativo tanto en el negocio como en los usuarios finales. Esto se traduce en:

- Menor pérdida financiera por interrupciones no previstas o tiempo fuera de servicio.
- Reducción de riesgos legales y de sanciones por brechas de seguridad o pérdida de datos.

- Preservación de la confianza del cliente y de la reputación corporativa.

Cabe agregar que, al incorporar herramientas automatizadas de diagnóstico, se fomenta la mejora continua en la calidad del software, mitigando riesgos de seguridad, problemas de rendimiento y errores de configuración.

De esta manera, el proyecto no solo contribuye a la optimización de los procesos de soporte y mantenimiento, sino que también fortalece la resiliencia de las soluciones tecnológicas, mejorando la competitividad y la satisfacción del cliente. En un mercado cada vez más exigente con la confiabilidad y la transparencia, esta solución diferenciará a la organización frente a competidores que aún dependen de procesos manuales de diagnóstico.

Análisis de Requerimientos

El análisis de requerimientos constituye un paso fundamental en el ciclo de vida del desarrollo del software (SDLC), pues permite definir lo que debe hacer el sistema antes de establecer cómo se implementará. Para este proyecto, el análisis se orienta hacia el diseño de un prototipo capaz de procesar de manera inteligente los registros de ejecución (logs) de aplicaciones empresariales y apoyar la recuperación temprana de servicios críticos.

El proceso de levantamiento de requerimientos se realizó tomando en cuenta tres aspectos principales:

a. La intención del producto.
b. La verificación de parámetros de diseño.
c. La estimación de características técnicas esperadas.

Intención del producto

El prototipo tiene como propósito principal automatizar la gestión y el diagnóstico de fallas en aplicaciones empresariales mediante el análisis inteligente de logs. La intención es dar apoyo a la recuperación de servicios críticos, asegurando la continuidad operativa de las organizaciones y fortaleciendo la confianza de los usuarios.

Las metas principales de este producto son:

- Detectar y clasificar errores de forma automática en los registros del sistema.
- Proporcionar al equipo de soporte sugerencias de soluciones basadas en patrones históricos y modelos de IA.
- Facilitar la integración con sistemas empresariales distribuidos y basados en la nube.
- Centralizar el conocimiento en una Base de Conocimiento.

- Servir como base para una futura solución SaaS, escalable y de integración con múltiples plataformas de logs.

Verificación de parámetros de diseño

Con base en los problemas identificados (vulnerabilidades, fallos de rendimiento, ausencia de controles de calidad) y en el estado del arte sobre detección de anomalías en registros, los parámetros que deben verificarse en el diseño del prototipo son:

- Disponibilidad y resiliencia: el sistema debe procesar logs en tiempo real sin convertirse en un punto único de fallo.
- Escalabilidad: el prototipo debe poder operar tanto en entornos locales como en arquitecturas en la nube, ajustándose al volumen de datos generado.
- Seguridad: la manipulación de logs debe garantizar la confidencialidad e integridad de la información, por lo que el acceso a dichos registros debe estar restringido mediante mecanismos de autenticación y cifrado.
- Interoperabilidad: capacidad de integrarse con diferentes fuentes de logs (servidores de aplicaciones, bases de datos, contenedores, etc.).
- Usabilidad: interfaz de notificación clara para los equipos de soporte, con reportes comprensibles y accionables.

Estimación de características de diseño

Se plantean las siguientes estimaciones iniciales de diseño:

- Volumen de procesamiento: capacidad mínima para analizar ≥ 10.000 líneas de logs por minuto, considerando escenarios empresariales reales.

- Latencia de respuesta: los incidentes críticos deben ser detectados y reportados en un máximo de 5 segundos desde su ocurrencia.
- Almacenamiento de logs: integración con un motor de base de datos NoSQL o de búsqueda (ElasticSearch, MongoDB o similar) que permita indexación y consultas rápidas.
- Modelo de inteligencia artificial: entrenamiento inicial sobre un conjunto de datos históricos (logs de errores conocidos) con capacidad de aprendizaje incremental.
- Módulo de notificación: integración con canales de comunicación existentes (correo electrónico, Slack, Microsoft Teams) para alertas automáticas al equipo de soporte.
- Recomendaciones de recuperación: el sistema debe ofrecer, junto con la alerta, al menos una acción correctiva sugerida, derivada del análisis de patrones previos.
- Escenarios de prueba: el prototipo debe validarse con casos reales de logs de fallos en aplicaciones empresariales para garantizar la aplicabilidad del modelo.

Requerimientos Funcionales y no Funcionales del sistema prototipo

Para efectos de trazabilidad, los requerimientos se clasifican en requerimientos funcionales y no funcionales:

Requerimientos funcionales

Los requisitos funcionales describen con precisión las acciones y los comportamientos que debe cumplir el sistema; determinan el alcance de lo que hará, desde las interacciones con la persona usuaria hasta las respuestas del propio sistema. Por ello, es indispensable que se formulen con claridad y consistencia para orientar el diseño, la implementación y la verificación del aplicativo. En la **Tabla 1** se presentan los requisitos funcionales definidos.

Tabla 1.

Requerimientos Funcionales V1

# de Req. Funcional	***Descripción***
RF-001	*El sistema debe permitir la ingesta automática de logs provenientes de múltiples fuentes*
RF-002	*El sistema debe aplicar técnicas de procesamiento de lenguaje natural y algoritmos de aprendizaje automático para detectar anomalías en los registros.*
RF-003	*El sistema debe generar reportes automáticos con la descripción del error, su criticidad y posibles causas.*
RF-004	*El sistema debe enviar notificaciones al equipo de soporte en tiempo real con las acciones sugeridas*
RF-005	*El sistema debe almacenar los incidentes diagnosticados para su consulta histórica.*

Nota: Elaborado por autor

Requerimientos no funcionales

Los requerimientos no funcionales establecen las cualidades y restricciones del sistema (como rendimiento, disponibilidad, seguridad, usabilidad, mantenibilidad y escalabilidad) que

condicionan su operación y experiencia de uso. Estos criterios deben expresarse con claridad y consistencia, pues orientan el diseño arquitectónico, las decisiones tecnológicas y la verificación de calidad. En la siguiente tabla, se presentan los requerimientos no funcionales definidos.

Tabla 2.

Requerimientos No Funcionales V1

# de Req. Funcional	***Descripción***
RNF-001	*El sistema debe garantizar un tiempo de respuesta menor a 10 segundos en la detección de eventos críticos*
RNF-002	*El sistema debe estar protegido mediante autenticación y control de accesos.*
RNF-003	*El sistema debe ser escalable horizontalmente, soportando crecimiento en volumen de datos.*
RNF-004	*El sistema debe ser interoperable con sistemas empresariales distribuidos en la nube o en entornos on-premises.*
RNF-005	*La arquitectura del sistema debe ser modular, permitiendo futuras mejoras o integraciones.*

Nota: Elaborado por autor

Marco Teórico

En esta sección se ofrece una perspectiva general de los conceptos teóricos que sustentan el diseño del prototipo propuesto. Su objetivo es brindar una visión clara y accesible sobre los elementos tecnológicos que intervienen en el desarrollo del prototipo; de este modo, se facilita la comprensión de los fundamentos técnicos y metodológicos a lectores no especializados.

En la estructura del contenido se abordan definiciones para los *logs* en el contexto tecnológico, definiciones relacionadas a los procesos de gestión y análisis de estos registros, conceptos de computación en la nube, relacionados a la Inteligencia Artificial, relacionados a centralización y gestión del conocimiento y otros conceptos esenciales:

Log: Los registros (*logs*) son usados para analizar los estados de ejecución de los sistemas, la supervisión del rendimiento y el diagnóstico de fallas, mediante el registro de eventos que representan estados del sistema en tiempo real, sin importar su causa: ya sea hardware (parte física del sistema) o software (relativo a las aplicaciones dentro del sistema), (Chuah et al., 2022). Son requeridos para el desarrollo y el mantenimiento de los sistemas informáticos porque documenta la operación y el estado de estos (Hu et al., 2025).

Procesamiento de logs. La información puede ser registrada en una amplia variedad de formatos dependiendo de la aplicación o del proceso, del sistema operativo u otros factores. El procesamiento de los logs centraliza y flexibiliza la extracción de esas líneas de registro sin procesar. Una vez configuradas las reglas de procesamiento y la información es procesada, esta es enviada a almacenamiento y está disponible para posterior análisis (manual o automatizado). (Dynatrace, 2025).

Análisis de logs. Es el análisis de los datos de registro extraídos del procesamiento de logs, en busca de orígenes sobre por qué un sistema o aplicación en particular no funciona (para los *logs* de tipo error). Dada la implementación de inteligencia artificial (IA) y Machine Learning (ML) y sus capacidades para el reconocimiento de patrones, se acerca el tiempo de análisis y se mejora la precisión de los resultados.(Flinders & Smalley, 2025)

Base de conocimiento (KB). Es un repositorio centralizado de información que se organiza y almacena para facilitar el acceso y el uso eficiente del conocimiento. (Slack.com, 2024).

Arquitectura distribuida. Modelo de diseño de sistemas en el que los componentes de software residen en computadoras distintas, pero se comunican y coordinan para lograr un objetivo común. Este enfoque distribuye la carga de procesamiento y los recursos, en lugar de depender de un único servidor central, lo que resulta en mayor escalabilidad, flexibilidad, concurrencia y tolerancia a fallos. (Zettler, 2025). Ver SOA.

Arquitectura orientada a servicios (SOA). Es un método de diseño de software que tiene como enfoque la construcción de software funcional y escalable a partir de componentes individuales, llamados servicios. (Chen, 2024)

Automatización. Aplicación de tecnologías que permiten ejecutar procesos sin intervención humana constante.(RedHat.com, 2025)

Inteligencia Artificial. La inteligencia artificial (IA) es una tecnología que permite a las computadoras y máquinas simular el aprendizaje humano, la comprensión, la resolución de problemas, la toma de decisiones, la creatividad y la autonomía. (Stryker & Kavlakoglu, 2025)

Modelo de Lenguaje Grande (LLM). Tipo de inteligencia artificial (IA) entrenado con grandes cantidades de datos de texto para comprender y generar lenguaje humano. (AWS.com, 2025)

La **computación basada en la nube** es la disponibilidad a pedido de recursos de procesamiento (como infraestructura y almacenamiento), así como servicios a través de Internet. Elimina la necesidad de que las personas y las empresas administren ellos mismos los recursos físicos y solo paguen por lo que usan. (Google Cloud, 2025b)

Arquitectura de nube. Se refiere al diseño y conecta todos los componentes y tecnologías necesarios para la computación en la nube. (Google Cloud, 2025a)

Recuperación ante desastres (DRP). En el contexto de IT, es la forma en que se reanudan las operaciones regulares después de un desastre. Esto generalmente se logra mediante la reanudación de las actividades esenciales y los procesos y sistemas utilizados para respaldarlas. (Fortinet.com, 2023)(IBM.com, 2025)

Considerando estos términos, a continuación, se realiza una introducción a la literatura correspondiente a los procesos que aquí se relacionan:

> En recientes años, con la integración de la computación basada en la nube y las tecnologías de redes inteligentes, la complejidad de los sistemas ha ido aumentando, lo que plantea desafíos importantes para los sistemas tradicionales de detección de anomalías, que presentan baja precisión, baja escalabilidad e interpretabilidad limitada (Ji et al., 2025). Teniendo esto en cuenta y de acuerdo con el artículo, Ji et al. (2025), “los investigadores han explorado varias direcciones, incluida la detección de anomalías de registros” (Zhang et al., 2022), el diagnóstico colaborativo utilizando sistemas multi-agente y la aplicación de modelos de lenguaje preentrenados en el monitoreo del sistema (Lee et al., 2025; Peddinti et al., 2023)”

Para facilitar el rastreo de anomalías dentro de la densidad de los sistemas actuales, se han implementado varios enfoques que, en muy grandes rangos, se clasifican en modelos tradicionales de aprendizaje automático, arquitecturas de aprendizaje profundo basadas en secuencias y agentes autónomos basados en LLM. Los modelos tradicionales se han basado principalmente en Máquina de Vectores de Soporte (SVM, por sus siglas en inglés), Random Forest (RF) y los árboles de decisión (DT, por sus siglas en inglés), usando timestamps (marcas de tiempo), direcciones IP y número de puertos como características. Sin embargo, a menudo presentan dificultades para generalizar en entornos heterogéneos y frágiles con formatos de registro desconocidos (Lee et al., 2025). En las arquitecturas basadas en secuencia, se centra en anomalías logarítmicas, plantillas y parámetros para capturar, pero limitadas a estas mismas (Song et al., 2024). Modelos posteriores requieren de constante reentrenamiento para el conjunto de datos y presentan dificultades con registros no estructurados o ruidosos. (Guo et al., 2021)

Todos los casos expuestos anteriormente se relacionan con el área de la ciberseguridad, en la que, para localizar la información de anomalías específicas, se implementan técnicas de minado de datos en los registros del sistema para: “detección de anomalías, diagnóstico de fallas, la monitorización del rendimiento y otras operaciones” (Hu et al., 2025).

Las conclusiones que detalla Batoun et al., relaciona los retos que enfrentan los profesionales en el proceso de creación, mantenimiento y uso de estos registros. En el que recalca que uno de los retos más importantes es la toma de decisiones informadas sobre dónde registrar, qué registrar y qué nivel de detalle utilizar en los registros para lograr un equilibrio entre la recopilación de suficientes detalles en los registros y la minimización del rendimiento y la sobrecarga de almacenamiento.

Análisis de restricciones

Este análisis se desarrolla en el contexto del prototipo de diagnóstico inteligente de logs para aplicaciones empresariales, desplegado en entornos híbridos (on-premise y nube), con énfasis en el lenguaje Java/Spring e integraciones con servicios externos y modelos de IA.

A partir de este marco se identifican y documentan las restricciones que condicionan el diseño, la implementación, la operación y la validación del sistema:

Legales y normativas

- Protección de datos personales y hábeas data: De acuerdo con la Ley 1581 de 2012 y su Decreto 1377 de 2013, el tratamiento de logs que puedan contener datos personales exige minimización, enmascaramiento, políticas de retención y acceso con trazabilidad. La SIC recomienda, además, gestionar y documentar incidentes de seguridad y accesos, según su guía oficial

- Cumplimiento institucional y sectorial (seguridad de la información, continuidad del negocio, gestión de incidentes): Para el cumplimiento interno y sectorial se adoptan controles de acceso por rol (RBAC) y auditoría del repositorio de eventos, en línea con las obligaciones de seguridad de la Ley 1581 y su Decreto 1377 y las buenas prácticas de la SIC para gestión y registro de incidentes.

- Propiedad y custodia de la evidencia: Cuando los logs deban preservarse como evidencia técnica, se garantiza su integridad conforme a la Ley 527 de 1999 (mensajes de datos con valor probatorio) y se aplican principios de cadena de custodia conforme a la Ley 906 de 2004 y el manual de la Fiscalía.

Técnicas

- Compatibilidad e integración: fuentes heterogéneas (Bases de datos, contenedores, sistemas legados) con formatos distintos. Se requieren normalización (parsing/plantillas) y correlación temporal.
- Capacidad y desempeño: volumen esperado mayor o igual a 10.000 líneas/min; latencia objetivo menor o igual a 5–10 segundos para detección y notificación; dimensionamiento de almacenamiento e índices (rotación, particionado por tiempo).
- Disponibilidad: despliegue sin punto único de fallo (workers de ingesta, colas/streaming, réplicas del índice). Estrategias de backpressure y reintentos idempotentes.
- Modelo de IA: necesidad de dataset representativo, etiquetado mínimo y reentrenamiento/actualización. Riesgo de deriva del modelo ante cambios en las aplicaciones que generan logs.
- Dependencias externas: límites de APIs/servicios (correo/chatops) y restricciones de red/firewall en entornos gubernamentales o corporativos.

Económicas y financieras

- Costos de infraestructura (on-premise y/o nube) para cómputo, almacenamiento, licencias del stack de búsqueda/observabilidad y monitoreo.
- Costo de entrenamiento/ejecución del modelo (CPU/GPU) y del pipeline de ingesta e índices a medida que crece el volumen de datos.
- Sostenibilidad: presupuesto para operación 24/7, copias, retención, archivado y soporte.

Salud y seguridad ocupacional

- Procedimientos de operación segura en centros de datos y manejo de incidentes; seguridad digital y gestión de turnos para evitar errores humanos.

Sociales y culturales

- Adopción por equipos de soporte/operaciones: curva de aprendizaje para interpretar alertas/explicaciones del sistema; se requieren guías de uso y capacitación.
- Gestión del cambio: evitar dependencia exclusiva del sistema (*human-in-the-loop*) y documentar lecciones aprendidas.

Políticas y gubernamentales / organizacionales

- Alineación con planes de TI, políticas de continuidad/DRP y niveles de servicio (SLA) exigidos por las entidades usuarias.
- Restricciones de contratación, compras y uso de nube pública según políticas institucionales.

Limitaciones del equipo de trabajo

- Capital: recursos financieros vs. alcance de infraestructura y licencias.
- Tecnología: disponibilidad de entornos de prueba similares a producción; acceso a herramientas de observabilidad y a hardware de cómputo para IA.
- Talento: experiencia en ingeniería de datos/observabilidad y en MLOps para mantener modelos y pipelines.

Estrategia de mitigación: Para gestionar las restricciones identificadas se adoptará una arquitectura orientada a eventos y escalable.

- Control de acceso basado en roles (RBAC) y enmascaramiento/minimización de datos en logs.
- Políticas de retención por niveles (caliente → tibio → archivo) con rotación y borrado seguro.
- Dimensionamiento mediante pruebas de carga y ajuste continuo de capacidad;
- Un ciclo de vida del modelo con monitoreo de deriva, versionamiento y reentrenamiento programado;
- Plan de adopción que incluya capacitación, manuales de uso y operación *human-in-the-loop*.

En la tabla 3, se muestra un resumen de estas restricciones identificadas y su respectivo impacto en los requerimientos funcionales y no funcionales:

Tabla 3.

Tabla de restricciones identificadas y su relación al impacto descrito en los requerimientos funcionales y no funcionales.

Restricción	*Impacto en RF/RNF*	*Mitigación/táctica*	*Evidencia por entregar*
Datos personales en logs (Legales)	RNF-002 Seguridad; RF-005 Historial consultable	Anónimo/mascar datos, RBAC, auditoría de accesos	Política de enmascaramiento, bitácora de auditoría
Heterogeneidad de fuentes (Técnicas)	RF-001 Ingesta RNF-004 Interoperabilidad	Parsing/plantillas, correlación temporal, contratos de datos	Catálogo de plantillas y pruebas de correlación
Capacidad/latencia (Técnicas)	RNF-001: Tiempo de respuesta RNF-003: Escalabilidad	Escalado horizontal, colas/streaming, particionado por tiempo	Pruebas de carga con métricas de ≥10k lpm y ≤10 s

Deriva del modelo (IA)	RF-002 Detección de anomalías; RNF-005 Modularidad/ observabilidad	Monitoreo de deriva, reentrenamiento programado	Registro de versiones del modelo y métricas por versión
Restricciones de nube/políticas	RNF-004 Interoperabilidad; despliegue	Alternativas on-prem, redes privadas, acuerdos de servicio	Plan de despliegue y matriz de cumplimiento
Capital y talento (Equipo)	Viabilidad del prototipo y piloto	Alcance por fases, capacitación, alianzas	Plan de proyecto, cronograma y plan de formación

Nota: Elaborado por autor

Marco Metodológico

Tipo de investigación

El presente proyecto se enmarca en una investigación aplicada, dado que su propósito es la construcción de un prototipo funcional con un fin práctico y de utilidad tecnológica. Según Hernández Sampieri et al. (2014), la investigación aplicada se orienta a la solución de problemas específicos e inmediatos. De acuerdo con el Manual de Frascati OECD (2015), se describe que la investigación aplicada se orienta principalmente hacia un objetivo o meta específico y práctico (p. 45). En este caso, la problemática se centra en la necesidad de optimizar la gestión y el análisis de *logs* empresariales mediante inteligencia artificial.

El carácter de la investigación es exploratorio-descriptivo. Por un lado, es exploratorio porque indaga en un campo en constante evolución (el uso de agentes de IA para el procesamiento de registros de sistemas) y en el que aún no existe una aplicación consolidada en entornos académicos locales. Por otro lado, es descriptivo porque pretende detallar el diseño y desarrollo de un prototipo, documentando sus componentes técnicos y las decisiones metodológicas adoptadas (Dankhe (1986) citado por Hernández Sampieri et al. (2014)).

Enfoque metodológico

La investigación se desarrollará bajo un enfoque mixto, que combina la perspectiva cuantitativa con la cualitativa, lo cual es coherente con lo planteado por (Creswell & Creswell, 2017), quien sostiene que este tipo de enfoque “integra datos numéricos y narrativos con el propósito de obtener una comprensión más completa del fenómeno estudiado” (pg 52):

- Desde la perspectiva cuantitativa, se emplearán métricas derivadas del uso de la metodología Kanban, tales como tiempos de ciclo y número de tareas completadas, lo que permitirá medir objetivamente el avance del proyecto.
- Desde la perspectiva cualitativa, se documentará la experiencia del equipo en el desarrollo del prototipo mediante cortas bitácoras elaboradas en las reuniones de seguimiento (*Team Sync*), con el fin de interpretar las dinámicas de trabajo, los obstáculos encontrados y las lecciones aprendidas.

Metodología de desarrollo del proyecto: Kanban

La elección de la metodología Kanban responde a la necesidad de implementar un esquema de trabajo ágil que sea flexible, visual y de baja carga administrativa, lo cual resulta especialmente pertinente para este proyecto académico de prototipado. A diferencia de metodologías ágiles más estructuradas como Scrum, Kanban no requiere la definición estricta de roles ni la planificación de iteraciones cerradas, sino que permite un flujo continuo de tareas adaptado a la capacidad del equipo (Martins, 2025)(Radlgan 2025).

En la siguiente tabla, se describe brevemente la comparación de Scrum y Kanban:

Figura 1.

Tabla comparativa Scrum vs Kanban

	Scrum	Kanban
Metodología de publicación	Sprints de longitud fija periódicos (por ejemplo, dos semanas)	Flujo continuo
Roles	Propietario del producto, experto en scrum, equipo de desarrollo	Entrega continua o a discreción del equipo
Métricas clave	Velocidad	Duración del ciclo
Filosofía de cambios	Los equipos deben evitar cambios en la previsión durante el sprint. De lo contrario, se sacrifica el aprendizaje sobre la estimación.	Los cambios pueden suceder en cualquier momento.

Tomado de Kanban | Atlassian

En este contexto, Kanban es esencial para este proyecto porque:

- Facilita la visualización del progreso mediante estos tableros de fácil comprensión.
- Permite identificar cuellos de botella en el desarrollo a través del control del trabajo en progreso.
- Se adapta de manera eficiente a equipos pequeños y con recursos limitados.
- Genera evidencias tangibles (capturas del tablero, métricas de flujo) que fortalecen la documentación del proceso.

El proyecto adopta estos componentes de Kanban, de la siguiente forma:

- *Tablero Kanban*: dividido en columnas básicas (Por hacer, En progreso, Terminado), que reflejan el estado de las tareas en todo momento.
- Tarjetas de tareas: cada tarjeta representa una actividad vinculada a los objetivos específicos y las subtareas que se generen de estos, lo que permite mantener la trazabilidad entre la planificación y la ejecución.
- Límites de WIP: se aplicarán restricciones al número de tareas simultáneamente en ejecución para fomentar la finalización de actividades antes de iniciar nuevas.
- Revisiones periódicas: se realizarán sesiones de revisión del tablero (semanales o quincenales), donde el equipo evaluará el avance y ajustará prioridades.
- Tomando referencia de Scrum, solo se incluirán los conceptos de Sprint para la organización de los tiempos del backlog y el Sprint Retrospective para la inspección de los procesos, las personas, las herramientas y la planificación de las mejoras que se llevarán a cabo en el siguiente Sprint. (Scrum.org, s/f)

La herramienta digital que se ha estipulado para el desarrollo de la visualización del *Tablero Kanban* será Trello.com.

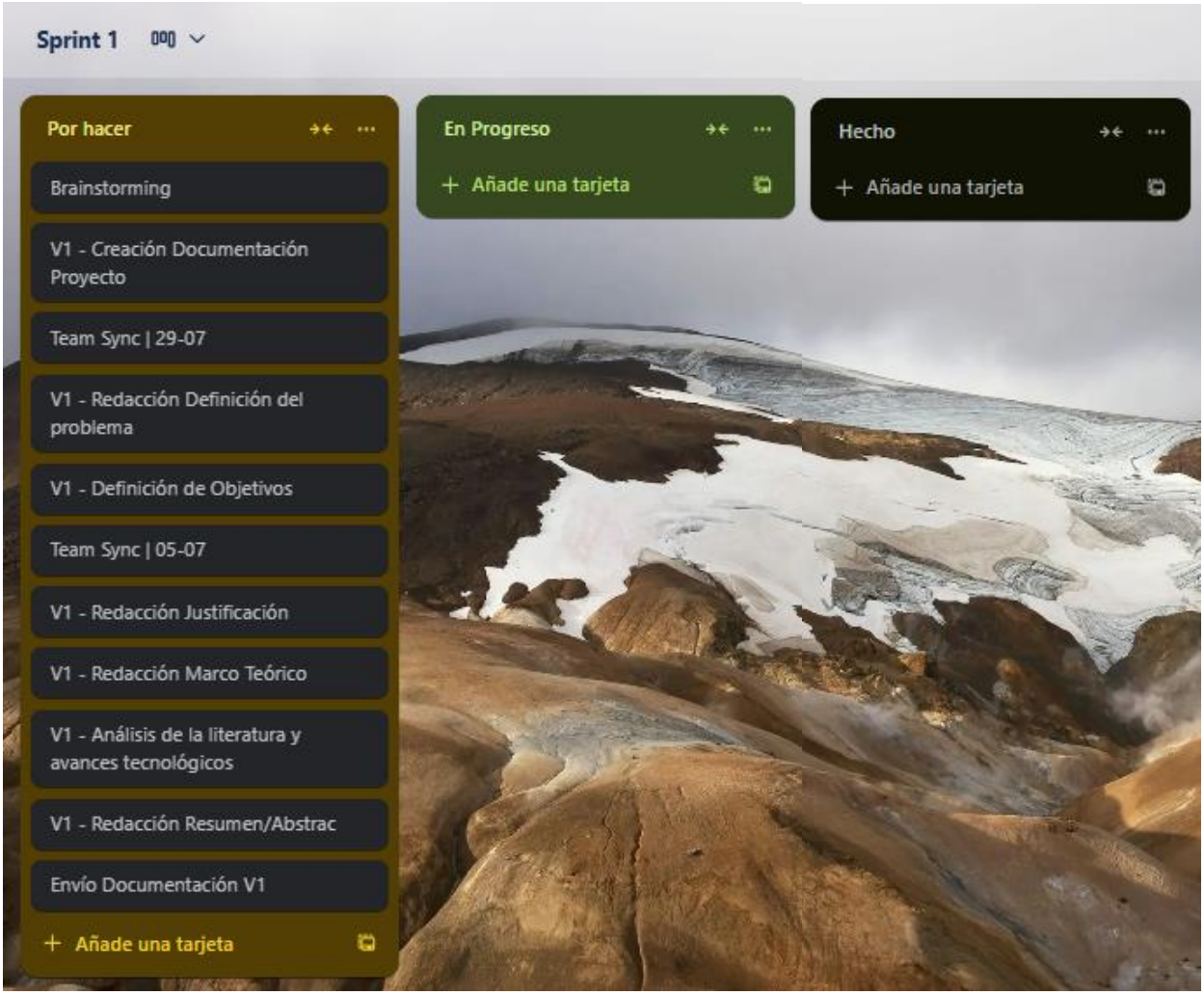
Desarrollo de la metodología

En la primera fase del desarrollo se ha estipulado la construcción del backlog inicial del Proyecto que consiste en la identificación y descomposición de los objetivos específicos en tareas definidas.

Dichas tareas se representan en las tarjetas dentro del tablero y se clasifican en la columna *Por hacer,* con el objetivo de establecer un mapa visual del trabajo pendiente. Tal y como se muestran en la **Figura 1**:

Figura 2.

Tablero Kanban versión preliminar de las tareas pendientes en la columna *Por Hacer* en el Sprint 1



Tomado de tablero [Sprint 1 | Trello]

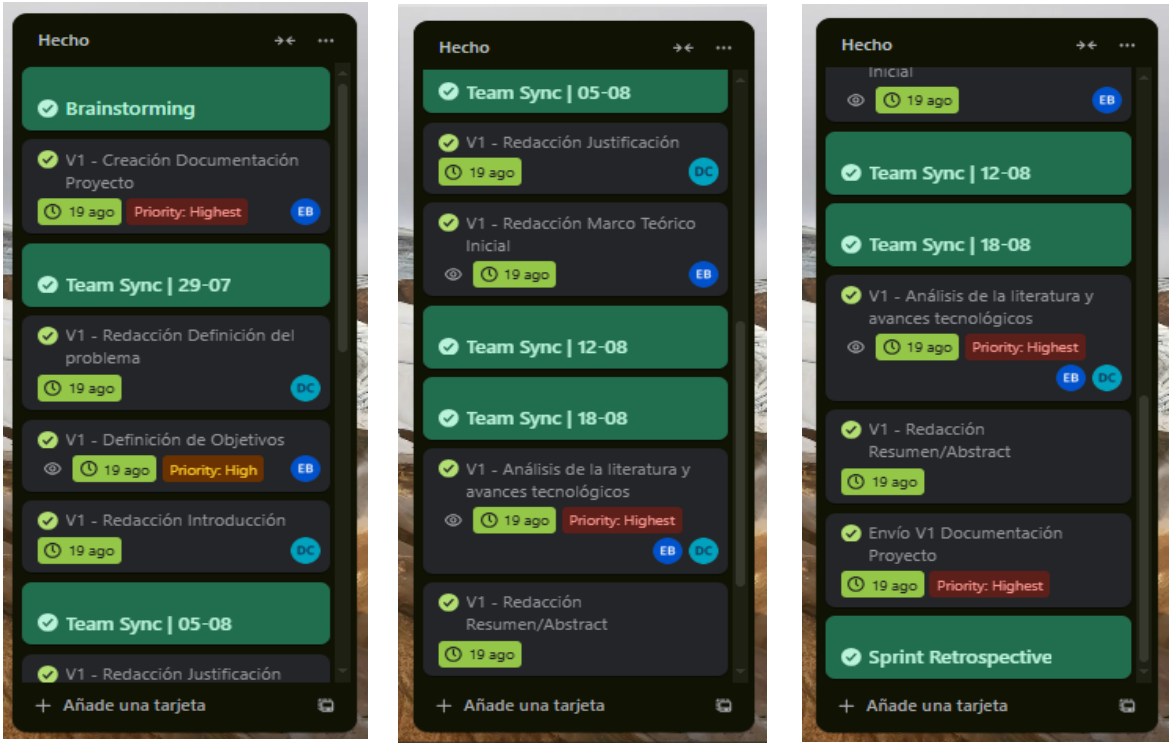
A medida que las tareas son atendidas por el equipo, las tarjetas se trasladan a la columna *En Progreso.* En esta fase interactiva, se mantendrá continuo y flexible, lo que permitirá iniciar nuevas actividades en la medida en la que se liberen capacidades, siempre respetando los límites de trabajo en progreso (WIP). Durante esta fase se desarrollan los componentes del prototipo de manera modular, así como la respectiva gestión del proyecto.

Semanalmente, se han establecido revisiones del tablero con el equipo (*Team Sync*), que tienen como objetivo verificar el estado de las tareas, identificar bloqueantes y/o cuellos de botella y plantear ajustes. Como evidencia académica, se generarán bitácoras de seguimiento que recojan estos objetivos.

Para las semanas de trabajo resumidas para el Sprint 1, la siguiente Figura 2 representa las tarjetas de tareas en *Hecho:*

Figura 3.

Tablero Kanban versión final de las tareas completadas en la columna *Hecho* en el Sprint 1



Tomado de tablero [Sprint 1 | Trello]

Como se puede apreciar, las tareas del Sprint 1 han sido enfocadas a la gestión del proyecto (al desarrollo de los objetivos, de la introducción, del marco teórico, etc.). Para el desarrollo del Sprint 2, se han diseñado y en el momento de la Figura 3, la ejecución de las tareas relacionadas al prototipo de la solución tecnológica que se ha propuesto:

Figura 4.

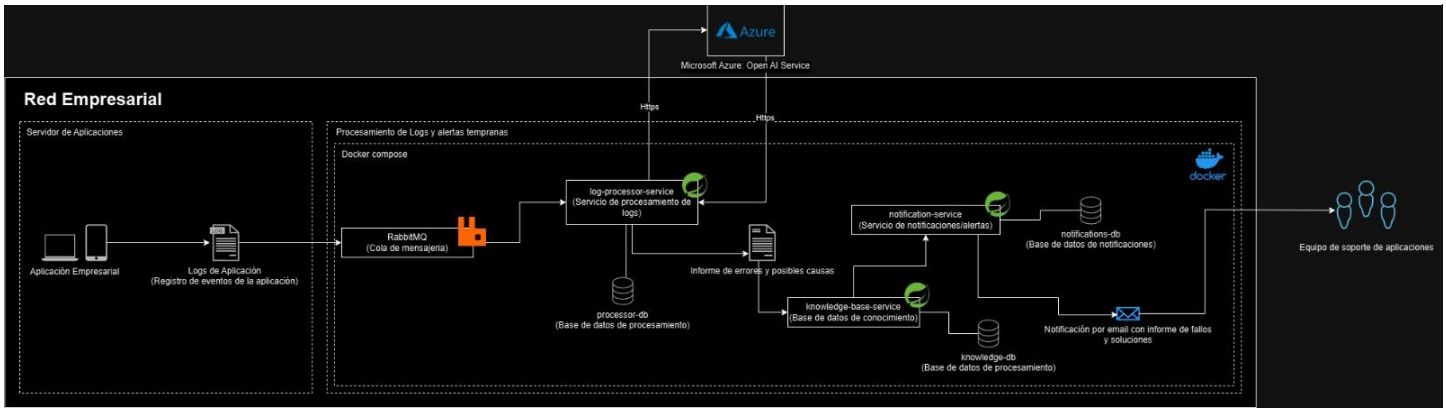
Tablero Kanban versión preliminar de las tareas en curso de la columna *En Progreso* del Sprint 2

Tomado de tablero Sprint 1 | Trello

Otro de las MVP significativos del Sprint 2, es el diseño preliminar de la Arquitectura de Software del sistema distribuido, que se muestra a continuación en la Figura 4:

Figura 5.

Esquema preliminar de la arquitectura de software del sistema distribuido



Nota: Elaborado por autor

La metodología que usó para el desarrollo del sistema distribuido es la arquitectura orientada a servicios (SOA). Este diagrama de arquitectura describe el comportamiento ideal del sistema distribuido, con el siguiente funcionamiento de soporte a una aplicación empresarial en producción:

- Cuando la aplicación presenta algún error, emite su descripción a una cola de mensajería.
- El error se procesa y, con la ayuda de Amazon Bedrock (servicio de cómputo en la nube de AWS), se obtiene un informe de las posibles causas y soluciones.
- En función de la retroalimentación de la IA, se genera un informe que se guarda en una base de datos de conocimiento para el equipo de desarrollo/soporte de la organización.
- Posteriormente, se emite una notificación con el informe adjunto por correo al equipo de desarrollo/soporte o a las personas interesadas.

En el transcurso del Sprint 3, con la arquitectura preliminar y la metodología definidas en el Sprint 2, el enfoque principal fue las tareas relacionadas a la herramienta prototipo: construir y consolidar los componentes críticos del prototipo y pruebas internas relacionadas a la validación de su funcionalidad mínima. Por otro lado, el último Sprint tuvo como carácter la consolidación de un E2E funcional, empaquetado de la solución (contenedores, scripts de arranque, dataset de ejemplo), junto con la preparación de un DEMO reproducible. También el diseño del plan de implementación que incluye la estrategia de plan de pruebas y el diseño de Pruebas de Aceptación de Usuario (UAT), para la parte de ambientes QA, Staging y Producción.

Resultados

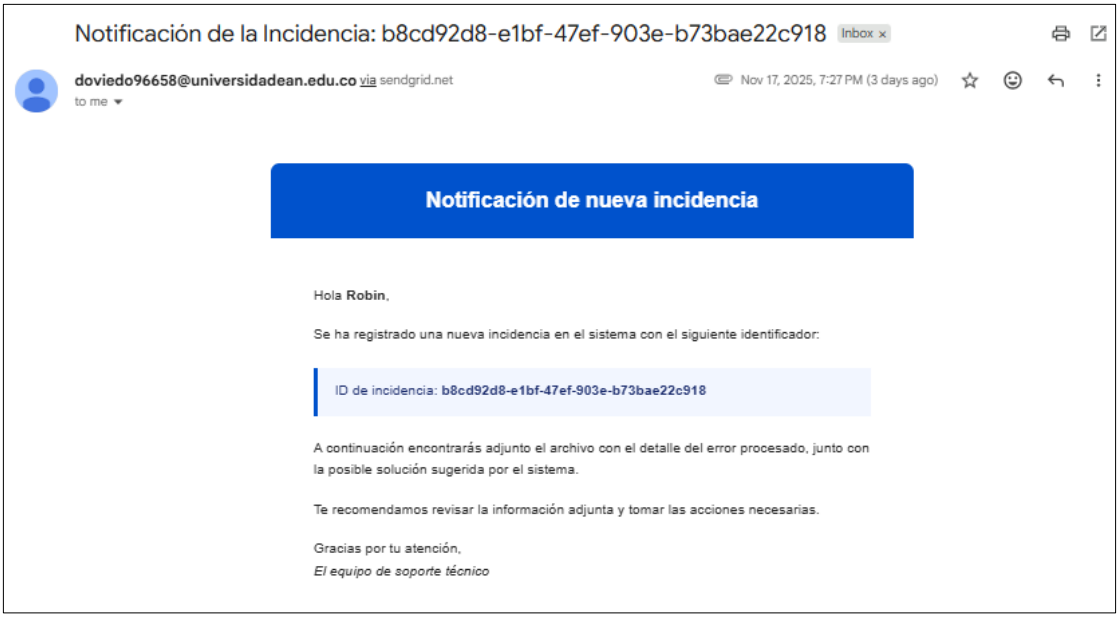
El desarrollo del prototipo de sistema distribuido para la gestión de logs en aplicaciones empresariales permitió validar el objetivo central del proyecto: desarrollar la solución tecnológica para gestionar y analizar logs, que genere una base de conocimiento que apoye los procesos asociados a la planificación y recuperación ante desastres (DRP).

Los resultados más relevantes se resumen a continuación:

- *Cumplimiento funcional:* El prototipo logró recibir y almacenar registros de error en tiempo real. Mediante un agente de inteligencia artificial (IA), se realizan el análisis y el procesamiento de estos registros que generan sugerencias para soluciones, con el desarrollo de un repositorio de conocimiento (KB), destinado a almacenar los logs procesados, las interpretaciones del agente y las recomendaciones generadas.

Figura 6.

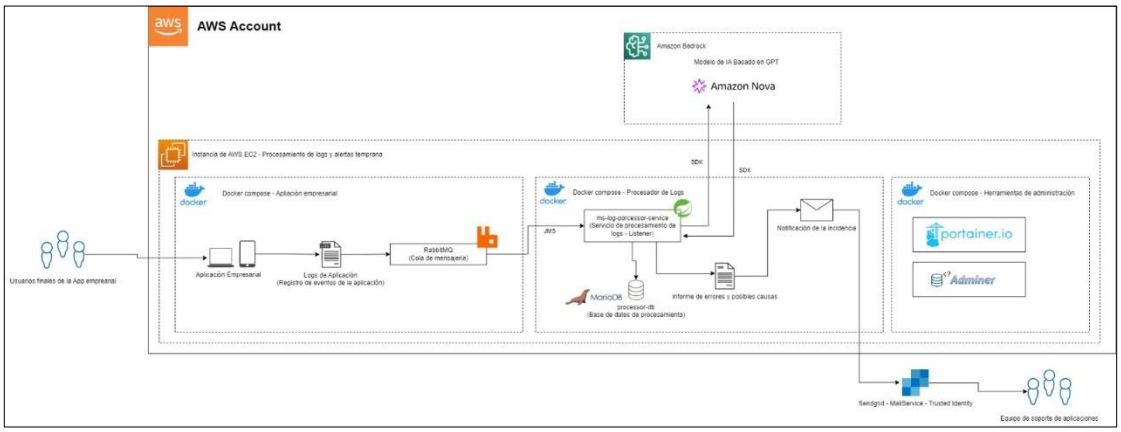
Notificación de correo electrónico luego de la gestión del registro de evento (Log)



- *Eficiencia técnica*: La arquitectura orientada a servicios demostró la integración de servicios independientes para la ingesta de logs, su procesamiento, la generación de respuestas y el almacenamiento de información, permitiendo manejar mayores volúmenes de eventos mediante el incremento controlado de instancias del backend.

Figura 7.

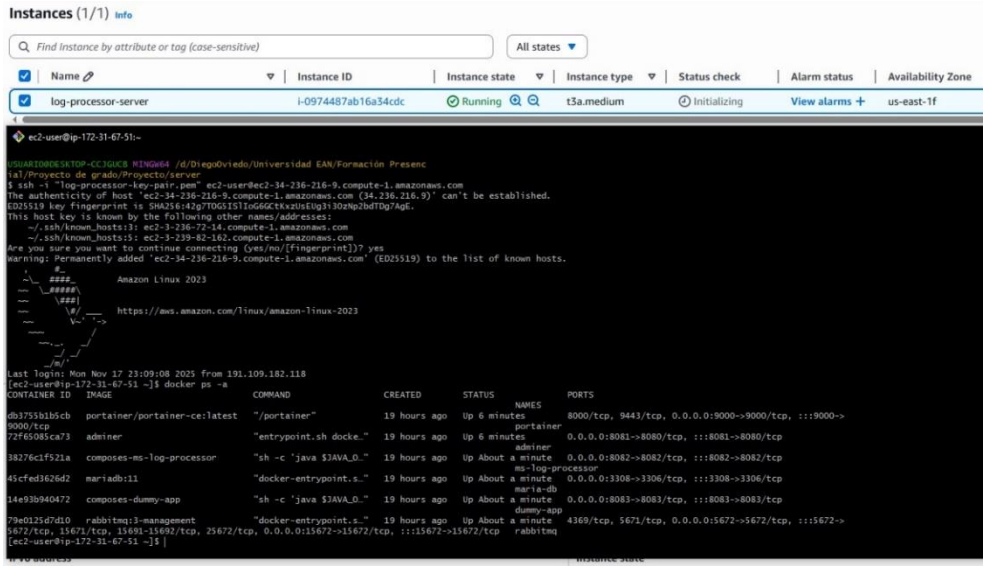
Arquitectura final para ProypeLog



- *Integración tecnológica:* El sistema se integró exitosamente con servicios de mensajería y monitoreo (servicios de AWS), cumpliendo los requerimientos de alta disponibilidad y seguridad.

Figura 8.

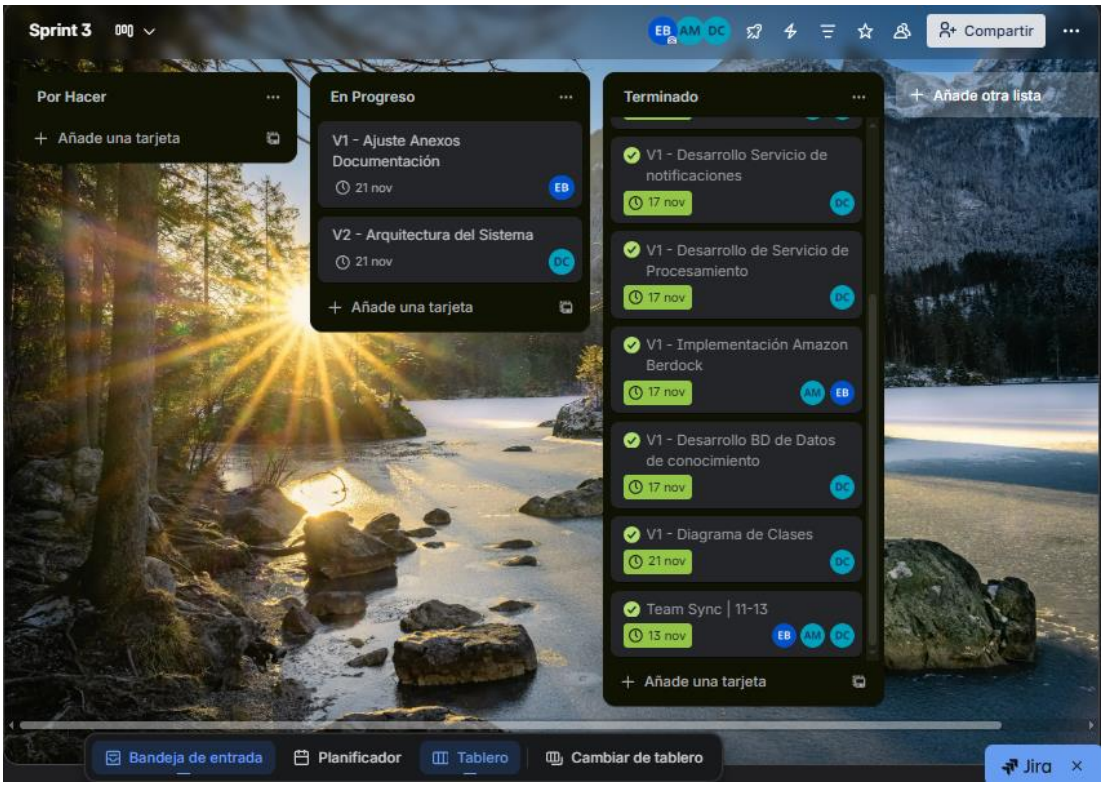
Consola de versionamiento de código GIT y servicio de AWS para ProtypeLog



- *Desarrollo ágil:* Mediante el uso de la metodología ágil Kanban, el desarrollo del proyecto facilitó la gestión progresiva de la herramienta, permitiendo organizar las tareas, priorizar funcionalidades y mantener la trazabilidad del proceso.

Figura 9.

Tablero que relaciona tareas diseñadas y ejecutadas para el Sprint 3 del proyecto



En conjunto, estos resultados demuestran que el prototipo cumple con los objetivos técnicos planteados: integra un agente de inteligencia artificial, que analiza los logs de forma automatizada y genera un repositorio de conocimiento que contribuye al propósito general de dar soporte a los procesos relacionados a la recuperación ante desastres (DRP).

Aunque el sistema requiere ampliación y pruebas más extensas para validar su aplicabilidad en escenarios empresariales reales, constituye un aporte significativo en términos conceptuales, técnicos y metodológicos dentro del alcance académico definido

Análisis de Costos

En un proyecto de ingeniería, como el desarrollo de un prototipo de sistema distribuido para la gestión inteligente de logs, no basta con demostrar su viabilidad técnica; también es necesario evidenciar que la solución puede implementarse bajo condiciones económicas razonables y que resulta sostenible en el tiempo. El presente análisis tiene como propósito establecer la viabilidad económica del desarrollo e implementación de dicho prototipo, el cual se enfoca en la integración con servicios de inteligencia artificial.

Para ello, se realiza un análisis de costos que considera:

- Costos directos: asociados de forma inmediata al desarrollo del prototipo.
- Costos indirectos: gastos de apoyo necesarios para ejecutar el proyecto.
- Costos generales (overhead): costos administrativos, de gestión y contingencias.

Supuestos Generales

- Duración del proyecto: 4 meses (desarrollo, validación y entrega).
- Equipo técnico: pequeño y especializado (sin interfaz gráfica).
- Enfoque técnico: backend distribuido con Spring Boot, mensajería asíncrona, almacenamiento en SQL y consumo de servicio de IA.
- Moneda: pesos colombianos (COP).
- Entorno de ejecución: infraestructura híbrida (máquinas virtuales y servicios cloud).

Costos Directos

1. Costos de personal:

   El componente más relevante del costo del proyecto corresponde al tiempo de dedicación del equipo técnico. En la Tabla 4 se desglosa el cálculo considerando salario mensual, porcentaje de dedicación y número de meses.

Tabla 4.

Tabla para el análisis en costos de persona2020)

Rol	*Descripción*	*Dedicación*	*Meses*	*Salario mensual (COP)*	*Costo total (COP)*
Arquitecto de software / líder técnico	Diseña la arquitectura distribuida, define APIs, flujos de mensajería, y modelo de IA	50%	4	$ 6.000.000	$ 12.000.000
Desarrollador backend (Java/Spring Boot)	Implementa la lógica de negocio, controladores, conexión con colas y base de datos	100%	4	$ 5.000.000	$ 20.000.000
Ingeniero DevOps / Infraestructura	Configura contenedores, despliegues y seguridad de servicios	50%	3	$ 5.000.000	$ 7.500.000
Ingeniero QA / Analista de pruebas	Diseña pruebas de rendimiento y validación de detección de errores	50%	3	$ 4.000.000	$ 6.000.000
Total, costos de personal					**$ 45.500.000**

Nota: Elaborado por autor

Este valor incluye:

- Diseño de la arquitectura distribuida (microservicios, colas de mensajería, componentes de ingesta y análisis de logs).
- Desarrollo de servicios backend en Java / Spring Boot para la recepción de eventos y comunicación con el agente de IA.
- Definición y entrenamiento inicial del modelo de IA / prompts para diagnóstico de errores.
- Configuración de infraestructura en la nube (pipelines, contenedores, monitoreo).
- Diseño y ejecución de pruebas funcionales, de rendimiento y de regresión.

2. Infraestructura en la nube y servicios externos

Para la ejecución y pruebas del prototipo se utiliza infraestructura proporcionada por Amazon Web Services (AWS), la cual ofrece flexibilidad, disponibilidad y escalabilidad para entornos distribuidos. El proyecto hace uso de una instancia de cómputo en la nube (EC2) con capacidad intermedia y un volumen de almacenamiento adicional para respaldos y registros del sistema.

En la siguiente tabla se presenta el detalle de los costos asociados:

Tabla 5.

Costos infraestructura Cloud (AWS Services)

Servicio	Descripción	Costo mensual (USD)	Meses	Costo total (COP)
Amazon EC2 (8 GB RAM, 4 vCPUs)	Instancia principal del backend y motor de análisis	43	4	$ 722.400
Amazon EBS (almacenamiento adicional)	Espacio adicional para almacenamiento de logs y copias de seguridad	5	4	$ 84.000
Total infraestructura Cloud (AWS)				**$ 806.400**

Nota: Elaborado por autor

El uso de una instancia Amazon EC2 con almacenamiento EBS resultó suficiente para soportar la carga de trabajo del prototipo, garantizando estabilidad, control de costos y disponibilidad continua durante el ciclo de pruebas.

3. Herramientas, licencias y equipos:

Aunque gran parte de las herramientas de desarrollo utilizadas son de código abierto (por ejemplo, VS Code, IntelliJ Community, Spring Boot, Git, Trello), se considera una amortización parcial del hardware y algunos servicios complementarios:

Tabla 6.

Tabla para el análisis en costos herramientas digitales, licencias en programas y equipos de cómputo

Concepto	*Costo estimado (COP)*
Licencias de desarrollo (IntelliJ IDEA, Postman Pro, Docker Desktop Pro, entre otros)	$ 1.200.000
Amortización de equipos (2 estaciones de desarrollo de $5.000.000, uso de 6 meses)	$ 1.700.000
Servicios de repositorio y CI/CD (GitHub Pro, Trello Premium)	$ 600.000
Dominio, SSL y DNS (anual)	$ 400.000
Total herramientas y equipos	**$ 3.900.000**

Nota: Elaborado por autor

Costos Indirectos

Para los costos indirectos, se establecieron costos con un porcentaje de holgura, considerando la máxima productividad posible durante el periodo de tiempo definido (4 meses). En la tabla 6 se encuentran los ítems definidos como costos indirectos:

Tabla 7.

Tabla para costos indirectos

Concepto	*Detalle*	*Costo estimado (COP)*
Servicios públicos y energía	Consumo adicional por trabajo en oficina / laboratorio	$ 2.400.000
Conectividad e infraestructura de red	Internet de alta disponibilidad, routers, switches	$ 4.800.000
Transporte y reuniones presenciales	Desplazamientos del equipo para sesiones de trabajo, revisión con stakeholders	$ 1.200.000
Capacitaciones y material de apoyo	Cursos cortos, documentación, talleres en IA / observabilidad	$ 1.500.000
Total costos indirectos		**$ 9.900.000**

Nota: Elaborado por autor

Estos costos garantizan:

- Condiciones mínimas para el **trabajo colaborativo** y la operación de los entornos de prueba.
- Actualización del equipo en **herramientas de IA aplicada a logs, DevOps y observabilidad**, que son claves en el éxito del prototipo.

Conclusiones

El desarrollo del prototipo del sistema para la gestión inteligente de registros permitió validar la viabilidad técnica del enfoque propuesto, demostrando que es posible integrar técnicas de inteligencia artificial y arquitecturas distribuidas con orientación a servicios para mejorar la gestión y el análisis de eventos en aplicaciones empresariales. Durante la construcción y las primeras etapas del sistema, fue posible implementar una arquitectura distribuida básica, integrar servicios de procesamiento de registros y realizar pruebas iniciales de análisis automatizado de logs con apoyo de modelos de lenguaje.

Los resultados preliminares evidencian que el sistema es capaz de recibir registros, procesarlos y generar respuestas iniciales basadas en el contenido, demostrando la viabilidad técnica de combinar procesamiento de datos, modelos de IA para apoyar la interpretación de eventos del sistema. A su vez, la arquitectura propuesta permitió separar funciones como ingesta de datos, análisis y respuesta, lo cual favorece una visión modular del sistema y sienta las bases para futuras iteraciones.

Adicionalmente, se implementaron los mecanismos iniciales para almacenar las interacciones y hallazgos relevantes en una base de conocimiento, con el propósito de conservar información útil sobre patrones de error y respuestas generadas. Este componente busca contribuir, en fases posteriores del proyecto, al fortalecimiento del sistema mediante consulta histórica y aprovechamiento incremental del conocimiento registrado.

Desde el punto de vista económico, el análisis de costos sugiere que la solución es viable en un escenario de prototipado controlado y que su escalabilidad futura dependerá principalmente de los recursos destinados a la infraestructura en la nube y a las capacidades

avanzadas de la Inteligencia Artificial (IA), siendo el personal técnico y servicios tecnológicos especializados la mayor parte de la inversión se destinada.

El uso del marco metodológico Kanban facilitó la planificación gradual del desarrollo, permitiendo visualizar el progreso del proyecto, priorizar tareas según necesidades emergentes y ajustar el flujo de trabajo durante la construcción del prototipo. La aplicación de esta metodología fue especialmente útil en un contexto académico, al ofrecer claridad sobre las fases del proceso y permitir documentar de manera estructurada los avances e iteraciones realizadas.

No obstante, se reconoce que los resultados obtenidos se circunscriben a un entorno experimental y controlado. El prototipo ha alcanzado un nivel funcional que permite demostrar el concepto académico, pero aún requiere fases posteriores de ampliación de casos de prueba, optimización del procesamiento y validación de la base de conocimiento en contextos más diversos.

En términos generales, se puede concluir que el prototipo en construcción representa un avance significativo hacia la automatización del análisis de logs con apoyo de IA en el contexto académico, pero aún se requieren etapas posteriores de experimentación, refinamiento y ampliación de casos de prueba para determinar su alcance final dentro del contexto empresarial y tecnológico planteado.

Referencias

American Psychological Association. (2020). *Publication manual of the American Psychological Association : the official guide to APA style*. American Psychological Association.

AWS.com. (2025). *¿Qué es un LLM (modelo de lenguaje de gran tamaño)?* Amazon AWS. https://aws.amazon.com/es/what-is/large-language-model/

Chuah, E., Jhumka, A., Malek, M., & Suri, N. (2022). A Survey of Log-Correlation Tools for Failure Diagnosis and Prediction in Cluster Systems. *IEEE Access*, *10*, 133487–133503. https://doi.org/10.1109/ACCESS.2022.3231454

Creswell, J. W., & Creswell, J. D. (2017). *Research design. Qualitative, quantitative, and mixed methods approaches* (SAGE Publications, Ed.; 5a ed.). SAGE Publications.

Dynatrace. (2025, octubre 15). *Log processing*. https://docs.dynatrace.com/docs/analyze-explore-automate/logs/lma-log-processing

Flinders, M., & Smalley, I. (2025). *What is log analysis?* IBM. https://www.ibm.com/think/topics/log-analysis

Fortinet.com. (2023, abril 27). *Disaster Recovery: Definition, Types & Planning*.

Google Cloud. (2025a). *¿Qué es la arquitectura de nube?* https://cloud.google.com/learn/what-is-cloud-architecture

Google Cloud. (2025b). *¿Qué es la computación en la nube?* https://cloud.google.com/learn/what-is-cloud-computing

Guo, H., Yuan, S., & Wu, X. (2021). LogBERT: Log Anomaly Detection via BERT. *Proceedings of the International Joint Conference on Neural Networks*, *2021-July*. https://doi.org/10.1109/IJCNN52387.2021.9534113

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). *Metodología de la investigación* (6a edición.). McGraw Hill.

Hu, J., Long, L., Sui, H., Gu, Z., & Zheng, G. (2025). CFTL: System Log Parsing Method Driven from Clustering According to First Token and Length for Anomaly Detection. *Applied Sciences (Switzerland)*, *15*(4). https://doi.org/10.3390/APP15041740

IBM.com. (2025). *What is a disaster recovery plan (DRP)?* What is a disaster recovery plan (DRP)?

Ji, X., Zhang, L., Zhang, W., Peng, F., Mao, Y., Liao, X., & Zhang, K. (2025). LEMAD: LLM-Empowered Multi-Agent System for Anomaly Detection in Power Grid Services. *Electronics 2025, Vol. 14, Page 3008*, *14*(15), 3008. https://doi.org/10.3390/ELECTRONICS14153008

Lee, J., Jeong, Y., Han, T., & Lee, T. (2025). LogRESP-Agent: A Recursive AI Framework for Context-Aware Log Anomaly Detection and TTP Analysis. *Applied Sciences 2025, Vol. 15, Page 7237*, *15*(13), 7237. https://doi.org/10.3390/APP15137237

Martins, J. (2025, enero 19). *What is Kanban? A Beginner's Guide for Agile Teams*. Asana.com. https://asana.com/en/resources/what-is-kanban

OECD. (2015). *Frascati Manual 2015 : Guidelines for Collecting and Reporting Data on Research and Experimental Development*. OECD Publishing.

Peddinti, S. R., Katragadda, S. R., Pandey, B. K., & Tanikonda, A. (2023). Utilizing Large Language Models for Advanced Service Management: Potential Applications and Operational Challenges. *SSRN Electronic Journal*. https://doi.org/10.2139/SSRN.5119925

Radlgan, D. (2025). *Kanban*. Atlassian.com. https://www.atlassian.com/agile/kanban

RedHat.com. (2025, enero 29). *Understanding automation*. Red Hat. https://www.redhat.com/en/topics/automation

Scrum.org. (s/f). *Introduction to the Sprint Retrospective*. Scrum.org. Recuperado el 20 de septiembre de 2025, de https://www.scrum.org/resources/introduction-sprint-retrospective

Slack.com. (2024, octubre 27). *Knowledge Base: comparte el conocimiento de tu empresa*. https://slack.com/intl/es-es/blog/productivity/knowledge-base-comparte-el-conocimiento-de-tu-empresa

Song, C., Ma, L., Zheng, J., Liao, J., Kuang, H., & Yang, L. (2024). *Audit-LLM: Multi-Agent Collaboration for Log-based Insider Threat Detection*. https://arxiv.org/pdf/2408.08902

Stryker, C., & Kavlakoglu, E. (2025). *What is artificial intelligence (AI)?* IBM. https://www.ibm.com/think/topics/artificial-intelligence

Zettler, K. (2025, noviembre 21). *What is a distributed system?* Atlassian. https://www.atlassian.com/microservices/microservices-architecture/distributed-architecture

Zhang, M., Chen, J., Liu, J., Wang, J., Shi, R., & Sheng, H. (2022). *LogST: Log Semi-supervised Anomaly Detection Based on Sentence-BERT*. https://doi.org/10.1109/ICSIP55141.2022.9886069