



Universidad
Nacional
de Loja

Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los de Recursos Naturales No Renovables

Maestría en Ingeniería en Software

Arquitectura de Gestión Centralizada de Logs con OpenSearch e IA en los Sistemas Institucionales de la Universidad Nacional de Loja

Trabajo de Titulación previo, a la
obtención del título de Magíster
en Ingeniería en Software

AUTOR:

Edgar Manuel Macas Loja

DIRECTOR:

Ing. Pablo Fernando Ordoñez Ordoñez, Mg. Sc.

Loja - Ecuador

2025

Certificación

Loja, 18 de marzo de 2025

Ing. Pablo Fernando Ordóñez Ordóñez, Mg. Sc.

DIRECTOR DE TRABAJO DE TITULACIÓN

CERTIFICO:

Que he revisado y orientado todo proceso de la elaboración del Trabajo de Titulación denominado: **Arquitectura de Gestión Centralizada de Logs con OpenSearch e IA en los Sistemas Institucionales de la Universidad Nacional de Loja**, previo a la obtención del título de **Magíster en Ingeniería en Software**, de autoría del estudiante **Edgar Manuel Macas Loja** cédula de identidad Nro. **1104929078**, una vez que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja para el efecto, autorizo la presentación para la respectiva sustentación y defensa.

Ing. Pablo Fernando Ordóñez Ordóñez, Mg. Sc.

DIRECTOR DEL TRABAJO DE TITULACIÓN

Autoría

Yo, **Edgar Manuel Macas Loja**, declaro ser autor del Trabajo de Titulación y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos y acciones legales, por el contenido del mismo. Adicionalmente, acepto y autorizo a la Universidad Nacional de Loja la publicación del Trabajo de Titulación en el Repositorio Digital Institucional - Biblioteca Virtual.

Firma:

Cédula de Identidad: 1104929078

Fecha: 18/03/2025

Correo electrónico: emmacasl@unl.edu.ec

Teléfono: 0968264982

Carta de autorización por parte del autor, para consulta, reproducción parcial o total y/o publicación electrónica de texto completo, del Trabajo de Titulación

Yo, **Edgar Manuel Macas Loja**, declaro ser autor del Trabajo de Titulación denominado: **Arquitectura de Gestión Centralizada de Logs con OpenSearch e IA en los Sistemas Institucionales de la Universidad Nacional de Loja** como requisito para optar el título de **Magíster en Ingeniería en Software**, autorizo al sistema Bibliotecario de la Universidad Nacional de Loja para que con fines académicos muestre la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Titulación que realice un tercero.

Para constancia de esta autorización, suscribo en la ciudad de Loja, a los dieciocho días del mes de marzo, de dos mil veinticinco.

Firma:

Autor: Edgar Manuel Macas Loja

Cédula de Identidad: 1104929078

Fecha: 18/03/2025

Correo electrónico: emmacasl@unl.edu.ec

Teléfono: 0968264982

DATOS COMPLEMENTARIOS:

Director del Trabajo de Titulación: Ing. Pablo Fernando Ordóñez Ordóñez, Mg. Sc.

Dedicatoria

A Dios, por iluminar mi camino y darme la fortaleza para superar cada desafío; a mis padres, cuyo amor, sacrificio y apoyo incondicional han sido el cimiento de mis logros; a mi esposa, mi compañera de vida, por su cariño, paciencia y por ser un pilar fundamental en mi camino profesional; a mis hijos, quienes con su amor y alegría dan sentido y propósito a cada uno de mis esfuerzos y a mis hermanos quienes han sido un apoyo incondicional que permitieron cumplir mis metas.

Edgar Manuel Macas Loja

Agradecimiento

Agradezco a Dios por brindarme la salud, fortaleza y sabiduría necesarias a lo largo de mi vida, permitiéndome alcanzar esta meta y convertirme en profesional. A mis padres, por su amor y apoyo incondicional, y a mi esposa e hijos, quienes han sido mi mayor motivación y respaldo en cada paso de este camino.

Expreso mi gratitud a la Universidad Nacional de Loja, a su cuerpo administrativo y docente por su dedicación y profesionalismo durante mi formación académica; y especialmente al Ing. Pablo Ordóñez, por su valiosa dirección, dedicación e interés en el desarrollo de este Trabajo de Titulación.

Asimismo, agradezco a mis compañeros de trabajo de la Dirección de Tecnologías, por compartir su conocimiento y experiencias que han enriquecido mi aprendizaje y crecimiento profesional. Su colaboración ha sido fundamental en este proceso.

Edgar Manuel Macas Loja

Índice de contenidos

Portada

Certificación.....	ii
Autoría.....	iii
Carta de autorización	iv
Dedicatoria.....	v
Agradecimiento	vi
Índice de contenidos.....	vii
Índice de tablas.....	ix
Índice de figuras	x
Índice de anexos.....	xi
1. Título	1
2. Resumen.....	2
Abstract	3
3. Introducción.....	4
4. Marco Teórico.....	5
4.1. Antecedentes	5
4.2. Tecnología	6
4.2.1. Recolección y procesamiento de logs	6
4.2.2. Almacenamiento y búsqueda	7
4.2.3. Modelos de Inteligencia Artificial aplicados a la gestión de logs	9
4.2.4. Sistemas de mensajería y colas	12
4.2.5. Herramientas de visualización.....	14
4.2.6. Notificaciones y alertas.....	15
4.3. Trabajos Relacionados	16
5. Metodología	18
5.1. Área de estudio.....	18

5.2.	Procedimiento	18
5.2.1.	Objetivo 1: Desarrollar casos de prueba para evaluar la integración de OpenSearch con modelos de IA que recomienden la mitigación de incidencias en los sistemas institucionales, utilizando una metodología experimental.....	19
5.2.2.	Diseñar la arquitectura para la gestión centralizada de logs en los sistemas institucionales de la Universidad Nacional de Loja, mediante un estilo arquitectónico que represente la integración de OpenSearch e inteligencia artificial.	20
5.3.	Recursos.....	22
5.3.1.	Recursos de Hardware y Software	22
6.	Resultados.....	24
6.1.	Objetivo 1: Desarrollar casos de prueba para evaluar la integración de OpenSearch con modelos de IA que recomienden la mitigación de incidencias en los sistemas institucionales, utilizando una metodología experimental.	24
6.1.1.	R1: Configuración del entorno de pruebas	24
6.1.2.	R2: Casos de Prueba	26
6.1.3.	R3: Visualización de Notificaciones y Logs.....	67
6.2.	Objetivo 2: Diseñar la arquitectura para la gestión centralizada de logs en los sistemas institucionales de la Universidad Nacional de Loja, mediante un estilo arquitectónico que represente la integración de OpenSearch e inteligencia artificial.....	71
6.2.1.	R4: Diseño de la Arquitectura.....	71
6.2.2.	Arquitectura Centralización Logs.....	79
7.	Discusión.....	80
	Objetivo 1: Evaluación de Casos de Prueba y Modelos de IA.....	80
	Objetivo 2: Diseñar la arquitectura para la gestión centralizada de logs en los sistemas institucionales de la Universidad Nacional de Loja, mediante un estilo arquitectónico que represente la integración de OpenSearch e inteligencia artificial.	84
8.	Conclusiones.....	85
9.	Recomendaciones.....	87
10.	Bibliografía.....	88
11.	Anexos.....	90

Índice de tablas

Tabla 1. Comparativa de Herramientas para la gestión de logs.	8
Tabla 2. Herramientas LLM para entorno locales.	11
Tabla 3. Comparativa de Herramientas de Sistemas de Mensajería y Colas.	13
Tabla 4. Comparativa de OpenSearch Dashboard y Grafana.	15
Tabla 5. Características de los Modelos LLM.	25
Tabla 6. Caso de Prueba 1 para la aplicación SIAAF.	27
Tabla 7. Caso de Prueba 2 para la aplicación SIAAF.	27
Tabla 8. Caso de Prueba 3 para la aplicación SIAAF.	28
Tabla 9. Caso de Prueba 4 para la aplicación SIAAF.	28
Tabla 10. Caso de Prueba 5 para la aplicación SIAAF.	29
Tabla 11. Casos de prueba aplicación SIAAF.	30
Tabla 12. Caso de Prueba 1 para la aplicación Trámites.	37
Tabla 13. Caso de Prueba 2 para la aplicación Trámites.	38
Tabla 14. Caso de Prueba 3 para la aplicación Trámites.	38
Tabla 15. Caso de Prueba 4 para la aplicación Trámites.	38
Tabla 16. Caso de Prueba 5 para la aplicación Trámites.	39
Tabla 17. Casos de prueba aplicación Trámites.	40
Tabla 18. Caso de Prueba 1 para la aplicación Admisiones.	50
Tabla 19. Caso de Prueba 2 para la aplicación Admisiones.	50
Tabla 20. Caso de Prueba 3 para la aplicación Admisiones.	51
Tabla 21. Caso de Prueba 4 para la aplicación Admisiones.	51
Tabla 22. Caso de Prueba 5 para la aplicación Admisiones.	52
Tabla 23. Casos de prueba aplicación Admisiones.	54
Tabla 24. Comparación de Tiempos de Respuesta SIAAF.	81
Tabla 25. Comparación de Tiempos de Respuesta TRÁMITES.	82
Tabla 26. Comparación de Tiempos de Respuesta TRÁMITES.	83

Índice de figuras

Figura 1. Notificación en Slack de un error crítico en SIAAF con detalles y recomendaciones sobre problemas en la base de datos.....	67
Figura 2. Notificación en Slack de un error en Admisiones relacionado con archivos faltantes y soluciones propuestas.	68
Figura 3. Notificación en Slack de un error crítico en una ruta no encontrada dentro de un sistema Django, con trazas y recomendaciones generadas por el modelo de IA.....	68
Figura 4. Visualización en OpenSearch Dashboards, mostrando gráficos de tendencias de logs, volúmenes de datos y análisis detallado para monitoreo y diagnóstico.	69
Figura 5. Versión inicial de la arquitectura propuesta para la gestión de logs, integrando recolección, procesamiento, visualización y notificaciones.	70
Figura 6. Ubicación de logs de los sistemas institucionales	71
Figura 7. Proceso de recolección de logs con Fluent Bit.....	72
Figura 8. Arquitectura de visualización y almacenamiento con OpenSearch y Grafana.	73
Figura 9. Integración de Apache Kafka para la gestión de logs.....	74
Figura 10. Normalización y Almacenamiento en Caché de Logs Críticos.	75
Figura 11. Integración de modelos de IA en Ollama.	76
Figura 12. Arquitectura de procesamiento y notificación de logs críticos.	77
Figura 13. Arquitectura para la gestión centralizada de logs.....	79

Índice de anexos

Anexo 1. Autorización de acceso a registro de logs.....	89
Anexo 2. Manual de Instalación de herramientas para la gestión de logs.....	90
Anexo 3: Certificación resumen.....	113

1. Título

**Arquitectura de Gestión Centralizada de Logs con OpenSearch e IA en los Sistemas
Institucionales de la Universidad Nacional de Loja**

2. Resumen

Este proyecto aborda la problemática de la gestión descentralizada de logs en los sistemas institucionales de la Universidad Nacional de Loja (UNL), la cual dificulta la detección temprana y resolución de incidencias, impactando negativamente en la seguridad, operatividad y continuidad de los servicios tecnológicos. Para ello, el presente Trabajo de Titulación (TT) tiene como objetivo el diseño de una arquitectura centralizada para la gestión de logs en los sistemas de la UNL, utilizando herramientas de código abierto como OpenSearch e inteligencia artificial (IA).

Para lograr este objetivo, la metodología integró tecnologías modernas como Fluent Bit, Apache Kafka, Redis, OpenSearch y modelos LLM(Large Language Models) de IA como (LLaMA3, Mistral y Gemma2), diseñadas para recolectar, procesar y analizar grandes volúmenes de datos. Los casos de prueba, realizados en entornos controlados, demostraron la capacidad del sistema para centralizar datos, identificar anomalías y proporcionar recomendaciones para la toma de decisiones. Se midió el impacto de la centralización de logs mediante los tiempos de respuesta ante incidencias, mostrando una mejora significativa en comparación con los procesos manuales. En concreto, los tiempos de respuesta y resolución de incidencias fueron menores con la arquitectura propuesta.

La implementación de esta arquitectura centralizada, basada en herramientas de código abierto, no solo mejora la eficiencia operativa y la capacidad de respuesta ante incidencias, sino que también ofrece un marco sostenible y accesible para entornos educativos. Este proyecto aporta a la comunidad académica al demostrar la viabilidad de integrar tecnologías de código abierto e inteligencia artificial para la gestión de logs ofreciendo un marco de referencia para otras instituciones educativas que busquen mejorar sus capacidades de monitoreo y respuesta ante incidentes en sus sistemas tecnológicos.

Palabras Claves: Gestión centralizada de logs, OpenSearch, Fluent Bit, Arquitectura centralizada, Análisis de logs, Inteligencia Artificial, Sistemas institucionales, Procesamiento de datos en tiempo real.

Abstract

This project addresses the issue of decentralized log management in the institutional systems of the Universidad Nacional de Loja (UNL), which hinders early detection and resolution of incidents, negatively impacting the security, operability, and continuity of technological services. To tackle this, the present Graduation Project (TT) aims to design a centralized architecture for log management in UNL's systems using open-source tools such as OpenSearch and artificial intelligence (AI).

To achieve this objective, the methodology integrated modern technologies like Fluent Bit, Apache Kafka, Redis, OpenSearch, and Large Language Models (LLMs) such as LLaMA3, Mistral, and Gemma2, designed to collect, process, and analyze large volumes of data. The test cases, conducted in controlled environments, demonstrated the system's ability to centralize data, identify anomalies, and provide recommendations for decision-making. The impact of log centralization was measured based on incident response times, showing a significant improvement compared to manual processes. Specifically, incident response and resolution times were shorter with the proposed architecture.

The implementation of this centralized architecture, based on open-source tools, not only improves operational efficiency and incident response capabilities but also provides a sustainable and accessible framework for educational environments. This project contributes to the academic community by demonstrating the feasibility of integrating open-source technologies and artificial intelligence for log management, offering a reference framework for other educational institutions seeking to enhance their monitoring and incident response capabilities in their technological systems.

Keywords: Centralized log management, OpenSearch, Fluent Bit, Centralized architecture, Log analysis, Artificial Intelligence, Institutional systems, Real-time data processing.

3. Introducción

En la era de la transformación digital, la gestión centralizada de logs se ha convertido en un proceso esencial para garantizar la seguridad, operatividad y continuidad de los servicios informáticos. Este proceso implica la recolección, almacenamiento y análisis de los registros generados por los diversos sistemas de una organización, conocidos como "logs", los cuales contienen información clave sobre accesos, errores y actividades críticas del sistema. La centralización de estos datos permite un monitoreo eficiente, una respuesta temprana a incidentes y una mejor capacidad de diagnóstico ante problemas técnicos [9].

En el contexto de la Universidad Nacional de Loja (UNL), la gestión de logs enfrenta una importante limitación: su carácter descentralizado. Los sistemas institucionales, como es el Sistema de Información Académico, Administrativo y Financiero (SIAAF), Sistema de Admisiones y Sistema de Trámites, manejan sus registros de logs de forma independiente, con tecnologías diversas como Python, Django, Node.js y ReactJs. Esta dispersión de los logs dificulta su análisis y el diagnóstico oportuno de problemas, limitando la capacidad de los equipos técnicos para identificar y resolver incidencias de forma eficiente. Además, esta falta de centralización compromete la seguridad y la continuidad operativa, elementos esenciales para garantizar la calidad de los servicios tecnológicos ofrecidos por la institución.

Ante esta problemática, surge la necesidad de implementar una solución de gestión centralizada de logs que integre tecnologías avanzadas, como OpenSearch e inteligencia artificial (IA) [18], para optimizar el monitoreo y análisis de datos. Este proyecto no solo aborda las limitaciones actuales, sino que también representa un avance significativo que fortalecerá la estabilidad de los sistemas institucionales de la UNL y mejorará su capacidad de respuesta ante incidentes, sentando las bases para una gestión tecnológica más eficiente y segura.

4. Marco Teórico

4.1. Antecedentes

La gestión de logs ha sido una parte integral de la administración de sistemas informáticos desde sus inicios. A medida que las organizaciones han crecido en complejidad y tamaño, la necesidad de gestionar de manera eficiente los registros de actividad de sus sistemas se ha vuelto crucial. Históricamente, la gestión de logs se realizaba de manera manual o con herramientas limitadas, lo que presentaba varios desafíos, incluyendo la falta de estandarización y la dificultad para detectar incidencias en tiempo real. En este contexto, propuestas como la implementación de sistemas de monitoreo de red con agentes han demostrado ser eficaces para la gestión centralizada de logs y la detección temprana de incidentes [7].

En instituciones educativas como la Universidad Nacional de Loja (UNL), donde se manejan grandes volúmenes de datos a diario, la implementación de soluciones avanzadas para la gestión centralizada de logs es fundamental para garantizar la operatividad y seguridad de los sistemas. Ortega Falces (2023) exploró la aplicación de Elastic Stack en plataformas educativas, demostrando que la monitorización de logs es esencial para mejorar la seguridad y optimización de recursos en sistemas institucionales [4].

Para la ejecución de la arquitectura de gestión centralizada de logs en este proyecto, se emplearán herramientas clave como OpenSearch, OpenSearch-Dashboard, modelos LLM de inteligencia artificial y FluentBit. Saiz Gutiérrez (2024) destacó la importancia de la integración de Kafka con ELK en plataformas Big Data, lo que refuerza la elección de Kafka en esta solución para la transmisión eficiente de registros en tiempo real [8]. Además, Farooq (2022) enfatiza la relevancia de la persistencia de logs en entornos Kubernetes mediante Elasticsearch, lo cual se alinea con el uso de OpenSearch en este proyecto para garantizar la disponibilidad y accesibilidad de datos críticos [2].

OpenSearch proporcionará la infraestructura necesaria para el almacenamiento y búsqueda de logs, mientras que los modelos de IA facilitarán la detección de anomalías y la recomendación de soluciones. FluentBit se encargará de la unificación y enrutamiento de los logs provenientes de diversas fuentes, y Apache Kafka permitirá la transmisión de datos en tiempo real con alta disponibilidad. Estas herramientas, en conjunto, formarán la base de la arquitectura propuesta, la cual será diseñada para optimizar la gestión de logs y mejorar la seguridad y operatividad de los sistemas. A continuación, se procederá a la definición detallada de cada uno de estos conceptos, destacando sus características y el papel que desempeñarán en la arquitectura.

4.2. Tecnología

La gestión de logs es fundamental para la observabilidad completa de los sistemas tecnológicos, permitiendo monitorear y analizar el rendimiento de aplicaciones e infraestructuras. Centralizar los logs en una única plataforma facilita su análisis y correlación, optimizando la detección de incidentes. Para ello, es crucial estructurar la información de manera clara, incorporando mensajes de error detallados y metadatos útiles que permitan identificar rápidamente la causa raíz de los problemas. Además, estandarizar los niveles de severidad y el formato de los logs garantiza coherencia y facilita su interpretación.

Para la implementación de la arquitectura propuesta en este proyecto, se emplean herramientas avanzadas de código abierto que permiten la recopilación, procesamiento y análisis eficiente de grandes volúmenes de logs. A continuación, se detallan las principales tecnologías utilizadas y su función dentro del sistema.

Para implementar una arquitectura eficiente de gestión centralizada de logs, se han seleccionado diversas herramientas que cumplen funciones clave en la recolección, almacenamiento, análisis, visualización y notificación de eventos. Estas tecnologías se agrupan en las siguientes categorías:

- **Recolección y procesamiento de logs:** Herramientas utilizadas para capturar y procesar logs en tiempo real.
- **Almacenamiento y búsqueda:** Plataformas que permiten indexar, buscar y estructurar grandes volúmenes de datos de logs.
- **Modelos de Inteligencia Artificial:** Algoritmos de aprendizaje automático aplicados a la detección y análisis de logs.
- **Sistemas de mensajería y colas:** Tecnologías que facilitan la transmisión eficiente de logs en entornos distribuidos.
- **Visualización de logs:** Herramientas que proporcionan interfaces gráficas para el monitoreo y análisis de logs.
- **Notificaciones y alertas:** Sistemas utilizados para alertar a los administradores sobre eventos críticos en los logs.

A continuación, se presentan en detalle las principales tecnologías utilizadas en cada una de estas categorías:

4.2.1. Recolección y procesamiento de logs

4.2.1.1. Fluent Bit

Fluent Bit es un procesador y reenviador de registros de código abierto diseñado para recopilar, filtrar y enviar logs a múltiples destinos [23]. Su diseño ligero y eficiente lo convierte en una herramienta ideal para entornos con recursos limitados, como sistemas embebidos y microservicios en contenedores Docker o Kubernetes. Fluent Bit es capaz de integrarse de

manera efectiva con OpenSearch o Elasticsearch para proporcionar una solución robusta de gestión centralizada de logs. En este contexto, Fluent Bit actúa como un recolector de datos que procesa los logs de diversas fuentes y los envía a OpenSearch o Elasticsearch, donde los datos son indexados y pueden ser analizados y visualizados en tiempo real. Esta integración permite a las organizaciones gestionar de manera eficiente grandes volúmenes de logs, mejorar la visibilidad del estado del sistema, y facilitar la detección y resolución de incidencias [6].

4.2.2. Almacenamiento y búsqueda

Las herramientas para la gestión, análisis y búsqueda de logs, como OpenSearch y Elasticsearch, desempeñan un papel fundamental en el procesamiento de grandes volúmenes de datos en tiempo real. Estas soluciones permiten centralizar, buscar y analizar información de manera eficiente, facilitando la detección de incidentes, el monitoreo de sistemas y la generación de reportes. Su capacidad para indexar y consultar datos estructurados y no estructurados las convierte en componentes esenciales en arquitecturas modernas enfocadas en la observabilidad y el análisis de datos críticos [4][7].

El procesamiento en tiempo real de grandes volúmenes de datos en registros de sistemas es fundamental para la detección temprana de incidentes y la optimización de recursos. Herramientas de Big Data han sido ampliamente adoptadas para mejorar este flujo de trabajo, permitiendo mayor eficiencia en la recolección, almacenamiento y análisis de logs [5].

La monitorización de logs no solo permite optimizar el rendimiento de los sistemas, sino que también juega un papel fundamental en la detección de intrusiones y ataques de seguridad. Estudios recientes han demostrado que la implementación de sistemas de monitoreo de logs enfocados en la seguridad mejora la identificación temprana de amenazas.

A continuación, se presentan los conceptos clave de cada herramienta, seguidos de una comparativa de sus características principales.

4.2.2.1. OpenSearch

OpenSearch es un conjunto distribuido, basado en la comunidad, con licencia de Apache 2.0, totalmente de código abierto para búsqueda y análisis. Se usa para una amplia gama de situaciones, como el monitoreo de aplicaciones en tiempo real, el análisis de registros y la búsqueda en sitios web. OpenSearch proporciona un sistema altamente escalable para proporcionar acceso y respuesta rápida a volúmenes grandes de datos con una herramienta de visualización integrada, OpenSearch Dashboards [20], [21], lo que facilita a los usuarios analizar sus datos. OpenSearch cuenta con tecnología de la biblioteca de búsqueda Apache Lucene y admite diversas capacidades de búsqueda y análisis, como la búsqueda de k vecinos más cercanos (KNN), SQL, detección de anomalías, Machine Learning Commons, análisis de rastreos, búsqueda de texto completo y más.

Los desarrolladores suelen adoptar software de código abierto por la libertad que ofrece para usar y modificar el software sin restricciones. El 21 de enero de 2021, Elastic NV anunció un cambio en su estrategia de licencias, dejando de lanzar nuevas versiones de Elasticsearch y Kibana bajo la licencia Apache 2.0 (ALv2) y optando por licencias más restrictivas como la Elastic License y la Server Side Public License (SSPL), que no cumplen con los criterios de código abierto [10]. Este cambio llevó a Amazon Web Services (AWS) y la comunidad de código abierto a desarrollar OpenSearch, una bifurcación basada en la última versión ALv2 de Elasticsearch (7.10.2). OpenSearch mantiene la licencia Apache 2.0, asegurando que siga siendo una opción libre y accesible para aquellos que dependen de estas tecnologías para la gestión y búsqueda de datos, sin quedar atados a un único proveedor.

4.2.2.2. Elasticsearch

Elasticsearch es un motor de búsqueda y analítica de RESTfull distribuido, un almacén de datos escalables y una base de datos de vectores capaz de abordar un número creciente de casos de uso. Como núcleo del Elastic Stack, almacena de forma central tus datos para una búsqueda a la velocidad de la luz, relevancia refinada y analíticas poderosas que escalan con facilidad. Su implementación ha sido utilizada en la monitorización de plataformas educativas, permitiendo la trazabilidad de eventos clave y el análisis de incidencias en tiempo real [4].

En la Tabla 2 se presentan las características principales de las herramientas utilizadas en la gestión de logs, destacando aspectos clave que facilitan su integración con modelos de lenguaje (LLM) de manera local. Estas características incluyen compatibilidad, escalabilidad, y licencias, elementos esenciales para garantizar un entorno eficiente y adaptable para la ejecución y análisis de grandes volúmenes de datos.

Tabla 1. Comparativa de Herramientas para la gestión de logs.

Características	Elasticsearch	OpenSearch
Tipo de Licencia	Dual (Elastic License 2.0 y SSPL). Restrictiva para usos comerciales.	Apache 2.0. Completamente libre y de código abierto.
Compatibilidad	Integración sólida con Kibana.	Compatible con versiones anteriores de Elasticsearch.
Escalabilidad	Altamente escalable para grandes volúmenes de datos.	Similar, mantiene la escalabilidad de Elasticsearch.
Comunidad y Soporte	Gran comunidad, pero soporte oficial es pago.	Comunidad en crecimiento con soporte activo.

Ventajas	Madurez del ecosistema. Mayor cantidad de plugins y herramientas disponibles.	Completamente gratuito y sin restricciones. Transparencia en el desarrollo.
Desventajas	Restricciones de licencia. Costos altos para el soporte comercial.	Ecosistema aún en desarrollo. Comunidad más pequeña que Elasticsearch.

4.2.3. Modelos de Inteligencia Artificial aplicados a la gestión de logs

4.2.3.1. Ollama

Ollama es una herramienta para ejecutar localmente grandes modelos de redes neuronales. Las empresas suelen percibir el uso de servicios públicos como un riesgo potencial de fuga de datos confidenciales y sensibles. Por ello, desplegar LLM en un servidor controlado permite gestionar de forma independiente los datos depositados en él, al tiempo que se aprovechan los puntos fuertes de LLM.

Esto también ayuda a evitar la desagradable situación de "vendor lock-in", en la que cualquier servicio público puede dejar de prestar servicios unilateralmente. Por supuesto, el objetivo inicial es permitir el uso de redes neuronales generativas en lugares donde no hay acceso a Internet o éste es difícil (por ejemplo, en un avión) [22].

La idea era simplificar el lanzamiento, control y puesta a punto de las LLM. En lugar de complejas instrucciones de varios pasos, Ollama permite ejecutar un simple comando y recibir el resultado final al cabo de un tiempo. Se presentará simultáneamente en forma de modelo de red neuronal local, con el que podrá comunicarse mediante una interfaz web y una API para integrarlo fácilmente en otras aplicaciones [11].

Para muchos desarrolladores, esto se convirtió en una herramienta muy útil, ya que en la mayoría de los casos era posible integrar Ollama con el IDE utilizado y recibir recomendaciones o código ya escrito directamente mientras se trabajaba en la aplicación.

En un principio, Ollama estaba destinado únicamente a ordenadores con el sistema operativo macOS, pero más tarde fue portado a Linux y Windows. También se ha lanzado una versión especial para trabajar en entornos contenerizados como Docker. Actualmente, funciona igual de bien tanto en ordenadores de sobremesa como en cualquier servidor dedicado con una GPU. Ollama admite la posibilidad de cambiar entre distintos modelos de forma inmediata y maximiza todos los recursos disponibles. Por supuesto, estos modelos pueden no rendir tan bien en un escritorio normal, pero funcionan bastante adecuadamente.

4.2.3.2. HuggingFaces

Podemos definir qué es Hugging Face como una empresa de tecnología que se dedica al desarrollo de herramientas y plataformas de procesamiento de lenguaje natural o NLP basadas en inteligencia artificial. Su enfoque se centra en crear modelos de aprendizaje profundo de última generación para tareas como la comprensión del lenguaje natural, la generación de lenguaje natural, el análisis de sentimientos o la traducción automática, entre otras.

Además, hugging-face ha creado una plataforma de código abierto llamada Transformers, la cual le permite a los desarrolladores de todo el mundo acceder y utilizar modelos de NLP preentrenados de última generación. Esta plataforma huggingface se ha convertido en una herramienta fundamental para el desarrollo de aplicaciones basadas en NLP y ha sido adoptada por una amplia comunidad de desarrolladores y empresas [12].

En resumen, al hablar sobre qué es Hugging Face hay que destacar que se trata de una empresa que está liderando la revolución de la inteligencia artificial en el procesamiento del lenguaje natural. Asimismo, está haciendo que sea más accesible y fácil para los desarrolladores y empresas de todo el mundo aprovechar las últimas tecnologías de NLP.

4.2.3.3. LM Studio

LM Studio es una herramienta que permite a los usuarios trabajar con modelos de lenguaje de gran tamaño (LLM) de manera local, facilitando su uso sin necesidad de conexión a Internet. Esto es especialmente útil para experimentar con modelos de código abierto como Llama 2, Vicuna y Mistral, entre otros. Al ejecutar estos modelos localmente, se garantiza la privacidad de los datos, ya que no se envían a la nube, lo que resulta crucial para manejar información confidencial. Además, LM Studio ofrece la posibilidad de personalizar y entrenar modelos específicos para tareas concretas, lo que puede ser más eficiente y económico en comparación con el uso de modelos comerciales.

La herramienta busca simplificar el proceso de ejecución de LLM, que a menudo requiere conocimientos técnicos avanzados. Con LM Studio, los usuarios pueden aprovechar el potencial de la inteligencia artificial generativa sin complicaciones técnicas excesivas, permitiendo un acceso más amplio a estas tecnologías.

4.2.3.4. GPT4All

Es otra herramienta que nos permite ejecutar modelos de lenguaje grandes (LLM) de forma privada en computadoras de escritorio y portátiles de uso diario.

Con esta herramienta, podemos usar inteligencia artificial de manera totalmente offline, ideal para cuando queremos mantener la información segura y privada.

En la Tabla 1 se presentan las características de cada herramienta que nos permiten ejecutar

modelos (LLM) de manera local.

Tabla 2. Herramientas LLM para entorno locales.

Características	Ollama	LM Studio	GPT4All
Tipo de Licencia	MIT License. Completamente de código abierto.	Licencia gratuita para uso personal; se requiere contacto para uso comercial.	Apache 2.0. Completamente de código abierto.
Compatibilidad de Modelos	Soporta modelos como LLaMA, GPT-3, GPT-4 y otros LLMs populares.	Compatible con una amplia gama de modelos, incluyendo LLaMA, Falcon, MPT, StarCoder, Replit, GPT-Neo-X, y más.	Diseñado para trabajar con modelos como Vicuna, Alpaca y LLaMA, enfocándose principalmente en la arquitectura GPT.
Interfaz de Usuario	Ofrece una interfaz amigable con un constructor de conversaciones de arrastrar y soltar, facilitando la creación y diseño de conversaciones de chatbot.	Proporciona una interfaz intuitiva que simplifica el proceso de desarrollo, accesible tanto para principiantes como para desarrolladores experimentados.	Ofrece una interfaz de línea de comandos y una interfaz gráfica de usuario, permitiendo a los usuarios interactuar con los modelos de manera flexible.
Soporte de GPU	Compatible con aceleración GPU, mejorando el rendimiento en tareas intensivas.	Utiliza CUDA, OpenCL, cuBLAS y Metal para aceleración, asegurando alto rendimiento en diferentes configuraciones de hardware.	No especifica métodos de aceleración, lo que puede afectar el rendimiento en máquinas menos potentes.
Sistema Operativo Soportado	Disponible para macOS, Linux y Windows (en vista previa), con soporte para procesadores que admiten AVX.	Disponible para Mac con M1/M2/M3, o PC con Windows con procesadores que soportan AVX2. Linux está disponible en versión beta.	Compatible con Windows, macOS y Linux, ofreciendo flexibilidad en la elección del sistema operativo.
Comunidad y Soporte	Cuenta con una comunidad en crecimiento y soporte activo, promoviendo la transparencia y la participación comunitaria.	Ofrece documentación oficial completa con guías detalladas y ejemplos, facilitando una curva de aprendizaje más suave.	Respaldado por una comunidad activa que comparte conocimientos, consejos de solución de problemas y proyectos colaborativos.

4.2.4. Sistemas de mensajería y colas

4.2.4.1. Redis

Redis es un almacén de clave/valor NoSQL de código abierto que opera en memoria, diseñado para ofrecer un rendimiento excepcional como caché de aplicaciones o base de datos de respuesta rápida [15]. A continuación, se presentan los conceptos principales sobre Redis:

- **Almacenamiento en Memoria:** Redis almacena datos en la memoria principal del servidor, lo que permite tiempos de respuesta significativamente más rápidos en comparación con los sistemas que almacenan datos en discos duros o unidades de estado sólido (SSD).
- **Estructuras de Datos:** Aunque es un almacén de clave/valor, Redis es considerado un servidor de estructuras de datos, ya que soporta múltiples tipos de datos como listas, conjuntos y hashes, facilitando operaciones complejas.
- **Alta Disponibilidad y Escalabilidad:** Redis incluye características como Redis Sentinel para la supervisión y failover automático, y Redis Cluster para la distribución de datos entre varios nodos, lo que asegura alta disponibilidad y escalabilidad.
- **Persistencia:** A pesar de su naturaleza en memoria, Redis ofrece opciones de persistencia que permiten la conservación de datos a través de instantáneas periódicas y registros de cambios, garantizando la durabilidad e integridad de la base de datos.
- **Integración y Soporte:** Con más de 100 clientes disponibles, Redis permite a los desarrolladores interactuar fácilmente con sus datos y agregar nuevas integraciones según sea necesario.
- **Casos de Uso:** Es comúnmente utilizado para mejorar el rendimiento en aplicaciones web mediante el almacenamiento en caché, así como para implementar sistemas de mensajería y chat debido a sus capacidades.

4.2.4.2. Apache Kafka

Apache Kafka es una plataforma distribuida de transmisión de datos que permite la publicación, almacenamiento y procesamiento en tiempo real de flujos de eventos. Diseñada originalmente por LinkedIn, Kafka gestiona grandes volúmenes de datos, facilitando la integración entre aplicaciones y microservicios mediante métodos sincrónicos y asíncronos [16]. A continuación, se presentan los conceptos clave sobre Apache Kafka:

- **Transmisión de Datos:** Kafka permite la transferencia eficiente de grandes cantidades de datos entre múltiples fuentes y destinos, facilitando la integración de sistemas distribuidos.
- **Arquitectura Distribuida:** Su diseño permite que múltiples productores y consumidores

interactúen con el sistema simultáneamente, lo que mejora la escalabilidad y la resiliencia.

- Integración Ágil: Kafka es fundamental para implementar integraciones ligeras y rápidas en entornos de microservicios, promoviendo una arquitectura más flexible y adaptable.
- Alto Rendimiento: Capaz de manejar millones de eventos por segundo, Kafka es ideal para aplicaciones que requieren procesamiento en tiempo real, como el Internet de las cosas (IoT) y el comercio electrónico.
- Compatibilidad con Kubernetes: Kafka se integra bien con Kubernetes, lo que permite una gestión eficiente de los recursos y una implementación simplificada en entornos en contenedores. [13]

En la Tabla 3 se presentan las características principales de Apache Kafka y Redis, dos herramientas ampliamente utilizadas en sistemas de mensajería y colas.

Tabla 3. Comparativa de Herramientas de Sistemas de Mensajería y Colas.

Características	Apache Kafka	Redis
Tipo de Licencia	Apache 2.0. Libre y de código abierto.	BSD 3-Clause. Libre y de código abierto.
Modelo de Uso	Sistema de mensajería distribuida basado en logs.	Almacenamiento en memoria con soporte para pub/sub.
Rendimiento	Diseñado para manejar grandes volúmenes de datos y altas tasas de <u>transferencia</u> .	Alto rendimiento para tareas en tiempo real con baja latencia.
Persistencia	Manejo nativo de persistencia a disco.	Opcional, generalmente en memoria con snapshots persistentes.
Ventajas	Ideal para flujos de datos complejos. Soporta procesamiento en tiempo real.	Simplicidad de configuración. Baja latencia para operaciones en tiempo real.
Desventajas	Mayor complejidad en la configuración. Latencia ligeramente superior en comparación con Redis.	No está diseñado específicamente para flujos complejos. Capacidad limitada para grandes volúmenes de datos.

4.2.5. Herramientas de visualización

4.2.5.1. OpenSearch Dashboard

OpenSearch Dashboards es una herramienta de visualización de código abierto que se utiliza para interactuar con datos almacenados en OpenSearch. Esta herramienta permite a los usuarios crear y gestionar paneles de control que visualiza datos en tiempo real, facilitando el análisis y la interpretación de grandes volúmenes de información [21]. Se integra directamente con Amazon OpenSearch Service, lo que permite a los usuarios acceder a sus dominios de OpenSearch a través de una interfaz gráfica intuitiva.

Las características clave de OpenSearch Dashboards incluyen:

- Visualización de datos: Permite crear gráficos, tablas y otros tipos de visualizaciones personalizadas.
- Interactividad: Los usuarios pueden explorar los datos mediante filtros y consultas dinámicas.
- Gestión de índices: Proporciona herramientas para administrar índices, facilitando la supervisión y el mantenimiento de los datos almacenados.
- Acceso seguro: Incluye opciones para controlar el acceso a los paneles mediante autenticación y políticas de seguridad, asegurando que solo los usuarios autorizados puedan acceder a la información sensible

4.2.5.2. Grafana

Grafana es una plataforma interactiva y open source de visualización de datos desarrollada por Grafana Labs. Dicha plataforma permite a los usuarios ver sus datos a través de tablas y gráficos que se unifican en un panel de control (o en varios) para facilitar la interpretación y la comprensión. También permite realizar consultas y configurar avisos sobre la información y los indicadores desde el lugar en el que se almacena dicha información, ya sea en entornos de servidores tradicionales, clústeres de Kubernetes y varios servicios de nube, entre otros. De esta forma, podrá analizar los datos e identificar las tendencias y las inconsistencias con mayor facilidad, por lo que sus procesos serán más eficientes. Grafana se diseñó según los principios del open source y con la creencia de que los datos deberían ser accesibles para toda la empresa, no solo para unas pocas personas. Esto fomenta una cultura por la cual cualquier persona que necesite los datos puede encontrarlos y utilizarlos fácilmente, lo cual permite que los equipos sean más abiertos, innovadores y colaborativos [14][18].

En la Tabla 4 se presentan las características principales de OpenSearch Dashboard y Grafana, dos herramientas clave para la visualización y análisis de datos.

Tabla 4. Comparativa de OpenSearch Dashboard y Grafana.

Características	OpenSearch Dashboard	Grafana
Tipo de Licencia	Apache 2.0. Libre y de código abierto.	AGPLv3. Libre y de código abierto.
Compatibilidad	Diseñado específicamente para OpenSearch.	Compatible con múltiples bases de datos, incluido OpenSearch.
Interfaz de Usuario	Simplicidad y enfoque en logs y métricas.	Altamente personalizable con soporte para visualizaciones avanzadas.
Ventajas	Integración nativa con OpenSearch. Optimizado para logs y búsquedas específicas.	Mayor flexibilidad. Soporte para datos de múltiples fuentes en un solo panel.
Desventajas	Menor personalización en visualizaciones. Exclusivo para OpenSearch.	Requiere configuración más avanzada para algunos usos. No optimizado exclusivamente para logs.

4.2.6. Notificaciones y alertas

4.2.6.1. SLACK

Slack es una plataforma de comunicación y colaboración que permite la integración con múltiples herramientas, facilitando la administración y monitoreo de eventos en tiempo real. En el contexto de la **gestión de logs**, Slack se convierte en un canal clave para la notificación de incidencias, alertas automatizadas y el análisis colaborativo de eventos críticos.

Principales usos de Slack en la gestión de logs

- **Notificaciones en tiempo real:** Integrado con sistemas de monitoreo como **Fluent Bit**, **OpenSearch** y **Grafana**, Slack permite recibir alertas instantáneas sobre eventos críticos en la infraestructura.
- **Automatización con Webhooks:** Mediante la configuración de **webhooks**, es posible enviar logs relevantes a canales específicos en Slack para su análisis inmediato.
- **Centralización de información:** Slack facilita la organización de **canales temáticos**, donde los equipos de desarrollo, operaciones y seguridad pueden discutir eventos específicos y coordinar respuestas ante incidentes.

La integración con herramientas como **Google Docs y Trello** permite documentar incidentes y seguir la resolución de problemas dentro del mismo entorno.

- **Análisis colaborativo de logs:** Equipos distribuidos pueden analizar en conjunto registros de errores, realizar diagnósticos rápidos y compartir hallazgos en tiempo real. Slack se convierte en una interfaz eficiente para coordinar acciones correctivas y planificar mejoras en la infraestructura tecnológica [24].

4.3. Trabajos Relacionados

El presente proyecto se basa en diversas investigaciones previas sobre la gestión centralizada de logs, la integración de inteligencia artificial en el análisis de logs y la implementación de herramientas avanzadas de monitoreo. A continuación, se presentan los estudios más relevantes y su relación con la solución propuesta:

- **New Relic (2022):** Este estudio analiza estrategias avanzadas de observabilidad en entornos empresariales, destacando la importancia de la estructuración de logs para mejorar la correlación de eventos y facilitar la detección de incidentes. La relación con este proyecto radica en que la implementación de OpenSearch y modelos de IA permitirá estructurar y analizar logs críticos, asegurando la identificación de patrones y posibles fallos en sistemas institucionales [9].
- **Saiz Gutiérrez (2024):** Su propuesta de integración del stack ELK con Kafka en plataformas Big Data es fundamental para optimizar la ingesta y procesamiento de logs en arquitecturas escalables. Este enfoque es altamente relevante para el presente trabajo, ya que la combinación de OpenSearch y Kafka garantiza un manejo eficiente de grandes volúmenes de registros, permitiendo un monitoreo centralizado y la reducción de latencias en la detección de incidentes [8].
- **Stefan Michis (2022):** En su comparativa entre ELK y Fluentd, destaca las ventajas de cada herramienta en distintos escenarios de implementación. Su análisis sugiere que Fluentd es más adecuado para entornos con restricciones de recursos, mientras que ELK es más robusto en la gestión de grandes volúmenes de datos. Esta investigación respalda la selección de Fluent Bit en este proyecto, asegurando una recolección de logs eficiente y con menor consumo de recursos antes de su almacenamiento en OpenSearch [7].
- **Ortega Falces (2023):** Examina la aplicación de Elastic Stack en plataformas educativas, resaltando su impacto en la detección de incidentes y la optimización de infraestructuras tecnológicas. Su análisis evidencia cómo una solución centralizada de logs mejora la administración y seguridad de sistemas en entornos educativos. La relación con este estudio radica en que el uso de OpenSearch y modelos de IA en este proyecto permitirá mejorar la detección y mitigación de incidentes en sistemas

institucionales mediante un análisis detallado y automatizado de registros [4].

- **Farooq Nargis (2022):** Se enfoca en la persistencia y gestión de logs en entornos Kubernetes, donde la volátil naturaleza de los contenedores dificulta el almacenamiento continuo de registros. Su propuesta de integrar Elasticsearch para garantizar la disponibilidad de datos se alinea con la solución planteada en este trabajo, ya que OpenSearch desempeñará un papel central en la retención y análisis de logs en tiempo real. La combinación de Fluent Bit y Kafka permitirá asegurar una recolección eficiente de logs en entornos distribuidos, facilitando su análisis mediante modelos de IA [2].

5. Metodología

5.1. Área de estudio

El presente proyecto se llevó a cabo en el contexto de los sistemas institucionales desarrollados en la Dirección de Tecnologías (DTI) de la Universidad Nacional de Loja (UNL), específicamente los sistemas Sistema de Información Académico, Administrativo y Financiero (SIAAF), Sistema de Admisiones y Sistema de Trámites. Este trabajo se enmarca en el desarrollo de una solución tecnológica para la gestión centralizada de logs, con el objetivo de mejorar la capacidad de monitoreo, detección y respuesta ante incidencias. El desarrollo del proyecto se realizó bajo la supervisión del Ing. Pablo Fernando Ordóñez Ordóñez, en un ámbito estrictamente académico e investigativo.

El presente proyecto se desarrolló siguiendo un enfoque metodológico que combinó la metodología experimental y la metodología de modelado para alcanzar los objetivos establecidos. Este enfoque permitió validar la integración de las herramientas seleccionadas, así como también diseñar una arquitectura robusta para la gestión centralizada de logs.

5.2. Procedimiento

El desarrollo de este proyecto se basó en el cumplimiento de dos objetivos específicos, los cuales fueron abordados mediante un enfoque metodológico que combina aspectos experimentales y de modelado. Este enfoque permitió garantizar una implementación robusta, validando la integración de herramientas tecnológicas mediante casos de prueba en entornos controlados y facilitando el diseño de una arquitectura centralizada eficiente. Estudios previos han resaltado la importancia del monitoreo de red mediante agentes para optimizar la detección de incidencias y mejorar la respuesta ante eventos críticos [10].

La metodología experimental permitió evaluar la integración de componentes clave, como Fluent Bit, OpenSearch, Apache Kafka, Redis y modelos de inteligencia artificial [3], [7], [18], [10], mediante la ejecución de casos de prueba diseñados para simular situaciones reales. Por otro lado, la metodología de modelado facilitó la representación estructurada del sistema mediante diagramas UML, asegurando la flexibilidad, escalabilidad y robustez de la arquitectura propuesta.

Para la recolección y visualización de logs en este proyecto, se han tomado en cuenta enfoques previos como la implementación de OpenSearch en entornos institucionales, así como experiencias en la monitorización de plataformas educativas mediante Elastic Stack [4].

A continuación, se describe en detalle el procedimiento seguido para alcanzar los objetivos del proyecto:

5.2.1. Objetivo 1: Desarrollar casos de prueba para evaluar la integración de OpenSearch con modelos de IA que recomienden la mitigación de incidencias en los sistemas institucionales, utilizando una metodología experimental.

- Se realizó la configuración e instalación de las herramientas necesarias para la recolección, almacenamiento y análisis de logs, las cuales incluyen Fluent Bit para la recolección de logs desde los diferentes sistemas institucionales [17], OpenSearch para la centralización y OpenSearch Dashboard para la visualización de los logs, se llevó a cabo la instalación de modelos LLM de IA (LLaMA 3, Mistral, Gemma2) para el procesamiento de las recomendaciones. Ver anexo 2.
- Se llevó a cabo la instalación de Apache Kafka para procesar grandes volúmenes de datos en tiempo real. Esto permitió garantizar la entrega confiable de mensajes y desacoplar los sistemas emisores (Fluent Bit) de los consumidores (API Flask y OpenSearch). Esto permitió priorizar logs críticos para análisis rápido por el modelo de IA, optimizando el tiempo de respuesta ante incidentes y asegurar la escalabilidad del sistema.
- Se desarrolló un servicio utilizando APIFlask para recibir los logs críticos enviados por Apache Kafka, procesarlos mediante un modelo de IA (LLaMA 3, Mistral o Gemma2) y generar recomendaciones o soluciones a los errores detectados. Este servicio actúa como un intermediario clave en el flujo de gestión de logs, normalizando los datos recibidos, enviándolos al modelo de IA para su análisis, y notificando las posibles soluciones mediante Slack. Su finalidad es automatizar el análisis y la generación de recomendaciones para incidentes críticos, optimizando la detección y mitigación de problemas en los sistemas institucionales y mejorando la eficiencia de respuesta del administrador del sistema.
- Se definieron escenarios específicos basados en los logs generados previamente por los sistemas institucionales de la Universidad Nacional de Loja, enfocándose en la recolección, procesamiento y análisis de logs críticos. Estos escenarios incluyeron la identificación y priorización de logs críticos, su gestión mediante OpenSearch, la generación automatizada de recomendaciones a través de modelos de IA (LLaMA 3, Mistral, o Gemma2), y la visualización de registros en tiempo real utilizando OpenSearch Dashboard. Esto permitió establecer un flujo integral que reflejara las necesidades reales del sistema.
- Se ejecutaron casos de prueba en un entorno controlado diseñado para simular condiciones reales de operación de los sistemas institucionales. Este entorno permitía emular la generación de logs críticos, su transmisión mediante Fluent Bit y Kafka, el

análisis por el modelo de IA y la visualización en OpenSearch Dashboard. Esto garantizó la validez de las pruebas, asegurando la consistencia y la reproducibilidad de los resultados en un contexto seguro.

- Se llevó a cabo la validación del flujo de notificaciones críticas mediante Slack, verificando que los logs críticos fueran analizados correctamente por el modelo de IA y que las recomendaciones generadas fueran entregadas de forma clara y oportuna al canal de notificaciones.

Estas actividades permitieron validar la integración de las herramientas seleccionadas, evaluar su eficacia y generar un flujo robusto de gestión de logs críticos, cumpliendo así con los requisitos del Objetivo 1. **Ver anexo 3:** Instalación de herramientas para la gestión de logs.

5.2.2. Diseñar la arquitectura para la gestión centralizada de logs en los sistemas institucionales de la Universidad Nacional de Loja, mediante un estilo arquitectónico que represente la integración de OpenSearch e inteligencia artificial.

- Se seleccionaron herramientas tecnológicas clave para la implementación de la arquitectura centralizada de logs, asegurando su capacidad de integración y cumplimiento de los objetivos del proyecto. Las herramientas incluyen:
 - **Fluent Bit:** Para la recolección de logs desde los sistemas institucionales.
 - **OpenSearch:** Para el almacenamiento, búsqueda y análisis centralizado de los registros.
 - **OpenSearch Dashboard:** Para la visualización gráfica en tiempo real de los logs procesados.
 - **Modelos de IA (LLaMA 3, Mistral, o Gemma2):** Para el análisis de los logs y la generación de recomendaciones de mitigación.
 - **Redis:** Como sistema de almacenamiento en caché rápido para acelerar las consultas y procesar los datos del modelo de IA.
 - **ChromaDB:** Para la búsqueda semántica de logs similares y enriquecimiento de datos históricos.
 - **Apache Kafka:** Para la transmisión y manejo de logs críticos entre los diferentes componentes de la arquitectura.
 - **Slack:** Para enviar notificaciones automatizadas al equipo administrativo sobre incidentes críticos y sus posibles soluciones. Estas herramientas fueron seleccionadas por su capacidad de interoperabilidad, escalabilidad y rendimiento, fundamentales para cumplir con las necesidades del diseño arquitectónico.

- Se diseñó una arquitectura modular para garantizar flexibilidad, interoperabilidad y escalabilidad. Este diseño incluyó los siguientes módulos principales:
 - **Recolección de logs:** Configurado con Fluent Bit para capturar registros desde los sistemas SIAAF, Admisiones y Trámites.
 - **Almacenamiento y procesamiento:** Implementado con OpenSearch y OpenSearch Dashboard para la centralización y análisis de grandes volúmenes de logs.
 - **Análisis inteligente:** Integrado con modelos de IA para procesar los registros y proporcionar recomendaciones en tiempo real.
 - **Manejo de logs críticos:** Configurado con Apache Kafka para priorizar y transmitir logs críticos al modelo de IA mediante Redis como caché de consultas rápidas y ChromaDB para búsquedas semánticas de logs históricos.
 - **Notificaciones:** Configurado con Slack para la entrega automatizada de alertas y soluciones. Este enfoque modular aseguró que cada componente cumpliera su función específica, manteniendo la cohesión y permitiendo futuras ampliaciones.
- Se utilizaron herramientas de modelado para elaborar diagramas representativos de la arquitectura, detallando la interacción entre los componentes y el flujo de datos. Los diagramas incluyen:
 - El flujo desde la recolección de logs hasta su almacenamiento en OpenSearch.
 - El análisis de los registros por modelos de IA y la generación de recomendaciones.
 - El envío automatizado de notificaciones mediante Slack. Estos diagramas facilitaron la comprensión visual del diseño y sirvieron como referencia para la implementación de la solución.
- Se evaluó la viabilidad de la arquitectura mediante simulaciones en entornos controlados que emulaban las condiciones de los sistemas institucionales. Estas pruebas confirmaron que la arquitectura diseñada puede:
 - Manejar grandes volúmenes de logs sin pérdida de datos.
 - Generar recomendaciones de mitigación en tiempo real.
 - Integrarse eficientemente con los sistemas existentes y con las herramientas seleccionadas. Los resultados de las simulaciones demostraron que la arquitectura propuesta cumple con los requisitos establecidos y es robusta ante escenarios de alta demanda.

Estas actividades permitieron diseñar una arquitectura modular y robusta, validar la integración de las herramientas seleccionadas y evaluar su viabilidad mediante simulaciones

en entornos controlados, cumpliendo así con los requisitos establecidos en el Objetivo 2.

5.3. Recursos

Para el desarrollo del presente proyecto, se emplearon diversos recursos que facilitaron la implementación de las metodologías y el cumplimiento de los objetivos planteados. A continuación, se detallan los recursos utilizados:

5.3.1. Recursos de Hardware y Software

5.3.1.1. Hardware

- **Laptop Asus Core i7 2.2GHz, 16 de RAM, NVIDIA GeForce GTX 1060:** Este equipo fue fundamental para el desarrollo del proyecto, sirviendo como plataforma para la instalación, configuración y ejecución de las herramientas necesarias, así como para realizar pruebas, analizar resultados y validar los casos de prueba.

5.3.1.2. Software

- **Fluent Bit:** Para la recolección y envío de logs desde los sistemas institucionales.
- **OpenSearch:** Para el almacenamiento, búsqueda y análisis centralizado de los registros.
- **OpenSearch Dashboard:** Para la visualización gráfica en tiempo real de los logs procesados.
- **Apache Kafka:** Para la transmisión y manejo de logs críticos, garantizando la priorización de eventos importantes.
- **Redis:** Para el almacenamiento en caché y la optimización del procesamiento de datos.
- **ChromaDB:** Para la búsqueda semántica de logs similares y el enriquecimiento de datos históricos.
- **Slack:** Como medio de notificación automatizada de incidentes críticos y sus posibles soluciones.
- **Modelos de IA (LLaMA 3, Mistral, Gemma2):** Para el análisis avanzado de logs y generación de recomendaciones de mitigación.
- **ChatGPT:** Para consultas y asistencia técnica durante el desarrollo y configuración del proyecto.

Además, se emplearon entornos de desarrollo integrados (IDEs) para programar y configurar las herramientas:

- **PyCharm:** Para el desarrollo en Python, incluyendo el servicio Flask y otros scripts relacionados.
- **Visual Studio Code:** Para la configuración y edición de archivos de

configuración, como los de Docker y Fluent Bit.

- **StarUML:** Para la elaboración de diagramas de la arquitectura y el flujo de datos del sistema.

6. Resultados

En esta sección, se presentan los resultados obtenidos de la investigación, organizados y numerados como “Rn”, donde “n” indica el número del resultado (R1, R2, R3, etc.), para facilitar su comprensión y seguimiento.

6.1. Objetivo 1: Desarrollar casos de prueba para evaluar la integración de OpenSearch con modelos de IA que recomienden la mitigación de incidencias en los sistemas institucionales, utilizando una metodología experimental.

6.1.1. R1: Configuración del entorno de pruebas

La arquitectura para la gestión centralizada de logs se diseñó teniendo en cuenta la portabilidad, escalabilidad y replicación en diversos entornos. Para garantizar la validación de la arquitectura propuesta, todas las herramientas necesarias fueron instaladas y configuradas en una máquina virtual con la distribución Debian 12.9. Esta decisión se tomó para asegurar que la solución pudiera ser desplegada en otros servidores sin depender de configuraciones específicas de hardware o software local.

La máquina virtual proporciona ventajas clave, como independencia del sistema operativo anfitrión, flexibilidad para ajustar recursos según necesidades futuras, y un entorno controlado para realizar pruebas de rendimiento y funcionalidad. Esto permitió realizar evaluaciones preliminares de la arquitectura en condiciones reproducibles, similares a las de producción.

6.1.1.1. Entorno de Pruebas

Hardware

El entorno de pruebas fue configurado sobre una máquina virtual diseñada para simular un ambiente controlado y reproducible. Esto permitió evaluar el desempeño del sistema en escenarios críticos. Las especificaciones técnicas del hardware fueron las siguientes:

- Procesador: Intel Xeon E5-2678 v3 (2.50 GHz).
- Memoria RAM: 16 GB DDR4.
- Almacenamiento: 100 GB SSD asignados.
- Virtualización: Proporcionada por VMware ESXi 7.0.

Software

Para garantizar la estabilidad y compatibilidad con los sistemas institucionales, se utilizó la distribución **Debian 12.9**. Esta elección asegura que las configuraciones realizadas sean replicables en otros entornos de producción. El software instalado incluyó:

- **Sistema Operativo:** Debian 12.9.

- **Docker:** Versión 24.0.2, utilizado para contenerización de servicios clave como Fluent Bit, OpenSearch y Apache Kafka.
- **Fluent Bit:** Versión 3.1, para la recolección y procesamiento de logs.
- **Apache Kafka:** Versión 3.5.1, empleado como sistema de mensajería.
- **OpenSearch:** Versión 2.8, utilizado para el almacenamiento y visualización de logs.
- **Python:** Versión 3.10, para la ejecución de la API Flask.

6.1.1.2. Configuración de modelos LLM

Como parte del desarrollo del proyecto, se evaluaron diversos modelos de lenguaje (LLM) para analizar su desempeño en la detección y resolución de logs críticos generados por los sistemas institucionales. Los modelos seleccionados fueron LLaMA3, Mistral y Gemma2, debido a sus características técnicas, términos de licencia y capacidad para generar recomendaciones en el contexto del análisis de logs.

El objetivo de esta evaluación fue determinar la efectividad de cada modelo en tres aspectos: su capacidad para identificar problemas a partir de los logs, su velocidad de respuesta y la calidad de las recomendaciones generadas. Este análisis permite identificar fortalezas y limitaciones de cada modelo y seleccionar el más adecuado para integrarlo en el flujo de gestión centralizada de logs.

En la Tabla 5, se resumen las características clave de los modelos evaluados, incluyendo la versión utilizada, el número de parámetros, la técnica de cuantización aplicada y los términos de licencia que los acompañan. Este nivel de detalle proporciona una base técnica sólida para sustentar las decisiones relacionadas con la integración y uso de estos modelos en el proyecto.

Tabla 5. Características de los Modelos LLM.

Modelo LLM	Llama	Mistral	Gemma
Versión	llama3.1	mistral v0.3	gemma2
Parámetros	8.03 mil millones (B) de parámetros	7.25 mil millones (B)	9.24 mil millones (B)
Quantization	Q4_K_M	Q4_0	Q4_0
Licence	LLAMA 3.1 COMMUNITY LICENSE AGREEMENT Llama 3.1 Version Release Date: July 23,	Apache License Version 2.0, January 2004	Gemma Terms of Use Fecha: 21 de febrero de 2024

	2024		
Prompt	<p>Eres un asistente de desarrollo de software experto en análisis de logs y resolución de problemas técnicos.</p> <p>Se te proporcionará un log de error generado por un sistema en producción. Tu tarea es:</p> <ol style="list-style-type: none"> 1. Analizar detalladamente el log y explicar el problema de manera clara y concisa. 2. Identificar en qué archivo y línea se generó el error para proceder con la revisión. 3. Proporcionar posibles causas del error con base en el contenido del log. 4. Sugerir una solución directa y práctica en caso de existir para resolver el problema. <p>Brindar la solución en español.</p>		

6.1.2. R2: Casos de Prueba

Los casos de prueba desarrollados tienen como objetivo evaluar la efectividad de la arquitectura propuesta para la gestión centralizada de logs, enfocándose en la detección y resolución de errores críticos en los sistemas institucionales. Para este propósito, se llevaron a cabo pruebas experimentales en aplicaciones, como el Sistema de Información Académico, Administrativo y Financiero (SIAAF), el Sistema de Trámites y el Sistema de Admisiones. Estas aplicaciones representan escenarios críticos en la gestión administrativa, académica y de procesos estudiantiles.

Durante las pruebas, se simularon y replicaron algunas incidencias que utilizaron logs críticos, los cuales fueron analizados mediante modelos de inteligencia artificial (LLaMA3, Mistral y Gemma2) con el fin de validar las recomendaciones proporcionadas por los modelos, determinando su efectividad en la resolución de los errores identificados. Cabe destacar que, para preservar la confidencialidad del entorno de ejecución, se reemplazó la ruta real de los archivos por el marcador "ruta_proyecto". A continuación, se presenta el análisis de cada aplicación evaluada y los resultados obtenidos en los casos de prueba.

6.1.2.1. Aplicación evaluada: SIAFF

Objetivo de los casos

Evaluar la capacidad de detección y resolución de logs críticos mediante el análisis de los modelos de IA seleccionados (LLaMA3, Mistral y Gemma2).

Descripción

En esta evaluación se utilizan logs críticos registrados por el sistema SIAAF en su archivo de

logs correspondiente, los cuales representan incidencias reales que requieren intervención inmediata.

A continuación, se detallan los siguientes casos de prueba:

Tabla 6. Caso de Prueba 1 para la aplicación SIAAF.

Caso de Prueba	Registro de Logs
<p>Error de nombre de view invalido</p>	<pre>[ERROR] [2024-11-15 20:21:38,666] [log django.request 241] [Internal Server Error: /admin/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php] Traceback (most recent call last): File "/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site- packages/django/core/handlers/exception.py", line 56, in inner response = get_response(request) File "/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site- packages/django/core/handlers/base.py", line 185, in _get_response response = middleware_method(File "/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site- packages/django_cas_ng/middleware.py", line 65, in process_view return HttpResponseRedirect(reverse(settings.CAS_LOGIN_URL_NAME) + '?' + params) File "/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site- packages/django/urls/base.py", line 88, in reverse return resolver._reverse_with_prefix(view, prefix, *args, **kwargs) File "/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site- packages/django/urls/resolvers.py", line 802, in _reverse_with_prefix raise NoReverseMatch(msg) django.urls.exceptions.NoReverseMatch: Reverse for 'cas_ng_login' not found. 'cas_ng_login' is not a valid view function or pattern name.</pre>

Tabla 7. Caso de Prueba 2 para la aplicación SIAAF.

Caso de Prueba	Registro de Logs
<p>Error en query con id incorrecto</p>	<pre>[ERROR] [2024-11-18 12:01:05,049] [log django.request 241] [Internal Server Error: /portafolio/adjunto/lista] Traceback (most recent call last): File "/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site- packages/django/core/handlers/exception.py", line 56, in inner response = get_response(request) File "/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site- packages/django/core/handlers/base.py", line 197, in _get_response response = wrapped_callback(request, *callback_args, **callback_kwargs) File "/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site-</pre>

	<pre> packages/django/contrib/auth/decorators.py", line 23, in _wrapped_view return view_func(request, *args, **kwargs) File "/ruta_aplicacion/portafolio/services/views/views.py", line 225, in adjunto_lista if PortafolioSecService.puede_ver(portafolio, request.user) is False: File "/ruta_aplicacion/portafolio/security/portafolio.py", line 139, in puede_ver return qset.filter(id=portafolio.id).exists() AttributeError: 'NoneType' object has no attribute 'id' </pre>
--	---

Tabla 8. Caso de Prueba 3 para la aplicación SIAAF.

Caso de Prueba	Registro de Log
Error en template incorrecta	<pre> [ERROR] [2024-11-19 08:52:41,178] [log django.request 241] [Internal Server Error: /seguimiento- silabo/seguimiento/crear/22917] Traceback (most recent call last): File "/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site- packages/django/core/handlers/exception.py", line 56, in inner response = get_response(request) File "/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site- packages/django/core/handlers/base.py", line 220, in _get_response response = response.render() File "/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site- packages/django/template/response.py", line 114, in render self.content = self.rendered_content File "/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site- packages/django/template/response.py", line 90, in rendered_content template = self.resolve_template(self.template_name) File "/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site- packages/django/template/response.py", line 72, in resolve_template return select_template(template, using=self.using) File "/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site- packages/django/template/loader.py", line 47, in select_template raise TemplateDoesNotExist(", ".join(template_name_list), chain=chain) django.template.exceptions.TemplateDoesNotExist: seguimiento_silabo/seguimiento/crear.html </pre>

Tabla 9. Caso de Prueba 4 para la aplicación SIAAF.

Caso de Prueba	Registro de Log
Permiso denegado al iniciar sesión	<pre> [ERROR] [2024-11-15 14:42:21,456] [log django.security 403] [Forbidden: /seguridad/accounts/login] Traceback (most recent call last): File "/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site- packages/django/core/handlers/exception.py", line 47, in inner response = get_response(request) File </pre>

	<pre> "/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site- packages/django/middleware/common.py", line 53, in process_request if not self.check_permissions(user): File "/ruta_aplicacion/proyectos/siaaf/app/seguridad/middleware.py", line 112, in check_permissions raise PermissionDenied("User does not have permission to access this view.") django.core.exceptions.PermissionDenied: User does not have permission to access this view. </pre>
--	---

Tabla 10. Caso de Prueba 5 para la aplicación SIAAF.

Caso de Prueba	Registro de Log
Host inválido	<pre> [ERROR] [2024-11-15 18:32:19,333] [log django.request 400] [Bad Request: /] Traceback (most recent call last): File "/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site- packages/django/core/handlers/exception.py", line 47, in inner response = get_response(request) File "/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site- packages/django/core/handlers/base.py", line 181, in _get_response self.check_host(request) File "/ruta_aplicacion/proyectos/siaaf/app/core/utils/host_check.py", line 45, in check_host raise SuspiciousOperation("Invalid HTTP_HOST header: '0.0.0.0'") django.core.exceptions.SuspiciousOperation: Invalid HTTP_HOST header: '0.0.0.0' </pre>

Casos de Prueba - Aplicación SIAAF

Para la aplicación de los casos de prueba, se utilizaron los modelos LLM LLaMA3, Gemma2 y Mistral, seleccionados por sus características específicas. Cada modelo fue evaluado utilizando el prompt correspondiente, con el objetivo de generar respuestas más precisas, rápidas y optimizadas.

Tabla 11. Casos de prueba aplicación SIAAF.

ID Caso	Log Inicial (Django /Nginx)	Descripción del Caso	Modelo de IA	Recomendación Generada	Tiempo de Respuesta (s)	Validación
CP-SIAAF-001	Tabla 6. Error generado por django	El sistema no encuentra la vista cas_ng_login debido a una configuración incorrecta en CAS_LOGIN_URL_NAME.	Llama3	<p>***Análisis del log** El log indica un error de tipo `Internal Server Error` que se produjo en el servidor al intentar acceder a la ruta `/admin/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php` . El error se generó en el marco de Django y específicamente en el middleware `django_cas_ng/middleware.py` .</p> <p>**Identificación del archivo y línea** La excepción se produjo en la función `process_view` del middleware `django_cas_ng/middleware.py`, específicamente en la línea 65.\n\n**Posibles causas del error** La causa más probable del error es que el nombre de la vista `cas_ng_login` no existe o no está definido correctamente. Esto se confirma por el mensaje de error `NoReverseMatch: Reverse for 'cas_ng_login' not found` .</p> <p>**Solución directa y práctica** Para resolver este problema, debemos asegurarnos de que la vista `cas_ng_login` esté bien definida en nuestro código Django. Puede ser que se haya eliminado o renombrado accidentalmente.\n\nRevisar los archivos de código y verificar si existe una vista con ese nombre. Si no es así, debemos agregar una vista con ese nombre o reemplazarla por una que exista.\n\nAdemás, también podemos intentar solucionarlo mediante la modificación del archivo `urls.py` para asegurarnos de que la ruta correspondiente esté bien definida y disponible. En resumen, la solución pasa por localizar y corregir el problema en la vista `cas_ng_login`, ya sea creando una nueva o reemplazándola por una existente."</p>	38.8 s	Validado
			Mistral	<p>1. El problema identificado en este log es que no se ha encontrado la URL especificada `cas_ng_login` para ser utilizada por el middleware CAS (Central Authentication Service). Esto significa que el sistema no puede acceder al URL de inicio de sesión del servicio de autenticación central.\n 2. El error se originó en la línea 241 del archivo `log django.request` .\n 3. Las posibles</p>	34.87 s	Validado

				<p>causas del error incluyen:\n - No tener definida la URL `cas_ng_login` en las configuraciones de los enlaces de tu aplicación Django.\n - La url `cas_ng_login` no está siendo reconocida debido a una inconsistencia en su nombre o que se encuentre en un módulo distinto del que se esperaba.\n 4. Para resolver el problema, revisa la configuración de tus URLs en Django y asegúrate de que esté definida la url `cas_ng_login`. Si no la encuentras, agregala correctamente según su nombre y módulo correspondiente. Luego reinicia el servidor para que los cambios tomen efecto.</p>		
			Gemma2	<p>## Análisis del log de error **Problema:** El sistema está enfrentando un error al intentar redirigir al usuario a la página de inicio de sesión CAS (Central Authentication Service). Se produce una excepción `NoReverseMatch` indicando que Django no puede encontrar la vista o patrón URL llamado "cas_ng_login". **Archivo y Línea del Error:** * El error se originó en el archivo: `ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site-packages/django_cas_ng/middleware.py` * La línea específica es el número `65`. **Posibles Causas:** 1. **Configuración Incorrecta de django_cas_ng:** Es posible que la configuración del módulo `django_cas_ng` no esté definida correctamente, especificando incorrectamente el nombre de la vista o patrón URL para la página de inicio de sesión CAS. 2. **Vista Ausente o Mal Nombreada:** La vista "cas_ng_login" puede estar ausente en el proyecto o su nombre podría haber cambiado. 3. **Error en el Patrón de URL:** El patrón de URL definido en los archivos `urls.py` del proyecto no coincide con el que espera django_cas_ng para la página de inicio de sesión CAS. **Solución Práctica (En Español):** 1. **Revisar la Configuración de django_cas_ng:** Verifica si las opciones como `CAS_LOGIN_URL_NAME` están configuradas correctamente en tu archivo `settings.py`. Asegúrate que coincida con el nombre de la vista o patrón URL utilizado para la página de inicio de sesión CAS. 2. **Confirmar la Existencia y Nombre de la Vista:** Busca la vista "cas_ng_login" dentro de tus archivos de vistas (generalmente en `views.py`). Si no existe, debes crearla. ``python from django.shortcuts import render def cas_ng_login(request): # Aquí iría el código para la página de inicio de sesión CAS return render(request, 'cas_login.html') # Ejemplo de plantilla `` 3. **Verificar el Patrón de URL:** Asegúrate que el patrón de URL en tu archivo `urls.py` coincida con la vista "cas_ng_login" y la configuración de django_cas_ng. ``python from django.urls import path urlpatterns = [path('admin/', admin.site.urls), # ... Otras rutas path('accounts/login/', cas_ng_login, name='cas_ng_login'), # Asegúrate que coincida con la vista y configuración de django_cas_ng]</p>	274.93 segundos	Validado
CP-SIAAF-002	Tabla 7. Error generado por django	Error en query con ID nulo	LLaMA3	<p>**Análisis del log de error** El log indica un error interno (500 Internal Server Error) en la aplicación Django. El error se produce cuando se intenta acceder a la ruta `/portafolio/adjunto/lista`. La excepción se genera en el archivo `views.py` y línea 225. **Identificación del archivo y línea** El error se produce en el archivo `views.py`, línea 225, dentro de la función `adjunto_lista`. **Análisis posible de causas** La causa principal del error es un intento de acceder al atributo `id` de un objeto que es `None`. Esto sugiere que la variable `portafolio` no tiene un valor asignado. En el contexto de la función `puede_ver`, esto puede deberse a que la consulta SQL (`qset.filter(id=portafolio.id).exists()`) intenta filtrar por un ID que no existe. **Solución práctica** Para resolver este problema, debemos asegurarnos de que la variable `portafolio` tenga un valor asignado antes de intentar acceder a su atributo `id`. Esto puede lograrse mediante validaciones adicionales o comprobando que el objeto sea instancia de una clase determinada. En concreto, se podría</p>	64.15 segundos	Validado

				<p>agregar un condicional para verificar si `portafolio` es `None` antes de intentar acceder a su atributo `id`: ``python if portafolio is not None and portafolio.id: if PortafolioSecService.puede_ver(portafolio, request.user) is False: # ... `` De esta manera, se evita el acceso a un objeto que no tiene un valor asignado y se minimiza la posibilidad de errores futuros.</p>		
			Mistral	<p>1. El error presentado es un `AttributeError` y se muestra en la línea 225 del archivo "/opt/siaaf/siaaf-web/app/portafolio/services/views/views.py" dentro de la función "adjunto_lista". La causa del error es que el objeto "portafolio" pasado como argumento a la función "puede_ver" no tiene un atributo llamado "id", lo que provoca una excepción. 2. El error se generó en la línea 225 del archivo "/opt/siaaf/siaaf-web/app/portafolio/services/views/views.py". 3. Posibles causas del error: - La variable "portafolio" no ha sido inicializada o asignada correctamente antes de ser pasada como argumento a la función "puede_ver". - El código contiene un bug que hace que el objeto "portafolio" no se cargue correctamente con los datos correspondientes. - El acceso al objeto "portafolio" desde otra clase o método puede estar siendo realizado antes de que sea cargado, lo que provoca que en tiempo de ejecución el atributo "id" aún no haya sido asignado. 4. Una solución práctica para resolver este problema es: - Revisar la inicialización y asignación del objeto "portafolio" en su contexto correcto antes de ser pasado como argumento a la función "puede_ver". - Verificar que el código contenga una lógica correcta para cargar el objeto "portafolio" con los datos correspondientes. - Si se están accediendo a "portafolio" desde otra clase o método, verificar que se carga antes de ser utilizado en "puede_ver".</p>	41.69 segundos	Validado
			Gemma2	<p>## Análisis del Log de Error El log indica un error interno del servidor (500) que se produjo al acceder a la URL `/portafolio/adjunto/lista`. Problema: Se está produciendo un `AttributeError` al intentar acceder al atributo `id` de un objeto `NoneType`. Esto significa que una variable o propiedad esperada para tener el atributo `id` no contiene ningún valor. Ubicación del Error: El error se genera en el archivo `/opt/siaaf/siaaf-web/app/portafolio/security/portafolio.py`, línea 139. Posibles Causas: Variable `portafolio` nula: La variable `portafolio` no contiene un objeto válido antes de llamar al método `puede_ver`. Esto puede ocurrir por diferentes razones: * El método que devuelve `portafolio` podría estar retornando `None` si no se encuentra ningún portafolio relevante. * Podría haber una falla en la lógica del código que busca el portafolio, como una consulta mal formulada o un filtro incorrecto. Solución: 1. Revisar la lógica del método que devuelve `portafolio`. Verificar que realiza correctamente la búsqueda y filtrado del portafolio deseado, y que retorna un objeto válido en caso de éxito. 2. Comprobar la variable `portafolio` antes de acceder a su atributo `id`. Agregar una condición dentro del bloque `if` para verificar si `portafolio` es diferente de `None`: ``python if portafolio is not None and PortafolioSecService.puede_ver(portafolio, request.user) is False: # ... código restante ... `` Esto evitará que se produzca el `AttributeError` si `portafolio` es `None`. Implementar esta solución puede ayudar a resolver el problema de forma directa y práctica.</p>	158.84 segundos	Validado

CP-SIAAF-003	Tabla 8. Error generado por django	Error en template inexistente	LLaMA3	<p>**Análisis del log** El log indica que se produjo un error de tipo "Internal Server Error" (Error Interno del Servidor) en la aplicación Django. El error se generó al intentar renderizar una plantilla HTML en la ruta "/seguimiento-silabo/seguimiento/crear/22917".</p> <p>**Archivo y línea** El error se generó en el archivo `django/template/loader.py` en la línea 47.</p> <p>**Posibles causas del error** Basándome en el contenido del log, puedo identificar algunas posibles causas del error: 1. La plantilla HTML "seguimiento_silabo/seguimiento/crear.html" no existe o no se encuentra en el directorio correcto. 2. Hay un problema de routing en la aplicación Django que impide la renderización de la plantilla correctamente.</p> <p>**Solución** Para resolver este error, puedo sugerir las siguientes acciones: 1. Verificar si la plantilla HTML "seguimiento_silabo/seguimiento/crear.html" existe y se encuentra en el directorio correcto. 2. Revisar la configuración de routing de la aplicación Django para asegurarme de que esté correctamente configurada para renderizar esta plantilla. 3. Si no hay problemas con la ruta, puedo intentar acceder a la página directamente desde el navegador para ver si el error se reproduce.</p> <p>**Sugerencia práctica** Puedo sugerir una solución directa y práctica como la siguiente: * Revisar la estructura de directorios y archivos en el proyecto para asegurarme de que la plantilla HTML esté correctamente ubicada. * Verificar la configuración de routing en el archivo `urls.py` de la aplicación Django para asegurarme de que esté correctamente definido. * Si el error persiste, puedo intentar acceder a la página directamente desde el navegador para ver si el error se reproduce.</p>	71.64 segundos	Validado
			Mistral	<p>1. El error se debe a que no existe el archivo de plantilla `seguimiento_silabo/seguimiento/crear.html` en el directorio especificado para las plantillas del proyecto Django. Esto produce un Internal Server Error cuando se intenta renderizar la página en la URL `/seguimiento-silabo/seguimiento/crear/22917`. 2. El error se generó en el archivo `django.template.loader.py` en la línea 47. 3. Las causas más comunes de este error son: - La plantilla especificada no existe en el directorio correcto de las plantillas del proyecto Django. - El nombre de la plantilla está mal escrito o contiene errores en su ruta. - No se ha definido el directorio de plantillas para el proyecto Django en settings.py. 4. Una solución práctica para resolver este problema es asegurarse de que el archivo de plantilla `seguimiento_silabo/seguimiento/crear.html` se encuentre en el directorio correcto de las</p>	38.14 segundos	Validado

				plantillas del proyecto Django y esté escrito sin errores en su ruta. También se recomienda comprobar que el nombre del archivo de plantilla coincida con la plantilla especificada en la URL de la página en error. Finalmente, asegúrese de que el directorio de plantillas se haya definido correctamente en settings.py del proyecto Django.		
			Gemma2	<p>## Análisis del Log de Error El log indica un error interno del servidor (Internal Server Error) que se produjo al intentar acceder a la URL <code>/seguimiento-silabo/seguimiento/crear/22917</code>. Error específico: <code>django.template.exceptions.TemplateDoesNotExist: seguimiento_silabo/seguimiento/crear.html</code> Ubicación del error: El error ocurrió en el archivo <code>/ruta_aplicacion/.virtualenvs/siaafenv/lib/python3.9/site-packages/django/template/loader.py</code>, específicamente en la línea 47 durante la ejecución de <code>select_template(template, using=self.using)</code>. Posibles causas: 1. Archivo de plantilla inexistente: El archivo <code>seguimiento_silabo/seguimiento/crear.html</code> no existe en la ubicación donde Django lo busca. 2. Ruta incorrecta: La ruta a la plantilla está escrita incorrectamente, hay un error en el nombre del archivo o en la organización de las carpetas dentro de la aplicación 'seguimiento_silabo'. Solución: 1. Verificar la existencia del archivo: Asegúrate de que el archivo <code>seguimiento_silabo/seguimiento/crear.html</code> exista en la ubicación correcta dentro de tu proyecto Django. 2. Revisar la ruta: Comprueba que la ruta a la plantilla esté escrita correctamente. La estructura común es <code>app_name/templates/app_nombre/nombre_del_archivo.html</code>. En este caso, debería ser <code>seguimiento_silabo/templates/seguimiento_silabo/crear.html</code>. 3. Reiniciar el servidor: Después de realizar los cambios, reinicia el servidor web (por ejemplo, Apache o Nginx) para que Django cargue las nuevas configuraciones. Si sigues teniendo problemas después de estas comprobaciones, puedes revisar el archivo <code>settings.py</code> para asegurarte de que la configuración del directorio de plantillas está correcta.</p>	188.01 segundos	Validado
CP-SIAAF-004	Tabla 9. Error generad o por djang	Error en la autenticación de usuarios a través de la vista <code>/seguridad/accounts/login</code> debido a permisos.	Llama 3	<p>Análisis del log El log de error indica que se produjo una petición HTTP con código de estado 403 (Forbidden) al intentar acceder a la ruta <code>/seguridad/accounts/login</code>. El mensaje de error indica que el usuario no tiene permiso para acceder a ese view. Archivo y línea donde se generó el error El error se originó en el archivo <code>middleware.py</code> del proyecto, en la línea 112. Posibles causas del error La principal causa de este error es que el middleware de seguridad (<code>seguridad/middleware.py</code>) está verificando los permisos del usuario y no lo encuentra con los permisos necesarios para acceder a ese view. Es posible que la configuración de permisos esté incorrecta o que el usuario no tenga las credenciales adecuadas. Solución práctica Para resolver este problema, se recomienda revisar la configuración de permisos en el middleware de seguridad y asegurarse de que el usuario tenga los permisos necesarios para acceder a ese view. También es posible que sea necesario revisar la autenticación del usuario o ajustar las configuraciones de permisos en general. En resumen, la solución es: * Revisar la configuración de permisos en <code>seguridad/middleware.py</code> * Asegurarse de que el usuario tenga los permisos necesarios para acceder a ese view * Revisar la autenticación del usuario y ajustar las configuraciones de permisos si necesario.</p>	58.08 segundos	Validado

			Mistral	<p>1. El error es un `PermissionDenied` (Negado de permisos) que se generó en la ruta "/seguridad/accounts/login". Esto indica que el usuario no tiene los permisos necesarios para acceder a esta vista. 2. La línea donde se generó el error es `line 112` del archivo `middleware.py` ubicado en la carpeta `seguridad` del proyecto `siaaf` . 3. Posibles causas de este error incluyen: no tener configurados los permisos adecuados para el usuario, una implementación incorrecta de las funciones que controlan los permisos o un problema con la autenticación del usuario. 4. Una solución práctica podría ser verificar si se está configurando correctamente los permisos de acceso a las vistas en el sistema. También es importante revisar la implementación de las funciones que controlan los permisos y asegurarse de que el usuario esté autenticado antes de acceder a la vista "/seguridad/accounts/login". Además, se puede asegurar que los permisos estén configurados en la base de datos y que coincidan con las funciones que controlan los permisos.</p>	32.98 segundos	Validado
			Gemma2	<p>## Análisis del Log de Error **Problema:** El usuario que intenta acceder a la URL `/seguridad/accounts/login` no tiene los permisos necesarios para hacerlo. Se genera un error 403 (Forbidden) y se detiene el proceso de solicitud. **Ubicación del Error:** * **Archivo:** `/ruta_proyecto/siaaf/proyectos/siaaf/app/seguridad/middleware.py` * **Línea:** 112 **Posibles Causas:** * **Configuración incorrecta de permisos:** El grupo o usuario que intenta acceder no tiene los permisos asignados para la vista `/seguridad/accounts/login`. * **Problema en el middleware `check_permissions`:** El código en la línea 112 del archivo mencionado podría estar funcionando mal y devolviendo falso incluso para usuarios con permisos. * **Fallo en la verificación de roles o usuarios:** Si se utiliza un sistema de autenticación basado en roles, es posible que el usuario no pertenezca al rol necesario para acceder a esa vista. **Solución:** 1. **Verificar permisos del usuario:** Asegúrate de que el usuario que intenta acceder a `/seguridad/accounts/login` tenga los permisos necesarios asignados. Esto puede implicar agregarlos a un grupo con permisos específicos o asignarles permisos directamente. 2. **Revisar el código del middleware `check_permissions`:** Comprueba la lógica en la línea 112 del archivo `middleware.py`. Asegúrate de que la función esté verificando correctamente los permisos del usuario y retornando True si tiene acceso a la vista. Recuerda que estas son solo algunas posibles soluciones basadas en el log proporcionado. Para una solución definitiva, se necesita investigar más a fondo el código y la configuración del sistema.</p>	165.96 segundos	Validado
CP-SIAAF-005	Tabla 10. Error generado por django	Error en la configuración de ALLOWED_HOSTS con encabezados no permitidos, como '0.0.0.0' o '190.15.133.65'.	Llama 3	<p>**Análisis del log** El log de error indica que se produjo un error en el sistema al recibir una petición HTTP con código de estado 400 (Petición malformada). El error se originó en la aplicación Django y específicamente en el archivo `host_check.py` en la línea 45. **Descripción del problema** El error se produce cuando la solicitud HTTP contiene un valor inválido para el campo `HTTP_HOST`, que es '0.0.0.0'. Esto desencadena una excepción de tipo `SuspiciousOperation`, que indica que la petición es sospechosa y no puede ser procesada. **Posibles causas del error** 1. Configuración incorrecta en el servidor: Es posible que haya una configuración incorrecta en el servidor web o en el archivo de configuración de Django que está generando este error. 2. Petición malformada: La petición HTTP puede estar siendo enviada con un valor inválido para el campo `HTTP_HOST`, lo que desencadena el error. 3. Problema en la aplicación: Es posible que haya un problema en la lógica de negocio o en la configuración de la aplicación que está generando este error. **Solución**</p>	78.73 segundos	Validado

				<p>Para resolver este problema, se recomienda: 1. Verificar la configuración del servidor web y el archivo de configuración de Django para asegurarse de que no hay una configuración incorrecta. 2. Analizar las peticiones HTTP en el sistema para determinar si alguna petición está enviando un valor inválido para el campo `HTTP_HOST`. 3. Revisar la lógica de negocio y la configuración de la aplicación para asegurarse de que no hay un problema en la implementación. **Código de resolución** Para solucionar este error, se puede agregar un tratamiento especial para el valor '0.0.0.0' en el archivo `host_check.py`, como por ejemplo: <code>python def check_host(request): if request.META['HTTP_HOST'] == '0.0.0.0': raise SuspiciousOperation("Invalid HTTP_HOST header: '0.0.0.0") # Procesar la petición de manera normal ``` De esta forma, se puede evitar que el error se produzca cuando se reciba una petición con un valor inválido para el campo `HTTP_HOST`.</code></p>		
			Mistral	<p>1. El problema que presenta el sistema en producción es de tipo "Bad Request". Esto indica que el servidor no pudo interpretar la solicitud HTTP recibida debido a una sintaxis incorrecta o un error de argumentos en la URL. La excepción lanza por el código de Django (<code>django.core.exceptions.SuspiciousOperation</code>) indica que hay un problema con el encabezado HTTP_HOST, específicamente que no es válido '0.0.0.0'. 2. El error se generó en la línea 45 del archivo <code>"/ruta_proyecto/siaaf/app/core/utills/host_check.py"</code>. 3. Las posibles causas de este problema pueden ser: - Que la aplicación de Django no esté configurada correctamente en el archivo de settings para que el encabezado HTTP_HOST contenga el dominio o IP válida, o bien que haya un error en los middlewares o Views que manipulen este encabezado. - Que se esté accediendo al sistema desde una URL no válida (que no coincide con la configurada). 4. Para resolver el problema, revisar la configuración del dominio en el archivo de settings y asegurarse que esté correctamente definido y coincida con el encabezado HTTP_HOST recibido por el servidor. También es importante revisar cualquier código en middlewares o Views que manipulen este encabezado para asegurarse de que no esté causando errores. También puede ser necesario hacer una actualización del sistema y reiniciarlo para que se vean los cambios. Además, verificar la URL desde la cual se está accediendo al sistema para asegurarse de que es válida y coincide con la configuración.</p>	41.66 segundos	Validado
			Gemma2	<p>## Análisis del log de error El log indica un error "Bad Request" (Solicitud incorrecta) con código HTTP 400. El problema se encuentra en la validación del encabezado "HTTP_HOST" durante el manejo de una solicitud a la raíz del sitio ("/"). **Ubicación del error:** * Archivo: `host_check.py` * Línea: 45 **Posibles causas:** 1. **Configuración incorrecta del servidor web:** El servidor web (Apache, Nginx) podría estar enviando un encabezado "HTTP_HOST" con el valor "0.0.0.0", lo cual Django considera inválido. 2. **Ataque de falsificación de solicitudes:** Un atacante podría intentar enviar una solicitud con un encabezado "HTTP_HOST" manipulado para obtener acceso a recursos no autorizados. **Solución práctica:** Verifica la configuración del servidor web y asegúrate de que el encabezado "HTTP_HOST" se configure correctamente. * Para Apache, revisa la configuración del VirtualHost asociado al sitio. Asegúrate de que el valor del atributo `ServerName` coincida con el nombre de dominio o dirección IP pública que quieres utilizar. * Para Nginx, verifica la configuración del bloque `server`. El atributo `server_name` debe contener el nombre de dominio o dirección IP que deseas utilizar. Además, implementa medidas de seguridad para prevenir ataques de falsificación de solicitudes.</p>	127.15 segundos	Validado

6.1.2.2. Aplicación evaluada: Módulo de Trámites

Objetivo de los casos: Evaluar la detección y resolución de logs críticos mediante los modelos de IA (LLaMA3, Mistral, Gemma2).

Descripción: En esta evaluación se utilizan logs críticos registrados por el módulo de Trámites en su archivo de logs correspondiente, los cuales representan incidencias reales que requieren intervención inmediata.

A continuación, se detallan los siguientes casos de prueba:

Tabla 12. Caso de Prueba 1 para la aplicación Trámites.

Caso de Prueba	Registro de Log
<p>Error al enviar notificación correo electrónico</p>	<pre> 2022-12-06 08:55:29,020: ERROR/ForkPoolWorker-4] Task app.seguridad.tasks.notificaciones_enviar_correo[cec07675-692a- 4464-a7bc-8f465c414a90] raised unexpected: SMTPServerDisconnected('Connection unexpectedly closed') Traceback (most recent call last): File "/ruta_aplicacion/.virtualenvs/tramites_env/lib/python3.7/site- packages/celery/app/trace.py", line 385, in trace_task R = retval = fun(*args, **kwargs) File "/ruta_aplicacion/.virtualenvs/tramites_env/lib/python3.7/site- packages/celery/app/trace.py", line 650, in __protected_call__ return self.run(*args, **kwargs) File "/ruta_aplicacion/proyectos/tramites/app/seguridad/tasks.py", line 27, in notificaciones_enviar_correo enviar_correo(notificacion.asunto, texto, destinatarios) File "/ruta_aplicacion /proyectos/tramites/app/core/utils/enviar_correo.py", line 16, in enviar_correo html_message=mensaje, File "/ruta_aplicacion/.virtualenvs/tramites_env/lib/python3.7/site- packages/django/core/mail/__init__.py", line 60, in send_mail return mail.send() File "/ruta_aplicacion/.virtualenvs/tramites_env/lib/python3.7/site- packages/django/core/mail/message.py", line 291, in send return self.get_connection(fail_silently).send_messages([self]) File "/ruta_aplicacion/.virtualenvs/tramites_env/lib/python3.7/site- packages/django/core/mail/backends/smtp.py", line 103, in send_messages new_conn_created = self.open() File "/ruta_aplicacion/.virtualenvs/tramites_env/lib/python3.7/site- packages/django/core/mail/backends/smtp.py", line 70, in open self.connection.login(self.username, self.password) File "/usr/lib/python3.7/smtplib.py", line 721, in login initial_response_ok=initial_response_ok) File "/usr/lib/python3.7/smtplib.py", line 631, in auth (code, resp) = self.docmd("AUTH", mechanism + " " + response) File "/usr/lib/python3.7/smtplib.py", line 421, in docmd </pre>

	<pre> return self.getreply() File "/usr/lib/python3.7/smtplib.py", line 394, in getreply raise SMTPServerDisconnected("Connection unexpectedly closed") smtplib.SMTPServerDisconnected: Connection unexpectedly closed </pre>
--	---

Tabla 13. Caso de Prueba 2 para la aplicación Trámites.

Caso de Prueba	Registro de Log
Error de tipos al realizar operaciones.	<pre> [2022-12-06 08:55:32,547: ERROR/ForkPoolWorker-4] Task app.tramite.tasks.orden_pago_consultar_siaaf[398e3343-007b-4c61- 8a08-cc9664fd264d] raised unexpected: TypeError('string indices must be integers') Traceback (most recent call last): File "/ruta_aplicacion/.virtualenvs/tramites_env/lib/python3.7/site- packages/celery/app/trace.py", line 385, in trace_task R = retval = fun(*args, **kwargs) File "/ruta_aplicacion/.virtualenvs/tramites_env/lib/python3.7/site- packages/celery/app/trace.py", line 650, in __protected_call__ return self.run(*args, **kwargs) File "/ruta_aplicacion/proyectos/tramites/app/tramite/tasks.py", line 21, in orden_pago_consultar_siaaf print('Orden pago ID: %s - Orden pago ID-SIAAF: %s - Respuesta: %s' % (orden_pago.id, orden_pago.orden_pago_id, respuesta['estado'])) TypeError: string indices must be integers </pre>

Tabla 14. Caso de Prueba 3 para la aplicación Trámites.

Caso de Prueba	Registro de Log
Error al concatenar string.	<pre> [2022-12-06 08:55:32,779: WARNING/ForkPoolWorker-4] Orden pago no actualizada [2022-12-06 08:55:32,779: WARNING/ForkPoolWorker-4] Orden pago ID: 3690 - Orden pago ID-SIAAF: None - Respuesta: can only concatenate str (not "NoneType") to str </pre>

Tabla 15. Caso de Prueba 4 para la aplicación Trámites.

Caso de Prueba	Registro de Log
Conexión interrumpida con el servidor al consultar detalles de proceso	<pre> 2021/04/19 10:13:03 [error] 4810#4810: *28 upstream prematurely closed connection while reading response header from upstream, client: 190.11.7.193, server: tramites.unl.edu.ec, request: "GET /tramite/proceso/editar/65 HTTP/1.1", upstream: "http://127.0.0.1:8000/tramite/proceso/editar/65", host: "tramites.unl.edu.ec" </pre>

Tabla 16. Caso de Prueba 5 para la aplicación Trámites.

Caso de Prueba	Registro de Log
Timeout al consultar detalles de proceso	2021/04/19 10:19:56 [error] 4810#4810: *284 upstream timed out (110: Connection timed out) while reading response header from upstream, client: 45.161.33.150, server: tramites.unl.edu.ec, request: "GET /tramite/proceso/lista/5 HTTP/1.1", upstream: "http://127.0.0.1:8000/tramite/proceso/lista/5", host: "tramites.unl.edu.ec"

Tabla 17. Casos de prueba aplicación Trámites.

ID Caso	Log Inicial (Django/Nginx)	Error	Modelo de IA	Recomendación Generada	Tiempo de Respuesta (ms)	Validación
CP-TRAMITE S-001	Tabla 12. Error generado por django	Error al enviar notificación correo electrónico	Llama3	<p>**Análisis del log** El log indica que un error inesperado (SMTPServerDisconnected) se generó en el sistema a las 8:55:29 horas del 6 de diciembre de 2022. El error se produjo mientras se ejecutaba la tarea `notificaciones_enviar_correo` perteneciente al módulo `seguridad/tasks.py`.</p> <p>**Identificación del archivo y línea** El error se generó en el archivo `ruta_proyecto/tramites/app/seguridad/tasks.py`, línea 27, en la función `notificaciones_enviar_correo`. Luego, el error se propagó a los archivos `ruta_proyecto/tramites/.virtualenvs/tramites_env/lib/python3.7/site-packages/django/core/mail/__init__.py` (línea 60) y `usr/lib/python3.7/smtplib.py` (líneas 721, 631 y 421).</p> <p>**Posibles causas del error**</p> <ol style="list-style-type: none"> 1. Problemas con la configuración de SMTP: es posible que el servidor de correo electrónico esté desconnectando inesperadamente o no esté configurado correctamente. 2. Conexiones indebidas a la base de datos: si hay problemas con las conexiones a la base de datos, puede provocar errores en la envío de correos electrónicos. 3. Problemas con el módulo `celery`: es posible que haya un problema en la configuración o ejecución del worker de Celery. <p>**Solución directa y práctica** Para resolver este problema, te sugiero las siguientes acciones:</p> <ol style="list-style-type: none"> 1. Verificar la configuración de SMTP: asegúrate de que la configuración de SMTP esté correcta y que no haya problemas con el servidor de correo electrónico. 2. Revisar los logs de error para encontrar más información sobre el error: puede haber otros errores o warning relacionados con el problema que se están produciendo. 3. Probar el envío de correos electrónicos desde la línea de comandos: si puedes enviar correos electrónicos manualmente, es posible que sea un problema específico de la tarea `notificaciones_enviar_correo`. 4. Verificar la configuración del worker de Celery: asegúrate de que el worker esté correctamente configurado y ejecutándose sin problemas. <p>En resumen, este error se debe a una conexión SMTP inesperadamente cerrada. Para resolver el problema, debemos investigar y verificar las configuraciones de SMTP, revisar los logs de error y probar el envío de correos electrónicos desde la línea de comandos.</p>	87.24 segundos	Validado

			Mistral	<p>1. El problema que se presenta en el log de error es la conexión inesperada cerrada al enviar un correo electrónico por medio del servidor SMTP (Simple Mail Transfer Protocol). En otras palabras, no se pudo establecer una comunicación estable y sólida con el servidor SMTP. 2. El error se generó en la línea 16 del archivo <code>`/ruta_proyecto/tramites/app/core/utills/enviar_correo.py`</code>, ya que es aquí donde se llama a la función <code>`send()`</code> de un objeto <code>`message`</code>. 3. Las posibles causas del error incluyen: - Errores en la configuración del servidor SMTP, como la URL o el usuario y contraseña incorrectos. - Problemas con la conexión a Internet o con el servidor SMTP específico (por ejemplo, si el servidor está bloqueado). - Errores en el código que llama a la función <code>`send()`</code>, como errores de sintaxis o lógica. 4. Una solución práctica para resolver este problema es revisar las configuraciones del servidor SMTP (URL, usuario y contraseña) y verificar que sean correctas. Si las configuraciones son correctas, puede ser necesario comprobar el estado del servidor SMTP o corregir eventuales errores de código en la función <code>`enviar_correo()`</code>. Además, puede ser útil intentar enviar un correo electrónico manualmente desde su aplicación para verificar que todo esté funcionando correctamente.</p>	43.43 segundos	
			Gemma2	<p>## Análisis del Log de Error El log indica que se produjo un error al intentar enviar un correo electrónico mediante la tarea <code>`notificaciones_enviar_correo`</code>. La excepción específica es <code>`SMTPServerDisconnected('Connection unexpectedly closed')`</code>, lo cual significa que la conexión con el servidor SMTP se cerró inesperadamente.</p> <p>**1. Ubicación del Error:** El error se generó en el archivo <code>`/ruta_proyecto/tramites/app/core/utills/enviar_correo.py`</code>, específicamente en la línea 16.</p> <p>**2. Posibles Causas:** * **Problemas de configuración del servidor SMTP:** Las credenciales de acceso (usuario y contraseña) pueden ser incorrectas, el host SMTP puede estar mal configurado o el puerto utilizado no sea el correcto. * **Conexión a internet inestable:** La conexión a internet podría estar intermitente o interrumpida durante el proceso de envío del correo electrónico. * **Servidor SMTP sobrecargado o inaccesible:** El servidor SMTP podría estar recibiendo un volumen excesivo de correos electrónicos o estar temporalmente fuera de servicio. * **Problema en la aplicación:** Un error interno en la aplicación <code>`tramites`</code> podría estar causando que se cierre la conexión con el servidor SMTP inesperadamente.</p> <p>**3. Solución Directa:** 1. **Verificar la configuración del servidor SMTP:** Asegúrate de que las</p>	137.52 segundos	Validado

				<p>credenciales (usuario y contraseña) son válidas, el host SMTP es correcto y el puerto utilizado coincide con la configuración del servidor.</p> <p>2. **Revisar la conexión a internet:** Comprueba si hay problemas de conectividad a internet durante el proceso de envío de correos electrónicos. Si es posible, intenta enviar un correo electrónico desde otro servicio o aplicación para verificar la conexión.</p> <p>3. **Contactar al proveedor del servidor SMTP:** Si sospechas que el problema reside en el servidor SMTP, contacta con tu proveedor de servicios para informar del error y solicitar ayuda.</p> <p>4. **Debuguear el código de la aplicación:** Utiliza herramientas de depuración para identificar si existe algún error dentro del código que esté causando que la conexión se cierre inesperadamente.</p>		
CP-TRAMITE S-002	Tabla 13. Error generado por django	Error de tipos al realizar operaciones	Llama3	<p>**Análisis del log** El log indica que el sistema ha generado un error en la aplicación "Task" (probablemente una tarea de trabajo en segundo plano) denominada "notificaciones_enviar_correo". El error es de tipo `SMTPServerDisconnected('Connection unexpectedly closed')`, lo que sugiere problemas al conectarse a un servidor de correo electrónico. **Archivo y línea donde se generó el error** El error se generó en la función `enviar_correo` del archivo `ruta_proyecto/tramites/app/core/utills/enviar_correo.py` en la línea 16. Esta función es llamada desde la tarea `notificaciones_enviar_correo` definida en el archivo `ruta_proyecto/tramites/app/seguridad/tasks.py` en la línea 27. **Posibles causas del error** 1. Problemas de conexión con el servidor de correo electrónico: Es posible que el servidor de correo electrónico esté experimentando problemas o no esté disponible. 2. Configuración incorrecta de la cuenta de correo electrónico: La configuración de la cuenta de correo electrónico puede estar incorrecta, lo que impide la conexión al servidor de correo electrónico. 3. Problemas con la autenticación SMTP: El error `SMTPServerDisconnected` sugiere problemas con la autenticación SMTP, posiblemente debido a una contraseña o usuario incorrecto. **Solución directa y práctica** Para resolver el problema, se recomienda: 1. Verificar la configuración de la cuenta de correo electrónico y asegurarse de que sea correcta. 2. Revisar los logs del servidor de correo electrónico para detectar posibles problemas o errores. 3. Probar la conexión a un servidor de correo electrónico diferente para determinar si el problema es específico de una cuenta o un servidor en particular. En resumen, se recomienda realizar una revisión exhaustiva de la configuración de la cuenta de correo electrónico y los logs del servidor de correo electrónico para determinar la causa raíz del error.</p>	71.60 segundos	Validado
			Mistral	1. El problema en el log indica que se ha producido un error inesperado al enviar	49.43 segundos	Validado

				<p>un correo electrónico por parte de la tarea `notificaciones_enviar_correo`. La excepción específica que se generó es `SMTPServerDisconnected('Connection unexpectedly closed')`, lo que sugiere que el servidor SMTP del correo electrónico se cerró de manera inesperada. 2. El error fue generado en la línea 27 del archivo `ruta_proyecto/tramites/app/seguridad/tasks.py` y se produjo al llamar a la función `enviar_correo()`. 3. Al analizar el trazado de errores, es posible que las siguientes causas estén involucradas en el problema: - El servidor SMTP del correo electrónico está sufriendo problemas o no se encuentra disponible. - La configuración del correo electrónico (usuario y contraseña) proporcionada es incorrecta o incompleta. 4. Como solución práctica a este problema, puedes intentar lo siguiente: - Verifica que el servidor SMTP del correo electrónico esté disponible y funcione correctamente. Si el servidor está sufriendo problemas, puedes ponerlo en mantenimiento o esperar a que los responsables del servidor corrijan el problema. - Comprueba que la configuración de correo electrónico (usuario y contraseña) proporcionada sea correcta e iniciales correctas. Si es necesario, actualiza esta información en el código o consulte al administrador del servidor SMTP si tiene dudas sobre la configuración. - También puedes intentar enviar un correo electrónico manualmente desde tu sistema para verificar si el problema se replica en otras acciones que involucren el uso del servidor SMTP. Si el problema persiste, puede ser necesario investigar sobre soluciones alternativas al servicio de correo electrónico actual o consultar a un expertos para obtener una opinión profesional sobre el problema y cómo resolverlo.</p>		
			Gemma2	<p>## Análisis del log de error El log indica un problema con la conexión SMTP durante el envío de correos electrónicos. Error: `SMTPServerDisconnected('Connection unexpectedly closed')` Ubicación del error: * Archivo: `ruta_proyecto/tramites/app/core/utills/enviar_correo.py` Línea: 16 (dentro de la función `enviar_correo`) Posibles causas: * Problemas con el servidor SMTP: El servidor de correo electrónico puede estar caído, sobrecargado o tener problemas técnicos. * Credenciales incorrectas: Las credenciales de usuario y contraseña para acceder al servidor SMTP pueden ser incorrectas. * Firewall o configuración de seguridad: Un firewall u otra configuración de seguridad puede estar bloqueando la conexión a través del puerto SMTP (puerto 25 por defecto). * Configuración incorrecta en el código: Puede haber un error en la configuración del cliente SMTP, como una dirección de servidor incorrecta o un puerto diferente al predeterminado. Solución práctica: 1. Verificar el estado del servidor SMTP: Contacta con tu proveedor de hosting o verifica el estado del servidor SMTP en línea. 2. Revisa las credenciales: Asegúrate de que las credenciales de usuario y contraseña para acceder al servidor SMTP sean correctas. 3. Comprueba la configuración de seguridad: Revisa tu firewall y configuraciones de seguridad para asegurarte de que no estén bloqueando conexiones a través del puerto SMTP. Si es necesario, crea excepciones para el dominio o IP del servidor</p>	237.60 segundos	Validado

				SMTP. 4. **Verifica la configuración en el código:** Asegúrate de que la configuración del cliente SMTP en `enviar_correo.py` sea correcta: dirección del servidor SMTP, puerto, credenciales, etc. Puedes utilizar herramientas como telnet o nc (netcat) para probar la conexión al servidor SMTP desde tu máquina local.		
CP-TRAMITE S-003		Error al concatenar string	Llama3	<p>**Análisis del log** El log de error indica que se produjo un problema relacionado con la orden de pago no actualizada. El mensaje de error especifica que se intentó concatenar un string con un valor de tipo "NoneType" (nulo), lo que no es permitido.</p> <p>**Archivo y línea donde se generó el error** No hay información específica sobre el archivo y línea donde se generó el error, ya que el log no proporciona esa información. Sin embargo, podemos deducir que el error se produjo en algún punto de la lógica del sistema relacionada con la orden de pago.</p> <p>**Posibles causas del error** 1. El valor `None` se está utilizando donde no debería, probablemente en un campo que espera un string. 2. La función o método que intenta concatenar el string y el valor nulo no está tratando adecuadamente los valores nulos. 3. Existe una dependencia con otro sistema o servicio que no está disponible (o no devuelve un valor) y eso hace que el flujo de la aplicación fallen.</p> <p>**Solución** Para resolver este problema, se recomienda:</p> <ol style="list-style-type: none"> 1. Revisar la lógica del sistema relacionada con la orden de pago y asegurarse de que se estén utilizando correctamente los valores nulos. 2. Verificar si existe algún servicio o dependencia externa que no esté disponible y ajustar el código para manejar esos casos de error. 3. Probar la aplicación con diferentes escenarios y datos para garantizar que el problema no se repita. <p>**Código sugerido** Para evitar errores como este, se puede implementar un simple check antes de intentar concatenar strings:</p> <pre> python if value is not None: # Concatenar strings aquí else: </pre>	68.39 segundos	Validado

				<pre># Manejar el caso de valor nulo aquí ... </pre> <p>De esta manera, se evita la tentativa de concatenar un string con un valor nulo y se puede manejar el error de manera adecuada.</p>		
			Mistral	<p>1. El problema que se presenta en el log es que la actualización de una orden de pago no se pudo realizar correctamente debido a un error de tipo `NoneTypeError`. Este error sucede porque se intentó concatenar un string con un valor de tipo `None`, lo que no es posible. 2. El error se generó en la línea 779 del archivo 'OrdenPagoManager.py', donde está definida la función encargada de actualizar una orden de pago. 3. Las causas posibles del error pueden ser: a) Que alguna variable utilizada en la concatenación no se haya asignado un valor antes de ser usada, o b) Que algún método que devuelve `None` no esté funcionando correctamente y devuelva el valor `None`. 4. Una solución práctica para resolver este problema podría ser revisar la función encargada de actualizar una orden de pago en 'OrdenPagoManager.py', buscando posibles variables no inicializadas que se estén utilizando en la concatenación y asignándoles un valor adecuado, o bien revisar el método que devuelve `None` para corregir cualquier error que pueda estar ocurriendo. También podría ser necesario realizar pruebas unitarias para asegurarse de que todas las funciones estén funcionando correctamente.</p>	34.00 segundos	Validado
			Gemma2	<p>## Análisis del Log de Error **Problema:** El sistema está intentando concatenar una cadena de texto (`str`) con un valor `None`, lo cual genera un error de tipo. El mensaje "can only concatenate str (not "NoneType") to str" indica claramente este problema. **Ubicación del Error:** Este error se generó en el archivo que registra la advertencia "[WARNING] [2022-12-06 08:55:32,779] [Orden pago no actualizada]". Para localizar la línea exacta del código donde ocurre la concatenación incorrecta, necesitarías revisar el archivo fuente asociado con este log. **Posibles Causas:** * **Variable sin inicializar:** La variable que se intenta concatenar con la cadena de texto podría no estar inicializada correctamente y contener el valor `None`. * **Llamada a función fallida:** Una función llamada dentro del código podría estar devolviendo `None` en lugar del valor esperado (una cadena de texto). * **Fallo en la lógica condicional:** Un bloque de código que evalúa una condición y realiza acciones según el resultado podría estar generando `None` en un caso específico, lo cual lleva a la concatenación incorrecta. **Solución:** 1. **Identifica la variable o función:** Revisa el código alrededor del error para determinar qué variable o función está devolviendo `None`. 2. **Verifica la inicialización:** Si es una variable, asegúrate de que esté correctamente inicializada con un valor apropiado antes de ser utilizada en la concatenación. 3. **Corrige la lógica:** Si la causa es una llamada a función fallida o lógica condicional, analiza el código para identificar y corregir la fuente del problema. Asegúrate de que todas las funciones devuelvan valores válidos (como cadenas</p>	179.07 segundos	Validado

				de texto) cuando sea necesario. En resumen, el error se debe a una concatenación incorrecta de una cadena con un valor `None`. La solución implica encontrar el origen del valor `None` y asegurar que la variable o función devuelva un valor adecuado para la concatenación.		
CP-TRAMITE S-004		La conexión entre Nginx y el servidor backend (Django) se cierra inesperadamente al consultar un proceso específico.	Llama3	<p>**Análisis del log** El log indica un error de tipo [ERROR] que ocurre en el sistema en producción. El mensaje específico sugiere que la conexión con el upstream (probablemente un servicio o microservicio) se cierra prematuramente mientras se está leyendo el encabezado de respuesta desde ese upstream. **Ubicación del error** El archivo y línea exactos donde se generó el error no están especificados en el log, pero podemos deducir que el problema se produce en la parte de la aplicación que interactúa con el upstream (probablemente un servicio o microservicio). **Posibles causas del error** Algunas posibles causas del error podrían ser: 1. **Conexión interrumpida**: La conexión entre el cliente y el upstream se cierra prematuramente, lo que impide la recepción de la respuesta completa. 2. **Error en el upstream**: El servicio o microservicio upstream puede estar experimentando un problema o error que hace que no pueda enviar una respuesta correcta. 3. **Configuración incorrecta**: La configuración del upstream o la comunicación entre los servicios podría estar mal configurada, lo que provoca la conexión prematura. **Solución directa y práctica** Para resolver este problema, te sugiero: 1. **Verificar la configuración del upstream**: Asegúrate de que la configuración del upstream esté correcta y que no haya errores en el servicio o microservicio. 2. **Realizar un seguimiento de las conexiones**: Utiliza herramientas de monitoreo para seguir el flujo de conexiones entre el cliente y el upstream, lo que puede ayudarte a identificar patrones o problemas recurrentes. 3. **Implementar mecanismos de reintentos**: Considera implementar mecanismos de reintentos en la aplicación para tratar de reconectarse con el upstream en caso de que la conexión se cierre prematuramente. **Conclusión** En resumen, el error indica una conexión prematura con el upstream mientras se está leyendo el encabezado de respuesta. Es importante verificar la configuración del upstream y realizar un seguimiento de las conexiones para identificar posibles problemas y implementar mecanismos de reintentos para tratar de resolver el problema.</p>	70.17 segundos	Validado
			Mistral	<ol style="list-style-type: none"> 1. El error indica que se produjo un cierre prematuro de la conexión del cliente mientras se estaba leyendo una respuesta de origen remoto (upstream) en el servidor tramites.unl.edu.ec. Esto implica que el servidor no pudo enviar la respuesta completa a la solicitud HTTP que fue realizada por un cliente, con dirección IP 190.11.7.193. 2. El error se generó en la línea que contiene la etiqueta "[Upstream]". Más específicamente, el error se produjo en la línea "Upstream: "http://127.0.0.1:8000/tramite/proceso/editar/65"", donde se define la ruta de upstream que se está utilizando. 	42.05 segundos	Validado

				<p>3. Algunas posibles causas del error pueden ser un problema en la conexión entre el servidor y el origen remoto, una sobrecarga del servidor, problemas con la configuración o código del programa en ejecución, o un bloqueo de recursos (como memoria o CPU) por parte del servidor.</p> <p>4. Una solución práctica para resolver el problema puede ser:</p> <ul style="list-style-type: none"> a. Verificar la conexión entre el servidor y el origen remoto para asegurarse que no haya problemas de red o pérdida de conexión. b. Revisar la configuración de servicio en ejecución (como Nginx, Apache, u otro) para asegurarse que esté correctamente configurado y que no existan problemas en las rutas definidas o en los parámetros de configuración. c. Si se está utilizando algún programa para generar la respuesta del servidor, revisar el código del mismo para asegurarse que no haya errores que estén bloqueando recursos o causando problemas con las conexiones. d. Monitorizar y ajustar los recursos utilizados por el servidor para evitar sobrecargas. 		
			Gemma2	<p>## Análisis del Log de Error **Problema:** El servidor web "tramites.unl.edu.ec" recibió una solicitud HTTP ("GET /tramite/proceso/editar/65 HTTP/1.1") desde el cliente 190.11.7.193, pero la conexión con el servicio que atiende esa solicitud (en este caso "http://127.0.0.1:8000/tramite/proceso/editar/65") fue cerrada prematuramente mientras se leía la cabecera de respuesta del servidor. **Ubicación del Error:** El error ocurrió en el archivo responsable de manejar las solicitudes entrantes al servidor web "tramites.unl.edu.ec". Sin embargo, sin información adicional, es imposible determinar el nombre específico del archivo y la línea donde se generó el error. **Posibles Causas:** * **Problema en el servicio backend (http://127.0.0.1:8000):** El servicio backend podría estar caído, tener problemas de configuración o recursos insuficientes para atender la solicitud. * **Problemas de red:** Puede haber una interrupción en la conexión entre "tramites.unl.edu.ec" y el servicio backend ("http://127.0.0.1:8000"). * **Error en el código del servidor web:** Un error en el código del servidor web podría estar causando que cierre prematuramente la conexión con el servicio backend. **Solución Práctica:** 1. **Verificar el estado del servicio backend (http://127.0.0.1:8000):** * Intenta acceder al servicio backend desde otro dispositivo o navegador. * Revisa los logs del servicio backend en busca de errores relacionados con la solicitud. 2. **Revisar las configuraciones de red:** Asegúrate de que no haya problemas de firewall o configuración de red que estén impidiendo la comunicación entre "tramites.unl.edu.ec" y el servicio backend. 3. **Comprobar los logs del servidor web (tramites.unl.edu.ec):** Busca errores adicionales en los logs del servidor web que puedan proporcionar más información sobre la causa del problema. 4. **Debuggear el código del servidor web:** Si se sospecha de un error en el código, revisa cuidadosamente el código relacionado con el manejo de las solicitudes HTTP y las conexiones al servicio backend. Recuerda que para una</p>	150.38 segundos	Validado

				solución definitiva necesitas acceder a los logs del servidor web y del servicio backend para obtener información más precisa sobre la causa del error.		
CP-TRAMITE S-005		El servidor Nginx genera un timeout al no recibir respuesta del backend en el tiempo configurado.	Llama3	<p>**Análisis del log** El log de error indica que el sistema ha experimentado un timeout al intentar leer la respuesta del upstream (un servicio o API externo) mientras se está leyendo el encabezado de la respuesta. El mensaje de error específico es "upstream timed out (110: Connection timed out)". **Archivo y línea** No hay información disponible sobre el archivo y línea en el que se generó el error, ya que el log no proporciona esa información. **Posibles causas del error** 1. **Conexión a upstream fallida**: La conexión con el upstream puede haber fallado debido a un problema de red o al servidor upstream no estar disponible. 2. **Timeout en la lectura de la respuesta**: El sistema puede estar esperando demasiado tiempo para recibir una respuesta del upstream, lo que puede ser causado por un problema en el servicio upstream o una congestión en la red. 3. **Error en la configuración del upstream**: La configuración del upstream puede estar incorrecta, lo que puede llevar a errores de conexión o lectura de respuestas. **Solución** Para resolver este problema, se sugiere: 1. **Verificar el estado del upstream**: Comprueba si el servicio upstream está disponible y no hay problemas técnicos en él. 2. **Aumentar la timeout de lectura de respuesta**: Ajusta la configuración del sistema para aumentar el tiempo de espera antes de considerar un error de conexión. 3. **Revisar la configuración del upstream**: Revisa la configuración del upstream y asegúrate de que esté correcta y no haya errores de conexión. En resumen, el problema es un timeout en la lectura de la respuesta del upstream, lo que puede ser causado por una conexión fallida o una congestión en la red. La solución consiste en verificar el estado del upstream, ajustar la timeout de lectura de respuesta y revisar la configuración del upstream.</p>	54.45 segundos	Validado
			Mistral	<p>1. El error se produce porque la conexión entre el cliente y el servidor está demorando o ha fallado al leer los encabezados de respuesta del servidor en tiempo limitado. Esto sucede en la URL `http://127.0.0.1:8000/tramite/proceso/lista/5`. 2. La traza del error se generó en la línea 1019 de archivo principal (el número de línea puede variar si el código ha sido modificado después de que se produjo el error). 3. Las causas potenciales del error incluyen: - Excesiva carga en el servidor, haciendo que no pueda procesar las solicitudes a tiempo. - Problemas con la red o los recursos necesarios para atender la solicitud. - Errores en el código de aplicación que están causando demoras en la respuesta. 4. Una solución práctica para resolver este problema podría ser: - Asegurarse de que no hay problemas con la red y los recursos del servidor, como espacio en disco o memoria insuficientes. - Optimizar el código de la aplicación para reducir las demoras en las respuestas. - Ajustar la configuración del servidor para manejar mejores cargas, como aumentar los límites de tiempo de espera o conexiones permitidas.</p>	36.43 segundos	Validado

			Gemma2	<p>## Análisis del Log de Error Este log indica un problema con la comunicación entre el servidor web (tramites.unl.edu.ec) y un servicio upstream al que se conecta para procesar una solicitud. **Problema:** El error "upstream timed out (110: Connection timed out)" significa que la conexión al servicio upstream (http://127.0.0.1:8000) expiró mientras el servidor web esperaba una respuesta de la misma. **Ubicación del Error:** **Archivo:** El log no especifica el archivo donde se generó el error. **Línea:** El log tampoco indica la línea específica del código donde ocurrió. Se necesita investigar más a fondo en el código del servidor web para determinar la ubicación precisa del error. **Posibles Causas:** 1. **Servicio Upstream Inaccesible:** El servicio ubicado en "http://127.0.0.1:8000" puede estar caído, reiniciando o simplemente no respondiendo a las solicitudes del servidor web. 2. **Problemas de Red:** Puede haber problemas de red entre el servidor web y el servicio upstream, como una conexión lenta o interrupciones. 3. **Sobrecarga del Servicio Upstream:** El servicio upstream puede estar saturado con demasiadas solicitudes, lo que lleva a tiempos de respuesta excesivos y conexiones expiradas. 4. **Configuración incorrecta:** Puede haber un problema en la configuración del servidor web que esté causando el error de tiempo de espera, como un valor demasiado bajo para el timeout de conexión. **Solución Sugerida:** 1. **Verificar Disponibilidad del Servicio Upstream:** Intenta acceder al servicio upstream manualmente desde otro dispositivo o herramienta de línea de comandos. Si no está disponible, se debe investigar la causa del problema y solucionar el problema en ese servicio. 2. **Revisar Conexión a Red:** Comprueba la conectividad entre el servidor web y el servicio upstream utilizando herramientas como ping y traceroute para identificar posibles problemas de red. 3. **Ajustar Timeout de Conexión:** Si la conexión al servicio upstream tarda demasiado, puedes aumentar el timeout de conexión en la configuración del servidor web. **Recomendaciones adicionales:** * Implementar monitoreo proactivo del servicio upstream para detectar problemas de disponibilidad y rendimiento. * Considerar técnicas de caché para reducir la carga sobre el servicio upstream. * Registrar información adicional en los logs, como la duración de la conexión al servicio upstream y los detalles de la solicitud.</p>	214.49 segundos	Validado
--	--	--	--------	--	-----------------	----------

6.1.2.3. Aplicación evaluada: Admisiones

Objetivo de los casos: Evaluar la detección y resolución de logs críticos mediante los modelos de IA (LLaMA3, Mistral, Gemma).

Descripción: En esta evaluación se utilizan logs críticos registrados por el sistema Admisiones en su archivo de logs correspondiente, los cuales representan incidencias reales que requieren intervención inmediata.

A continuación, se detallan los siguientes casos de prueba:

Tabla 18. Caso de Prueba 1 para la aplicación Admisiones.

Caso de Prueba	Registro de Log
Error al ubicar recursos png	<pre>2024/09/02 15:07:10 [error] 584#584: *956104 open() "/var/www/admisiones/media/admisiones_app/banco_pregunta/186/galeria/bien_modelo_bord_dto.png" failed (2: No such file or directory), client: 172.16.32.158, server: ,request: "GET/back/media/admisiones_app/banco_pregunta/186/galeria/bien_modelo_bord_dto.png HTTP/1.0", host:"admisiones.unl.edu.ec"</pre>

Tabla 19. Caso de Prueba 2 para la aplicación Admisiones.

Caso de Prueba	Registro de Log
Error al conectarse a la base de datos	<pre>[2023-01-29 02:03:42,688: ERROR/MainProcess] Database gave error: OperationalError('could not connect to server: No route to host\n\tIs the server running on host "172.16.32.161" and accepting\n\tTCP/IP connections on port 3254?\n') Traceback (most recent call last): File "/ruta_aplicacion/.virtualenvs/admisiones_back_env/lib/python3.9/site- packages/django/db/backends/base/base.py", line 282, in ensure_connection self.connect() File "/ruta_aplicacion/.virtualenvs/admisiones_back_env/lib/python3.9/site- packages/django/db/backends/base/base.py", line 263, in connect self.connection = self.get_new_connection(conn_params) File "/ruta_aplicacion/.virtualenvs/admisiones_back_env/lib/python3.9/site- packages/django/db/backends/postgresql/base.py", line 215, in get_new_connection connection = Database.connect(**conn_params) File "/ruta_aplicacion/.virtualenvs/admisiones_back_env/lib/python3.9/site- packages/psycopg2/__init__.py", line 122, in connect conn = _connect(dsn, connection_factory=connection_factory, **kwasync) psycopg2.OperationalError: could not connect to server: No route to host</pre>

	Is the server running on host "172.16.32.161" and accepting TCP/IP connections on port 3254?
--	--

Tabla 20. Caso de Prueba 3 para la aplicación Admisiones.

Caso de Prueba	Registro de Log
Error al cargar la asignación ítem para el aspirante	<pre>No fue posible cargar la asignacion item para el aspirante., exception=get_por_aspirante_order_postulacion_item() missing 1 required positional argument: 'periodo_admision' Traceback (most recent call last): File "/ruta_aplicacion/proyectos/admisiones- back/app/asignacion/services/view/api/asignacion_item.py", line 19, in get_por_aspirante data = AsignacionItemAppService.get_asignacion_item_aspirante_data(request.user.aspi rante) File "/ruta_aplicacion/proyectos/admisiones- back/app/asignacion/application/asignacion_item.py", line 131, in get_asignacion_item_aspirante_data asignacion_item = AsignacionItemAppService.get_por_aspirante_order_postulacion_item(aspirante). first() TypeError: get_por_aspirante_order_postulacion_item() missing 1 required positional argument: 'periodo_admision'</pre>

Tabla 21. Caso de Prueba 4 para la aplicación Admisiones.

Caso de Prueba	Registro de Log
Error en atributo del modelo	<pre>No se pudo generar los datos para la postulación., exception='NoneType' object has no attribute 'estado_registro_nacional' Traceback (most recent call last): File "/ruta_aplicacion/proyectos/admisiones- back/app/postulacion/services/view/api/postulacion.py", line 117, in get_data_crear_view estado, data = PostulacionAppService.get_data_crear_visualizar_detalle_por_seleccion (request.user, kwargs.get('seleccion_id')) File "/ruta_aplicacion/proyectos/admisiones- back/app/postulacion/application/postulacion.py", line 382, in get_data_crear_visualizar_detalle_por_seleccion estado, respuesta = RegistroNacionalAppService.actualizar_info_registro_nacional_paa_oa(usuario, File "/ruta_aplicacion/proyectos/admisiones- back/app/inscripcion/application/registro_nacional.py", line 368, in actualizar_info_registro_nacional_paa_oa requiere_actualizar_registro_nacional = RegistroNacionalAppService.requiere_actualizar_registro_nacional(File "/ruta_aplicacion/proyectos/admisiones- back/app/inscripcion/application/registro_nacional.py", line 348, in requiere_actualizar_registro_nacional estado = registro_nacional.estado_registro_nacional == RegistroNacional.HABILITADOS_PENDIENTES or registro_nacional.titulo_bachiller == RegistroNacional.NO_REGISTRADO AttributeError: 'NoneType' object has no attribute 'estado_registro_nacional'</pre>

Tabla 22. Caso de Prueba 5 para la aplicación Admisiones

Caso de Prueba	Registro de Log
<p>Error al configurar variables en el settings.py</p>	<pre>[2024-06-24 08:04:17 -0500] [2212134] [ERROR] Exception in worker process Traceback (most recent call last): File "/ruta_aplicacion/.virtualenvs/admisiones_back_env/lib/python3.9/site- packages/gunicorn/arbiter.py", line 589, in spawn_worker worker.init_process() File "/ruta_aplicacion/.virtualenvs/admisiones_back_env/lib/python3.9/site- packages/gunicorn/workers/base.py", line 134, in init_process self.load_wsgi() File "/ruta_aplicacion/.virtualenvs/admisiones_back_env/lib/python3.9/site- packages/gunicorn/workers/base.py", line 146, in load_wsgi self.wsgi = self.app.wsgi() File "/ruta_aplicacion/.virtualenvs/admisiones_back_env/lib/python3.9/site- packages/gunicorn/app/base.py", line 67, in wsgi self.callable = self.load() File "/ruta_aplicacion/.virtualenvs/admisiones_back_env/lib/python3.9/site- packages/gunicorn/app/wsgiapp.py", line 58, in load return self.load_wsgiapp() File "/ruta_aplicacion/.virtualenvs/admisiones_back_env/lib/python3.9/site- packages/gunicorn/app/wsgiapp.py", line 48, in load_wsgiapp return util.import_app(self.app_uri) File "/ruta_aplicacion/.virtualenvs/admisiones_back_env/lib/python3.9/site- packages/gunicorn/util.py", line 359, in import_app mod = importlib.import_module(module) File "/usr/lib/python3.9/importlib/__init__.py", line 127, in import_module return _bootstrap._gcd_import(name[level:], package, level) File "<frozen importlib._bootstrap>", line 1030, in _gcd_import File "<frozen importlib._bootstrap>", line 1007, in _find_and_load File "<frozen importlib._bootstrap>", line 986, in _find_and_load_unlocked File "<frozen importlib._bootstrap>", line 680, in _load_unlocked File "<frozen importlib._bootstrap_external>", line 790, in exec_module File "<frozen importlib._bootstrap>", line 228, in _call_with_frames_removed File "/ruta_aplicacion/proyectos/admisiones-back/cfg/wsgi.py", line 16, in <module> application = get_wsgi_application() File "/ruta_aplicacion/.virtualenvs/admisiones_back_env/lib/python3.9/site- packages/django/core/wsgi.py", line 12, in get_wsgi_application django.setup(set_prefix=False) File "/ruta_aplicacion/.virtualenvs/admisiones_back_env/lib/python3.9/site- packages/django/__init__.py", line 24, in setup apps.populate(settings.INSTALLED_APPS) File "/ruta_aplicacion/.virtualenvs/admisiones_back_env/lib/python3.9/site- packages/django/apps/registry.py", line 124, in populate app_config.ready() File "/ruta_aplicacion/proyectos/admisiones-back/app/core/apps.py", line 15, in ready super().ready() File "/ruta_aplicacion/proyectos/admisiones-</pre>

```

back/django_ddd/config/clean_app_config.py", line 40, in ready
    self.__load_custom_admin_module(self.CUSTOM_ADMIN_MODULE,
register_to=site)
  File "/ruta_aplicacion/proyectos/admisiones-
back/django_ddd/config/clean_app_config.py", line 44, in
__load_custom_admin_module
    autodiscover_modules(admin_module, register_to=register_to)
  File
  File
"/ruta_aplicacion/.virtualenvs/admisiones_back_env/lib/python3.9/site-
packages/django/utils/module_loading.py", line 58, in autodiscover_modules
    import_module("%s.%s" % (app_config.name, module_to_search))
  File "/usr/lib/python3.9/importlib/__init__.py", line 127, in
import_module
    return _bootstrap._gcd_import(name[level:], package, level)
  File "<frozen importlib._bootstrap>", line 1030, in _gcd_import
  File "<frozen importlib._bootstrap>", line 1007, in _find_and_load
  File "<frozen importlib._bootstrap>", line 986, in
_find_and_load_unlocked
  File "<frozen importlib._bootstrap>", line 680, in _load_unlocked
  File "<frozen importlib._bootstrap_external>", line 790, in exec_module
  File "<frozen importlib._bootstrap>", line 228, in
_call_with_frames_removed
  File "/ruta_aplicacion/proyectos/admisiones-
back/app/oferta_academica/presentation/admin.py", line 5, in <module>
    from app.oferta_academica.application.programa_estudio import
ProgramaEstudioAppService
  File "/ruta_aplicacion/proyectos/admisiones-
back/app/oferta_academica/application/programa_estudio.py", line 3, in
<module>
    from app.oferta_academica.application.periodo_admision import
PeriodoAdmisionAppService
  File "/ruta_aplicacion/proyectos/admisiones-
back/app/oferta_academica/application/periodo_admision.py", line 13, in
<module>
    from app.inscripcion.application.inscripcion import
InscripcionAppService
  File "/ruta_aplicacion/proyectos/admisiones-
back/app/inscripcion/application/inscripcion.py", line 13, in <module>
    from app.inscripcion.application.notificacion import
NotificacionInscripcionAppService
  File "/ruta_aplicacion/proyectos/admisiones-
back/app/inscripcion/application/notificacion.py", line 12, in <module>
    from app.notificacion.layer.application.mensajeria import
MensajeriaAppService
  File "/ruta_aplicacion/proyectos/admisiones-
back/app/notificacion/layer/application/mensajeria.py", line 8, in <module>
    class MensajeriaAppService(object):
  File "/ruta_aplicacion/proyectos/admisiones-
back/app/notificacion/layer/application/mensajeria.py", line 33, in
MensajeriaAppService
    def enviar_uno_a_uno(mensajeid, telefono, tipo, datos=None,
ruta=settings.MENSAJERIA_SMS_RUTA_INFORMATIVO,
  File
"/ruta_aplicacion/.virtualenvs/admisiones_back_env/lib/python3.9/site-
packages/django/conf/__init__.py", line 94, in __getattr__
    val = getattr(_wrapped, name)
AttributeError: 'Settings' object has no attribute
'MENSAJERIA_SMS_RUTA_INFORMATIVO'

```

Tabla 23. Casos de prueba aplicación Admisiones.

ID Caso	Log Inicial (Django/Nginx)	Error	Modelo de IA	Recomendación Generada	Tiempo de Respuesta (ms)	Validación
CP-ADMISIONES-001	Fig. 1 Error generado por django	Error al ubicar recursos PNG	Llama3	<p>**Análisis del log** El log indica un error al intentar abrir un archivo (open()) en el sistema. El archivo que no existe es "/var/www/admisiones/media/admisiones_app/banco_pregunta/186/galeria/bien_modelo_bord_dto.png". La razón del error es "No such file or directory" (2), lo que sugiere que el archivo no se encuentra en el directorio especificado.</p> <p>**Ubicación del error** El error se generó en el archivo open() en la línea donde se intenta abrir el archivo mencionado arriba.</p> <p>**Posibles causas**</p> <ol style="list-style-type: none"> 1. El archivo no existe o fue eliminado recientemente. 2. La ruta del archivo es incorrecta, lo que puede deberse a un error de configuración o una actualización fallida. 3. Hay un problema de permisos en el directorio o en el archivo mismo. <p>**Solución práctica**</p> <ol style="list-style-type: none"> 1. Verificar la existencia del archivo en la ruta especificada y asegurarse de que no ha sido eliminado recientemente. 2. Revisar la configuración del sistema para asegurarse de que la ruta del archivo es correcta y no hay errores de configuración. 3. Comprobar los permisos en el directorio y en el archivo mismo para asegurarse de que no hay restricciones de acceso que impidan la lectura o escritura. <p>Recomiendo revisar los logs anteriores para ver si hubo algún problema previamente relacionado con este archivo o ruta. Además, es importante verificar los registros de sistema y configuración para detectar posibles problemas de permisos o configuración incorrecta.</p>	65.32 segundos	Validado

			Mistral	<p>1. El error se genera porque el sistema no encuentra el archivo especificado en la función `open()`. La ruta del archivo es "/var/www/admisiones/media/admisiones_app/banco_pregunta/186/galeria/bien_modelo_bord_dto.png" y el sistema no lo encuentra en el directorio indicado, lo que produce un error de tipo "No such file or directory".</p> <p>2. La línea en la que se generó el error es la primera que contiene la llamada a `open()`: [open() "/var/www/admisiones/media/admisiones_app/banco_pregunta/186/galeria/bien_modelo_bord_dto.png" failed (2: No such file or directory)].</p> <p>3. Una de las causas del error es que el archivo especificado en la función `open()` puede no existir en el directorio indicado o puede ser accesible sólo desde otro sistema, lo cual impide que sea leído por el servidor web. Otra posibilidad es que hay un problema con los permisos de acceso a ese archivo o directorio, lo cual impide que se lea o se escriba en él. También puede existir un fallo de configuración en la ruta del archivo o en el script que intenta abrirlo.</p> <p>4. Una solución práctica para resolver el problema es revisar el directorio y verificar que el archivo especificado en la función `open()` exista realmente y tenga los permisos de acceso necesarios. Si el archivo no existe, sería necesario crearlo o copiarlo al directorio indicado. En caso de que haya un problema con los permisos de acceso, sería necesario modificarlos para que se permita la lectura del archivo por el servidor web. También puede ser necesaria revisar la configuración de rutas y scripts que se están utilizando para asegurarse de que estén correctas y funcionan como esperado.</p>	43.78 segundos	Validado
			Gemma2	<p>## Análisis del Log de Error **Problema:** El sistema no puede encontrar el archivo "bien_modelo_bord_dto.png" en la ruta especificada "/var/www/admisiones/media/admisiones_app/banco_pregunta/186/galeria/". Esto provoca un error al intentar servirlo a un usuario que realizó una solicitud GET para acceder al recurso. **Ubicación del Error:** El error se generó en el archivo responsable de manejar las solicitudes HTTP y la apertura de archivos para servirlos. Es probable que se encuentre dentro del código que gestiona la ruta "/back/media/admisiones_app/banco_pregunta/186/galeria/" **Línea del Código:** La línea exacta del código donde ocurrió el error no está presente en el log proporcionado. **Posibles Causas:** * **Archivo eliminado o movido:** El archivo "bien_modelo_bord_dto.png" podría haber sido eliminado o movido de la ruta especificada anteriormente. * **Permisos incorrectos:** El servidor podría no tener permisos para acceder al archivo debido a configuraciones incorrectas de permisos en el sistema de archivos. * **Ruta incorrecta:** La ruta especificada en el código del servidor podría estar mal escrita o incompleta. * **Error de configuración:** Una configuración</p>	195.93 segundos	Validado

				<p>incorrecta dentro del servidor web podría causar problemas para encontrar y servir archivos. **Solución:** 1. **Verificar la existencia del archivo:** Dirígete al directorio "/var/www/admisiones/media/admisiones_app/banco_pregunta/186/galeria/" en el sistema y verifica si el archivo "bien_modelo_bord_dto.png" existe. 2. **Revisión de permisos:** Asegúrate de que el usuario o grupo que ejecuta el servidor web tenga permisos de lectura para acceder al archivo y su directorio padre. 3. **Confirmación de la ruta:** Comprueba cuidadosamente que la ruta especificada en el código del servidor coincide con la ubicación física del archivo. 4. **Revisión de configuración:** Verifica las configuraciones del servidor web para asegurarte de que no haya errores que impidan servir archivos desde esa ruta específica. Si después de verificar estos puntos el problema persiste, se puede buscar ayuda en los foros o documentación específica del servidor web que estés utilizando.</p>		
CP-ADMISIONES-002	Fig. 1 Error generado por django	Error al conectar se a la base de datos	Llama3	<p>**Análisis del log**</p> <p>El log de error indica que ha habido un problema al intentar conectarse a la base de datos PostgreSQL. El mensaje de error especifica que el servidor no puede ser alcanzado y se pregunta si el servidor está funcionando en la dirección IP "172.16.32.161" y aceptando conexiones TCP/IP en el puerto 3254.</p> <p>**Archivo y línea donde se generó el error**</p> <p>El error se generó en el archivo `base.py` (línea 215) dentro de la biblioteca psycpg2, que es utilizada por Django para conectarse a la base de datos PostgreSQL.</p> <p>**Posibles causas del error**</p> <ol style="list-style-type: none"> **Servidor de base de datos no disponible**: El servidor de base de datos PostgreSQL no está funcionando o no está aceptando conexiones en el puerto 3254. **Configuración incorrecta del archivo de configuración de la base de datos**: La configuración del archivo de configuración de la base de datos puede estar mal configurada, lo que impide la conexión con el servidor de base de datos. ** Problemas de red**: Hay problemas en la red que impiden la comunicación entre el servidor y el cliente. <p>**Solución**</p> <ol style="list-style-type: none"> Verificar si el servidor de base de datos PostgreSQL está funcionando correctamente y aceptando conexiones en el puerto 3254. 	63.36 segundos	Validado

				<p>2. Comprobar la configuración del archivo de configuración de la base de datos para asegurarse de que esté correctamente configurado.</p> <p>3. Si se determina que hay problemas de red, intentar resolver los problemas de red o contactar con el equipo de red para obtener ayuda.</p> <p>En resumen, la solución consiste en verificar si el servidor de base de datos está disponible y aceptando conexiones, y luego investigar posibles problemas de configuración o red.</p>		
			Mistral	<p>1. El error presentado en el log se refiere a una falta de conexión con un servidor PostgreSQL en la dirección IP "172.16.32.161" y el puerto 3254. Esta situación se produce porque el servidor no está ejecutándose o no está aceptando conexiones en dicha dirección y puerto especificados.</p> <p>2. La línea que contiene el error es la número 282 en el archivo "/ruta_proyecto/admisiones/.virtualenvs/admisiones_back_env/lib/python3.9/site-packages/django/db/backends/base/base.py".</p> <p>3. Las posibles causas del error son:</p> <ul style="list-style-type: none"> - El servidor PostgreSQL no está en ejecución. - El servidor no está aceptando conexiones TCP/IP en la dirección IP "172.16.32.161" y el puerto 3254. - La configuración del archivo de conexión a base de datos es incorrecta. <p>4. Para resolver el problema se recomienda:</p> <ol style="list-style-type: none"> 1. Verificar que el servidor PostgreSQL esté en ejecución y funcione correctamente. Si no lo está, reiniciarlo. 2. Comprobar que el servidor PostgreSQL esté aceptando conexiones TCP/IP en la dirección IP "172.16.32.161" y el puerto 3254. Si es necesario, editar la configuración para permitirlo. 3. Verificar que la configuración del archivo de conexión a base de datos sea correcta, incluyendo la dirección IP y el puerto del servidor PostgreSQL. 	43.82 segundos	Validado
			Gemma2	<p>## Análisis del Log de Error</p> <p>El log indica un problema de conexión con la base de datos PostgreSQL. El sistema no puede establecer una conexión con el servidor en la dirección IP `172.16.32.161` en el puerto `3254`.</p>	226.32 segundos	Validado

			<p>**1. Problema:** Imposibilidad de conectar con la base de datos PostgreSQL.</p> <p>**2. Ubicación del Error:** * El error se genera en el archivo: `/ruta_proyecto/admisiones/.virtualenvs/admisiones_back_env/lib/python3.9/site-packages/psycopg2/`__init__.py` * La línea específica es la `122`.</p> <p>**3. Posibles Causas:** * **Servidor PostgreSQL Desactivado:** El servidor de base de datos puede estar apagado o inactivo en el host `172.16.32.161`. * **Firewall Bloqueando Conexiones:** Un firewall podría estar bloqueando las conexiones entrantes al puerto `3254` en el servidor PostgreSQL. * **Error de Red:** Puede haber un problema de red que impide la comunicación entre el sistema que ejecuta el código Django y el servidor de base de datos. * **Datos de Conexión Incorrectos:** Las credenciales o la dirección del servidor podrían estar incorrectas en la configuración de Django para la base de datos.</p> <p>**4. Solución:**</p> <ol style="list-style-type: none"> **Comprobar Estado del Servidor:** Asegúrate de que el servidor PostgreSQL esté corriendo y aceptando conexiones en el host `172.16.32.161` en el puerto `3254`. Puedes verificar esto usando herramientas como `psql` o un monitor de red. **Revisar Configuración del Firewall:** Si se usa un firewall, revisa las reglas para permitir conexiones entrantes al puerto `3254` para la dirección IP del sistema que ejecuta Django. **Verificar Conexión Red:** Asegúrate de que hay una conexión estable entre el sistema Django y el servidor PostgreSQL. Puedes ejecutar pruebas de ping o traceroute para comprobar la conectividad. **Revisar Configuración de Django:** Comprueba la configuración de la base de datos en tu archivo `settings.py` y asegúrate de que las credenciales (usuario, contraseña, host y puerto) sean correctas. <p>**Importante:** Recuerda hacer una copia de seguridad de tu configuración antes de realizar cambios.</p>			
CP-		Error al	Llama3	**Análisis del log**	85.96 segundos	Validado

ADMISIONES-003		cargar la asignación ítem para el aspirante	<p>El log de error indica que no fue posible cargar la asignación ítem para un aspirante. La excepción se produce en el método <code>`get_por_aspirante`</code> dentro del archivo <code>`/ruta_proyecto/admisiones/proyectos/admisiones-back/app/asignacion/services/view/api/asignacion_item.py`</code>, línea 19.</p> <p>La traza de error muestra que el problema se origina en el método <code>`get_por_aspirante_order_postulacion_item`</code> (no especifica el archivo y línea exactos), ya que falta un argumento requerido (<code>`periodo_admision`</code>).</p> <p>**Posibles causas del error**</p> <ol style="list-style-type: none"> 1. El método <code>`get_por_aspirante_order_postulacion_item`</code> fue modificado recientemente para recibir un parámetro adicional, pero no se actualizó el llamado en el método <code>`get_por_aspirante`</code>. 2. Hay una incompatibilidad entre la versión de Python utilizada y la función <code>`get_por_aspirante_order_postulacion_item`</code>, lo que provoca un error al intentar pasar argumentos incorrectos. 3. El método <code>`get_por_aspirante_order_postulacion_item`</code> no está siendo llamado correctamente, lo que hace que falte el parámetro requerido. <p>**Solución**</p> <p>Para resolver este problema, se recomienda verificar las siguientes cosas:</p> <ol style="list-style-type: none"> 1. Revisar el método <code>`get_por_aspirante`</code> en <code>`/ruta_proyecto/admisiones/proyectos/admisiones-back/app/asignacion/services/view/api/asignacion_item.py`</code>, línea 19, y asegurarse de que esté llamando correctamente al método <code>`get_por_aspirante_order_postulacion_item`</code> con el parámetro <code>`periodo_admision`</code>. 2. Verificar si hay alguna compatibilidad de Python entre la versión utilizada en el sistema y la función <code>`get_por_aspirante_order_postulacion_item`</code>. Si es necesario, actualizar Python a una versión compatible. 3. Revisar el método <code>`get_por_aspirante_order_postulacion_item`</code> para asegurarse de que esté siendo llamado correctamente y que no haya problemas de incompatibilidad entre la función y su llamada. <p>En resumen, se recomienda revisar los métodos involucrados en el error y asegurarse de que estén llamados correctamente con los argumentos requeridos.</p>		
----------------	--	---	---	--	--

			Mistral	<p>1. El error se presenta en la carga de una asignación de item para un aspirante y está causado por que el método <code>get_por_aspirante_order_postulacion_item()</code> en el archivo <code>asignacion/services/view/api/asignacion_item.py</code> está faltando un argumento posicional llamado 'periodo_admision'.</p> <p>2. El error se generó en la línea 19 del archivo <code>/ruta_proyecto/admisiones/proyectos/admisiones-back/app/asignacion/services/view/api/asignacion_item.py</code> y en la línea 131 del archivo <code>/ruta_proyecto/admisiones/proyectos/admisiones-back/app/asignacion/application/asignacion_item.py</code></p> <p>3. El error puede deberse a que no se está enviando el argumento 'periodo_admision' en la llamada del método, o que este argumento no está declarado como necesario.</p> <p>4. Una solución práctica para resolver este problema sería agregar el parámetro 'periodo_admision' en todas las llamadas al método <code>get_por_aspirante_order_postulacion_item()</code> y declararlo como requerido, de esta manera se evita que falle la carga del item asignado al aspirante. Además, es posible que sea necesario revisar el código en caso de que se haya olvidado definir el parámetro 'periodo_admision' como requerido.</p>	36.88 segundos	Validado
			Gemma2	<p>## Análisis del log de error</p> <p>**Problema:** El sistema no pudo cargar la asignación item para el aspirante debido a una falta de argumento requerido en la función <code>`get_por_aspirante_order_postulacion_item()`</code>.</p> <p>**Ubicación del error:**</p> <p>**Archivo:** <code>`/ruta_proyecto/admisiones/proyectos/admisiones-back/app/asignacion/application/asignacion_item.py`</code></p> <p>**Línea:** 131</p> <p>**Posibles causas:**</p> <p>1. **Falta de parámetro 'periodo_admision':** La función <code>`get_por_aspirante_order_postulacion_item()`</code> requiere el parámetro <code>`periodo_admision`</code>, que no se está proporcionando al llamar a la función en la línea 131 del archivo <code>`asignacion_item.py`</code>.</p> <p>2. **Error en la lógica de llamada:** Existe un error en la forma en que se llama a la función <code>`get_por_aspirante_order_postulacion_item()`</code>, posiblemente por una condición o variable incorrecta que no asigna el valor del <code>`periodo_admision`</code>.</p>	194.20 segundos	Validado

				<p>**Solución:** Verificar si el parámetro `periodo_admision` está siendo definido correctamente antes de llamar a la función `get_por_aspirante_order_postulacion_item()`. Asegúrate de pasar el valor correcto como argumento.</p> <p>Por ejemplo, revise las líneas de código alrededor de la línea 131 en el archivo `asignacion_item.py` para asegurarse de que se está obteniendo y pasando el valor de `periodo_admision` correctamente a la función.</p> <pre> python # Ejemplo de cómo pasar el periodo_admision periodo_admision = get_current_admission_period() # Función o lógica para obtener el periodo actual asignacion_item = AsignacionItemAppService.get_por_aspirante_order_postulacion_item(aspirante, periodo_admision).first() </pre> <p>Si el problema persiste, se recomienda revisar más a fondo la documentación de la función `get_por_aspirante_order_postulacion_item()` para comprender completamente sus argumentos y uso.</p>		
CP-ADMISIONES-004	Error al acceder al atributo 'estado_registro_nacional' del aspirante	Llama3	<p>**Análisis del log de error**</p> <p>El log de error indica que se produjo un problema al intentar generar datos para la postulación. El mensaje de error es específico y sugiere que el problema se originó en la línea 348 del archivo `registro_nacional.py`. A continuación, te presento los detalles del análisis:</p> <ul style="list-style-type: none"> * La excepción `NoneType` object has no attribute 'estado_registro_nacional' indica que se intentó acceder a un atributo ('estado_registro_nacional') de un objeto cuyo valor es `None`. * El error se produjo en la función `requiere_actualizar_registro_nacional()` del archivo `registro_nacional.py`, línea 348. * La llamada a esta función proviene desde el archivo `postulacion.py`, línea 117, como parte de la función `get_data_crear_visualizar_detalle_por_seleccion()`. 	84.35 segundos	Validado	

				<p>**Posibles causas del error**</p> <ol style="list-style-type: none"> 1. Un objeto `RegistroNacional` no ha sido instanciado correctamente o está siendo tratado como `None`. 2. El atributo `estado_registro_nacional` no existe en el objeto `RegistroNacional`. <p>**Solución directa y práctica**</p> <p>La solución es identificar por qué se está produciendo este error y asegurarse de que el objeto `RegistroNacional` esté correctamente instanciado y tenga el atributo `estado_registro_nacional`. Para hacer esto, revise las líneas 348 y siguientes del archivo `registro_nacional.py` para determinar cómo se instancia el objeto `RegistroNacional` y si se está tratando como `None`.</p> <p>En particular, revise la línea que llama a la función `requiere_actualizar_registro_nacional()` y asegúrese de que el objeto `RegistroNacional` esté correctamente instanciado antes de intentar acceder a su atributo `estado_registro_nacional`. Si es necesario, agregue un verificador para verificar si el objeto `RegistroNacional` no está nulo o vacío antes de tratarlo como válido.</p> <p>**Sugerencia**</p> <p>Revisa las líneas 348 y siguientes del archivo `registro_nacional.py` para determinar cómo se instancia el objeto `RegistroNacional` y si se está tratando como `None`. Asegúrate de que el objeto esté correctamente instanciado antes de intentar acceder a su atributo `estado_registro_nacional`.</p>		
			Mistral	<ol style="list-style-type: none"> 1. El problema que se presenta en el log es un error de tipo `AttributeError`, específicamente en la línea 348 del archivo `/ruta_proyecto/admisiones/proyectos/admisiones-back/app/inscripcion/application/registro_nacional.py`. El error se produce porque se intenta acceder al atributo `estado_registro_nacional` de un objeto que es de tipo `NoneType`, lo que no tiene el método `estado_registro_nacional`. 2. El error se generó en la línea 348 del archivo `/ruta_proyecto/admisiones/proyectos/admisiones-back/app/inscripcion/application/registro_nacional.py`. 3. Posibles causas del error pueden ser: 	44.78 segundos	Validado

			<ul style="list-style-type: none"> - Que el objeto que se está tratando de acceder no fue creado correctamente o no tiene los atributos esperados, lo que podría estar produciendo que `estado_registro_nacional` sea `None`. - Un código fallido en la lógica que permite que un objeto sea asignado como `None` cuando debería tener un valor. <p>4. Una solución práctica para resolver el problema es:</p> <ul style="list-style-type: none"> - Revisar el código en las líneas previas a la línea 348 del archivo <code>`/ruta_proyecto/admisiones/proyectos/admisiones-back/app/inscripcion/application/registro_nacional.py`</code> para asegurarse de que el objeto que se está tratando de acceder no es `None`. - Verificar que los atributos esperados estén siendo asignados correctamente al objeto en cuestión. - Si la causa del problema no se puede identificar con esta inspección, podría ser necesario revisar el código en un nivel más alto para comprender mejor el comportamiento de los objetos y cómo son creados y manipulados. 		
		Gemma2	<p>## Análisis del Log de Error</p> <p>**Problema:** El sistema no pudo generar los datos para la postulación debido a un error en la cadena de ejecución. Se produce una excepción `AttributeError` que indica que se intenta acceder al atributo `estado_registro_nacional` de un objeto que es `None`.</p> <p>**Ubicación del Error:**</p> <pre>* **Archivo:** `/ruta_proyecto/admisiones/proyectos/admisiones-back/app/inscripcion/application/registro_nacional.py` * **Línea:** 348</pre> <p>**Posibles Causas:**</p> <ol style="list-style-type: none"> 1. **Objeto `registro_nacional` es `None`:** La variable `registro_nacional` no contiene un objeto válido en la línea 348 del archivo mencionado. Esto podría suceder si: <ul style="list-style-type: none"> * La consulta que obtiene el `registro_nacional` falla o no devuelve ningún resultado. * El proceso anterior a esta línea ha eliminado accidentalmente el valor de `registro_nacional`. 2. **Problema con la lógica de comparación:** La expresión <code>`estado = registro_nacional.estado_registro_nacional == RegistroNacional.HABILITADOS_PENDIENTES or registro_nacional.titulo_bachiller == RegistroNacional.NO_REGISTRADO`</code> podría tener un error lógico. 	290.23 segundos	Validado

				<p>* Es posible que los valores `RegistroNacional.HABILITADOS_PENDIENTES` o `RegistroNacional.NO_REGISTRADO` no estén definidos correctamente.</p> <p>**Solución Propuesta:**</p> <p>1. **Depurar el código:**</p> <ul style="list-style-type: none"> - Imprimir el valor de `registro_nacional` en la línea 348 para verificar si es `None`. - Investigar la consulta que obtiene `registro_nacional` para asegurarte de que está funcionando correctamente y devuelve un objeto válido. <p>2. **Revisar la lógica:** Verificar los valores de las constantes `RegistroNacional.HABILITADOS_PENDIENTES` y `RegistroNacional.NO_REGISTRADO`. Asegurarte que están definidas correctamente.</p>		
CP-ADMISIONES-005	Error al no encontrar la variable MENSAJERIA_SMS_RUTA_INFORMATIVO	Llama3	<p>**Análisis del log**</p> <p>El log indica que se produjo un error en un proceso de trabajo (worker process) en un entorno WSGI. El error se produce al intentar cargar el archivo `wsgi.py` en la carpeta `/ruta_proyecto/admisiones/proyectos/admisiones-back/cfg/`. La traza del error indica que el problema se origina en una línea específica dentro de ese archivo.</p> <p>**Archivo y línea generadora del error**</p> <p>El error se produce en la línea 16 del archivo `wsgi.py`.</p> <p>**Posibles causas del error**</p> <ol style="list-style-type: none"> 1. Error de configuración: Es posible que haya un error en la configuración de Django o Gunicorn. 2. Falta o no existe el atributo 'MENSAJERIA_SMS_RUTA_INFORMATIVO' en los settings de Django. 3. Problema con la carga de módulos o importaciones. <p>**Solución directa y práctica**</p> <p>Para resolver este problema, puedo sugerir las siguientes acciones:</p> <ol style="list-style-type: none"> 1. Verificar la configuración de Django y Gunicorn para asegurarme de que todo esté configurado correctamente. 	77.97 segundos	Validado	

				<p>2. Comprobar si el atributo 'MENSAJERIA_SMS_RUTA_INFORMATIVO' existe en los settings de Django y verificar su valor.</p> <p>3. Si no hay errores de configuración, puede ser necesario revisar las importaciones o carga de módulos en el archivo `wsgi.py` y en los archivos que se importan.</p> <p>En resumen, la solución más directa y práctica para resolver este problema es revisar la configuración de Django y Gunicorn, verificar si el atributo 'MENSAJERIA_SMS_RUTA_INFORMATIVO' existe en los settings de Django y revisar las importaciones o carga de módulos en el archivo `wsgi.py`.</p>		
			Mistral	<p>Este error se produce porque la variable `MENSAJERIA_SMS_RUTA_INFORMATIVO` no está definida en tus settings. Django utiliza los valores de las variables en tus settings para acceder a ciertas configuraciones en tu proyecto, y en este caso, la variable que no se encuentra define es `MENSAJERIA_SMS_RUTA_INFORMATIVO`.</p> <p>Para solucionar este problema debes definir esta variable en tus settings. Si estás utilizando el sistema de envío de SMS por medio de una API externa, tal como Twilio o Nexmo, sería necesario definir la ruta que usarás para enviar los mensajes y asegurarte de que esa variable se encuentre en tus settings.</p> <p>Aquí hay un ejemplo de cómo podrías definir esa variable en tus settings:</p> <pre>```python MENSAJERIA_SMS_RUTA_INFORMATIVO = 'https://api.twilio.com/2010-04-01/Accounts/{TU_CUENTA_TWILIO}/Messages.json' ```</pre> <p>Puedes revisar las documentaciones de Twilio para obtener más información sobre cómo funciona la API y cómo puedes establecer conexión en tu proyecto Django: https://www.twilio.com/docs/api</p>	42.96 segundos	Validado
			Gemma2	<p>## Análisis del log de error</p> <p>**Problema:** El sistema está experimentando un error al intentar enviar mensajes SMS a través del servicio de mensajería. Específicamente, el problema surge al acceder a la configuración `settings.MENSAJERIA_SMS_RUTA_INFORMATIVO`.</p> <p>**Archivo y línea:**</p>	179.34 segundos	Validado

				<p>El error ocurre en el archivo <code>`/ruta_proyecto/admisiones/proyectos/admisiones-back/app/notificacion/layer/application/mensajeria.py`</code>, en la línea 33 de este mismo archivo.</p> <p>**Causa probable:** La variable <code>`MENSAJERIA_SMS_RUTA_INFORMATIVO`</code> no está definida en las configuraciones del proyecto (<code>`settings.py`</code>). Esto significa que la aplicación no tiene información sobre dónde enviar los mensajes SMS.</p> <p>## Solución propuesta:</p> <ol style="list-style-type: none"> **Verificar la configuración**: Asegúrate de que la variable <code>`MENSAJERIA_SMS_RUTA_INFORMATIVO`</code> esté definida correctamente en el archivo <code>`settings.py`</code>. **Documentación del servicio de mensajería:** Consulta la documentación del servicio de mensajería que estás utilizando para saber cómo configurar correctamente la ruta de envío de mensajes SMS. <p>**Recomendaciones adicionales:**</p> <ul style="list-style-type: none"> * Agregar logs más detallados al código para identificar mejor el origen del error. * Implementar manejo de excepciones para evitar que el error cause un fallo completo del sistema. 		
--	--	--	--	---	--	--

Los resultados obtenidos en los casos de prueba demuestran la efectividad del sistema propuesto para detectar y analizar errores críticos en la aplicación SIAAF. Las recomendaciones generadas por los modelos de IA fueron aplicadas técnicamente, y en la mayoría de los casos se logró resolver los errores identificados de manera efectiva. Sin embargo, algunos escenarios, como el envío de mensajes SMS, no pudieron ser validados debido a la falta de acceso al API correspondiente, lo que limita la comprobación completa de las recomendaciones. Asimismo, ciertos errores similares no pudieron ser evaluados completamente por restricciones técnicas en el entorno de pruebas. A pesar de estas limitaciones, los resultados generales refuerzan la viabilidad de integrar modelos de lenguaje como parte del flujo de gestión centralizada de logs, optimizando la resolución de incidencias en entornos institucionales.

6.1.3. R3: Visualización de Notificaciones y Logs

Esta sección describe cómo los resultados del procesamiento de logs críticos se visualizan a través de notificaciones en Slack y mediante el almacenamiento de todos los logs en OpenSearch Dashboard. Estas visualizaciones son fundamentales para facilitar la identificación y resolución rápida de incidencias en los sistemas institucionales, proporcionando a los administradores un acceso directo a los detalles de los eventos y sus recomendaciones.

Notificaciones en Slack

Slack se configuró como el canal de notificación en tiempo real para alertar a los administradores sobre errores críticos detectados en los sistemas SIAAF, Trámites y Admisiones. Cada notificación enviada incluye información detallada sobre el evento, acompañada de recomendaciones generadas por los modelos de inteligencia artificial. Estas notificaciones proporcionan una visión inmediata de las incidencias, facilitando la intervención rápida de los equipos técnicos.

Estructura de la Notificación

Cada notificación enviada mediante Slack contiene los siguientes campos clave:

- **Nivel del Error:** Grado de severidad del evento.
- **Sistema:** Sistema de origen del log (SIAAF, Trámites o Admisiones).
- **Descripción del Error:** Mensaje del log original transformado y simplificado.
- **Recomendación Generada:** Sugerencia proporcionada por el modelo de IA para resolver el problema.
- **Timestamp:** Fecha y hora exactas del evento.

A continuación, se describen ejemplos de notificaciones enviadas en Slack:

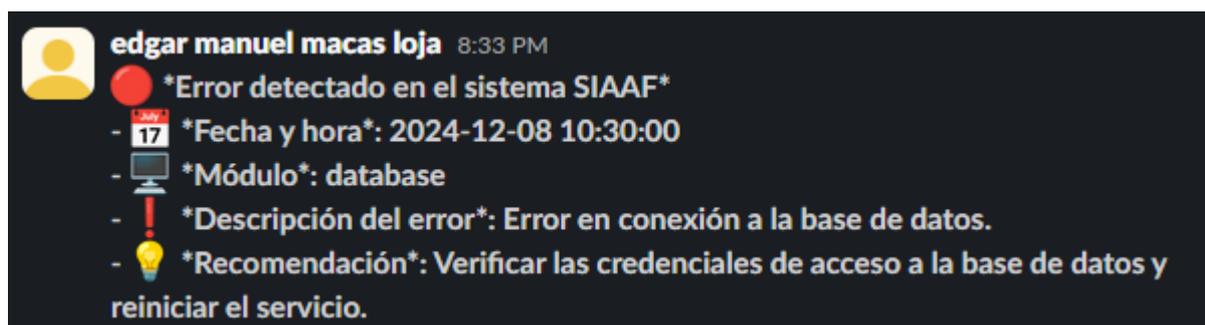


Figura 1. Notificación en Slack de un error crítico en SIAAF con detalles y recomendaciones sobre problemas en la base de datos

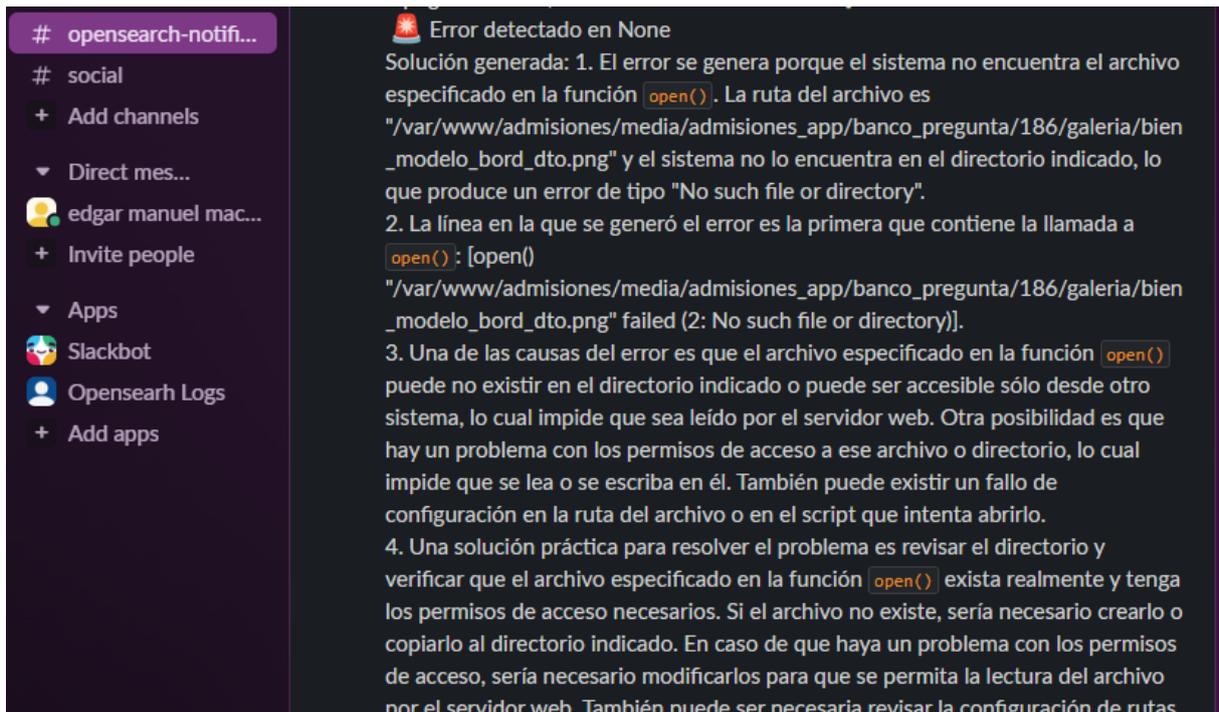


Figura 2. Notificación en Slack de un error en Admisiones relacionado con archivos faltantes y soluciones propuestas.

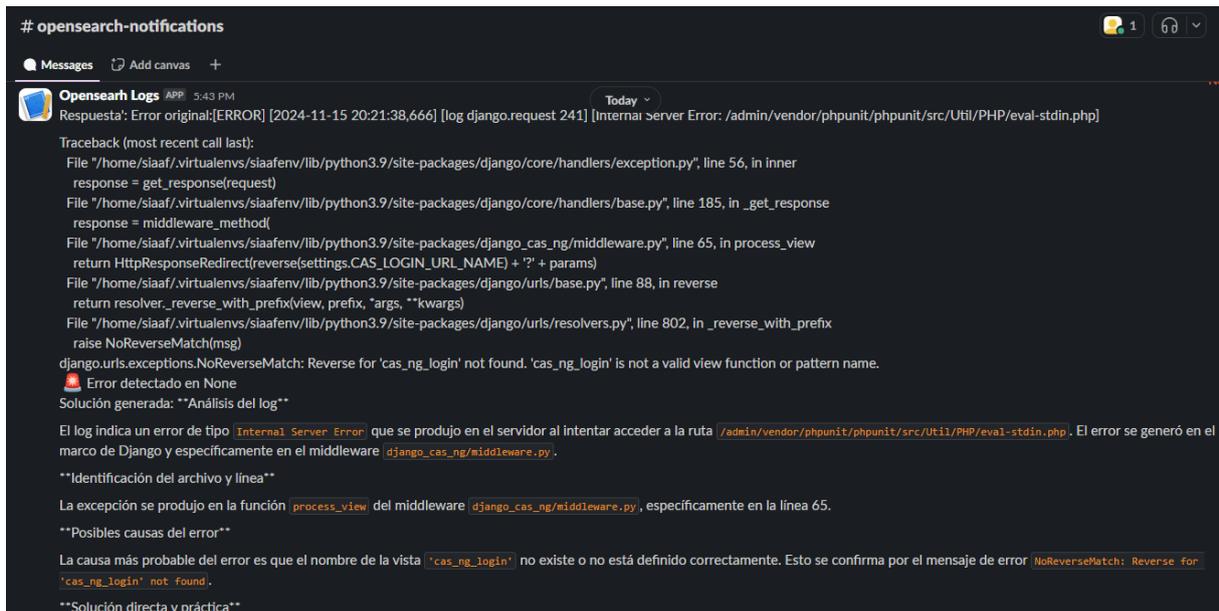


Figura 3. Notificación en Slack de un error crítico en una ruta no encontrada dentro de un sistema Django, con trazas y recomendaciones generadas por el modelo de IA.

Visualización en OpenSearch Dashboard

Además de las notificaciones en Slack, todos los logs (tanto críticos como no críticos) fueron almacenados y organizados en OpenSearch Dashboard, que actúa como un repositorio centralizado para su consulta y análisis detallado. Esta herramienta permite a los administradores explorar los logs mediante búsquedas específicas, aplicar filtros avanzados

y visualizar tendencias o patrones relacionados con incidencias en los sistemas institucionales.

La integración de Slack y OpenSearch Dashboard ofrece una solución integral para la gestión de logs en los sistemas institucionales. Mientras que Slack facilita una reacción inmediata ante errores críticos mediante notificaciones en tiempo real, OpenSearch Dashboard complementa esta funcionalidad con un análisis profundo de los eventos registrados. Esta combinación fortalece significativamente la capacidad de los administradores para monitorear, diagnosticar y resolver incidencias de manera eficiente y oportuna.

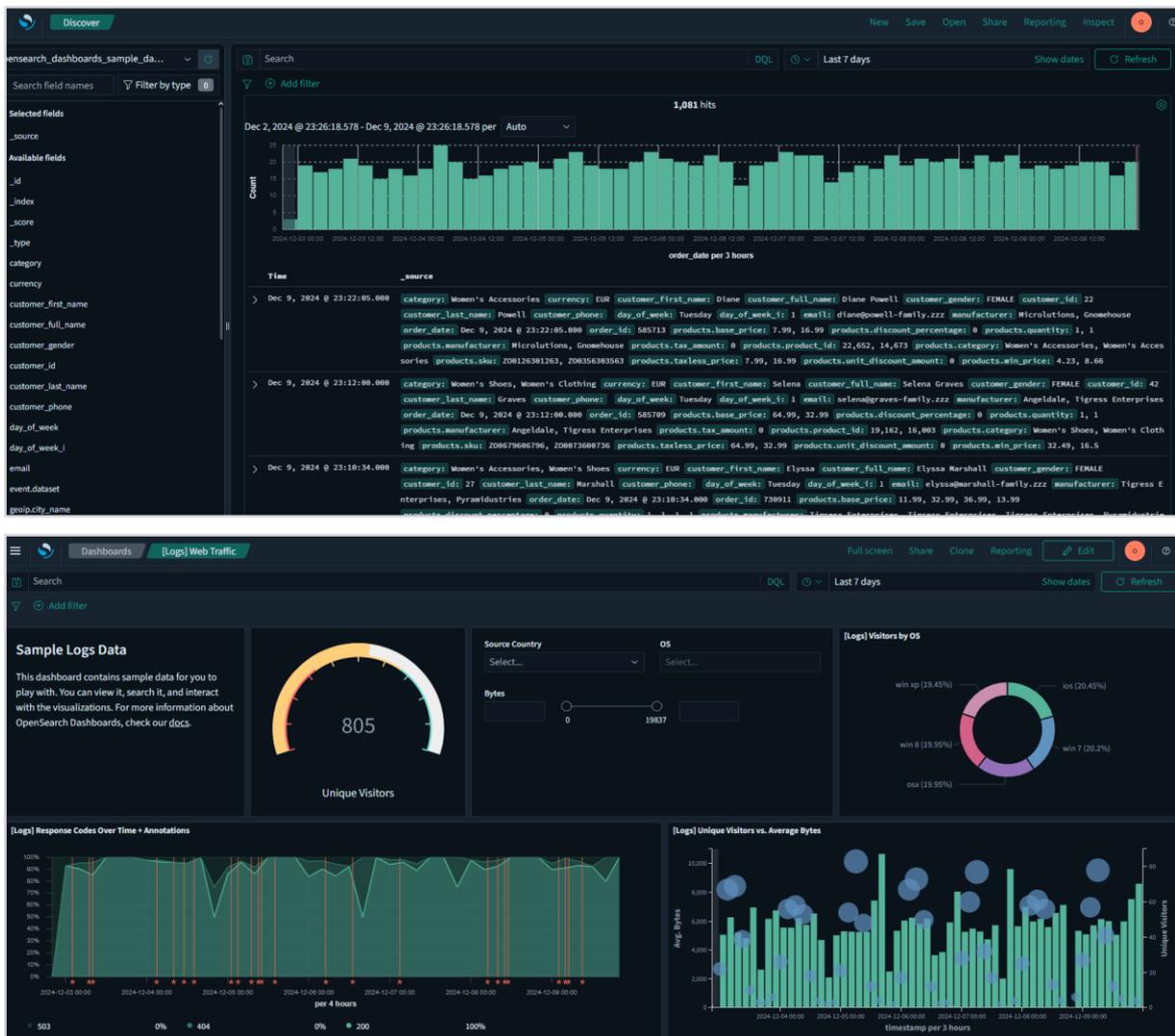


Figura 4. Visualización en OpenSearch Dashboards, mostrando gráficos de tendencias de logs, volúmenes de datos y análisis detallado para monitoreo y diagnóstico.

La arquitectura propuesta integra herramientas clave que trabajan en conjunto para recolectar, procesar y visualizar logs generados por los sistemas institucionales. En la Fig. 5, se presenta la versión inicial de esta arquitectura, que ilustra cómo los logs generados son procesados

mediante Fluent Bit y enviados a Apache Kafka para su gestión. Los logs críticos se analizan y procesan mediante una API Flask conectada a modelos de inteligencia artificial, generando notificaciones en Slack, mientras que todos los logs se almacenan en OpenSearch para análisis detallado.

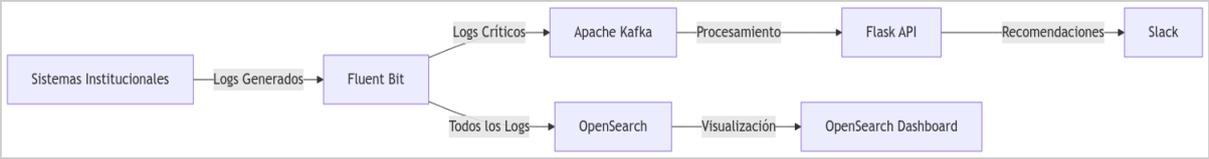


Figura 5. Versión inicial de la arquitectura propuesta para la gestión de logs, integrando recolección, procesamiento, visualización y notificaciones.

6.2. Objetivo 2: Diseñar la arquitectura para la gestión centralizada de logs en los sistemas institucionales de la Universidad Nacional de Loja, mediante un estilo arquitectónico que represente la integración de OpenSearch e inteligencia artificial.

6.2.1. R4: Diseño de la Arquitectura

A continuación, se presenta el diseño de la arquitectura propuesta para la gestión centralizada de logs en los sistemas institucionales de la Universidad Nacional de Loja, integrando herramientas como Fluent Bit, Apache Kafka, OpenSearch, y modelos de inteligencia artificial, lo que permite recolectar, analizar y visualizar logs de manera eficiente. Este apartado presenta cada componente de la arquitectura, su interacción y la propuesta final representada mediante diagramas UML.

6.2.1.1. Generación de Logs

Los sistemas institucionales de la Universidad Nacional de Loja (SIAAF, Trámites y Admisiones) generan logs en diferentes formatos, dependiendo de las tecnologías utilizadas, como Django y Nginx. Estos logs contienen eventos clave que incluyen errores críticos, advertencias y mensajes informativos que son esenciales para el monitoreo y diagnóstico de las aplicaciones.

La gestión de estos logs se inicia con su recolección desde las ubicaciones específicas configuradas en cada sistema. Como se muestra en la **Figura. 6**, los logs de cada aplicación se almacenan en rutas independientes dentro del sistema de archivos del servidor. Por ejemplo:

- Los logs del sistema **SIAAF** se encuentran en `/var/log/siaaff/*log`.
- Los logs del sistema **Admisiones** están en `/var/log/amisiones/*log`.
- Los logs del sistema **Trámites** se ubican en `/var/log/tramites/*log`.

Esta organización permite una recolección estructurada y eficiente de los datos, facilitando su procesamiento posterior por herramientas como Fluent Bit y su análisis en componentes adicionales de la arquitectura.

- Diagrama:



Figura 6. Ubicación de logs de los sistemas institucionales

6.2.1.2. Recolección de Logs

La recolección de logs en los sistemas institucionales de la Universidad Nacional de Loja se realiza mediante **Fluent Bit**, una herramienta ligera y eficiente diseñada para capturar, estructurar y transferir logs desde rutas específicas hacia sistemas de análisis y almacenamiento centralizados. Este proceso es crucial para garantizar que los datos sean procesados y enviados de manera estructurada a los componentes siguientes de la arquitectura.

Descripción del Proceso de Recolección

1. **Input:** Fluent Bit captura los logs desde las rutas configuradas en cada sistema, como se muestra en la **Figura. 7:**
 - o `/var/log/siaff/*log` para el sistema SIAAF.
 - o `/var/log/amisiones/*log` para el sistema Admisiones.
 - o `/var/log/tramites/*log` para el sistema Trámites.
2. **Parser:** Los datos recolectados son estructurados en un formato legible como **JSON**, facilitando su análisis y procesamiento.
3. **Output:** Los logs procesados son enviados a **Apache Kafka** para gestionar logs críticos y a **OpenSearch** para su almacenamiento y visualización.

Este flujo asegura una recolección eficiente y ordenada de los logs, facilitando su análisis y la detección temprana de incidencias en los sistemas institucionales. Fluent Bit actúa como un puente clave entre las aplicaciones y los sistemas de análisis, estandarizando los datos para un uso posterior.

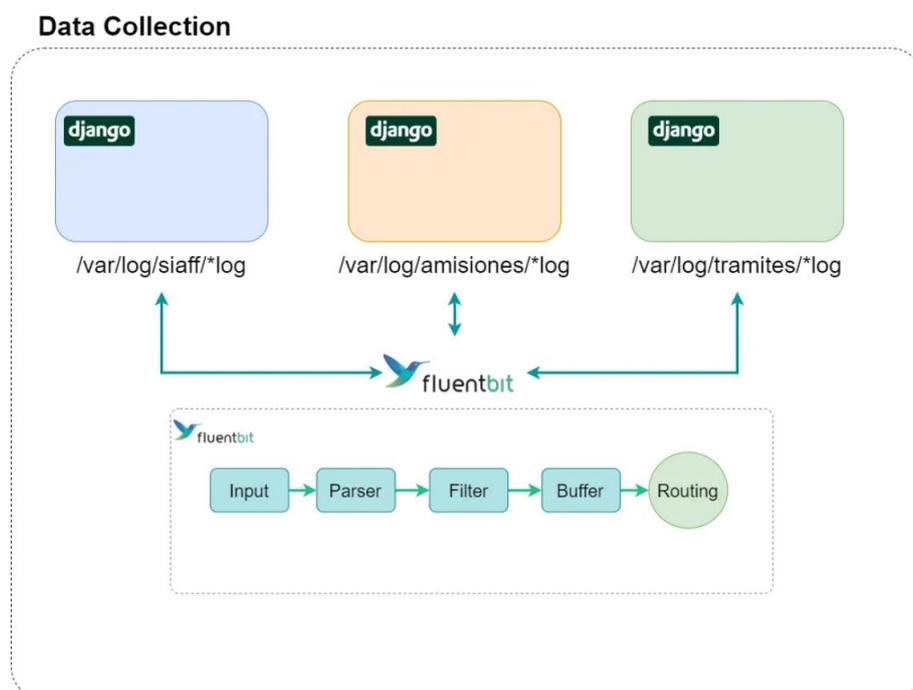


Figura 7. Proceso de recolección de logs con Fluent Bit.

6.2.1.3. Almacenamiento y Visualización

El almacenamiento y visualización de los logs recolectados es un componente clave de la arquitectura propuesta para la gestión centralizada de logs. Este proceso asegura que los datos capturados sean accesibles para un análisis avanzado y una visualización intuitiva, facilitando el monitoreo continuo y la identificación de patrones o incidencias.

OpenSearch

1. Recepción y Almacenamiento:

OpenSearch actúa como el repositorio centralizado que recibe todos los logs recolectados por Fluent Bit. Este sistema almacena tanto logs críticos como informativos, asegurando su disponibilidad para búsquedas posteriores y análisis detallados.

2. Análisis y Visualización:

Mediante OpenSearch Dashboard, los administradores tienen acceso a herramientas avanzadas de búsqueda y generación de visualizaciones. Esto permite analizar tendencias, identificar anomalías y explorar logs específicos mediante filtros personalizados. Además, OpenSearch Dashboard se integra fácilmente en entornos Docker, garantizando flexibilidad y escalabilidad.

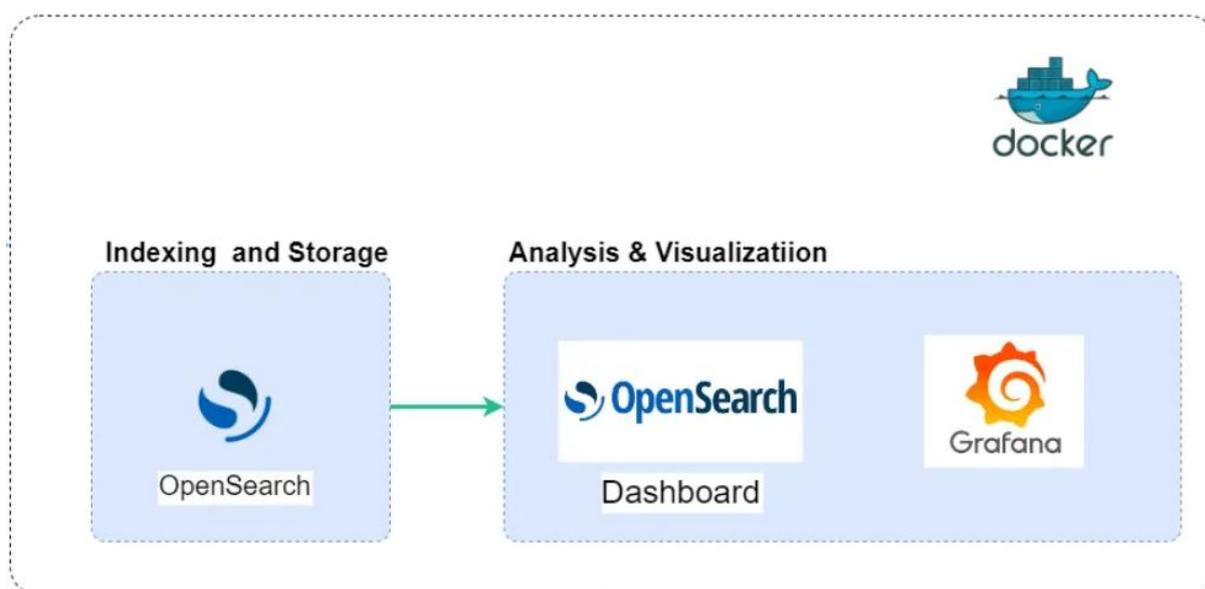


Figura 8. Arquitectura de visualización y almacenamiento con OpenSearch y Grafana.

Alternativas de Visualización: Grafana

Grafana también es considerado como una herramienta complementaria o alternativa para la visualización, especialmente en la representación gráfica de métricas y patrones. Ambas opciones (OpenSearch Dashboard y Grafana) ofrecen capacidades robustas para la visualización, aunque OpenSearch Dashboard está optimizado para la gestión de logs.

6.2.1.4. Procesamiento de Logs Críticos

El procesamiento de logs críticos es un elemento central de la arquitectura, diseñado para garantizar que los eventos más importantes sean analizados de manera eficiente y priorizados para una respuesta inmediata. Este flujo combina las capacidades de **Apache Kafka** como sistema de mensajería confiable y la API basada en **Flask** para gestionar la interacción con los modelos de inteligencia artificial.

Apache Kafka

- Actúa como un buffer confiable para los logs críticos, asegurando que estos sean transmitidos de manera ordenada y sin pérdida de datos.
- Kafka organiza los logs críticos en tópicos específicos, lo que facilita su consumo por parte de otros componentes del sistema.

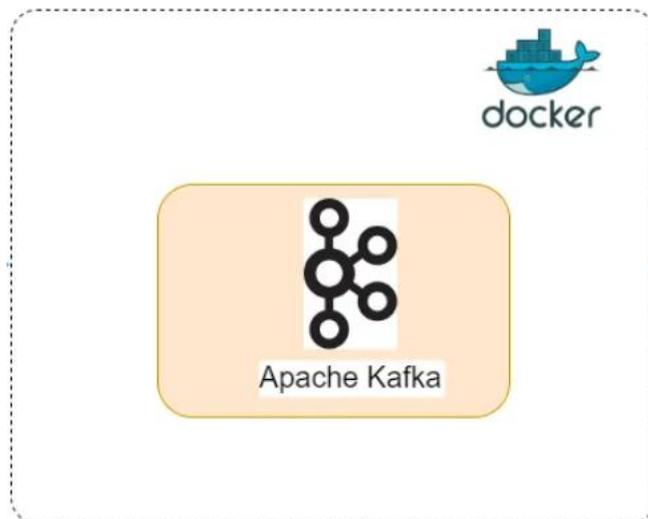


Figura 9. Integración de Apache Kafka para la gestión de logs.

Flask API

- **Recepción y procesamiento:** Consume los logs almacenados en Kafka, transformándolos y preparándolos para su análisis por los modelos de inteligencia artificial.
- **Integración con modelos de IA:** Envía los logs al modelo correspondiente (LLaMA3, Mistral o Gemma2) para su análisis, obteniendo recomendaciones específicas para resolver los errores detectados.
- **Notificaciones en tiempo real:** Las recomendaciones generadas son enviadas a **Slack**, proporcionando una herramienta de alerta inmediata para los administradores.

Normalización y Almacenamiento en Caché de Logs Críticos

Como parte del flujo de procesamiento de logs críticos, se incluye un proceso de normalización

y almacenamiento temporal que optimiza el análisis y la disponibilidad de los datos para su consulta. Este proceso utiliza una combinación de la **API Flask**, **ChromaDB** para búsquedas semánticas, y **Redis** como sistema de almacenamiento en caché.

1. Normalización de Logs Críticos

La API Flask transforma los logs críticos recolectados, estandarizando su formato para asegurar su compatibilidad con las herramientas de análisis posteriores. Esto incluye estructurar los datos en un formato uniforme y reducir la redundancia, facilitando el acceso y la búsqueda eficiente.

2. Búsquedas Semánticas con ChromaDB y Modelos de Embeddings

Los logs normalizados son enviados a **ChromaDB**, una base de datos optimizada para búsquedas semánticas. En este proceso, se utiliza el modelo de embeddings **mx-bai-embed-large** configurado mediante **OllamaEmbeddings** para representar los logs en un espacio vectorial. Esto permite realizar consultas basadas en similitudes con logs históricos, identificando patrones o errores repetitivos que pueden acelerar la resolución de incidencias.

Este modelo de embeddings mejora la precisión de las búsquedas semánticas, facilitando la identificación de incidencias relacionadas o recurrentes.

Almacenamiento Temporal con Redis

Una vez procesados, los logs son almacenados en Redis, un sistema de almacenamiento en caché de alto rendimiento. Redis actúa como un repositorio temporal para los datos más recientes, lo que permite su recuperación rápida por otros componentes de la arquitectura, como los modelos de IA o los sistemas de notificación.

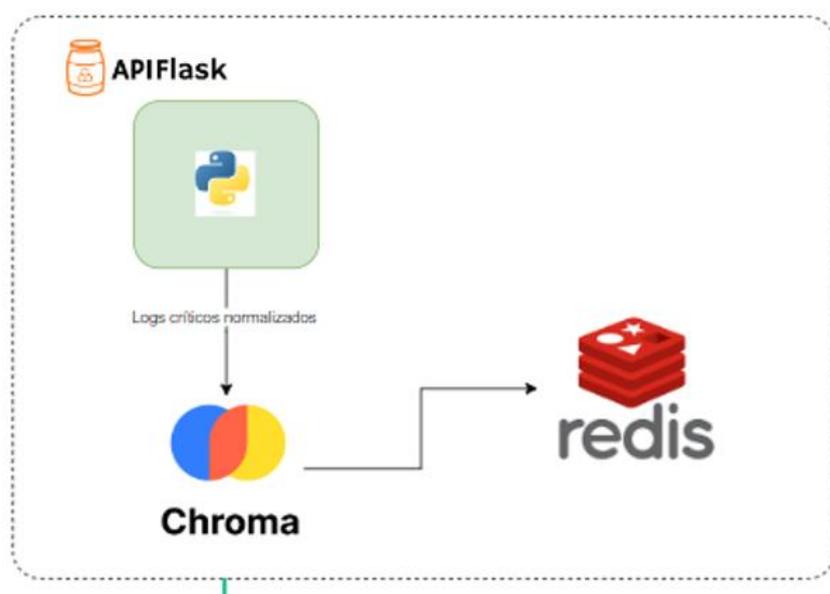


Figura 10. Normalización y Almacenamiento en Caché de Logs Críticos.

6.2.1.5. Análisis con Modelos de IA

La integración de modelos de inteligencia artificial (IA) es un componente clave en el análisis y resolución de logs críticos. Esta etapa utiliza modelos avanzados para interpretar los datos y generar recomendaciones efectivas que ayuden a los administradores a solucionar problemas en los sistemas institucionales.

Modelos Utilizados

1. **Llama3**: Especializado en análisis crítico, proporciona resultados precisos en entornos complejos.
2. **Claude**: Destacado por su balance entre velocidad y precisión.
3. **Mistral**: Enfocado en ofrecer soluciones rápidas, ideal para escenarios que requieren agilidad.

Proceso de Análisis

- Los logs críticos, previamente procesados y normalizados, son enviados desde la API Flask hacia los modelos de IA alojados en **Ollama_Server**.
- Los modelos analizan los datos, identifican posibles causas del error y generan recomendaciones específicas para su resolución.
- Estas recomendaciones son devueltas al cliente **Ollama_Client**, desde donde son reenviadas al sistema de notificaciones en Slack.



Figura 11. Integración de modelos de IA en Ollama.

6.2.1.6. Notificaciones

El sistema de notificaciones implementado utiliza **Slack** como canal principal para alertar al equipo técnico sobre errores críticos detectados en los sistemas institucionales. Este enfoque garantiza una comunicación en tiempo real y facilita una rápida respuesta ante incidencias. Las notificaciones en Slack complementan el análisis y almacenamiento en OpenSearch, proporcionando una herramienta adicional para la gestión eficiente de logs críticos y mejorando la coordinación entre los miembros del equipo técnico.

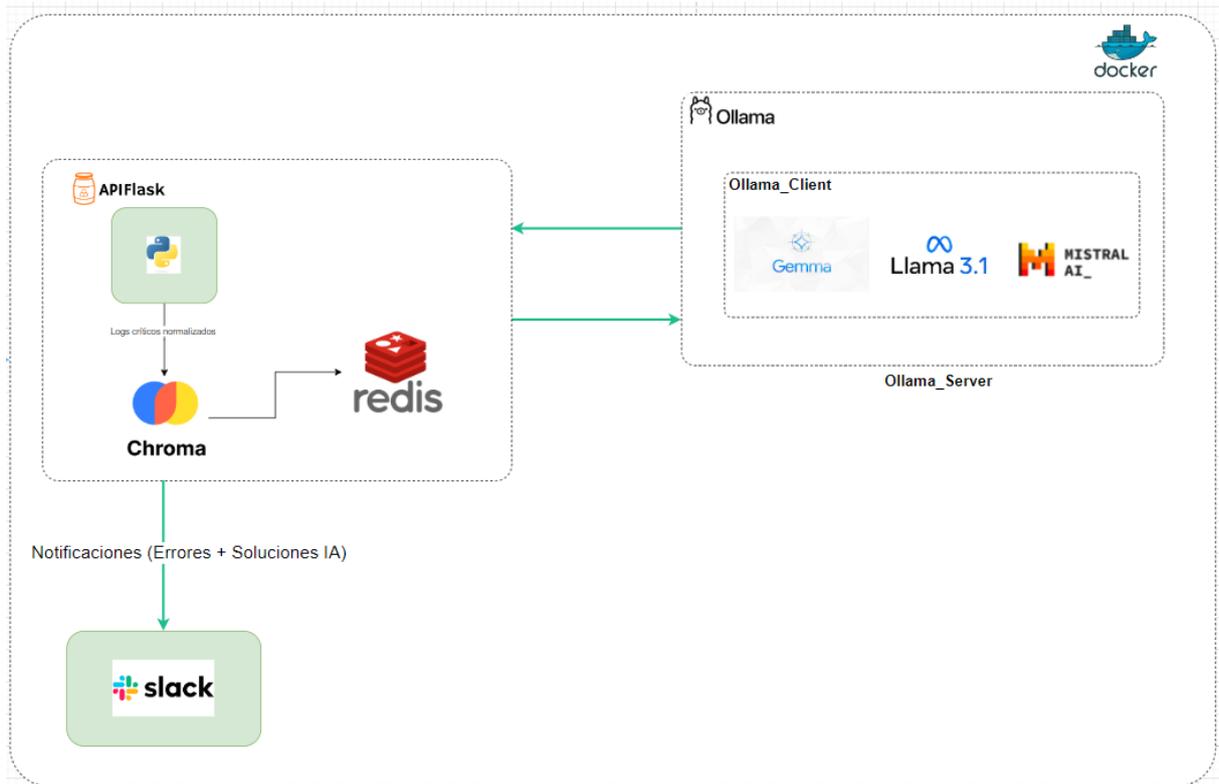


Figura 12. Arquitectura de procesamiento y notificación de logs críticos.

6.2.1.7. Propuesta Final de la Arquitectura

La arquitectura final integra de manera cohesiva todos los componentes descritos previamente, abarcando desde la generación de logs en los sistemas institucionales hasta su análisis y visualización. Este diseño asegura un flujo de datos eficiente, optimizando la recolección, procesamiento y almacenamiento para proporcionar a los administradores una herramienta robusta y escalable.

Con esta arquitectura, se logra:

- **Respuesta Rápida a Errores Críticos:** Los logs críticos son detectados y procesados en tiempo real, permitiendo alertas inmediatas y recomendaciones prácticas mediante herramientas como Slack.
- **Análisis Completo de Logs:** Todos los eventos registrados, tanto críticos como no críticos, se almacenan y visualizan en OpenSearch Dashboard, facilitando un monitoreo exhaustivo y la identificación de patrones.

Este diseño no solo responde a los requerimientos iniciales del proyecto, sino que también establece una base sólida para futuras mejoras y escalabilidad en la gestión de logs institucionales.

6.2.2. Arquitectura Centralización Logs

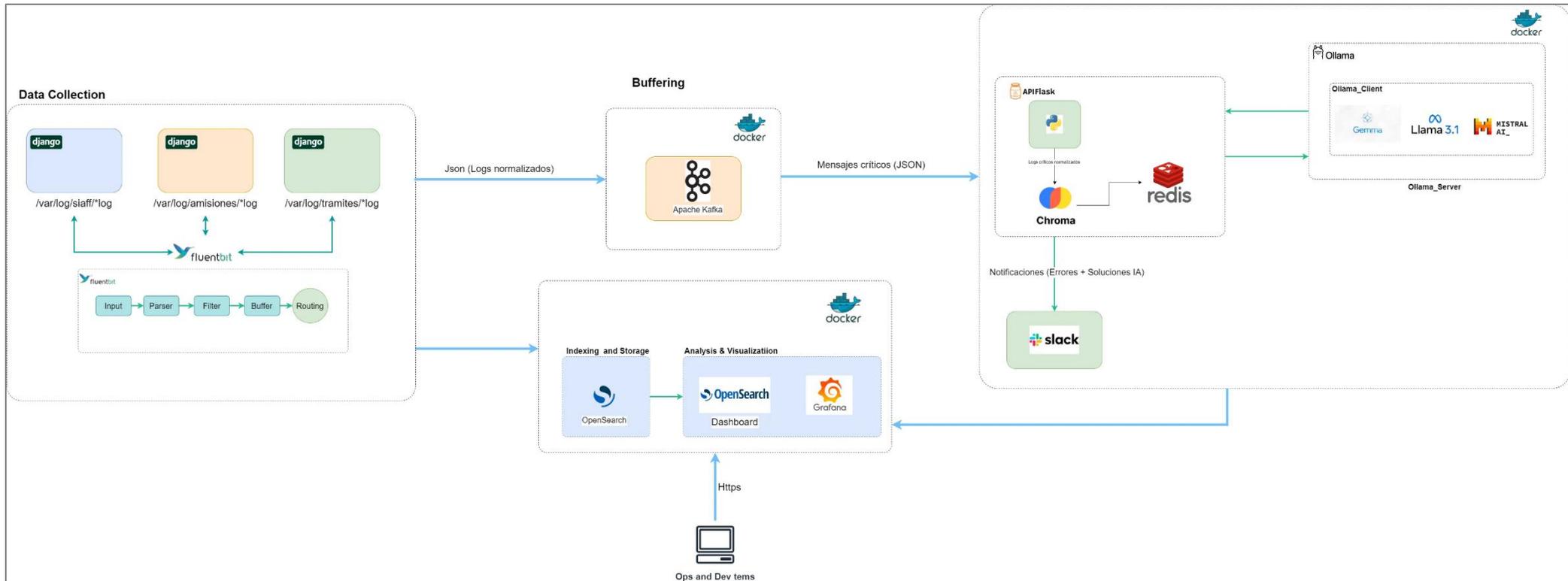


Figura 13. Arquitectura para la gestión centralizada de logs.

7. Discusión

Este trabajo se suma a los esfuerzos por optimizar la gestión centralizada de logs en entornos educativos, destacando el uso de herramientas de código abierto y modelos de inteligencia artificial para mejorar la detección y resolución de incidencias. A diferencia de otros estudios centrados en gestión de logs tradicionales, esta investigación integra un enfoque modular y escalable, capaz de adaptarse a las necesidades específicas de los sistemas institucionales de la Universidad Nacional de Loja (UNL).

Objetivo 1: Evaluación de Casos de Prueba y Modelos de IA

El diseño y configuración de la arquitectura para la gestión centralizada de logs permitió establecer una infraestructura inicial que garantiza portabilidad, escalabilidad y replicabilidad. Como parte del resultado **R1**, la configuración del entorno en una máquina virtual con Debian 12 demostró ser una solución viable para pruebas controladas. Esta etapa permitió identificar ventajas como la independencia del hardware anfitrión y la facilidad de replicación en otros entornos.

El resultado **R2** destaca la implementación de herramientas como Fluent Bit para la recolección de logs, Apache Kafka para la gestión de mensajes y OpenSearch Dashboard para la visualización. Aunque estas herramientas demostraron su eficacia en las pruebas, el uso de recursos compartidos resaltó la necesidad de optimizar el rendimiento en futuros entornos de producción. La modularidad del diseño también facilita futuras ampliaciones para adaptarse a sistemas más complejos.

Análisis de Resultados de Casos de Prueba

Para evaluar el desempeño de la integración propuesta, se llevaron a cabo 15 casos de prueba distribuidos en tres aplicaciones críticas: **SIAAF, Trámites y Admisiones**. En cada una de estas aplicaciones, se midieron los tiempos de respuesta de los modelos de IA, la precisión en las recomendaciones generadas y la efectividad de las herramientas utilizadas para la notificación y visualización de logs.

Las métricas obtenidas permitieron comparar tres modelos de IA: **LLaMA3, Mistral y Gemma2**, con el objetivo de analizar su capacidad de detección y resolución de logs críticos.

Se ejecutaron **15 casos de prueba** distribuidos en tres aplicaciones críticas:

- **SIAAF**: 5 casos de prueba

- **Trámites:** 5 casos de prueba
- **Admisiones:** 5 casos de prueba

El objetivo fue evaluar la **detección y resolución de logs críticos** mediante los modelos de IA **LLaMA3, Mistral y Gemma**.

Resultados Comparativos de Modelos de IA

Tabla 24. Comparación de Tiempos de Respuesta SIAAF

ID Caso	Descripción del Caso	LLaMA3 (s)	Mistral (s)	Gemma2 (s)
CP-SIAAF-001	Error de nombre de view inválido	38.8s	34.87 s	274.93 s
CP-SIAAF-002	Error en query con ID nulo	64.15 s	41.69 s	158.84 s
CP-SIAAF-003	Error en template inexistente	71.64 s	38.14 s	188.01 s
CP-SIAAF-004	Error de nombre de view inválido	58.08 s	32.98 s	165.96 s
CP-SIAAF-005	Error en template inexistente	78.73 s	41.66 s	127.15 s

El análisis de tiempos de respuesta mostró diferencias significativas entre los modelos:

- **Mistral:** Fue el modelo con los tiempos de respuesta más bajos en todas las aplicaciones evaluadas. Su rapidez lo convierte en una opción viable para entornos donde el tiempo de respuesta es un factor crítico.
- **LLaMA3:** Aunque presentó tiempos de respuesta ligeramente superiores a los de Mistral, destacó por su precisión en la detección y análisis de errores complejos. Esto sugiere que es más adecuado en escenarios donde la calidad del análisis es prioritaria sobre la velocidad.
- **Gemma2:** Mostró los tiempos de respuesta más altos en todas las pruebas, lo que indica que requiere optimización adicional antes de considerarse para una implementación en producción.

A nivel de precisión en la identificación de errores y generación de soluciones, **LLaMA3** obtuvo mejores resultados en problemas más complejos, mientras que Mistral mostró eficiencia en errores más comunes y recurrentes. En contraste, Gemma2 presentó inconsistencias en sus respuestas, lo que refuerza la necesidad de ajustes y optimización.

Tabla 25. Comparación de Tiempos de Respuesta TRÁMITES

ID Caso	Descripción del Caso	LLaMA3 (s)	Mistral (s)	Gemma2 (s)
CP-TRAMITES-001	Error al enviar notificación correo electrónico	87.24 s	43.43 s	137.52 s
CP-TRAMITES-002	Error de tipos al realizar operaciones	71.60 s	49.43 s	237.60 s
CP-TRAMITES-003	Error al concatenar string.	68.39 s	34.00 s	179.07 s
CP-TRAMITES-004	Conexión interrumpida con el servidor al consultar detalles de proceso	70.17 s	42.05 s	150.38 s
CP-TRAMITES-005	Timeout al consultar detalles de proceso	54.45 s	36.43 s	214.49 s

El análisis de tiempos de respuesta mostró diferencias significativas entre los modelos:

- **Mistral:** De manera similar a los resultados en la aplicación SIAAF, el modelo Mistral generalmente mostró los **menores tiempos de respuesta** en la mayoría de los casos de prueba para la aplicación Trámites. Esto indica que Mistral mantiene su eficiencia y rapidez en el procesamiento de logs, incluso en esta aplicación.
- **LLaMA3:** Este modelo, al igual que en la aplicación SIAAF, demostró tiempos de respuesta **ligeramente superiores** en comparación con Mistral. Sin embargo, se destaca su capacidad para ofrecer **mayor precisión** en la detección y análisis de errores complejos. Esto sugiere una consistencia en el rendimiento de LLaMA3, priorizando la calidad del análisis sobre la velocidad.
- **Gemma2:** El modelo Gemma también presentó los **mayores tiempos de respuesta** en comparación con los otros dos modelos en la aplicación Trámites. Este patrón consistente sugiere que Gemma requiere una **optimización adicional** en su tiempo de procesamiento para ser más eficiente en un entorno de producción.

En resumen, para la aplicación Trámites, Mistral sobresale por su rapidez, LLaMA3 por su precisión en errores complejos, y Gemma necesita mejoras en sus tiempos de respuesta. Es importante considerar que estos resultados se basan en los tiempos de respuesta, pero la precisión en las recomendaciones también es un factor clave para evaluar el rendimiento de cada modelo.

Al igual que en la aplicación SIAAF, la combinación de velocidad y precisión es fundamental para seleccionar el modelo más adecuado para cada escenario específico. Se debe considerar que, si bien Mistral es más rápido, LLaMA3 podría ser preferible para errores más complejos donde la precisión en el análisis es crucial. La optimización de Gemma podría ser necesaria si se desea implementar este modelo en un entorno donde la velocidad es una

prioridad.

Tabla 26. Comparación de Tiempos de Respuesta TRÁMITES

ID Caso	Descripción del Caso	LLaMA3 (s)	Mistral (s)	Gemma (s)
CP-ADMISIONES-001	Error al ubicar recursos .png	65.32 s	43.78 s	195.93 s
CP-ADMISIONES-002	Error al conectarse a la base de datos	63.36 s	43.82 s	226.32 s
CP-ADMISIONES-003	Error al cargar la asignación ítem para el aspirante	85.96 s	36.88 s	194.20 s
CP-ADMISIONES-004	Error al acceder a un atributo	8 4.3 5 s	44.78 s	290.23 s
CP-ADMISIONES-005	Error al configurar variables en el settings.py	77.97 s	42.96 s	179.34 s

El análisis de tiempos de respuesta mostró diferencias significativas entre los modelos:

- **Mistral:** Al igual que en las aplicaciones SIAAF y Trámites, el modelo Mistral generalmente mostró los **menores tiempos de respuesta** en la mayoría de los casos de prueba para la aplicación Admisiones. Esto confirma que Mistral mantiene su **eficiencia y rapidez** en el procesamiento de logs, independientemente de la aplicación.
- **LLaMA3:** Este modelo, de manera consistente con las otras aplicaciones, demostró tiempos de respuesta **ligeramente superiores** en comparación con Mistral en la aplicación Admisiones. Sin embargo, LLaMA3 se destaca por ofrecer **mayor precisión** en la detección y análisis de errores complejos, manteniendo su perfil de priorizar la calidad del análisis sobre la velocidad.
- **Gemma2:** El modelo Gemma2 también presentó los **mayores tiempos de respuesta** en comparación con los otros dos modelos en la aplicación Admisiones. Este patrón consistente en las tres aplicaciones indica que Gemma requiere **optimización adicional** en su tiempo de procesamiento para mejorar su eficiencia en entornos de producción.

En resumen, para la aplicación Admisiones, Mistral se destaca por su rapidez, LLaMA3 por su precisión en errores complejos, y Gemma necesita mejoras en sus tiempos de respuesta. Es importante considerar que estos resultados se basan en los tiempos de respuesta, pero la precisión en las recomendaciones también es un factor clave para evaluar el rendimiento de cada modelo.

Objetivo 2: Diseñar la arquitectura para la gestión centralizada de logs en los sistemas institucionales de la Universidad Nacional de Loja, mediante un estilo arquitectónico que represente la integración de OpenSearch e inteligencia artificial.

El análisis de casos de prueba en los sistemas SIAAF, Trámites y Admisiones permitió validar la funcionalidad de la arquitectura y los modelos de IA seleccionados. Como se describe en el resultado **R3**, los modelos LLaMA3, Mistral y Gemma2 generaron recomendaciones relevantes en la mayoría de los escenarios evaluados. Los tiempos de respuesta variaron según el modelo, siendo LLaMA3 el más balanceado en cuanto a precisión y velocidad.

El resultado **R4** aborda la visualización y notificación, destacando el uso combinado de OpenSearch Dashboard y Slack. Mientras que OpenSearch proporcionó un repositorio centralizado para el análisis de logs, Slack permitió una respuesta inmediata ante errores críticos. Esto facilitó a los administradores identificar y atender incidencias de manera eficiente. Sin embargo, algunos errores relacionados con APIs externas, como el envío de mensajes SMS, no pudieron ser validados debido a restricciones de acceso, lo que resalta la necesidad de gestionar dependencias externas de manera más robusta.

Este trabajo presenta una arquitectura modular que integra herramientas de código abierto y modelos de IA avanzados, demostrando su efectividad en la gestión centralizada de logs. Se aporta un marco replicable y escalable para instituciones educativas que busquen mejorar su capacidad de monitoreo, detección de errores y respuesta ante incidencias. Además, la introducción de embeddings semánticos y el uso combinado de OpenSearch y Slack fortalecen las posibilidades de integración y análisis de datos.

En los casos de prueba realizados, los modelos de IA mostraron diferencias claras en su desempeño:

- Mistral destacó por ser el más rápido en el procesamiento de logs, lo que lo hace ideal para entornos donde la velocidad es prioritaria.
- LLaMA3 demostró su precisión en errores complejos, siendo el modelo preferido para escenarios donde la calidad del análisis es crucial.
- Gemma2 requirió tiempos de procesamiento más largos, lo que sugiere la necesidad de optimización para entornos de producción.

Estas diferencias subrayan la importancia de seleccionar el modelo de IA en función del contexto específico y las necesidades del sistema.

8. Conclusiones

- Dando respuesta a la pregunta de investigación, se concluye que la implementación de una arquitectura centralizada de gestión de logs, basada en herramientas como Fluent Bit, OpenSearch y Apache Kafka, ha demostrado ser una estrategia efectiva para mejorar la capacidad de respuesta ante incidentes en los sistemas institucionales de la Universidad Nacional de Loja. Esta solución ha permitido recopilar, procesar y analizar grandes volúmenes de datos de manera estructurada, facilitando la identificación temprana de problemas y la toma de decisiones más informadas. Aunque la implementación en entornos de producción no fue parte del alcance del proyecto, los resultados de las pruebas controladas evidencian que la centralización de logs puede optimizar significativamente los tiempos de diagnóstico y resolución de incidentes.
- La integración de modelos de inteligencia artificial como LLaMA3, Mistral y Gemma2 ha aportado un valor significativo al análisis de logs y la mitigación de incidentes. Estos modelos demostraron ser capaces de interpretar errores complejos y generar recomendaciones prácticas para su resolución. Las pruebas realizadas revelaron diferencias clave en su desempeño: mientras que Mistral destacó por su rapidez, LLaMA3 sobresalió por su precisión en la resolución de problemas complejos. No obstante, se identificaron retos en el manejo de logs no estructurados, lo que resalta la necesidad de mejorar la normalización de datos para maximizar la efectividad de los modelos.
- La arquitectura modular propuesta, diseñada con herramientas de código abierto, ha probado ser eficaz y adaptable para gestionar logs de sistemas con estructuras diversas. La integración de Fluent Bit para la recolección, Apache Kafka para la gestión de mensajes y Redis para el almacenamiento en caché asegura una operatividad eficiente. La flexibilidad y escalabilidad de esta arquitectura permiten su futura expansión y adaptación a nuevos desafíos. La validación experimental llevada a cabo en este proyecto confirma que el diseño no solo satisface las necesidades actuales, sino que también ofrece un marco robusto para el desarrollo de capacidades adicionales.
- De cara al futuro, es recomendable entrenar los modelos de IA con datos históricos específicos de los sistemas institucionales para mejorar la precisión y adaptabilidad de sus recomendaciones. Además, se sugiere implementar pipelines de CI/CD para automatizar el despliegue de la arquitectura en entornos de

producción, minimizando errores manuales y facilitando actualizaciones. Estas mejoras no solo potenciarían la eficiencia del sistema, sino que también garantizarían una integración más fluida y consistente en el entorno operativo institucional. Con estas propuestas, la solución podría consolidarse como un estándar para la gestión de logs en entornos similares.

9. Recomendaciones

- Se recomienda llevar a cabo una implementación escalonada de la arquitectura propuesta en un entorno de producción. Iniciar con un sistema piloto permitirá validar su efectividad en condiciones reales, identificar posibles ajustes en la configuración y optimizar el rendimiento según las necesidades específicas de los sistemas institucionales. Este enfoque mitigará riesgos y facilitará una transición ordenada hacia una adopción completa.
- Para mejorar la precisión, adaptabilidad y relevancia de las soluciones generadas por los modelos de inteligencia artificial, se sugiere entrenarlos con datos históricos y específicos de los sistemas institucionales. Este entrenamiento personalizado permitirá que los modelos se ajusten a las particularidades de los incidentes más frecuentes y ofrezcan respuestas más robustas y contextualizadas, mejorando así la experiencia del administrador.
- Es recomendable implementar pipelines de integración y entrega continua (CI/CD) para automatizar el despliegue de la arquitectura en entornos de producción. Esto incluye la configuración automatizada de herramientas como Fluent Bit, Kafka y Redis, lo que reducirá errores manuales, garantizará la consistencia en la configuración y permitirá una actualización más ágil y eficiente del sistema. Además, esta automatización soportará expansiones futuras y aumentará la confiabilidad de la infraestructura.

10. Bibliografía

- [1] L. A. Herrera Lara, "Implementación de un Centro de Operaciones de Seguridad y Redes (NSOC) usando herramientas open source para la infraestructura industrial de la Empresa Eléctrica Quito," Tesis de Grado, Escuela Politécnica Nacional, Facultad de Ingeniería Eléctrica y Electrónica, Quito, Ecuador, 2022.
- [2] F. N. Tahir, "Gestión centralizada de Logs en clústeres Kubernetes," Trabajo de Fin de Máster, Universitat Politècnica de València, Valencia, España, 2022.
- [3] V. H. Figueroa Fernández, "Estructura de seguridad para el análisis y gestión de logs de aplicaciones multicapa," **Ciencia Sur**, vol. 6, no. 7, pp. 15-33, ene. 2021. [En línea]. Disponible en: [URL si aplica].
- [4] I. Ortega Falces, "Monitorización de una plataforma educativa utilizando Elastic Stack," Trabajo de Fin de Grado, Grado en Ingeniería en Tecnologías de Telecomunicación, Universidad Pontificia Comillas, Madrid, España, 2024.
- [5] Y. Herrera y F. Heredia, "Monitorización de Sistemas con Tecnologías de Big Data," **Ecuadorian Science Journal**, vol. 5, núm. esp. 3, pp. 152-161, 2021. [En línea]. Disponible en: [URL si aplica].
- [6] J. Montero, "Centralización de logs en microservicios con Fluent Bit," **Teldat Blog**, 27 de octubre de 2020. [En línea]. Disponible en: [URL si aplica].
- [7] S. V. Michis, "Gestión centralizada de Logs. Herramientas de código abierto," Trabajo de Fin de Grado, Universitat Politècnica de València, Valencia, España, 2022.
- [8] J. Saiz Gutiérrez, "Implementación del stack ELK en entornos Big Data," Tesis de Maestría, Universidad Tecnológica de Madrid, Facultad de Ingeniería Informática, Madrid, España, 2024.
- [9] New Relic, "Las mejores prácticas de la administración de logs," Whitepaper, ago. 2022. [En línea]. Disponible en: [URL si aplica].
- [10] C. Gantiva Duarte y C. Castellanos Calero, "Propuesta para implementar un sistema de monitoreo de red mediante uso de agentes y gestión de logs para la empresa Datamarket Solutions S.A.S.," Trabajo de Grado, Universidad Distrital Francisco José de Caldas, Facultad Tecnológica, Programa de Ingeniería Telemática, Bogotá D.C., Colombia, 2022.
- [11] F. Liu, Z. Kang y X. Han, "Optimizing RAG Techniques for Automotive Industry PDF Chatbots: A Case Study with Locally Deployed Ollama Models," **arXiv preprint arXiv:2408.05933**, 2024. [En línea]. Disponible en: <https://arxiv.org/abs/2408.05933>. [Accedido: 17-mar-2025].
- [12] M. A. Suryani, S. Karmakar y B. Mathiak, "Exploration of Hugging Face Models by Heterogeneous Information Network and Linking Across Scholarly Repositories," en **Social Networks Analysis and Mining: 16th International Conference, ASONAM 2024, Rende, Italy,*

September 2–5, 2024, Proceedings, Part III*, Lecture Notes in Computer Science, vol. 15213, Cham: Springer, 2024, pp. 371–386.

[13] T.-H. C. Locke, H. Li y W. Liu, "Logsassist: Assisting log analysis through log summarization," *IEEE Transactions on Software Engineering*, vol. 48, pp. 3227–3241, 2022.

[14] Red Hat, "What is Grafana," *Red Hat*, 2024. [En línea]. Disponible en: <https://www.redhat.com/es/topics/data-services/what-is-grafana>. [Accedido: 22-nov-2024].

[15] Y. Taleb, K. McGehee, N. Yan, S. Wang, S. C. Müller y A. Samuels, "Amazon MemoryDB: A Fast and Durable Memory-First Cloud Database," en *Companion of the 2024 International Conference on Management of Data (SIGMOD-Companion '24)*, Santiago, Chile, junio 2024, pp. 309-320.

[16] E. Elshoubary y T. Radwan, "Studying the Efficiency of the Apache Kafka System Using the Reduction Method, and Its Effectiveness in Terms of Reliability Metrics Subject to a Copula Approach," *Applied Sciences*, vol. 14, no. 15, p. 6758, 2024.

[17] S. H. Castillo, "Solución de centralización y visualización de logs para Nova Servidores," Trabajo de Diploma, Universidad de las Ciencias Informáticas, Facultad 1, La Habana, Cuba, 2017.

[18] E. Calderon, G. del Campo y A. Santamaría, "Monitoring framework for the performance evaluation of an IoT platform with Elasticsearch and Apache Kafka," *Information Systems Frontiers*, 2023.

[19] R. M. Castro, "Implementación de servidores con herramientas DevOps: creación de servidor de logs con software libre con Ansible," Trabajo de Fin de Grado, Escuela Politécnica Nacional, Facultad de Ingeniería de Sistemas, Quito, Ecuador, 2022.

[20] Logit.io, "What are OpenSearch dashboards?," *Logit.io*, 2023. [En línea]. Disponible en: <https://logit.io/blog/post/what-are-opensearch-dashboards>. [Accedido: 21-ago-2024].

[21] AWS, "What is OpenSearch?," *Amazon Web Services*, 2024. [En línea]. Disponible en: <https://aws.amazon.com/es/what-is/opensearch/>. [Accedido: 21-ago-2024].

[22] DataCamp, "Top open-source LLMs: The future of natural language processing," *DataCamp*, 2024. [En línea]. Disponible en: <https://www.datacamp.com/es/blog/top-open-source-llms>. [Accedido: 21-ago-2024].

[23] Fluent Bit, "Fluent Bit: Lightweight and High-Performance Log Processor," Documentación oficial, 2024. [En línea]. Disponible en: <https://fluentbit.io>. [Accedido: 21-ago-2024].

[24] M. K. Dahdal, "How Slack Facilitates Communication and Collaboration in University Courses," *Social Science Computer Review*, vol. 41, no. 1, pp. 123-139, 2023.

11. Anexos

Anexo 1: Autorización de acceso a registro de logs.



UNL

Universidad
Nacional
de Loja

Dirección de
Tecnología de Información

Memorando Nro.: UNL-DTI-2024-0290-M

Loja, 08 de noviembre de 2024

PARA: Sr. Edgar Manuel Macas Loja
Analista de Sistemas de Información 1

ASUNTO: Autorización el acceso al repositorio de los sistemas institucionales Admisiones, SIAAF y Trámites de la Universidad Nacional de Loja.

En atención al oficio s/n de fecha Loja, 5 de noviembre del 2024, remitido a través de correo institucional suscrito por el maestrante Ing. Edgar Macas, mismo que en su parte pertinente manifiesta "(...) *Por medio de la presente, me dirijo a usted para solicitar formalmente el acceso al repositorio de los sistemas institucionales Admisiones, SIAAF y Trámites de la Universidad Nacional de Loja, en el marco de mi proyecto de titulación de maestría, titulado "Arquitectura de Gestión Centralizada de Logs con OpenSearch e Inteligencia Artificial en los Sistemas Institucionales de la Universidad Nacional de Loja"*.

Este proyecto tiene como objetivo diseñar una propuesta de infraestructura para la gestión centralizada de logs, integrando OpenSearch y un modelo de inteligencia artificial que facilite la detección y mitigación de incidentes en los sistemas mencionados. Cabe aclarar que el alcance del proyecto se limita a la conceptualización y diseño de la arquitectura, sin llegar a una implementación en el entorno productivo.

En una conversación previa con el Ing. Danny Muñoz, Especialista de Software, se comentó de manera general sobre los objetivos del proyecto y el uso que se daría a la información recolectada. Para avanzar en el análisis y evaluación, es necesario levantar los sistemas en un ambiente local, lo cual permitirá realizar pruebas y simulaciones controladas. Me comprometo a tratar toda la información manejada en el desarrollo de este proyecto con la máxima confidencialidad, evitando la exposición de datos sensibles o cualquier situación que comprometa a la institución o a la Dirección de Tecnologías de la Información (...)"

En virtud de lo antes expuesto, autorizo el acceso al repositorio de los sistemas institucionales Admisiones, SIAAF y Trámites de la Universidad Nacional de Loja.

Sin otro particular, aprovecho la oportunidad para reiterarle el testimonio de mi más alta consideración.

Atentamente,

Anexo 2. Manual de Instalación de herramientas para la gestión de logs.

Introducción

La instalación de las herramientas se llevará a cabo en una máquina virtual configurada con VMware® Workstation 17 Pro, utilizando una licencia de uso personal. Esta máquina virtual servirá como un entorno controlado y portable para ejecutar los casos de prueba asociados al diseño de la arquitectura.

Una vez instalada la máquina virtual, continuamos con la configuración del usuario dentro del sistema operativo de la máquina virtual, asegurando que los permisos y ajustes iniciales estén correctamente establecidos. Posteriormente, se procede con la instalación de Docker, una herramienta esencial que permitirá gestionar y desplegar contenedores de manera eficiente, facilitando la instalación y configuración de los servicios necesarios para el proyecto.

Docker será la base para implementar componentes clave como Fluent Bit, OpenSearch, Apache Kafka, OpensearchDashboard o Grafana y el modelo de inteligencia artificial mediante Ollama o LM Studio, asegurando la modularidad y flexibilidad del entorno.

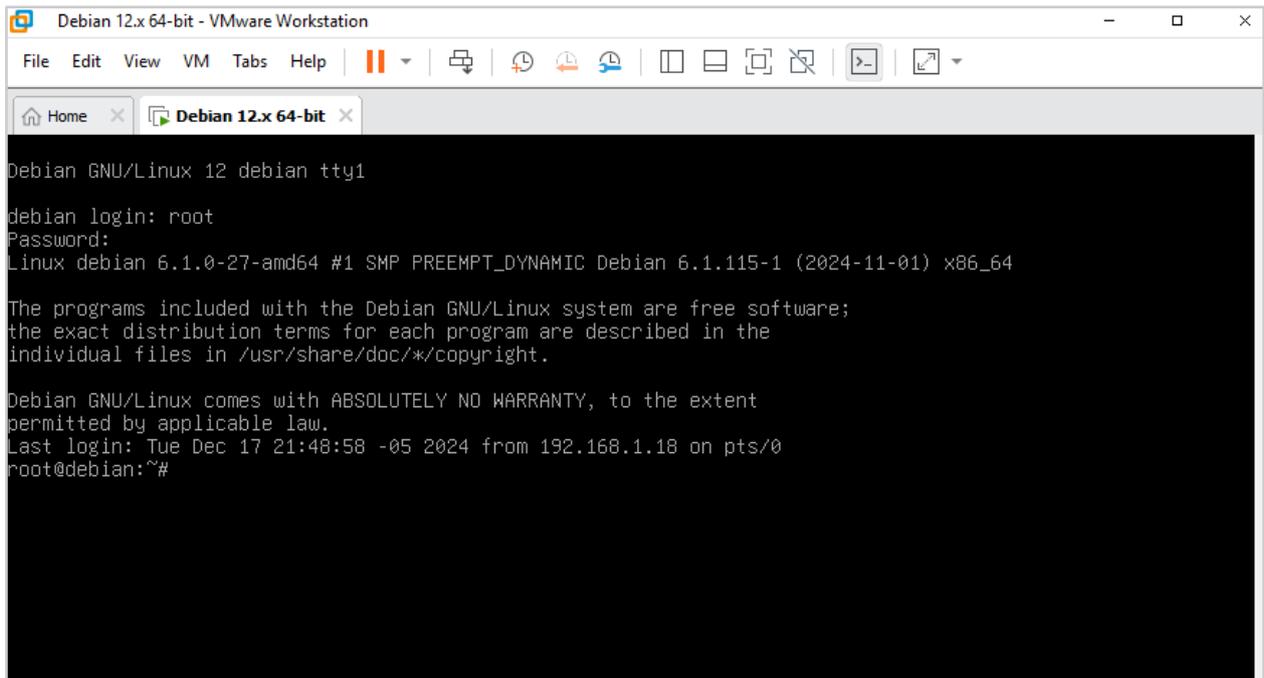


Figura. 1. Ejecución de Máquina virtual con Debian12

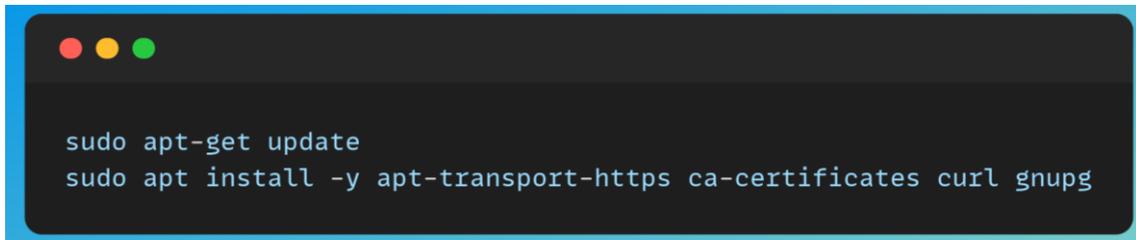
Requisitos Previos

- Sistema operativo Ubuntu 22.04 o superior.
- Acceso a una terminal con privilegios de administrador.
- Conexión a internet para la descarga de paquetes.
- Instalación de Docker y Docker Compose.
- Configuración adecuada del entorno virtual.

Instalación de Herramientas necesarias

Instalación de Docker

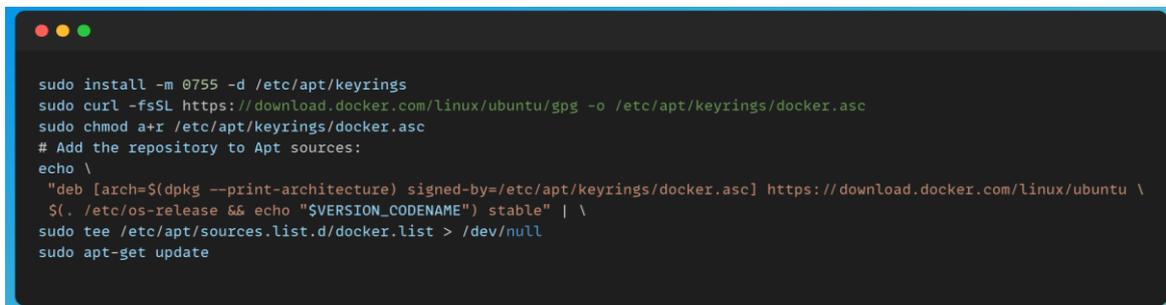
Paso 1: Actualización del Sistema y Dependencias

A terminal window with a dark background and a light blue border. It contains two lines of terminal commands: 'sudo apt-get update' and 'sudo apt install -y apt-transport-https ca-certificates curl gnupg'.

```
sudo apt-get update
sudo apt install -y apt-transport-https ca-certificates curl gnupg
```

Figura 2. Instalación de dependencias.

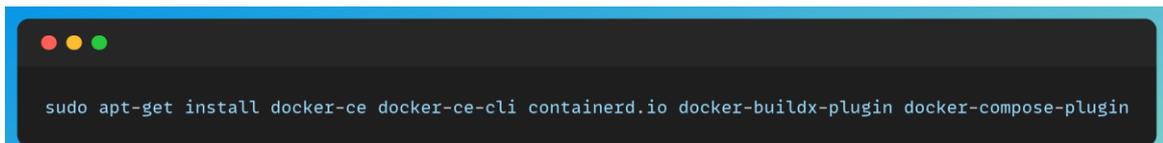
Paso 2: Importar Clave GPG y Repositorio Docker

A terminal window with a dark background and a light blue border. It contains several lines of terminal commands for installing GPG keys and adding the Docker repository to the sources list.

```
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
# Add the repository to Apt sources:
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

Figura 3. Configuración del repositorio de Docker.

Paso 3: Instalación de Docker

A terminal window with a dark background and a light blue border. It contains one line of terminal command for installing Docker and its related components.

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Figura 4. Instalación de Docker.

Instalación OpenSearch y OpensearchDashboard

A continuación, se detallan los pasos necesarios para la instalación de OpenSearch en la versión 2.18.0.

La instalación la realizaremos igualmente mediante la configuración de Docker

Paso 1: Optimizar el sistema y ajustar mapas de memoria

```
# Editamos el archivo sysctl config
sudo vi /etc/sysctl.conf
# Agregamos la nueva línea o actualizamos el valor si existe y guardamos los cambios
vm.max_map_count=262144
# Recargamos los parámetros del kernel usando sysctl
sudo sysctl -p
# Verificamos que los cambios han sido aplicados, verificando el nuevo valor asignado
cat /proc/sys/vm/max_map_count
```

Figura 5. Optimización del sistema.

Paso 2: Ejecutar OpenSearch en un Contenedor Docker

```
docker pull opensearchproject/opensearch:2
```

Para iniciar OpenSearch en un contenedor Docker, se ejecuta el siguiente comando:

```
docker run -d -p 9200:9200 -p 9600:9600 -e "discovery.type=single-node" opensearchproject/opensearch:latest
```

Figura 6. Inicialización del contenedor de OpenSearch.

Paso 3: Configuración de Contraseña de Administrador

Para OpenSearch 2.12 o superior, es necesario establecer una contraseña de administrador personalizada antes de la instalación mediante el siguiente comando:

```
docker run -d -p 9200:9200 -p 9600:9600 -e "discovery.type=single-node" -e "OPENSEARCH_INITIAL_ADMIN_PASSWORD=<custom-admin-password>" opensearchproject/opensearch:latest
```

Figura 7. Ajuste de contraseña inicial.

Paso 4: Verificación de la Instalación

Para verificar que OpenSearch se encuentra ejecutándose correctamente, enviamos una solicitud al puerto 9200 con el siguiente comando:

```
curl https://localhost:9200 -ku admin:<custom-admin-password>
```

Si la ejecución es correcta, obtenemos la siguiente respuesta:

```
{
  "name": "a937e018cee5",
  "cluster_name": "docker-cluster",
  "cluster_uuid": "GLAJAG6bTeWErFUy_d-CLw",
  "version": {
    "distribution": "opensearch",
    "number": <version>,
    "build_type": <build-type>,
    "build_hash": <build-hash>,
    "build_date": <build-date>,
    "build_snapshot": false,
    "lucene_version": <lucene-version>,
    "minimum_wire_compatibility_version": "7.10.0",
    "minimum_index_compatibility_version": "7.0.0"
  },
  "tagline": "The OpenSearch Project: https://opensearch.org/"
}
```

Figura 8. Verificación de ejecución de contenedor.

Paso 5: Configuración de un Clúster de OpenSearch usando Docker Compose

Aunque es técnicamente posible configurar un clúster de OpenSearch creando contenedores de uno en uno, resulta mucho más eficiente definir todo el entorno en un archivo YAML y utilizar Docker Compose para gestionar el clúster, compuesto por OpenSearch y OpenSearch Dashboards.

Creamos un archivo llamado docker-compose.yml en el directorio de inicio del host. Este archivo define un clúster que incluye tres contenedores: dos para ejecutar el servicio de OpenSearch y uno para OpenSearch Dashboards. Estos contenedores se comunican a través de una red de puente llamada opensearch-net y utilizan dos volúmenes persistentes, uno para cada nodo de OpenSearch. Dado que este archivo no desactiva explícitamente la configuración de seguridad de demostración, se instalarán certificados TLS auto firmados y se crearán usuarios internos con nombres de usuario y contraseñas predeterminadas.

Paso 6: Establecer una contraseña de administrador personalizada

A partir de OpenSearch 2.12, es necesario definir una contraseña de administrador personalizada para habilitar la configuración de seguridad de demostración.

Procedemos con la creación de un archivo .env en el mismo directorio donde se encuentra el archivo docker-compose.yml e incluimos la variable:

OPENSEARCH_INITIAL_ADMIN_PASSWORD junto con una contraseña segura como valor.

Paso 7: Configuración del Archivo docker-compose.yml

Generamos el archivo de configuración .yml con las siguientes configuraciones:

```
version: '3'
services:
  fluent-bit:
    image: cr.fluentbit.io/fluent/fluent-bit:latest
    container_name: fluent-bit
    volumes:
      - C:/Users/Edgar/PycharmProjects/giosoftware:/fluent-bit/logs
      - C:/Users/Edgar/PycharmProjects/giosoftware/fluent-bit.conf:/fluent-bit/etc/fluent-bit.conf
      - C:/Users/Edgar/PycharmProjects/fluentbit-db:/fluent-bit/logs/db
    networks:
      - opensearch-net
    environment:
      - FLB_CONFIG_FILE=/fluent-bit/etc/fluent-bit.conf
  opensearch-node1:
    image: opensearchproject/opensearch:latest
    container_name: opensearch-node1
    environment:
      - cluster.name=opensearch-cluster
      - node.name=opensearch-node1
      - discovery.seed_hosts=opensearch-node1,opensearch-node2
      - cluster.initial_cluster_manager_nodes=opensearch-node1,opensearch-node2
      - bootstrap.memory_lock=true # along with the memlock settings below, disables swapping
      - OPENSEARCH_JAVA_OPTS=-Xms512m -Xmx512m # minimum and maximum Java heap size, recommend setting both to 50% of system RAM
      - OPENSEARCH_INITIAL_ADMIN_PASSWORD=StrongPass1@ # Sets the demo admin user password when using demo configuration, required for OpenSearch 2.12 and higher
    ulimits:
      memlock:
        soft: -1
        hard: -1
      nofile:
        soft: 65536 # maximum number of open files for the OpenSearch user, set to at least 65536 on modern systems
        hard: 65536
    volumes:
      - opensearch-data1:/usr/share/opensearch/data
    ports:
      - 9200:9200
      - 9600:9600 # Required for Performance Analyzer
    networks:
      - opensearch-net
  opensearch-node2:
    image: opensearchproject/opensearch:latest
    container_name: opensearch-node2
    environment:
      - cluster.name=opensearch-cluster
      - node.name=opensearch-node2
      - discovery.seed_hosts=opensearch-node1,opensearch-node2
      - cluster.initial_cluster_manager_nodes=opensearch-node1,opensearch-node2
      - bootstrap.memory_lock=true
      - OPENSEARCH_JAVA_OPTS=-Xms512m -Xmx512m
      - OPENSEARCH_INITIAL_ADMIN_PASSWORD=StrongPass1@
    ulimits:
      memlock:
        soft: -1
        hard: -1
      nofile:
        soft: 65536
        hard: 65536
    volumes:
      - opensearch-data2:/usr/share/opensearch/data
    networks:
      - opensearch-net
  opensearch-dashboards:
    image: opensearchproject/opensearch-dashboards:latest
    container_name: opensearch-dashboards
    ports:
      - 5601:5601
    expose:
      - '5601'
    environment:
      OPENSEARCH_HOSTS: '["https://opensearch-node1:9200","https://opensearch-node2:9200"]'
      OPENSEARCH_USERNAME: "admin"
      OPENSEARCH_PASSWORD: "StrongPass1@"
      opensearch.ssl.verificationMode: "none" # Solo para pruebas, evita la verificación del SSL
    networks:
      - opensearch-net
volumes:
  opensearch-data1:
  opensearch-data2:
networks:
  opensearch-net:
```

Figura 9. Archivo de docker-compose.yml

Este archivo facilita la creación de un entorno completo para el clúster de OpenSearch y OpenSearch Dashboards, permitiendo además la integración de Fluent Bit para la gestión de logs.

Desde el directorio de inicio de nuestro host, que contiene el archivo docker-compose.yml, creamos e iniciamos los contenedores en modo separado con el siguiente comando:

```
docker-compose up -d
```

Luego, procedemos con la verificación de los contenedores se hayan iniciado correctamente con el siguiente comando:

```
docker-compose ps
```

Paso 8: Acceso a OpenSearch Dashboards

Para acceder a los paneles de OpenSearch, abrimos un navegador e ingresamos la siguiente dirección: `http://localhost:5601`. Debemos ingresar con el usuario `admin` y con la contraseña ingresada en el paso 4, las cuales son las siguientes:

- Usuario: `admin`
- Contraseña: `admin`

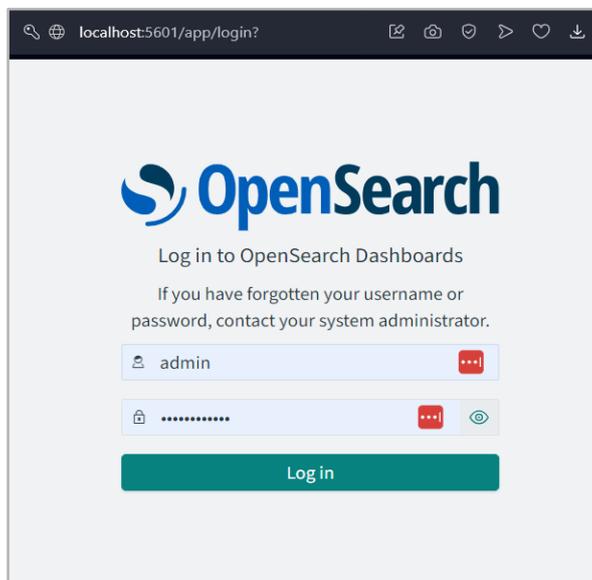


Figura 8. Inicio de sesión de interfaz OpenSearchDashboard.

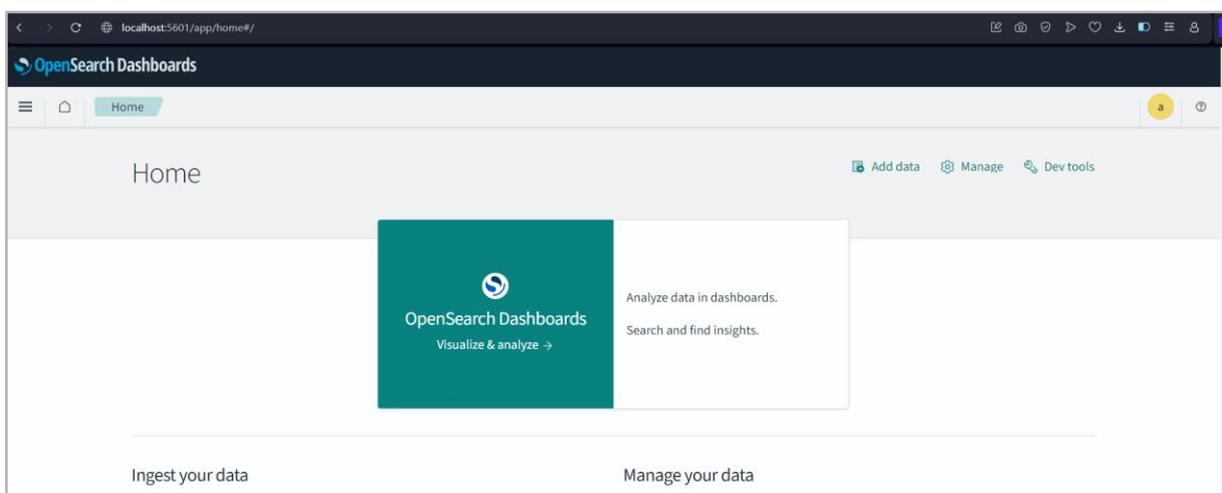


Figura 9. Interfaz principal de OpenSearchDashboard.

Configuración del Recolector de Logs: Fluent Bit

Paso 1: Instalación del contenedor FluentBit

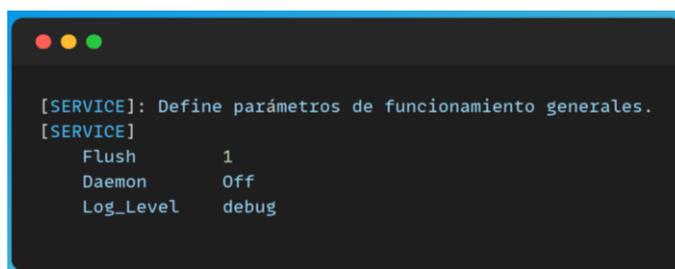
Las imágenes de contenedor de Fluent Bit están disponibles en Docker Hub para entornos de producción y pueden implementarse en múltiples arquitecturas. Para iniciar la instalación, ejecutamos el siguiente comando:

```
docker run -ti cr.fluentbit.io/fluent/fluent-bit
```

Paso2: Configuración de Fluent Bit

Fluent Bit permite la configuración mediante un archivo de configuración principal que controla las entradas de logs, filtros y salidas. Este archivo organiza la configuración en secciones clave: SERVICE, INPUT, FILTER y OUTPUT.

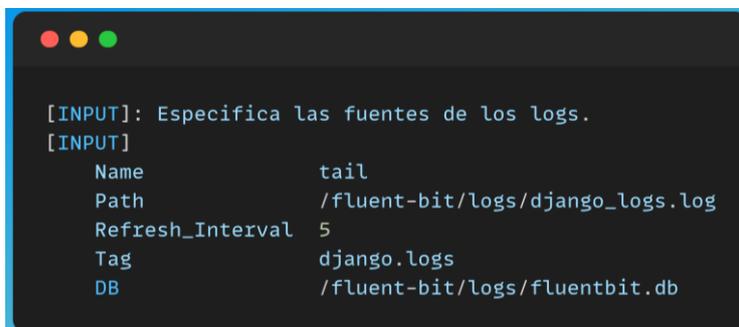
A continuación, detallamos la estructura de configuración de Fluent Bit:



```
[SERVICE]: Define parámetros de funcionamiento generales.
[SERVICE]
  Flush      1
  Daemon    Off
  Log_Level  debug
```

Figura 10. Sección SERVICE de la configuración de fluentbit.

- **Flush:** Intervalo en segundos para enviar los datos recolectados.
- **Daemon:** Define si Fluent Bit debe ejecutarse como un demonio.
- **Log_Level:** Nivel de registro, aquí configurado en debug para detalles adicionales durante la prueba.



```
[INPUT]: Especifica las fuentes de los logs.
[INPUT]
  Name          tail
  Path          /fluent-bit/logs/django_logs.log
  Refresh_Interval 5
  Tag           django_logs
  DB            /fluent-bit/logs/fluentbit.db
```

Figura 11. Sección INPUT de la configuración de fluentbit.

- **Name:** Tipo de entrada; aquí, tail para leer archivos.
- **Path:** Ruta al archivo de logs de la aplicación (en este ejemplo, logs de Django).

- **Refresh_Interval:** Frecuencia en segundos para revisar el archivo en busca de nuevos logs.
- **Tag:** Etiqueta asignada a estos logs para facilitar su filtrado.
- **DB:** Ubicación del archivo de estado para realizar seguimiento de logs procesados.

```
[FILTER]: Configura filtros para modificar o enriquecer los logs.
[FILTER]
  Name      grep
  Match     django.logs
  Regex     severity ERROR # Opcional: solo errores
```

Figura 12. Sección FILTER de la configuración de fluentbit.

- **Name:** Tipo de filtro; aquí grep para filtrar mensajes específicos.
- **Match:** Etiqueta de los logs a los que aplicar el filtro.
- **Regex:** Expresión regular para filtrar solo logs que contienen la palabra ERROR, si es necesario.

```
[OUTPUT]: Define el destino de los logs.
[OUTPUT]
  Name      opensearch
  Match     django.logs
  Host      host.docker.internal # IP o nombre del host de OpenSearch
  Port      9200
  Index     django_logs
  Type      _doc
  HTTP_User admin # Opcional si tu OpenSearch tiene autenticación
  HTTP_Passwd StrongPass1@ #Opcional si tu OpenSearch tiene autenticación
  tls       On
  tls.verify Off
```

Figura 13. Sección OUTPUT de la configuración de fluentbit.

- **Name:** Tipo de salida; en este caso opensearch.
- **Match:** Etiqueta de los logs que se enviarán a OpenSearch.
- **Host y Port:** Dirección y puerto de OpenSearch.
- **Index:** Nombre del índice en OpenSearch donde se almacenarán los logs.
- **HTTP_User y HTTP_Passwd:** Credenciales si OpenSearch requiere autenticación.
- **tls y tls.verify:** Configuración TLS para la conexión segura.

Instalación Apache Kafka



Figura 14. Apache Kafka como intermediario para la transmisión y procesamiento de datos en tiempo real entre sistemas.

Paso 1: Configuración de Apache Kafka con Docker Compose

Para incluir Apache Kafka, es necesario editar el archivo `docker-compose.yml` y definir dos servicios: `zookeeper` y `kafka`. Esta configuración permite el despliegue y la conexión entre ambos servicios de manera sencilla.

Estructura del archivo `docker-compose.yml`:

```
version: '2'
services:
  zookeeper:
    image:wurstmeister/zookeeper:latest
    ports:
      - "2181:2181"
  kafka:
    image:wurstmeister/kafka:latest
    ports:
      - "9092:9092"
    expose:
      - "9093"
    environment:
      KAFKA_ADVERTISED_LISTENERS: INSIDE://kafka:9093,OUTSIDE://localhost:9092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INSIDE:PLAINTEXT,OUTSIDE:PLAINTEXT
      KAFKA_LISTENERS: INSIDE://0.0.0.0:9093,OUTSIDE://0.0.0.0:9092
      KAFKA_INTER_BROKER_LISTENER_NAME: INSIDE
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_CREATE_TOPICS:"my-topic:1:1"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
```

Figura 15. Configuración de KAFKA en contenedor.

Esta sección del `docker-compose.yml` define dos servicios: `zookeeper` y `kafka`. El servicio Kafka está configurado para exponer los puertos 9092 y 9093.

Servicio Zookeeper

- Zookeeper actúa como un servicio de coordinación para Kafka.
- Está configurado para exponer el puerto 2181, que es el puerto de escucha predeterminado de Zookeeper.

Servicio Kafka

- Este servicio utiliza la imagen wurstmeister/kafka, que es ampliamente utilizada para entornos Docker de Kafka.
- Puertos: El puerto 9092 se mapea para el acceso externo, mientras que 9093 es un puerto expuesto para la comunicación interna en la red de contenedores.
- Variables de entorno:
 - KAFKA_ADVERTISED_LISTENERS: Define los listeners de Kafka para comunicaciones internas (INSIDE) y externas (OUTSIDE).
 - KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: Configura los protocolos de seguridad para cada listener.
 - KAFKA_LISTENERS: Especifica las direcciones de escucha para Kafka en cada listener (interno y externo).
 - KAFKA_INTER_BROKER_LISTENER_NAME: Define el listener de comunicación entre brokers (INSIDE).
 - KAFKA_ZOOKEEPER_CONNECT: Configura la conexión a Zookeeper.
 - KAFKA_CREATE_TOPICS: Crea un tema predeterminado, my-topic, con 1 partición y 1 réplica.
- Volúmenes:
 - Incluye el volumen /var/run/docker.sock para que Kafka pueda resolver dinámicamente las direcciones IP de los contenedores.

Pruebas y Verificación de Kafka

Para asegurarse de que Kafka está funcionando correctamente, es recomendable realizar una prueba de publicación y consumo de mensajes.

Conectar con Kafka:

- Usar herramientas como kafka-console-producer.sh y kafka-console-consumer.sh para enviar y recibir mensajes en el tema my-topic.

Ejemplo de Comandos de Prueba:

Productor:

```
docker exec -it <kafka_container_id> /opt/kafka/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic my-topic
```

Escribimos mensajes en la consola y los enviamos al tema my-topic.

Consumidor:

```
docker exec -it <kafka_container_id> /opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic my-topic --from-beginning
```

Este comando permite leer los mensajes enviados al tema my-topic.

Configuración e Instalación de Modelos LLM (LLaMA 3, Mistral)



Figura 16. Instalación de LLM (LLaMA 3) con Ollama para análisis avanzado de logs.

Paso 1: Instalación y Configuración con Ollama

Para instalar Ollama en el servidor o entorno local, utiliza el siguiente comando:

```
curl -fsSL https://ollama.com/install.sh | sh
```

Una vez instalada la herramienta, inicia el servidor Ollama con:

```
ollama serve
```

Esto permite que el modelo LLaMA 3 esté disponible para ejecución en el entorno local y que otras aplicaciones puedan conectarse a él.

Paso 2: Ejecución del Modelo LLaMA 3 en Local

Para ejecutar LLaMA 3 en el entorno local, asegúrate de tener el modelo descargado y configurado en Ollama. Puedes utilizar comandos específicos de Ollama para cargar y ejecutar el modelo.

```
emmacasl@AADM04104CP15:~$ ollama run llama3.1
>>> Hello! It's your friend Mario.
Wahoo! It's-a me, Luigi! Nice to see you, buddy! What's-a on your mind?
Got any adventures in the Mushroom Kingdom we should tackle? Or maybe some
Goomba-stompin' fun to be had? I'm-a all ears!
```

Figura 17. Ejecución del modelo Llama3.1 mediante ollama.

Implementación de la API REST con Flask

Paso 1: Elaboración de api con flask

Para realizar la comunicación entre el sistema de recolección de logs de Fluent Bit, el modelo de IA LLM y los servicios de OpenSearch y Slack, se implementa una API REST usando Flask. Esta API recibe los registros de logs, los envía al modelo LLM para procesamiento, y distribuye la respuesta resultante a OpenSearch y a Slack mediante un Webhook para notificaciones.

Paso 2: Estructura de la API

El siguiente código en Flask permite definir un endpoint que recibe datos en un formato JSON, los envía al modelo de IA, y luego distribuye los resultados:

Se definen dos clases de datos Item y Itemexample para la estructura de las solicitudes entrantes, que contienen información como el nombre del modelo, el prompt o la instrucción que se enviará al modelo LLM:

```
from fastapi import FastAPI
from typing import Union
from pydantic import BaseModel
import json
import requests
app = FastAPI(debug=True)

class Itemexample(BaseModel):
    name: str
    prompt: str
    instruction: str
    is_offer: Union[bool, None] = None

class Item(BaseModel):
    model: str
    prompt: str
```

Figura 18. Definición de clases para solicitudes.

Paso 3: Endpoints de la API

- Endpoint `/chat/{llms_name}`: Este es el endpoint principal que recibe una solicitud POST con datos para el modelo LLM, envía el prompt al modelo y distribuye la respuesta.

```
@app.post("/chat/{llms_name}")
def update_item(llms_name: str, item: Item):
    if llms_name == "llama3":
        url = urls[0]
        payload = {
            "model": "llama3",
            "prompt": item.prompt,
            "stream": False
        }
        response = requests.post(url, headers=headers, data=json.dumps(payload))

        if response.status_code == 200:
            model_response = response.json()
            result = model_response.get("response", "Sin respuesta")

            # Notificación a Slack
            slack_payload = {
                "text": f"Respuesta del modelo '{llms_name}': {result}"
            }
            slack_response = requests.post(slack_webhook_url, headers=headers, data=json.dumps(slack_payload))

            if slack_response.status_code == 200:
                return {"data": model_response, "llms_name": llms_name, "slack_notification": "success"}
            else:
                return {"data": model_response, "llms_name": llms_name, "slack_notification": "failed"}
        else:
            return {"item_name": item.model, "error": response.status_code, "data": response.text}

    return {"item_name": item.model, "llms_name": llms_name}
```

Figura 19. Definición de función para el consumo del API.

Explicación del Proceso

- **Recepción de Datos:** La API recibe un prompt a través del endpoint `/chat/{llms_name}`.
- **Envío al Modelo de IA:** Se envía el prompt al modelo LLM especificado (en este caso, llama3) utilizando la URL configurada.
- **Procesamiento de la Respuesta:** Se extrae la respuesta del modelo desde el JSON devuelto.
- **Notificación a Slack:** La respuesta del modelo se envía como mensaje al canal de Slack configurado.
- **Verificación de Resultados:** La API devuelve un JSON indicando si la notificación a Slack fue exitosa o fallida, junto con los datos del modelo.

```

from fastapi import FastAPI
app = FastAPI()
from typing import Union
from fastapi import FastAPI
from pydantic import BaseModel
import json
import requests
app = FastAPI(debug=True)
class Itemexample(BaseModel):
    name: str
    prompt: str
    instruction: str
    is_offer: Union[bool, None] = None

class Item(BaseModel):
    model: str
    prompt: str

urls = ["http://localhost:11434/api/generate"]

headers = {
    "Content-Type": "application/json"
}
# URL del webhook de Slack
slack_webhook_url = "https://hooks.slack.com/services/T07V78KUAEM/B07UUFJQ5QB/XXXXXX"

@app.get("/")
async def root():
    return {"message": "Hello World"}

@app.get("/hello/{name}")
async def say_hello(name: str):
    return {"message": f"Hello {name}"}

@asynccontextmanager
async def lifespan(app: FastAPI):
    print("Initializing API ...")
    # Este código se ejecuta al inicio de la aplicación
    consumer_task = asyncio.create_task(consume_logs())
    try:
        yield
    finally:
        # Este código se ejecuta al cerrar la aplicación
        consumer_task.cancel()
        try:
            await consumer_task
        except asyncio.CancelledError:
            pass

app = FastAPI(lifespan=lifespan)

try:
    consumer = Consumer(kafka_conf)
    consumer.subscribe(['test'])
    print("creo consumidor kafka")
except KafkaException as e:
    print(f"Error al crear el consumidor de Kafka: {e}")

async def consume_logs():
    print("Consumiendo logs...")
    consumer.subscribe(['test'])
    while True:
        msg = consumer.poll(timeout=1.0)
        if msg is None:
            await asyncio.sleep(1) # Esperar un segundo y volver a intentar
            continue
        if msg.error():
            print(f"Error: {msg.error()}")
            continue
        log_text = msg.value().decode('utf-8')
        print(f"Log recibido desde Kafka: {log_text}")

        # 1. Normalizar y extraer información
        log_normalizado, log_hash, metadatos = normalize_and_extract(log_text)

```

Figura 20. Código completo para implementación del API .

```
import json
import httpx
import requests

async def call_api_model_llm(model_llm, data_message):
    print(f'Inicio de petición a modelo LLM: {model_llm}')
    url = "http://localhost:11434/api/generate"
    headers = {
        "Content-Type": "application/json"
    }
    mensaje_inicial = (
        "Eres un asistente de desarrollo de software experto en análisis de logs y resolución de problemas técnicos. "
        "Se te proporcionará un log de error generado por un sistema en producción. Tu tarea es:\n"
        "1. Analizar detalladamente el log y explicar el problema de manera clara y concisa.\n"
        "2. Identificar en qué archivo y línea se generó el error para proceder con la revisión.\n"
        "3. Proporcionar posibles causas del error con base en el contenido del log.\n"
        "4. Sugerir una solución directa y práctica para resolver el problema. \n"
        "Dar la solución en español \n"
    )
    payload = {
        "model": model_llm,
        "temperature": 0.2,
        "prompt": mensaje_inicial + "\n\n" + data_message,
        "stream": False
    }
    print(f'Petición a modelo LLM: {payload}')
    try:
        complete_response = ""
        response = requests.post(url, headers=headers, data=json.dumps(payload), stream=True)
        if response.status_code == 200:
            response_text = response.text
            data = json.loads(response_text)
            actual_response = data.get("response")
            return {"message": "Petición enviada a modelo LLM", "response": actual_response}
        else:
            return {"message": "Fallo al enviar petición a modelo LLM", "error": response.text}
    except httpx.RequestError as e:
        print(f'Error al enviar petición a modelo LLM: {str(e)}')
        return {"message": "Error al enviar petición a modelo LLM", "error": str(e)}
```

Figura 21. Definición de función para el invocar al modelo LLM.

Configuración de webhook de SLACK

Para recibir notificaciones de eventos o registros de aplicaciones en un canal de Slack, es necesario configurar un Webhook de Slack. Esta integración permitirá enviar mensajes automáticamente a un canal específico, facilitando el monitoreo en tiempo real de los eventos importantes, a continuación, se detallan los pasos para configurar el Webhook de Slack

Paso 1: Acceder a la Configuración de Integraciones en Slack

- Abre Slack y ve a la sección de Aplicaciones o selecciona Administrar aplicaciones desde el menú principal.
- En la barra de búsqueda de aplicaciones, escribe "Incoming Webhooks" y selecciona esta opción cuando aparezca.

slack api

Search

Documentation Tutorials Your Apps

Your Apps Create New App

i If your app is (or will be) listed in the Slack Marketplace, please review our [Slack Marketplace Agreement](#). These terms are in addition to the existing Developer Policy, API TOS, and Brand Guidelines.

By keeping your app in the Slack Marketplace or review process, you're confirming your agreement to the Slack Marketplace Agreement and to providing additional information for security review, if requested. If you don't agree with this Agreement, please send an email to feedback@slack.com, and we'll remove your app from the Slack Marketplace or the review process.

I Agree

Filter apps by name or workspace

App Name	Workspace	App Type	Distribution Status
Opensearch Logs	EdgarMacas	Modern	Not distributed

Figura 22. Interfaz de Slack API para la creación y configuración de aplicaciones personalizadas, como OpenSearch Logs, en un espacio de trabajo.

Paso 2: Activar los Webhooks Entrantes

- Haz clic en Agregar a Slack para iniciar la configuración de los Webhooks.
- Selecciona el espacio de trabajo donde deseas configurar el Webhook y confirma haciendo clic en Permitir si se solicita.

Opensearch Logs

Incoming Webhooks

Settings

- Basic Information
- Collaborators
- Socket Mode
- Install App
- Manage Distribution

Features

- App Home
- Agents & Assistants **NEW**
- Workflow Steps **NEW**
- Org Level Apps
- Incoming Webhooks**
- Interactivity & Shortcuts

Activate Incoming Webhooks

Incoming webhooks are a simple way to post messages from external sources into Slack. They make use of normal HTTP requests with a JSON payload, which includes the message and a few other optional details. You can include [message attachments](#) to display richly-formatted messages.

Adding incoming webhooks requires a bot user. If your app doesn't have a [bot user](#), we'll add one for you.

Each time your app is installed, a new Webhook URL will be generated.

If you deactivate incoming webhooks, new Webhook URLs will not be generated when your app is installed to your team. If you'd like to remove access to existing Webhook URLs, you will need to [Revoke All OAuth Tokens](#).

Figura 23. Configuración de Incoming Webhooks en Slack para recibir mensajes desde fuentes externas, como notificaciones de logs procesados.

Paso 3: Configurar el Webhook para un Canal Específico

- Elige el canal donde deseas que se envíen las notificaciones (puedes seleccionar un canal existente o crear uno nuevo para esta integración).
- Haz clic en Permitir o Añadir Webhook para confirmar la integración.

Paso 4: Obtener la URL del Webhook

- Slack generará una URL única para el Webhook de tu canal. Copia esta URL, ya que la necesitarás para configurarla en el servicio o aplicación que enviará las notificaciones.
- Debemos mantener esta URL segura, ya que cualquier solicitud HTTP enviada a esta dirección publicará mensajes en el canal de Slack.

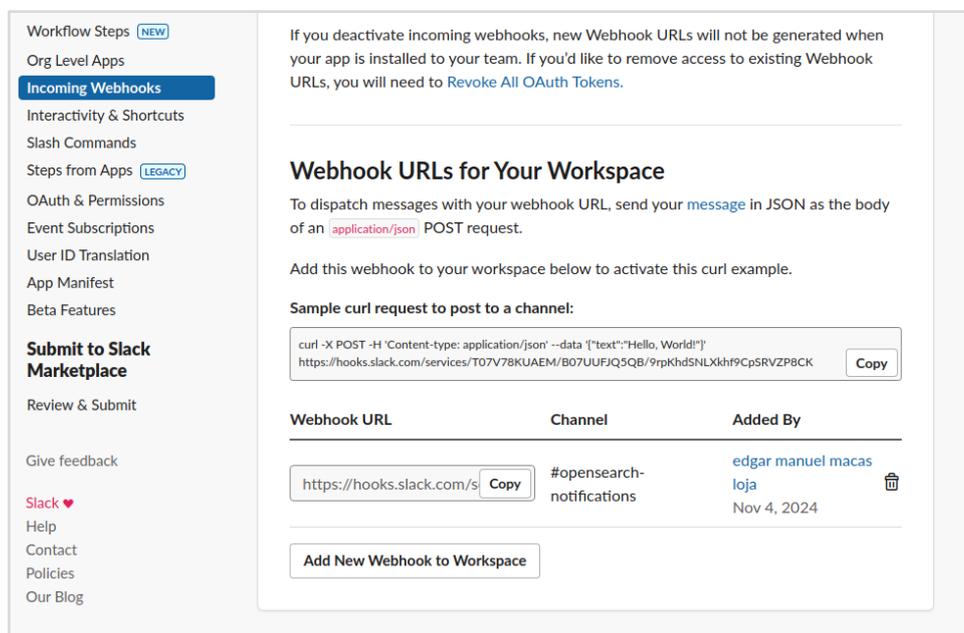


Figura 24. Configuración de URL de Webhook en Slack para enviar mensajes a un canal específico mediante solicitudes JSON.

Paso 5: Configurar el Webhook en la Aplicación o Servicio

- En la aplicación o servicio que utilizará el Webhook de Slack, busca la opción de configuración de Webhooks o notificaciones de Slack.
- Pega la URL del Webhook y guarda los cambios. Esto permitirá que el servicio envíe mensajes al canal de Slack seleccionado.

Paso 6: Probar la Configuración

- Para asegurarte de que el Webhook funciona correctamente, envía un mensaje de prueba. En la mayoría de los servicios, existe la opción de "Probar Webhook" o "Enviar mensaje de prueba".

- Verifica en Slack si el mensaje aparece en el canal configurado. Si el mensaje de prueba se recibe correctamente, la configuración está completa.

Paso 7: Prueba de Comunicación entre el Registro de Logs, el Modelo LLM, el Webhook de Slack

Para asegurar que la API desarrollada en Flask funciona correctamente y permite la comunicación entre los registros de logs, el modelo LLM y el Webhook de Slack, realizaremos una prueba desde Postman. En esta prueba, se enviará un registro de log con un error simulado, generado en una aplicación Django, y se verificará que el modelo de IA proporciona una respuesta adecuada, la cual se notifica en el canal de Slack.

- Configura la URL de la API en Flask para el endpoint /chat/llama3 (por ejemplo, http://localhost:8000/chat/llama3).
- En el cuerpo de la solicitud, selecciona el tipo de contenido JSON y proporciona un mensaje que simule un error en el formato JSON adecuado.

Ejemplo de Cuerpo de la Solicitud en JSON:



Figura 25. Solicitud POST en formato JSON enviada al endpoint /chat/llama3 para procesar un mensaje de error mediante el modelo LLaMA3.

- Enviar la Solicitud y Revisar la Respuesta
 - Haz clic en Enviar en Postman y espera la respuesta de la API.
 - La API debería procesar el log y enviar el prompt al modelo LLaMA 3 para obtener una sugerencia sobre cómo resolver el error.
 - En la respuesta JSON de la API, verifica que se incluye la sugerencia del modelo para solventar el error.
- Notificación en Slack
 - Tras recibir la respuesta del modelo, la API enviará una notificación al canal de Slack configurado, con el mensaje del error y la recomendación del modelo.

- Abre Slack y verifica en el canal correspondiente que se haya recibido la notificación. El mensaje debe contener:
 - La descripción del error detectado.
 - La sugerencia o posible solución proporcionada por el modelo.
- Captura de pantalla: Adjunta una imagen del mensaje de Slack para ilustrar cómo se muestra la notificación.

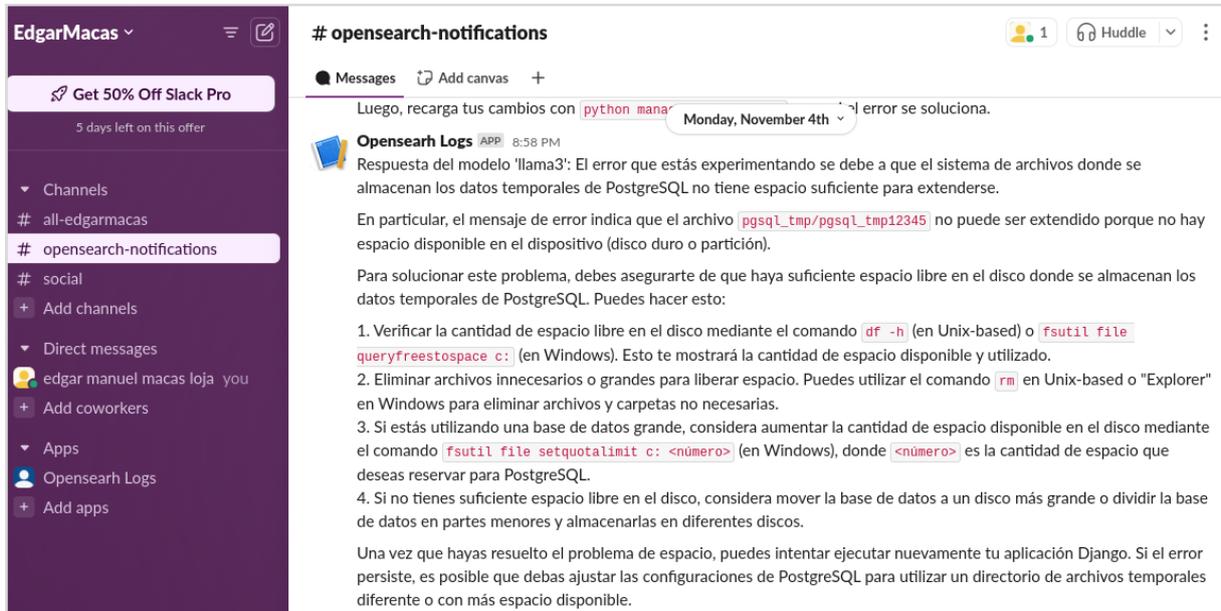


Figura 26. Notificación en Slack con solución generada por LLaMA3 para un error de PostgreSQL.

Envío de notificación al canal de SLACK:

```

import requests
import json
import httpx

async def send_to_slack(message):
    slack_webhook_url = "https://hooks.slack.com/services/XXXXXXXXXXXXXXXXX"
    headers = {
        "Content-Type": "application/json"
    }
    slack_payload = {
        "text": f"Respuesta': {message}"
    }
    try:
        async with httpx.AsyncClient() as client:
            slack_response = await client.post(slack_webhook_url, headers=headers, data=json.dumps(slack_payload))
            if slack_response.status_code == 200:
                return {"message": "Notificación enviada a Slack", "slack_response": slack_response.text}
            else:
                return {"message": "Fallo al enviar notificación a Slack", "error": slack_response.text}
    except httpx.RequestError as e:
        return {"message": "Error al enviar notificación a Slack", "error": str(e)}

```

Figura 27. Definición función para notificar a SLACK.

Repositorio con las configuraciones

[🔗 Configuraciones OpenSearch](#)

[🔗 OpenSearch LLM](#)

Nota: Los repositorios están configurados como privados. Para solicitar acceso, por favor contactar a la dirección de postgrado de la maestría en software: maestria.software@unl.edu.ec.

Anexo 3: Certificación resumen.

CERTIFICACIÓN DE TRADUCCIÓN

Loja, 05 de febrero de 2025

Lic. Viviana Valdivieso Loyola Mg. Sc.

DOCENTE DE INGLÉS

A petición verbal de la parte interesada:

CERTIFICA:

Que, desde mi legal saber y entender, como profesional en el área del idioma inglés, he procedido a realizar la traducción del resumen, correspondiente al Trabajo de Integración Curricular titulado **Arquitectura de Gestión Centralizada de Logs con OpenSearch e IA en los Sistemas Institucionales de la Universidad Nacional de Loja**, de la autoría de: **Edgar Manuel Macas**, portador de la cédula de identidad número **1104929078**

Para efectos de traducción se han considerado los lineamientos que corresponden a un nivel de inglés técnico, como amerita el caso.

Es todo cuanto puedo certificar en honor a la verdad, facultando al portador del presente documento, hacer uso del mismo, en lo que a bien tenga.

Atentamente. -



Lic. Viviana Valdivieso Loyola Mg. Sc.

1103682991

N° Registro Senescyt 4to nivel **1031-2021-2296049**

N° Registro Senescyt 3er nivel **1008-16-1454771**