UNIVERSIDAD NACIONAL AGRARIA DE LA SELVA

FACULTAD DE INGENIERÍA EN INFORMÁTICA Y SISTEMAS

ESCUELA PROFESIONAL DE CIENCIAS EN INFORMÁTICA Y SISTEMAS

"ACEPTACIÓN DE HERRAMIENTAS PROPUESTAS PARA IMPLEMENTAR DEVOPS REFORZADO EN PEQUEÑAS EMPRESAS DE DESARROLLO DE SOFTWARE."

Tesis

Para optar el título profesional de:

INGENIERO EN INFORMATICA Y SISTEMAS

PRESENTADO POR:

SILVA TRUJILLO ABEL ANTONIO

Asesor:

PANDO SOTO BRIAN CESAR

Tingo María – Perú

2026

Universidad Nacional Agraria de la Selva Facultad de Ingeniería en Informática y Sistemas	**ACTA DE SUSTENTACIÓN DE TESIS N.º 06-2025**	COMISION DE GRADOS Y TITULOS **Fecha:** 17/diciembre/2025

PARTE 1. FASE INICIAL

Siendo las 08:00 **horas del día 17 de diciembre de 2025**; en la Sala de Conferencias de la FIIS, se instala el jurado calificador conformado por:

Jurado 1: Mg. Ronald Eduardo Ibarra Zapata (presidente)

Jurado 2: Mg. José Orlando Castillo Cornelio

Jurado 3: Mg. Julio Cesar Gonzales Paico

Oficializado mediante **RESOLUCIÓN N° 217-2025-D-FIIS-UNAS** del 19 de noviembre de 2025, para el proceso de sustentación del informe final de Tesis del bachiller **ABEL ANTONIO SILVA TRUJILLO,** titulado: **"ACEPTACIÓN DE HERRAMIENTAS PROPUESTAS PARA IMPLEMENTAR DEVOPS REFORZADO EN PEQUEÑAS EMPRESAS DE DESARROLLO DE SOFTWARE".** ASESOR: **Mg. Brian Cesar Pando Soto.**

Se manifiesta que el bachiller cumple con los requisitos exigidos de Ley y se le invita a disertar su Tesis por espacio de 30 minutos, asimismo se dispondrá de igual tiempo para la absolver preguntas y sugerencias.

PARTE 2. FASE DE PREGUNTAS Y RESULTADO

Culminada la exposición se inicia la fase de preguntas por parte del jurado calificador; también se invita a los asistentes a formular preguntas sobre el tema de Tesis.

Absueltas todas las peticiones, el jurado calificador procede a deliberar en privado la calificación y resultado.

Concluida la deliberación y en presencia del público, el jurado calificador anuncia que el resultado de la Sustentación de Tesis es: APROBADO POR UNANIMIDAD.

(NOTA: consignar una de la siguientes: DESAPROBADO, APROBADO POR MAYORIA o APROBADO POR UNANIMIDAD)

Con calificativo de: EXCELENTE

(NOTA: consignar una de la siguientes: EXCELENTE, MUY BUENO, BUENO, DEFICIENTE, MUY DEFICIENTE)

Por lo que se comunicará a las instancias correspondientes para el trámite respectivo.

PARTE 3. CONFORMIDAD

De todo lo mencionado se firma al pie en señal de conformidad, siendo las 09:15. horas se da por finalizada la ceremonia de Sustentación de Tesis.

Firma:	Firma:	Firma:
Jurado 1: Ronald Eduardo Ibarra Zapata	**Jurado 2:** José Orlando Castillo Cornelio	**Jurado 3:** Julio Cesar Gonzales Paico
Firma:	Firma:	
Sustentante: Abel Antonio Silva Trujillo	**Asesor:** Brian Cesar Pando Soto	

"Decenio de la Igualdad de Oportunidades para Mujeres y Hombres"
"Año de la Esperanza y el Fortalecimiento de la Democracia"

CERTIFICADO DE SIMILITUD T.I. N 045 - 2026 - CS-RIDUNAS

El Jefe de la Unidad de Soporte Científico de la Universidad Nacional Agraria de la Selva, quien suscribe,

CERTIFICA QUE:

El Trabajo de Investigación; aprobó el proceso de revisión a través del software TURNITIN, evidenciándose en el informe de originalidad un índice de similitud no mayor del 25% y contenido generado por Inteligencia Artificial menor o igual al 20%. Según establece el Art. 29° y 30° del Acuerdo Nro.017-2025-CIUNAS-VRI-UNAS.

Programa de Estudio:

Ingeniería en Informática y Sistemas

Tipo de documento:

Tesis	X	Trabajo de Suficiencia Profesional	

TÍTULO	AUTOR	PORCENTAJE	
		SIMILITUD	CONTENIDO GENERADO POR INTELIGENCIA ARTIFICIAL
ACEPTACIÓN DE HERRAMIENTAS PROPUESTAS PARA IMPLEMENTAR DEVOPS REFORZADO EN PEQUEÑAS EMPRESAS DE DESARROLLO DE SOFTWARE	SILVA TRUJILLO ABEL ANTONIO	05 % Cinco	Menor a 20 %

Tingo María, 13 de febrero de 2026.

UNIVERSIDAD NACIONAL AGRARIA DE LA SELVA
UNIDAD DE SOPORTE CIENTIFICO

ING. EINSTEIN A. ORTIZ MORALES
JEFE

C.C. Archivo

UNIVERSIDAD NACIONAL AGRARIA DE LA SELVA
FACULTAD DE INGENIÉRIA EN INFORMÁTICA Y SISTEMAS
ESCUELA PROFESIONAL DE INGENIERÍA EN INFORMATICA Y SISMTEMAS

ACEPTACIÓN DE HERRAMIENTAS PROPUESTAS PARA IMPLEMENTAR DEVOPS REFORZADO EN PEQUEÑAS EMPRESAS DE DESARROLLO DE SOFTWARE.

Autor : Bach Silva Trujillo Abel Antonio.

Asesor(es) : Mg. Pando Soto Brian Cesar. (UNAS).

Área de investigación : Ingeniería de Software y Comunicación.

Línea de investigación : Ingeniería de Software

Eje Temático de investigación : Procesos de Software

Lugar de ejecución : Lead Working Partner (LWP)

Duración : Ocho meses

Financiamiento : Autofinanciado

Monto : S/. 8610.00

Tingo María – Perú . 2025

DEDICATORIA

A MIS PADRES, Abel Silva y Lupe Trujillo, y a **MI ABUELA**, **Camila Morales**, por ser parte de mi vida y por su papel fundamental en mi desarrollo profesional. A **MI HERMANA Maite Silva**, por tanto amor y apoyo incondicional. Su impacto y enseñanzas han moldeado mi desarrollo.

AGRADECIMIENTOS

A Dios, por haberme dado sabiduría, salud, voluntad y por siempre enseñarme el camino correcto.

A mi padre, **Abel Silva Ríos**, y a mi madre, **Lupe Liliana Trujillo Morales**, por su constante apoyo y sacrificios para permitirme mis estudios.

A mi abuela, **Camila Morales Peres**, gracias infinitas por siempre apoyarme y luchar por mí.

Gracias a mi asesor, **Mg. Brian Pando Soto**, por haberme orientado en mi investigación y por enriquecer al mundo de DevOps.

Agradezco a mi otro asesor, el **Mg. Abraham Eliseo Dávila Ramón**, por adentrarme al mundo de la investigación y el desarrollo operativo.

Agradezco al **Mg Ronal Ibarra Zapata** por su guía y conocimientos en mi vida académica.

A todos ellos, muchas gracias por ser parte fundamental para la culminación de esta tesis.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

En los últimos años, las empresas pequeñas desarrolladoras de software se han interesado por adoptar metodologías ágiles y DevOps que mejoren la comunicación entre equipos de desarrollo y operaciones. Pero su aplicación enfrenta barreras por falta de estandarización y recursos técnicos. En este contexto, surge el enfoque DevOps Reforzado, que combina los principios de DevOps con la norma ISO/IEC 29110, orientada a organizaciones con menos de 25 integrantes. La presente investigación tiene como objetivo determinar el grado de aceptación tecnológica de las herramientas propuestas para implementar DevOps Reforzado, tomando como caso de estudio la implementación del Stack ELK (Elasticsearch, Logstash y Kibana), utilizada en la etapa de monitoreo. Para dicha implementación se utilizó la metodología "Action - Research", esta metodología tuvo tres ciclos iterativos. Análogamente se aplicó el Modelo de Aceptación Tecnológica (TAM), que evalúa la utilidad percibida, la facilidad de uso percibida y la intención de uso de la herramienta implementada. La investigación es de tipo aplicada, con diseño no experimental y corte transversal, utilizando un cuestionario TAM con escala Likert de siete puntos aplicado a profesionales de desarrollo y operaciones. Los resultados evidencian una alta percepción de utilidad e intención de uso, aunque con cierta variabilidad en la facilidad de uso. De forma complementaria, se observó un incremento en el nivel de madurez DevOps en los resultados de la metodología "Action - Research" tras la implementación. Se concluye que el modelo TAM es adecuado para evaluar la aceptación tecnológica de soluciones en entornos DevOps Reforzados y que la adopción del stack ELK contribuye positivamente a la observabilidad y eficiencia operativa en pequeñas empresas de software.

Palabras clave: DevOps Reforzado, Modelo TAM, Aceptación Tecnológica, ELK, Observabilidad, ISO/IEC 29110, "Action - Research".

ABSTRACT

In recent years, small software development companies have shown growing interest in adopting agile methodologies and DevOps practices to improve communication between development and operations teams. However, their implementation faces barriers due to a lack of standardization and technical resources.In this context, the Reinforced DevOps approach emerges, combining DevOps principles with the ISO/IEC 29110 standard, which is oriented toward organizations with fewer than 25 members.The objective of this research is to determine the degree of technological acceptance of the tools proposed to implement Reinforced DevOps, using as a case study the implementation of the ELK Stack (Elasticsearch, Logstash, and Kibana), applied during the monitoring stage. For this implementation, the Action-Research methodology was used, consisting of three iterative cycles. Additionally, the Technology Acceptance Model (TAM) was applied to evaluate the perceived usefulness, perceived ease of use, and intention to use the implemented tool.This research is applied in nature, with a non-experimental and cross-sectional design, using a seven-point Likert TAM questionnaire administered to development and operations professionals. The results show a high perception of usefulness and intention to use, although with some variability in ease of use. Complementarily, an increase in DevOps maturity was observed based on the Action-Research results after the implementation.It is concluded that the TAM model is suitable for evaluating technological acceptance in Reinforced DevOps environments and that the adoption of the ELK Stack contributes positively to observability and operational efficiency in small software development companies.

Keywords: Reinforced DevOps, TAM Model, Technology Acceptance, ELK, Observability, ISO/IEC 29110, “Action - Research”.

I. INTRODUCCIÓN

El desarrollo de software en pequeñas empresas enfrenta un entorno donde la demanda de actualizaciones rápidas, servicios estables y procesos estandarizados supera con frecuencia su capacidad operativa. En estos contextos suelen encontrarse problemas como poca automatización, escaso control de versiones, baja trazabilidad, ausencia de monitoreo centralizado y procesos informales que dificultan la entrega continua (Karunarathne et al., 2024). Esta situación resulta en retrasos, reprocesos y decisiones intuitivas en vez de evidencia técnica, impactando directamente la calidad del software y la eficiencia del equipo.

Como solución a estas restricciones, DevOps se ha alzado como una forma de integrar desarrollo y operaciones a través de la automatización, la colaboración y la retroalimentación continua. Pero su implementación en pequeñas empresas falla por falta de guías, roles definidos y procesos mínimos de soporte. Ante esta situación, la norma ISO/IEC 29110, para organizaciones de hasta 25 personas, se ha erigido como un estándar para regularizar procesos, funciones y artefactos. La integración de ambos enfoques ha creado el DevOps Reforzado, en el cual DevOps proporciona las prácticas técnicas y la ISO/IEC 29110 el marco de procesos.

En la literatura se informan avances, pero también lagunas evidentes. Por ejemplo, Muñoz demuestra que una plataforma apoyada en el perfil base ISO/IEC 29110 puede fortalecer DevOps y mejorar la comprensión del proceso, descubriendo brechas de madurez y tareas cruciales para pequeñas empresas (Muñoz et al., 2021) . García ilustra que DevOps automatiza hasta en un 78% los productos de trabajo ISO/IEC 29110, aumentando la productividad y reduciendo los tiempos de entrega, aunque se enfoca en la automatización y no en la aceptación de herramientas (García et al., 2023). Otros estudios sobre DevOps Reforzado se centran en procesos, roles, métricas, madurez y automatización, pero no evalúan la aceptación tecnológica de las herramientas específicas, particularmente en actividades críticas como monitoreo y observabilidad (Muñoz & Rodríguez, 2021).

A pesar de contar con algunos procesos estandarizados y con prácticas mínimas de DevOps Reforzado, la empresa estudiada presenta una brecha evidente, no dispone de una herramienta funcional, ágil y eficiente para la fase de monitoreo. Se utilizaba Amazon CloudWatch, pero su uso es limitado, no es flexible para el análisis detallado de registros y no facilita una respuesta rápida ante incidentes. Para cerrar esta brecha, se implementó el Stack ELK como solución de

monitoreo centralizado, después de hacer una comparación con otras herramientas como Grafana y Prometheus, se eligió el Stack ELK por la factibilidad para la empresa, con el objetivo de mejorar la observabilidad del sistema y acelerar la detección y respuesta ante fallas. Sin embargo, antes de consolidar su adopción, es necesario verificar si esta herramienta realmente resulta práctica, útil y aceptada por el equipo técnico. Por ello, se aplicó el Modelo de Aceptación Tecnológica (TAM) para evaluar la utilidad percibida, la facilidad de uso y la intención de uso de la Stack ELK dentro del proceso de DevOps Reforzado.

La adopción de una herramienta tecnológica no depende únicamente de su capacidad técnica, sino de su aceptación por parte de quienes la operarán diariamente. Evaluar esta aceptación es util para evitar implementaciones fallidas, garantizar sostenibilidad y asegurar que la herramienta realmente contribuya a la mejora del proceso DevOps Reforzado. El Modelo de Aceptación Tecnológica (TAM) permite analizar la utilidad percibida, la facilidad de uso y la intención de uso, ofreciendo una base empírica sólida para determinar si el Stack ELK es viable dentro de la organización

Esta investigación se desarrolla en una pequeña empresa de software ubicada en Huánuco, Perú, bajo un enfoque aplicado y descriptivo, con diseño no experimental. Se evalúa exclusivamente la aceptación tecnológica de el Stack ELK dentro de la fase de monitoreo del DevOps Reforzado, sin abarcar otras fases del proceso, dado el ajuste de alcance basado en factibilidad y relevancia práctica. Los resultados obtenidos permiten comprender el nivel de aceptación de la herramienta y aportan evidencia útil para futuras decisiones de estandarización, automatización y mejora continua.

En la seccion 2 se presenta la literatura que esta confirmado por antecedentes y el marco teorico. En la seccion 3 se presentan materiales y metodos de igual manera, el tipo de investigacion. En la seccion 4 se presenta la ejecución bajo la metodología action – research, luego en la seecion 5 tenemos los resultados obtenidos, que posteriomente se discuten en la seccion 6, y en la seecion 7 tenemos las concluciones de la implementacion.

1.1. Problema General.

¿Cúal es el grado de aceptación de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas?

1.2. Problema Específico.

- ¿Cúal el grado de Utilidad Percibida de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas ?
- ¿Cúal el grado de Intención de uso de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas ?
- ¿Cúal el grado de Facilidad de uso de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas ?

1.3. Objetivo General.

Determinar el grado de aceptación de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas.

1.4. Objetivo Específico.

- Determinar el grado de Utilidad Percibida de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas.
- Determinar el grado de Intención de uso de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas.
- Determinar el grado de Facilidad de uso de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas.

1.5. Hipotesis General

La aceptación tecnológica de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas es alta

1.6. Hipotesis Especifico

- La Utilidad percibida de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas es alta
- La Intención de uso de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas es alta
- La Facilidad de uso de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas es alta

II. REVISIÓN DE LITERATURA.

2.1. Antecedentes de la Investigación.

A continuación se muestra de manera ordenada los trabajos relacionados con la investigacion , categorizados por temas en comun y se finaliza con un debate entre las investigaciones relacionadas.

2.1.1. Implementación de la ISO/IEC 29110: Metodologías, desafíos, resultados, y lecciones aprendidas.

- **Metodología de Implementación y Desafíos.**

En la literatura se observa que la implementación de la ISO/IEC 29110 en entidades muy pequeñas se ha abordado mediante distintos enfoques metodológicos, que coinciden en enfrentar desafíos recurrentes en los procesos de Gestión de Proyectos e Implementación de Software. En un estudio de mejora de procesos en una pequeña empresa peruana, Abarca utilizó la metodología de Action - Research, lo que permitió diagnosticar la situación inicial, aplicar cambios iterativos en procesos y artefactos, y recoger lecciones aprendidas directamente en el contexto real de trabajo (Abarca et al., 2015). Por su parte, Minero evaluaron la implementación del estándar en un centro de desarrollo de software universitario a través de encuestas a estudiantes y docentes, centrándose en cómo los procesos estandarizados apoyan la culminación de proyectos y la experiencia de seguir dichos procesos (Minero et al., 2020). En un contexto industrial, Muñoz describe un método de cinco pasos para implementar el perfil básico de la ISO/IEC 29110 en cuatro VSEs que ya trabajaban con Scrum, iniciando con la identificación de brechas en Gestión de Proyectos e Implementación de Software, el mapeo de los procesos reales al estándar y la adaptación de las prácticas a la forma de trabajo existente de cada organización (Muñoz et al., 2020). Complementariamente, Muñoz y Montoya realizaron una comparación sistemática entre el estado del arte y el estado de la práctica para identificar los problemas más frecuentes durante la implementación, encontrando dificultades asociadas a productos de trabajo críticos, así como a la comprensión y ejecución consistente de las prácticas del estándar (Munoz & Montoya-Mendez, 2021). Más recientemente, Mejía analizaron la estructura y trazabilidad de los productos de trabajo del perfil básico y, mediante lógica matemática, propusieron tautologías para explicitar los estados de dichos productos, abordando directamente problemas de falta de control de versiones,

documentación y desarrollo de pruebas señalados en implementaciones previas (Terron & Mejía, 2023). Estos estudios coinciden en que, más allá del contexto (industrial o académico), los principales desafíos de la ISO/IEC 29110 se concentran en la correcta comprensión del estándar, la gestión de evidencias y versiones de los productos de trabajo y la integración de sus prácticas en procesos ya existentes.

- **Resultados y Beneficios de la Implementación.**

A pesar de los desafíos mencionados, los antecedentes reportan beneficios consistentes tras la adopción del estándar. Abarca hizo la implementación en una empresa y permitió alinear sus procesos de desarrollo con prácticas internacionales, facilitando la mejora de la calidad del software y apoyándose en la experiencia previa con MoProSoft para reducir el esfuerzo de adopción; entre los beneficios observados destacan el incremento de la competitividad, la facilidad de migración desde otros modelos y el mantenimiento de las prácticas incluso ante la rotación de personal (Abarca et al., 2015). En el entorno académico, Minero señalan que la estandarización de actividades mediante ISO/IEC 29110 facilitó la conclusión exitosa de proyectos y fue percibida como útil por estudiantes y docentes, aunque persistieron retos en actividades específicas como arquitectura y diseño detallado (Minero et al., 2020). En el caso de las cuatro VSEs mexicanas que ya utilizaban Scrum, Muñoz indica que la implementación del estándar fue bien aceptada y relativamente sencilla, reforzando los procesos de gestión de proyectos y desarrollo sin cambiar la forma de trabajo, y resolviendo problemas relacionados con la ausencia de evidencias, el control de cambios, el diseño de software y la documentación de pruebas (Muñoz et al., 2020). Adicionalmente, la identificación de productos de trabajo con mayor número de problemas permitió proponer herramientas y técnicas específicas para apoyar su gestión en las VSEs, mientras que las mejoras basadas en lógica matemática contribuyen a clarificar los estados de los productos y fortalecer el control de versiones y la trazabilidad (Terron & Mejía, 2023) . En conjunto, estas investigaciones muestran que la ISO/IEC 29110 no solo refuerza la madurez de los procesos de Gestión de Proyectos e Implementación de Software en organizaciones pequeñas, sino que también genera evidencia de mejora en calidad, productividad y capacidad de seguimiento, lo que respalda su uso como marco de referencia para integrar y reforzar otros enfoques, como DevOps y DevOps Reforzado, en contextos de muy pequeñas organizaciones.

2.1.2. Estado del Arte de DevOps: metodologías de investigación, enfoque de implementación, y resultados reportados.

- **Metodologías de investigación sobre DevOps.**

Los estudios empíricos sobre DevOps han utilizado principalmente enfoques cualitativos, estudios de caso múltiples y revisiones sistemáticas de literatura.

Macarthy desarrolla una taxonomía empírica de DevOps a partir de entrevistas con profesionales y aplican Teoría Fundamentada para identificar variantes de implementación, analizando cómo se combinan prácticas, roles y herramientas en contextos reales (Macarthy & Bass, 2020) . De forma complementaria, Lwakatare realiza un estudio de casos múltiples en cinco empresas pequeñas y medianas, describiendo en detalle cómo se organiza el pipeline de despliegue, qué cambios se requieren en estructura organizacional y qué responsabilidades asume el equipo de desarrollo cuando también se hace cargo de producción (Lwakatare et al., 2019). Por otro lado, Paez propone una estrategia de versionado específica para contextos DevOps, basada en la identificación de artefactos, elección de herramientas, convenciones de versionado y trazabilidad entre versiones, validándola en tres proyectos reales de diferentes tipos de organización (Paez, 2018). Finalmente, Hamza desarrollan una revisión sistemática de 33 estudios sobre lineamientos, beneficios y desafíos de adopción de DevOps, construyendo un marco de referencia que integra conceptos, prácticas, principios y directrices de adopción a partir de evidencia distribuida en múltiples plataformas y dominios (Hamza et al., 2024). Estos ejemplos muestran que la investigación DevOps usa metodologías robustas (entrevistas, casos de estudio, revisión de literatura) para entender el "cómo" se hace y el "por qué" funciona o no en la práctica.

- **Enfoques de implementación y prácticas clave de DevOps.**

Como implementación, la literatura concuerda en que DevOps no es una herramienta, sino un conjunto de prácticas que deben engranarse a lo largo del ciclo de vida del software. Lwakatare señala que la implementación exitosa en PYMEs significa que el equipo de desarrollo se hace cargo de poner en producción, respaldado por una cadena de herramientas que automatizan la compilación, las pruebas, el despliegue, el aprovisionamiento de

infraestructura y la monitorizació (Lwakatare et al., 2019). En esa misma línea, se plantea un punto esencial de implementación: la administración de artefactos en un ambiente donde coexisten código de aplicación, infraestructura como código, scripts de implementación, configuraciones y binarios (Paez, 2018). Su propuesta de estrategia de versionado especifica qué artefactos deben controlarse por versiones, qué herramientas utilizar (por ejemplo, Git para código, repositorios de artefactos para binarios) y cómo mantener la trazabilidad entre las versiones que conforman un despliegue específico para permitir auditoría, rollback y consistencia entre entornos.

Felipe propone una vista de ciclo completo donde DevOps se estructura en las etapas iterativas de planificación, codificación, compilación, prueba, implementación, operación y monitoreo, siendo un ciclo constante donde la retroalimentación de producción impulsa la siguiente planificación. (Felipe Redondo & Núñez Cárdenas, 2022). La revisión sistemática de Hamza sintetiza estas prácticas en un modelo de "loop" continuo que integra planificación, desarrollo, pruebas, despliegue, liberación y monitorización, acompañado de principios CAMS (cultura, automatización, medición y compartición) (Hamza et al., 2024). Paralelamente, los estudios de casos de Lwakatare evidencian que la adopción de estas prácticas exige cambios organizacionales: redefinición de roles, mayor autonomía técnica del equipo, y una curva de aprendizaje pronunciada tanto para desarrolladores como para personal de operaciones (Lwakatare et al., 2019).

- **Resultados, beneficios y desafíos en la adopción de DevOps.**

En términos de resultados, los trabajos revisados coinciden en que las implementaciones maduras de DevOps mejoran tanto la velocidad como la estabilidad de la entrega. Alamin combina estudio de casos y simulaciones controladas, reportan que un entorno con DevOps puede incrementar la frecuencia de despliegue en un 45%, reducir el lead time de cambios en un 38 % y disminuir la tasa de fallos de cambio en un 32%, además de mejorar el tiempo medio de recuperación (Alamin et al., 2025). Estos resultados empíricos refuerzan la percepción de que DevOps no solo acelera las entregas, sino que también aporta mayor estabilidad operativa. Los estudios de caso analizados por Lwakatare muestran beneficios similares: ciclos de liberación más cortos, menos errores de despliegue y una colaboración más fluida entre desarrollo y operaciones cuando el equipo asume propiedad de extremo a extremo del pipeline (Lwakatare et al., 2019). En la misma línea, la estrategia de versionado de Paez se

percibe como una mejora frente a enfoques previos, especialmente en escenarios de resolución de incidentes y necesidad de revertir versiones de forma rápida y controlada (Paez, 2018).

Sin embargo, estos beneficios vienen acompañados de desafíos recurrentes. El estudio de Alamin identifica problemas persistentes para integrar seguridad (DevSecOps), gestionar deuda técnica y alinear la cultura entre equipos aún organizados en silos (Alamin et al., 2025). Hamza desde la perspectiva de la revisión sistemática, resaltan la ausencia de lineamientos estándar de adopción, la falta de claridad sobre las capacidades mínimas de procesos, prácticas y herramientas, el costo de la automatización y la escasez de personal con habilidades adecuadas como obstáculos frecuentes en organizaciones que intentan migrar hacia DevOps (Hamza et al., 2024).

2.1.3. Automatización y Herramientas Digitales para la Optimización de Procesos en DevOps Reforzado.

- **Fortalecimiento del Proceso DevOps con ISO/IEC 29110.**

Negrete proponen una guía para definir un proceso genérico de DevOps apoyado en prácticas del perfil base de ISO/IEC 29110, haciendo énfasis en la automatización de herramientas y procesos y esperando obtener beneficios como el control del proyecto y la mejora de la comunicación (Muñoz & Negrete, 2020). Este modelo pretende guiar la implementación de DevOps, pero haciendo hincapié en la adaptación a la organización. En la misma línea, Muñoz proponen una guía y una plataforma web para implementar y mejorar DevOps, a través del diagnóstico y adaptación de DevOps reforzado a los procesos de la organización (Muñoz et al., 2021). Esta investigación demuestra la capacidad de la plataforma web para disminuir el tiempo de entendimiento y adaptación del proceso DevOps, mediante un caso práctico en una pequeña empresa. La principal diferencia entre estos dos estudios es el enfoque metodológico; mientras que Negrete se enfocan en crear una guía paso a paso para implementar (Muñoz & Negrete, 2020), Muñoz dan un paso más y ofrecen una herramienta (plataforma web) para facilitar la aplicación de esta guía en las organizaciones. (Muñoz et al., 2021).

- **Automatización y Adaptación de Procesos con DevOps.**

García tratan la forma en que la implementación de DevOps automatiza procesos basados en el estándar ISO/IEC 29110, en su perfil básico, en un centro de desarrollo de software (García et al., 2023). En este caso se muestra cómo utilizar herramientas de Azure DevOps para automatizar flujos de trabajo, mejorar la productividad y disminuir el tiempo de entrega de software. Rodríguez por otro lado, se enfocan en la implementación de prácticas DevOps en una organización pequeña, identificando desafíos y proporcionando orientación para superarlos (Muñoz & Rodríguez, 2021). En este caso se usa un enfoque metodológico que involucra análisis comparativos y métricas específicas, dando como resultado una implementación de DevOps personalizada para la organización. La principal diferencia entre estos estudios es el área de aplicación: García se enfocan en la automatización de procesos de software con DevOps (García et al., 2023), mientras que Rodríguez analizan una implementación más completa de DevOps, abarcando procesos técnicos, culturales y organizacionales (Muñoz & Rodríguez, 2021).

- **Guías electrónicas y herramientas para la implementación.**

Acevedo crea una guía electrónica para la implementación del proceso DevOps, apoyándose en las tareas del perfil base de la norma ISO/IEC 29110, para micro y pequeñas empresas. (Acevedo et al., 2025). Esta guía organiza las etapas de planificación, codificación, compilación, pruebas, liberación, despliegue, operación y monitoreo, con las herramientas tecnológicas para cada etapa, automatizando, colaborando interdisciplinariamente y mejorando continuamente. Su beneficio es proporcionar un marco estandarizado y adaptable a empresas con pocos recursos, fortaleciendo los conceptos de DevOps con la estructura de la ISO/IEC 29110. Esta investigación es un aporte a los estudios de Negrete y Muñoz, creando una guía electrónica que materializa el DevOps Reforzado en la web, permitiendo la adopción de prácticas DevOps estandarizadas (Muñoz & Negrete, 2020), (Muñoz et al., 2021).

2.2. Marco Teórico.

A continuación, se presentan los constructos teóricos que enmarcan la importancia de la investigación.

2.2.1. DevOps.

DevOps integra desarrollo (Dev) y operaciones (Ops) para alinear personas, procesos y tecnología en la planificación, desarrollo, entrega y operación de aplicaciones. DevOps involucra la alineación y colaboración entre roles que antes estaban aislados, como desarrollo, operaciones de TI, control de calidad y seguridad. (Microsoft Learn, 2025).

Los equipos aprovechan la cultura, las prácticas y las herramientas de DevOps para tener más confianza en las aplicaciones que desarrollan, satisfacer mejor las necesidades de los clientes y alcanzar los objetivos de negocio más rápido. DevOps permite a los equipos entregar valor continuamente a los clientes, generando mejores productos, más confiables (Microsoft Learn, 2025).

2.2.1.1. Ciclo DevOps.

Es relevante destacar que algunos expertos combinan algunas de estas fases, resumiéndolas en 6 o 5 fases. Sin embargo existen investigaciones (Felipe Redondo & Núñez Cárdenas, 2022) donde se describe de manera completa desde su concepción inicial. DevOps es, entre muchas interpretaciones, un ciclo interminable compuesto por las fases que lo conforman. La metodología comienza en el equipo de Desarrollo (Dev), encargado de la Planificación (Plan), la Codificación (Code), la Compilación (Build) (algunos simplifican ambos procesos como Desarrollo) y las Pruebas (Test), para luego dar paso al equipo de Operaciones (Ops) que inicia con la Liberación (Release), el Despliegue (Deploy), el Funcionamiento (Operate) y el Monitoreo (Monitor). Finalmente, se regresa a la etapa de Planeación para reiniciar el ciclo (Felipe Redondo & Núñez Cárdenas, 2022).

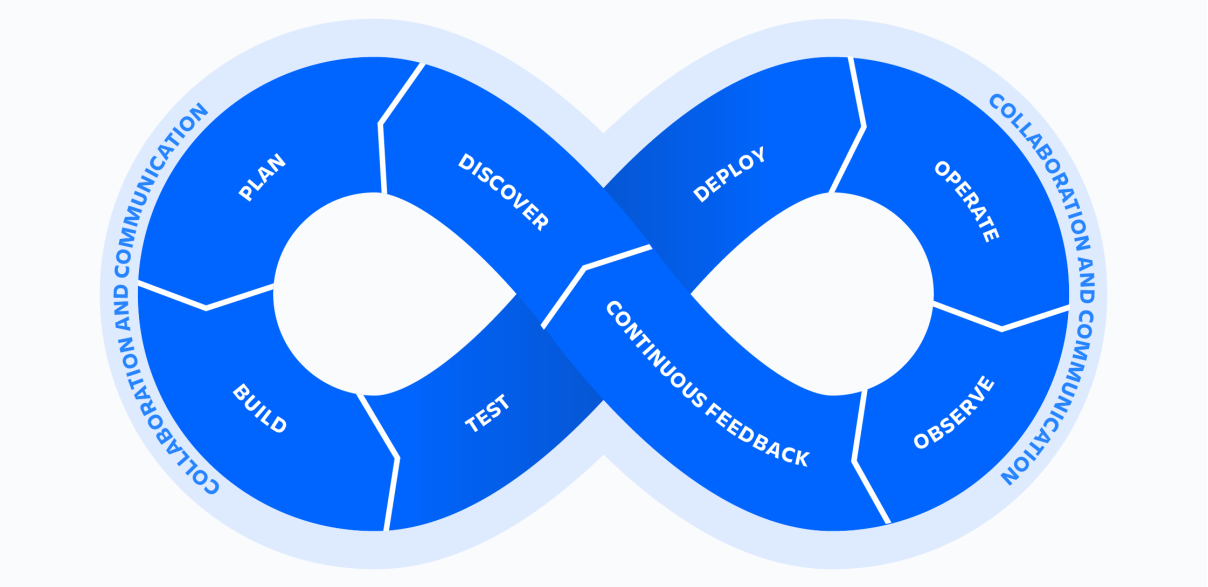


Figura 1. Ciclo DevOps
Fuente: (TRBL-SERVICES, 2021)

Cada fase se define :(Felipe Redondo & Núñez Cárdenas, 2022).

La **planificación** implica entender las necesidades de los clientes y establecer una comunicación constante. Su resultado es una hoja de ruta para el futuro. Aquí se definen las funcionalidades de mayor importancia, los riesgos y restricciones del proyecto, así como los criterios de aceptación que se utilizarán para verificar el producto. Además, se establecen hitos y fechas límites preliminares, se asignan responsables y se alinean las expectativas entre las áreas técnicas y de negocio. Todo ello hace que el equipo tenga una misma visión del producto y un camino definido para las siguientes fases del ciclo DevOps.

La **codificación**, implica desarrollar el software siguiendo un estilo de código uniforme y evitando errores comunes de seguridad y patrones de código no recomendados. En esta etapa se ponen en práctica las buenas prácticas de programación, control de versiones y revisión de código entre pares, lo que permite identificar errores de manera temprana y garantizar la calidad del producto. Además, se añaden pruebas unitarias y documentación básica del código para que los cambios sean comprensibles, reutilizables y ampliables en el futuro sin romper nada.

La **compilación** hace uso de DevOps, donde los desarrolladores envían su código a un repositorio común y se revisa de manera eficiente para encontrar problemas a

tiempo. También se ejecutan pruebas automatizadas para regresión e integración. Aquí es donde suelen entrar en juego herramientas de CI/CD que automáticamente compilan cada cambio al repositorio y dan retroalimentación inmediata sobre si rompe algo. También se lleva a cabo el empaquetado del software y la verificación de dependencias para garantizar que los artefactos resultantes sean reproducibles y puedan promoverse a los siguientes entornos.

Las **pruebas** abarcan tanto pruebas manuales como automatizadas posteriores a la compilación. Esto permite descubrir errores antes de la puesta en producción y medir el rendimiento, la seguridad y el cumplimiento de las mejores prácticas. En esta etapa se realizan diferentes tipos de pruebas, como pruebas funcionales, de regresión, de carga, de seguridad, entre otras, para validar el comportamiento del sistema en diferentes situaciones. Los resultados se registran y se priorizan las fallas encontradas para que el equipo de desarrollo pueda solucionarlas en ciclos cortos y tener siempre el control de la calidad.

La **liberación** es la etapa donde la compilación está lista para desplegarse en producción. Asegurarse de que los cambios de código estén bien probados y puedan desplegarse automáticamente (si el DevOps está maduro). En esta etapa se negocian las aprobaciones, se gestionan las versiones del software y se planifican las ventanas de cambio para minimizar la interrupción a los usuarios. Además, se actualiza la documentación técnica y operativa correspondiente a la nueva versión, facilitando la trazabilidad de los cambios y permitiendo que el despliegue a producción se realice de manera controlada y predecible.

El **despliegue** implica lanzar la compilación en producción, y se puede automatizar para tener un proceso continuo. Si se detecta algún problema, se pueden redirigir las solicitudes al entorno anterior mientras se solucionan. Además, el despliegue implica también toda la capilaridad de la bajada comercial, coordinando a las distintas áreas de la organización para que los cambios lleguen efectivamente al usuario final. En esta fase se ponen en marcha procesos de comunicación, soporte y seguimiento, lo que refleja que DevOps no solo se centra en herramientas tecnológicas, sino que involucra de manera integral los procesos de negocio asociados al producto.

El **funcionamiento** se refiere al momento en que la nueva versión del producto está en uso y el equipo de operaciones trabaja para garantizar un funcionamiento sin problemas. La retroalimentación del cliente es fundamental para mejorar el producto en el

futuro. En esta fase se gestionan la operación diaria del servicio, la atención de incidencias y la supervisión del cumplimiento de los acuerdos de nivel de servicio. Además, se coordinan acciones con el equipo de desarrollo para resolver problemas recurrentes, optimizar el rendimiento y ajustar funcionalidades según las necesidades del negocio, de manera que el sistema se mantenga estable y alineado con las expectativas de los usuarios.

El **monitoreo** implica recopilar datos y analizar el comportamiento, rendimiento y errores de los clientes en la aplicación. Esta información se utiliza para cerrar el ciclo del proceso y mejorar continuamente el producto. Además, el monitoreo involucra la identificación de defectos, vacíos funcionales y deuda técnica que puedan afectar la calidad del servicio; estos hallazgos, junto con los nuevos requerimientos de los usuarios, se priorizan y alimentan la planificación de un nuevo ciclo DevOps.

2.2.2. Principales practicas del DevOps:

Existe un gran cantidad de practicas que apoyan a la implementación de DevOps, una de las mas mencionadas son las siguiente:

2.2.2.1. Integración Continua.

La integración continua (CI) consiste en automatizar la unión de los cambios de código que realizan varios desarrolladores dentro de un mismo proyecto. Es una práctica fundamental en DevOps, ya que permite que las modificaciones se integren con frecuencia en un repositorio central, donde inmediatamente se ejecutan compilaciones y pruebas. Con este proceso, las herramientas automatizadas verifican que el nuevo código no cause problemas antes de integrarse al código principal. El corazón de la CI es un control de versiones, que lleva registro de cada modificación del código fuente. A ello añaden pruebas automatizadas, análisis de calidad, comprobaciones de estilo y demás controles para garantizar que el código permanezca limpio, estable y listo para integrar (Atlassian, 2021).

2.2.2.2. Entrega Continua.

La entrega Continua automatiza la preparación de cambios de código para su despliegue en producción, haciendo más eficiente y rápida la entrega de software.

Permite a los desarrolladores realizar pruebas más allá de las unidades, incluidas las de interfaz de usuario, carga, integración y fiabilidad de la API. Además, automatiza la creación y reproducción de entornos de pruebas en la nube, permitiendo una verificación completa de las actualizaciones (Amazon, 2016).

Esta práctica consiste en que los cambios se apliquen automáticamente en un entorno de prueba o producción una vez compilados, sin intervención humana. Es diferente a la entrega continua en que requiere aprobación humana antes de desplegar a producción. Entre las ventajas se encuentran la automatización del proceso de publicación, la optimización de la productividad, la identificación y corrección temprana de errores y entregas más rápidas y frecuentes (Amazon, 2016).

También puede decirse que la Entrega Continua es una extensión de la integración continua, automatizando todos los cambios de código en un entorno de prueba y/o producción tras la compilación. Se hace énfasis en tener una buena base de CI/CD, automatización de despliegues, marcas de características y obtener beneficios como eliminar la complejidad de despliegue de software y permitir lanzamientos más frecuentes (Atlassian, 2023).

Para finalizar, es importante mencionar que la Entrega Continua permite a los equipos de desarrollo automatizar el flujo de software a través del ciclo de vida del desarrollo, obteniendo beneficios como la reducción del tiempo de entrega, la disminución de costos del desarrollo tradicional, la escalabilidad y la implementación automatizada en cada etapa del desarrollo. Las mejores prácticas implican liberar cambios con documentación, usar desarrollo basado en tronco, tener canalizaciones automatizadas y buscar cero interrupciones en las actualizaciones (IBM, 2023).

2.2.2.3. Monitoreo Continuo.

En DevOps, el monitoreo continuo es la práctica de monitorear constantemente el rendimiento del sistema y los recursos, y detectar de forma proactiva cualquier cambio o anomalía que pueda afectar el rendimiento. En la infraestructura en la nube, el monitoreo continuo es fundamental para verificar la eficacia de los controles y mitigar los riesgos, ofreciendo datos casi en tiempo real para identificar posibles amenazas y resolverlas

antes de que causen problemas. El monitoreo continuo es esencial en un panorama más amplio, desde el desarrollo hasta las operaciones de TI, y Azure Monitor unifica la observabilidad integral en todas sus aplicaciones e infraestructura (Microsoft, 2023).

En la práctica, la supervisión continua se incorpora en todas las etapas de los ciclos DevOps y operaciones de TI para mantener una vigilancia constante del estado, el rendimiento y la fiabilidad de las aplicaciones e infraestructuras a medida que avanzan por las etapas de desarrollo, producción y usuarios. Hay siete mejores prácticas para el monitoreo continuo: monitorear todas las aplicaciones y partes relevantes de la infraestructura, habilitar el monitoreo en todos los entornos en que se implemente, configurar alertas procesables con notificaciones y correcciones, preparar paneles y libros de trabajo basados en roles para informes y mejorar continuamente usando el ciclo "Construir-Medir-Aprender" (Microsoft, 2023).

2.2.2.4. Automatización de pruebas

La automatización de pruebas, como se define, es el proceso de usar herramientas para ejecutar pruebas en un producto de software (por ejemplo, una aplicación web) de forma automática. Esto significa verificar que el software se ajuste a los estándares de calidad establecidos (estilo de código, funcionalidad, experiencia de usuario, etc.) sin intervención humana significativa. Esta metodología hace posible una revisión más temprana y eficiente de los productos de software a lo largo de su ciclo de vida, permitiendo identificar errores en una etapa temprana y mejorar la calidad de forma continua (Atlassian, 2024).

La automatización de pruebas es un paso esencial en DevOps para integrar y automatizar la fase de pruebas en el ciclo de vida de CI/CD (Atlassian, 2024). Al hacerlo, aceleran los ciclos de lanzamiento y mejoran la colaboración entre los equipos de desarrollo y operaciones, en línea con los principios DevOps de velocidad, eficiencia y alta calidad en la entrega de software. Se hace énfasis en tener en cuenta aspectos como la frecuencia de lanzamiento, las herramientas actuales de automatización y la madurez del producto para establecer una estrategia de automatización de pruebas y cómo esto transforma la función de control de calidad en una más integrada al ciclo de vida del software (Atlassian, 2024; Red Hat, 2023).

Herramientas para automatizar pruebas (Sentrio, 2022):

Para seleccionar las mejores herramientas de automatización, DevOps utilizó un enfoque de "todo en uno" con múltiples herramientas para propósitos específicos en lugar de una sola herramienta para todo. Estas herramientas fueron determinantes en todo el ciclo de vida de la producción de software, desde el desarrollo hasta la administración y entrega del software. Entre las herramientas automatizadas para DevOps más destacadas para acelerar el desarrollo de software y mejorar la calidad de las aplicaciones se encontraron:

- Jenkins, herramienta de CI (integración continua) que automatizó tareas como compilación, pruebas y despliegue continuo de aplicaciones. Su adaptabilidad y la gran cantidad de extensiones disponibles la hicieron muy personalizable.

- Git, que revolucionó el control de versiones, permitiendo rastrear cambios y colaborar en equipos, integrándose con otras herramientas de automatización.

- Docker, la herramienta de contenedores que transformó la manera en que se desarrolla y despliega el software, capaz de encapsular una aplicación y sus dependencias en unidades aisladas para asegurar la portabilidad y la consistencia a lo largo del ciclo de vida de la aplicación.

- Kubernetes, una solución para orquestar contenedores y así poder automatizar el despliegue, la supervisión y la escalabilidad..

- Ansible, herramienta de automatización para administración de configuración, servidores, despliegue de aplicaciones y administración de infraestructura, reconocida por su naturaleza declarativa y su capacidad de efectuar cambios de manera repetible y controlada.

- Puppet, herramienta para administrar la configuración y la infraestructura como código, automatizando la configuración de servidores y aplicaciones para la escalabilidad.

- JFrog Artifactory, que proporcionó una forma de administrar artefactos, almacenarlos, organizarlos y distribuirlos.

- Chef, que permitió configurar, aprovisionar y gestionar la infraestructura y las aplicaciones de manera consistente y escalable.

- SonarQube, herramienta de análisis estático de código, que ayudó a asegurar la calidad del código en el proceso de desarrollo, permitiendo identificar errores tempranamente y fomentando las buenas prácticas de programación.

Estas herramientas se volvieron fundamentales en la metodología DevOps, ayudando a automatizar procesos y mejorar la colaboración y eficiencia en el desarrollo y entrega de software.

2.2.2.5. Despliegue Continuo

La justificación de esta estrategia se basa en la automatización con pruebas predefinidas, lo que permite la liberación automática de cambios de código al entorno de producción una vez que han pasado las pruebas definidas. Este enfoque ayuda a acelerar el time-to-market, eliminando los cuellos de botella entre la programación y el valor entregado al cliente, y reduciendo drásticamente los costos de las pruebas manuales de regresión. También vale la pena mencionar la eliminación de mecanismos para controlar paquetes grandes de cambios a producción, tales como juntas de planificación y aprobación de publicaciones, en la mayoría de los casos. (IBM, 2022).

2.2.2.6. Infraestructura como codigo

La infraestructura como código (IaC) es una práctica clave en DevOps que usa archivos descriptivos y control de versiones para definir y aprovisionar infraestructuras

como redes, máquinas virtuales y equilibradores de carga. Este método garantiza que cada ejecución cree el mismo entorno, lo que fomenta la confianza en el desarrollo y la entrega de aplicaciones a escala (Microsoft, 2022).

Según IBM, la IaC usa un lenguaje de codificación declarativo para automatizar el aprovisionamiento de la infraestructura de TI sin tener que manipular servidores, sistemas operativos, etc. manualmente. En los lugares donde se despliegan muchas aplicaciones al día, la IaC es una necesidad para controlar los costos, disminuir los riesgos y adaptarse rápidamente a las nuevas oportunidades y amenazas competitivas (IBM, 2022).

Amazon destaca que la IaC permite aprovisionar y respaldar la infraestructura con código, en lugar de procesos manuales, acelerando el desarrollo, las pruebas y la implementación de aplicaciones. La automatización resultante elimina errores de configuración, facilita la replicación de entornos y acelera la detección y resolución de problemas (Amazon, 2023).

En cuanto a las ventajas, la IaC permite replicar entornos, disminuir errores de configuración y mejorar la iteración en entornos de prácticas recomendadas. La capacidad de especificar el estado deseado de la infraestructura en código libera a los desarrolladores para que se enfoquen en mejorar continuamente las aplicaciones en lugar de administrar entornos manualmente (Amazon, 2023).

La IaC es una parte integral de DevOps que proporciona consistencia, automatización y eficiencia en la implementación y gestión de infraestructuras de TI. Estas características son fundamentales para superar los obstáculos de la implementación continua y ágil de aplicaciones empresariales.

2.2.3. ISO/IEC 29110

En la norma oficial de la ISO/IEC 29110 (7. SO/IEC TR 29110-1:2016(es), Ingeniería de Software y Sistemas — Perfiles de ciclo de vida para Pequeñas Organizaciones (VSEs) — Parte 1: Visión general, s. f.). Indican que es un conjunto de directrices y especificaciones elaboradas para asistir a las pequeñas organizaciones (VSEs) en la mejora de la calidad de los sistemas, software o servicios que generan y el rendimiento de sus procesos.

Esta norma considera las necesidades propias de las VSEs, entendiendo por tales a las empresas, organizaciones, departamentos o proyectos con no más de 25 personas. La norma se apoya en perfiles que agrupan subconjuntos de procesos, actividades, tareas y resultados de otras normas internacionales aplicables a las VSEs, tales como la ISO/IEC/IEEE 12207 para software y la ISO/IEC/IEEE 15288 para sistemas. Estos perfiles se ajustan a las necesidades de las VSEs y sirven de guía para establecer procesos de ciclo de vida de sistemas y software de calidad. Al establecer un perfil basado en la norma ISO/IEC 29110, las VSEs pueden ser reconocidas y auditadas bajo esta norma. La norma se puede aplicar en cualquier etapa de un ciclo de vida de desarrollo de sistemas o software, como cascada, iterativo, incremental, evolutivo o ágil.

De manera general, la ISO/IEC 29110 es una guía desarrollada específicamente para ayudar a las pequeñas organizaciones a mejorar la calidad de sus sistemas, software y servicios, y a optimizar sus procesos de desarrollo. Esta norma proporciona pautas claras y específicas para ayudar a estas organizaciones a alcanzar estándares más altos en términos de calidad y eficiencia en sus operaciones relacionadas con el desarrollo de sistemas.

2.2.4. Reinforced DevOps (DevOps Reforzado).

DevOps tiene muchos beneficios cuando se implementa correctamente en dos casos: evolucionando el proceso actual dentro de una organización o implementándolo en una organización. Para abordar los problemas identificados, se propone una guía que refuerza el proceso genérico de DevOps con prácticas provenientes de la norma ISO/IEC 29110 (Muñoz et al., 2021; Muñoz & Rodríguez, 2021).

En otra investigación se menciona que DevOps se centra en la colaboración del equipo para automatizar el proceso utilizando flujos de trabajo visibles. Sin embargo, algunas organizaciones no están dispuestas a adoptar este enfoque debido a los riesgos o problemas específicos clasificados en tres categorías: orientación, procesos y equipo. Se señala que la serie ISO/IEC 29110 puede mejorar el proceso DevOps al garantizar la calidad del producto, reducir el retrabajo y proporcionar orientación para la definición de procesos, la gestión de riesgos y la provisión de roles para mantener un equipo auto coordinado (Muñoz & Negrete, 2020). Además, señalan que se utilizaron artefactos, roles y tareas del perfil Básico de ISO/IEC 29110 en casos de estudio de organizaciones con problemas similares a los detectados en entornos DevOps. La propuesta de la guía consiste en utilizar las prácticas, tareas, productos de trabajo

y roles del perfil Básico de la serie ISO/IEC 29110 para obtener un enfoque DevOps Reforzado que asegure la calidad, evite el retrabajo y reduzca los riesgos (Muñoz & Negrete, 2020).

En cuanto a los roles en DevOps Reforzado se tomó en cuenta tanto los roles de la ISO/IEC 29110 como los roles del proceso DevOps genérico para desarrollar una matriz comparativa. Esta matriz permite revisar las actividades realizadas por cada rol y establecer la relación entre ellos. También se menciona que el equipo de trabajo se divide en dos grupos: el equipo de desarrollo y el equipo de operaciones, los cuales tienen diferentes actividades, pero deben trabajar juntos (Muñoz & Negrete, 2020).

2.2.4.1. Proceso DevOps Reforzado.

El DevOps Reforzado menciona que mediante una revisión de literatura se llegó a definir un proceso de DevOps genérico, Se identificaron analizaron y clasificaron las fases, actividades y tareas de un enfoque DevOps (Muñoz & Negrete, 2020).

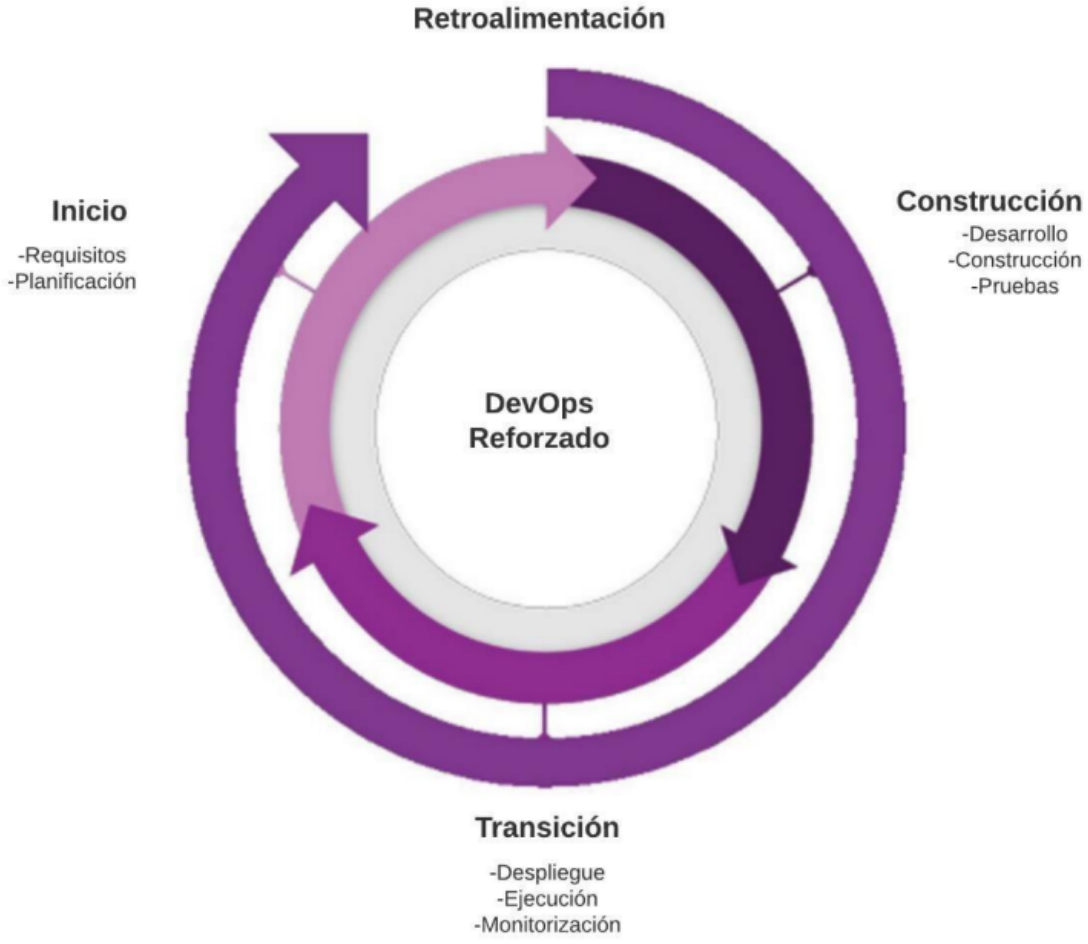


Figura 2. Ciclo DevOps Reforzado
Fuente: (Muñoz et al., 2021)

En la fase de "Inicio", se busca elaborar la planificación del proyecto y establecer los requerimientos, el cronograma, las herramientas y el equipo esencial para la ejecución del proyecto en su totalidad. Esta fase implica dos pasos principales: la identificación de los requisitos y la planificación (Muñoz & Negrete, 2020).

En "Construcción" se busca controlar el flujo de trabajo desde la confirmación en la etapa hasta la aprobación de cambios, pasando por diferentes tipos de pruebas (validación, exploratorias, unitarias, modulares, regresión, entre otras) para verificar las piezas de software que se desplegarán en el ambiente productivo.

El objetivo es automatizar estas tareas según ciertas reglas para lograr las propiedades deseadas en el software. Esta etapa se divide en tres partes: desarrollo, compilación y pruebas. (Muñoz & Negrete, 2020).

La etapa de "Transición" tiene como objetivo instalar el software en un servidor o en el hardware apropiado para que funcione. Esta entrega puede realizarse de forma manual o automatizada. Una vez que el software está instalado, la fase de lanzamiento consiste en generar los entregables de software en base a las nuevas funcionalidades incorporadas o errores corregidos en funcionalidades previas. Luego, el software se entrega al cliente para su uso. Luego, el equipo operativo toma el control para monitorear los servidores y las aplicaciones en busca de problemas. Esta fase consta de tres componentes esenciales: implementación, ejecución y supervisión. (Muñoz & Negrete, 2020).

La etapa "Retroalimentación" busca proporcionar información sobre cómo resolver problemas y mejorar el proceso. Para alcanzar este propósito, las partes involucradas y el equipo de trabajo dan sus ideas para evitar problemas y mantener en funcionamiento el software. Es importante mencionar que el proceso genérico de DevOps consta de cuatro fases, ocho actividades y cuarenta tareas que deben ser ejecutadas en el orden establecido para realizar un proceso de desarrollo de software con pensamiento DevOps. (Muñoz & Negrete, 2020) .

2.2.4.1. Roles DevOps Reforzado.

Para la elección de los roles adecuados en el contexto del enfoque DevOps Reforzado, se tomaron en cuenta los roles descritos en la norma ISO/IEC 29110 y los roles del proceso genérico de DevOps. Dicho análisis dio como resultado una matriz comparativa, donde se comparan las funciones que realiza cada rol y así identificar las relaciones que existen entre ellos. (Muñoz & Negrete, 2020) Para la elección de los roles se utilizaron los siguientes criterios: Cuando un determinado rol DevOps no tiene ninguna asociación con ningún rol definido en la norma ISO/IEC 29110 (Muñoz & Negrete, 2020).

En cuanto al equipo humano, se dividió en dos: el equipo de desarrollo y el equipo de operaciones. Cada uno de estos grupos realiza funciones diferentes, pero deben trabajar en conjunto y coordinados (Muñoz & Negrete, 2020).

Tabla 1. Roles de DevOps Reforzado

ISO / IEC 29110	DEVOPS	REINFORCED DEVOPS
Analista (AN)	Gestor del flujo de valor (VSM)	Analista (AN)
Jefe de proyecto (PM)	Maestro de procesos (PM)	Maestro de procesos (PM)
Líder técnico (TL)	Ingeniero de DevOps (DE)	Ingeniero de DevOps (DE)
Equipo de trabajo (WT)	Equipo de Desarrollo (DT) y Operaciones (OP)	Equipo de Desarrollo (DT) y Equipo de Operaciones (OP)
Cliente (CUS)	Partes interesadas	Partes interesadas
	Garantía de calidad(QA)	Garantía de calidad(QA)
	Infosec (IS)	Infosec (IS)
	Portero (GK)	Portero (GK)

Ingeniero de fiabilidad (RE)	Ingeniero de fiabilidad (RE)
Propietario del producto (PO)	Propietario del producto (PO)

Fuente : (Muñoz & Negrete, 2020).

2.2.4.2. Tareas de DevOps Reforzado

El proceso DevOps reforzado tiene 4 fases, 9 actividades y 87 tareas, que deben ser implementadas para realizar un proceso de desarrollo de software con cultura DevOps(Muñoz & Negrete, 2020). Ver **ANEXO A**.

2.2.4.3. Pirámide DevOps Reforzado.

Con el propósito de simplificar la implementación del proceso DevOps Reforzado, se desarrolló una estrategia que se alinea con la estructura jerárquica de necesidades de la pirámide de Maslow. Con el fin de llevar a cabo la materialización del proceso DevOps reforzado, se elaboró un manual de orientación. La concepción de esta guía se fundamentó en el marco conceptual de la pirámide de necesidades de Maslow, se denominó a esta representación "pirámide DevOps". Su creación se sustentó en las demandas inherentes a la organización en relación con la adopción de un enfoque DevOps (abarcando aspectos como cultura, colaboración, automatización y medición).

Dentro de esta pirámide DevOps, se atendieron las necesidades que guían a la organización hacia la ejecución de un enfoque DevOps eficaz. Dependiendo de estas necesidades identificadas, la organización puede asumir tareas específicas que le permitan avanzar en la dirección correcta para implementar DevOps de manera óptima (Muñoz et al., 2021; Muñoz & Rodríguez, 2021).

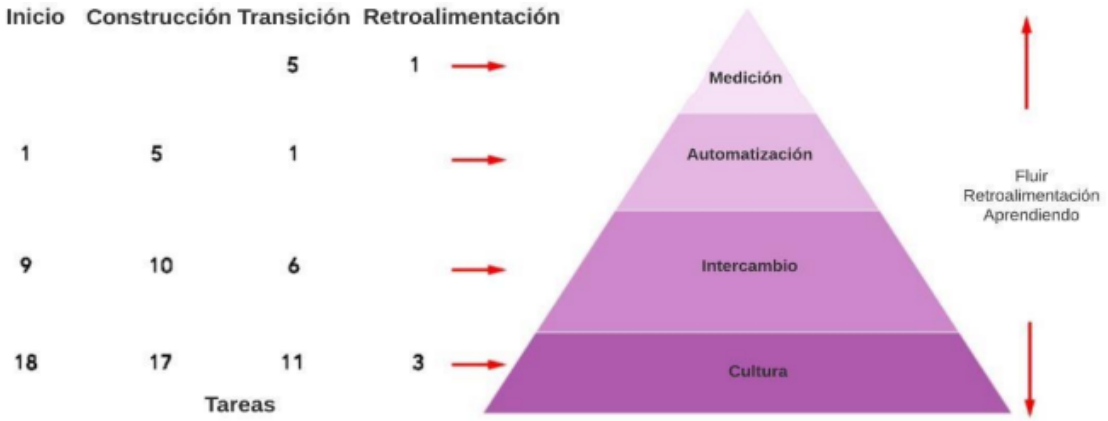


Figura 3. Piramide DevOps Reforzado
Fuente: (Muñoz et al., 2021)

2.2.5. Modelo de Aceptación Tecnológica (TAM).

El Modelo de Aceptación Tecnológica (TAM), propuesto inicialmente por Davis, se basa en que el uso de la tecnología está determinado por la motivación del individuo y directamente por estímulos externos, en función de las propiedades y capacidades del sistema. (F. Davis, 1985)

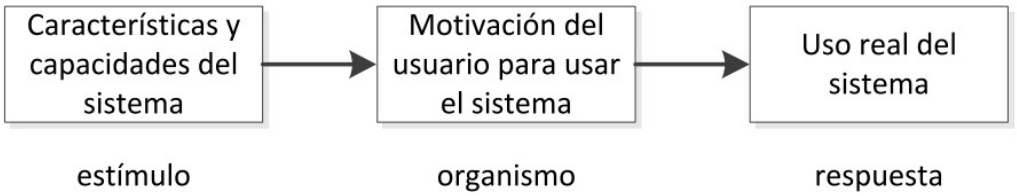


Figura 4. Raíz de TAM
Fuente: (F. Davis, 1985)

Este modelo ha sido reconocido por su uso para predecir la adopción y el uso de tecnologías de información y comunicación (TIC) en diferentes contextos, tales como motores de búsqueda, sitios web, compras en línea, plataformas de aprendizaje como Moodle (F. Davis, 1985).

En sus inicios en 1985, Davis propuso el TAM como una forma de explicar las causas detrás de la aceptación de las tecnologías por parte de los usuarios. Basándose en la Teoría de la Acción Razonada, Davis sugirió que la motivación del usuario se explica a través de tres factores: la utilidad percibida, la facilidad de uso percibida y la actitud hacia el uso del sistema(F. Davis, 1985).

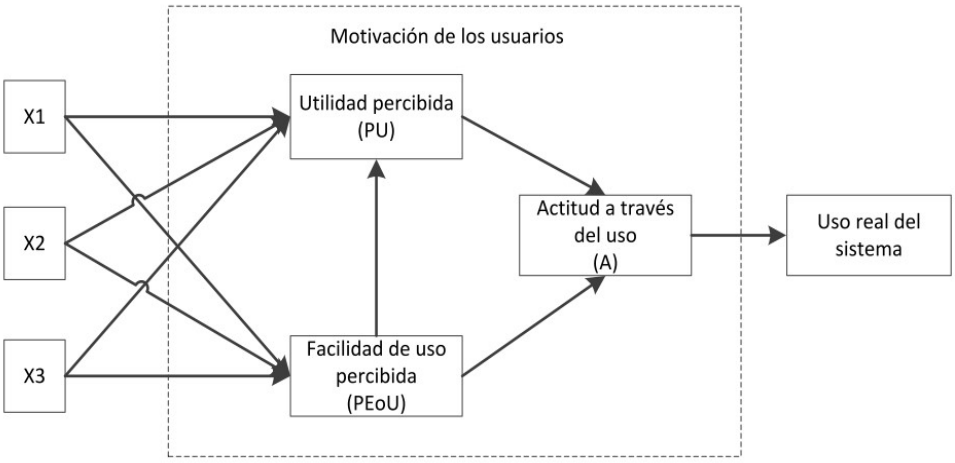


Figura 5. Primera propuesta de TAM
Fuente: (F. Davis, 1985)

Posteriormente, en una versión mejorada dos años después, se destacó que la utilidad percibida y la facilidad de uso percibida tienen una influencia directa en la intención de uso, eliminando el factor de actitud hacia a través de uso (F. Davis, 1985).

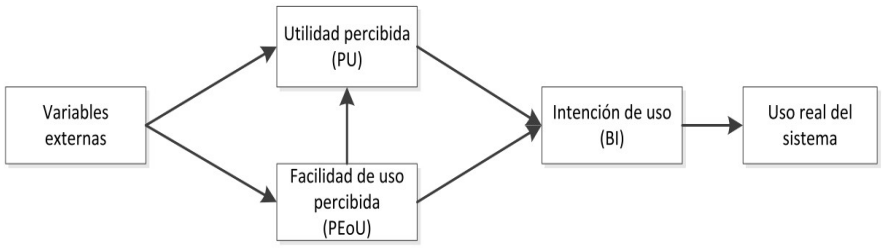


Figura 6. Versión final de TAM
Fuente: (F. Davis, 1985)

2.2.6. Modelo de madurez.

El modelo de madurez Bucena(Zarour et al., 2020) es un marco diseñado para evaluar el nivel de adopción de DevOps dentro de una organización, especialmente en empresas pequeñas o muy pequeñas. Su propósito es medir qué tan preparadas están las organizaciones para implementar prácticas DevOps y qué tan efectivas son al aplicarlas.

El modelo está estructurado en cuatro dimensiones clave:

1. Cultura: nivel de transparencia colaboración, comunicación y mentalidad DevOps.

2. Personas: competencias, aprendizaje y organización del equipo.
3. Procesos: nivel de automatización, integración continua, pruebas, documentación y gestión del proyecto.
4. Tecnología: uso de herramientas, automatización, monitoreo, gestión de entornos y control de versiones.

El modelo permite diagnosticar el estado actual de DevOps en una organización, identificar fortalezas y debilidades en cada dimensión, determinar el nivel de madurez general de la organización, guiar mejoras futuras, mostrando qué capacidades deben reforzarse para subir de nivel. El modelo se usa mediante un cuestionario estructurado, donde cada factor tiene posibles respuestas asociadas a un nivel (1 a 5).

Cálculo del nivel de capacidad de cada dimensión, usando la fórmula:

$$Nivel\ de\ Capacidad\ de\ la\ Dimensión = \frac{\sum_{i=1}^{n} f_i\ Puntaje\ Obtenido}{\sum_{i=1}^{n} f_i\ Puntaje\ Maximo}\ X\ l \qquad (1)$$

$f = \ El\ factor\ dentro\ de\ la\ dimensión\ evaluada$
$N = Número\ de\ factores\ en\ la\ dimensión\ evaluada$
$l = 5\ (numero\ de\ niveles)$
Fuente: (Zarour et al., 2020).

Aplicarlo permite conocer el nivel de madurez actual (ejemplo del estudio: organizaciones entre nivel 3 y 4), las áreas más débiles, como gestión de datos, documentación, monitoreo, gestión de proyectos, seguimiento de incidencias, qué dimensiones requieren intervención inmediata, y cuáles están desarrolladas. Si la organización necesita fortalecer cultura, capacitación, automatización o procesos.

III. MATERIALES Y MÉTODOS

En este capítulo se presentan los materiales y métodos utilizados en la presente investigación.

3.1. Lugar de ejecución.

El escenario de esta investigación se localiza en la región de Huánuco, más específicamente en el distrito de Rupa-Rupa, que forma parte de la provincia de Leoncio Prado. La ubicación geográfica de esta área se describe mediante las coordenadas de latitud y longitud, que son -9.29833 grados al sur y -76.0003 grados al oeste, respectivamente, equivalentes a -9° 17′ 54″ de latitud sur y -76° 0′ 1″ de longitud oeste. La extensión total de esta región abarca 36,900 hectáreas, equivalente a 369 kilómetros cuadrados. La altitud promedio de la zona se sitúa en 648 metros sobre el nivel del mar. En cuanto al clima predominante en esta área, se trata de un clima ecuatorial, que se caracteriza por temperaturas elevadas y una humedad constante a lo largo de todo el año.

3.2. Materiales y Métodos.

En este estudio se emplearon dos tipos de recursos: equipos y software, cuyas características se describen a continuación.

3.2.1. Materiales.

- Equipos

Tabla 2. Tabla de Materiales (Equipos).

Equipo	Utilidad
Laptop	Se necesito una laptop para investigar sobre el monitoreo, y las herramientas que puedan ayudar a este proceso
Conexión a Internet	Se necesito conectividad a internet debido a que se rrequirio un entorno en la nube para hacer el monitoreo
Proyector Data	Se necesito un proyector multimedia para las capacitaciones de la herramienta

Fuente: Elaboración propia

- Software

Tabla 3. Tabla de Materiales (Software)

Software	Utilidad
Microsoft Forms	Se utiliza un formulario en línea para evaluar la madurez del equipo
Elastic ELK (8.10.3)	Se usa esta Herramienta para el monitoreo

Fuente: Elaboración propia

3.2.2. Metodología

En esta sección se detalla el tipo de investigación llevada a cabo, el alcance abarcado, el diseño empleado, el enfoque seguido, la población objeto de estudio, la selección de la muestra, la forma en que se definieron y midieron las variables analizadas, así como las técnicas e instrumentos utilizados para recopilar y analizar datos, y se evalúa la validez y confiabilidad del instrumento aplicado.

3.2.3. Población : Indeterminada

La población de esta investigación abarca individuos con rasgos comunes, y su extensión es indeterminada (Roberto Hernández Sampieri et al., 2014). Esta conformada por todos los miembros de las empresas pequeñas de desarrollo de software**.**

3.2.3.1. Unidad de análisis

La unidad de análisis está constituida por cada integrante del equipo técnico, considerado individualmente, ya que son los usuarios directos quienes permiten evaluar su aceptación mediante el modelo TAM.

3.2.3.2. Muestra : No probabilística

La muestra es no probabilística por conveniencia, ya que se seleccionó a los trabajadores directamente involucrados con el uso de la herramienta implementada. La muestra estuvo conformada por 6 participantes de una sola empresa.

3.2.3.3. Tipo de investigación: Aplicada

Esta investigación se clasifica como aplicada, de acuerdo con la definición proporcionada por Hernández Sampieri (Roberto Hernández Sampieri et al., 2014). Esta categorización se fundamenta en la naturaleza del estudio, que se centra en la resolución de problemas específicos en un contexto particular. La investigación no solo crea conocimiento teórico, sino que también busca resolver problemas prácticos. Usando teorías ya existentes, se usan métodos experimentales para profundizar en el entendimiento y solución de problemas particulares.

3.2.3.4. Diseño de Investigación : Preexperimental

En la investigación se utilizará un diseño preexperimental, según la metodología de Sampieri (Roberto Hernández Sampieri et al., 2014). Ya que al plantear nuevas herramientas, vamos a tener poco control, además de que recolecta datos de adopción tecnológica en un solo momento en el tiempo y no manipula grupos de muestra. Pero antes de poner en marcha estas herramientas, mediremos la madurez DevOps para apoyar nuestras conclusiones con más fuerza. Nuestra principal línea de trabajo continuará siendo la medición de la aceptabilidad de las herramientas que proponemos.

$$G: X1 - \mathrm{Y}1 - Z2$$

G: Trabajadores

X1: Diagnóstico de madurez y mapeo empirico de procesos

Y1: Aplicación de las Herramientas

Z2: Medición mediante cuestionario TAM

3.2.3.5. Alcance de Investigación : Descriptivo

La revisión de las herramientas planteadas para hacer un "DevOps Reforzado" es de alcance descriptivo. (Roberto Hernández Sampieri et al., 2014). En este tipo de estudio lo que se busca es describir de manera exacta cuánta aceptación tienen las herramientas propuestas. El propósito es proporcionar una visión clara y completa de las herramientas en este contexto, lo que permitirá a los profesionales y organizaciones comprender mejor su utilidad y aplicabilidad.

3.2.3.6. Variable : Aceptación Tecnológica

Definición Conceptual: Es una teoría de la psicología social que determina el nivel de aceptación tecnológica de una sociedad ante la llegada de nuevas tecnologías. Este modelo asume que a través del análisis se puede deducir si una sociedad es más propensa a aceptar innovaciones o si, por el contrario, es conservadora. Es un instrumento para sondear las expectativas de una sociedad sobre lo que una tecnología puede ofrecer. (F. Davis, 1985).

Definición Operacional: Es un modelo que explica y predice la aceptación y el uso de la tecnología en el trabajo. Según TAM, la intención de uso de un sistema depende de dos creencias: la utilidad percibida, que es el grado en que una persona cree que usar el sistema mejorará su rendimiento laboral, y la facilidad de uso percibida, que es el grado en que una persona cree que usar el sistema será libre de esfuerzo. TAM también incorpora otras variables que influyen en la utilidad percibida y la intención de uso, como los procesos de influencia social y los procesos instrumentales cognitivos. TAM es un modelo robusto, poderoso y parsimonioso que ha sido ampliamente validado por numerosos estudios empíricos.

3.2.3.7. Dimensiones.

En el siguiente cuadro se muestra los indicadores por cada dimensión. Es valido mendiconar que los indicadores se volvieron preguntas para el cuestionario.

Tabla 4. Dimensiones e Indicadores de TAM

Dimensiónes	Indicadores
Utilidad percibida	Incremento del rendimiento laboral percibido
	Aumento de la productividad
	Mejora de la eficacia en las tarea
	Percepción de utilidad general de la herramienta
Facilidad de Uso	Claridad y comprensión en la interacción con la herramienta
	Nivel de esfuerzo mental requerido
	Percepción de facilidad de uso general
	Capacidad para lograr objetivos con la herramienta
Intención de Uso	Intención declarada de uso futuro
	Predicción de uso basada en disponibilidad.
	Planificación de uso sostenido en el tiempo.

Fuente: (F. D. Davis et al., 1989)

3.2.3.8. Técnicas e Instrumentos de investigación

La técnica utilizada en esta investigación fue la encuesta, mientras que el instrumento correspondió a un cuestionario estructurado, el cual empleó una escala Likert de 1 a 7 para registrar las respuestas. Este cuestionario fue diseñado específicamente para medir el grado de aceptación tecnológica.

- 1 – Totalmente en desacuerdo
- 2 – En desacuerdo
- 3 – Ligeramente en desacuerdo
- 4 – Neutral/ Ni de acuerdo ni en desacuerdo
- 5 – Ligeramente de acuerdo
- 6 – De acuerdo
- 7 – Totalmente de acuerdo

Se utilizaron las siguientes preguntas para el cuestionario, las preguntas fueron elaboradas en base a investigaciones originales de TAM que propone las preguntas para la utilizacion de otra herramienta (F. D. Davis et al., 1989). En esta ocacion lo adaptamos para el uso del ELK, siendo este cuestionario ya validado.

Tabla 5. Preguntas para Utilidad Percibida

#	Pregunta
1.1	Usar ELK mejora mi rendimiento en el trabajo
1.2	Usar ELK incrementa mi productividad
1.3	Usar ELK mejora mi eficacia en el trabajo
1.4	Encuentro util el uso de ELK en mi trabajo

Fuente : Elaboración propia.

Tabla 6. Preguntas Facilidad de Uso

#	Pregunta
2.1	Mi interacción con ELK es clara y comprensible
2.2	Usar ELK no requiere mucho esfuerzo metal
2.3	Encuentro que ELK es fácil de usar
2.4	Me resulta facil lograr que ELK haga lo que necesito

Fuente : Elaboración propia.

Tabla 7. Preguntas para Intencion de Uso

#	Pregunta
3.1	Si tengo acceso a ELK, tengo la intención de usarlo
3.2	Dado que tengo acceso a ELK, predigo que lo usaré
3.3	Planeo usar ELK en los proximos meses

Fuente : Elaboración propia.

3.2.3.9. Metodologia de Implementación (Investigación – Acción)

La metodología de investigación-acción, conocida en inglés como Action Research, facilitó el desarrollo de la implementación del monitoreo y de las herramientas ELK, proporcionando un enfoque estructurado. Esta metodología permitió establecer un camino claro a seguir, desde el punto de partida hasta el objetivo final, brindando orientación sobre cómo iniciar y culminar el proceso.

IV. IMPLEMETACIÓN DEL STACK ELK

4.1. Investigación – Acción (Fase 1)

1. Diagnóstico: Se hizo una evaluación de la madurez con el modelo propuesto por Mohammad (Zarour et al., 2020)(Ver **ANEXO B**), para tener una visíon general de cual era el grado de madurez en el que estaba la empresa . Esta evaluación brindo de manera general las deficiencias de la empresa. Se tuvo una reunión con los involucrados y se identificó una dificultad en el seguimiento de errores en los servicios debido a las limitaciones de CloudWatch, correspondientes al proceso de Monitoreo del DevOps Reforzado. Esta herramienta tiene dificultades para buscar logs y carece de automatización, lo que dificulta encontrar y solucionar problemas. Es lento y no encaja en un mundo DevOps moderno en el que necesitas ser eficiente y rápido con los logs.

2. Planificación:

Tabla 8. Comparación de Herramientas

Criterio	Prometheus	Grafana	ELK
Legibilidad de los logs	Si	Si	Si
Dockerización	Si	Si	Si
Se ajusta al contexto del negocio	No	No	Si
Facilidad de aprendizaje	No	Si	Si

Fuente: Elacoración propia

En la Tabla 8 se pueden observar las comparaciones que realizamos entre las herramientas que nos podían solucionar el problema del monitoreo de logs. Se miró la legibilidad de logs, si era dockerizable, si era fácil de aprender y lo más importante, si se adaptaba al negocio.

Tras evaluar distintas herramientas, se decidió implementar el stack ELK (Elasticsearch, Logstash y Kibana) para la gestión y visualización de logs. Esta solución se eligió por su capacidad para procesar grandes cantidades de datos, optimizar el almacenamiento y ofrecer una interfaz más amigable para el análisis de registros. La implementación se realizó en los tres principales servicios: Lobby-BFF, UniversalAPI y API_Usoft, garantizando que los logs no solo se recopilen, sino que también se etiqueten con el nombre del servicio para una identificación y seguimiento más sencillos.

3. Ejecución: Para realizar esta implementación, primero se desplegó una versión dockerizada del stack ELK, lo que permitió probar sin incurrir en costos adicionales. Luego, se generó una rama en Git que clonara el entorno de producción para poder probar el nuevo código sin interrumpir el servicio. Esta estrategia aseguró una implementación controlada, minimizando riesgos antes de su implementación completa. Ver **ANEXO C**

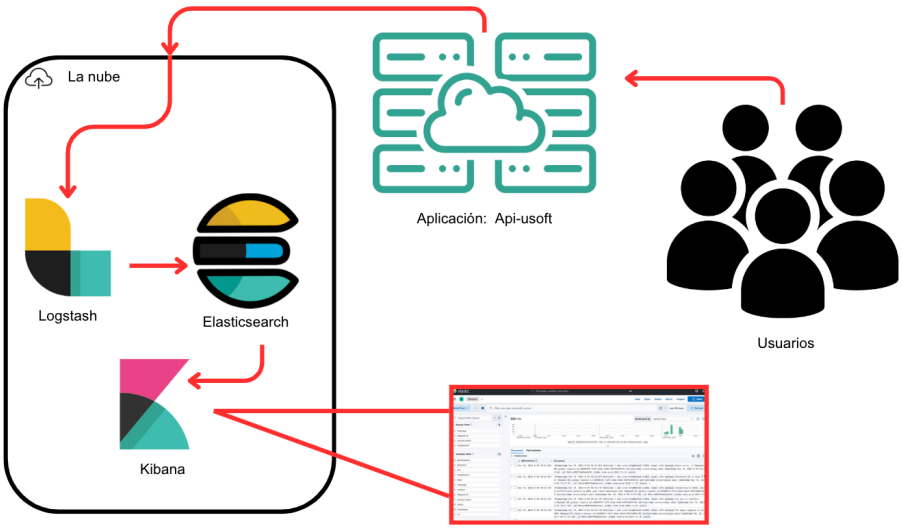


Figura 7. Arquitectura del Stack ELK
Fuente: Elaboración propia

En la **Figura 7,** se evidencia la integración del Stack ELK con la aplicación: los usuarios utilizan la aplicación desplegada en la nube, mientras que el Stack ELK se encuentra

implementado en un entorno de nube independiente. En el código fuente de la aplicación se configuró el envío de logs hacia dicho entorno, de modo que los registros son recibidos por Logstash, procesados y almacenados en Elasticsearch, y finalmente visualizados mediante Kibana.

4. Evaluación: Se puede observar la integración del Stack ELK con la aplicación: los usuarios interactúan con la aplicación alojada en la nube, y el Stack ELK está desplegado en la nube aparte. En el código se hizo que la aplicación mandara logs a ese ambiente, por lo que Logstash los recibe, los procesa y los guarda en Elasticsearch, para visualizarlos en Kibana.

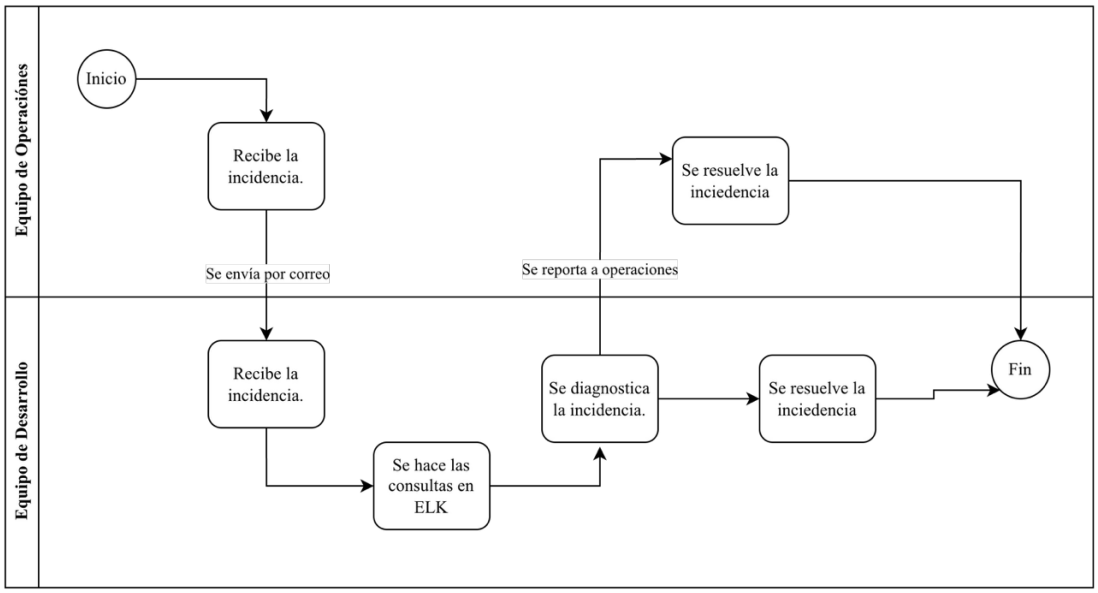


Figura 8. Diagrama de Procesos de Indicencias
Fuente: Elaboración propia

En la **Figura 8** muestra como se están utilizando ahora los procesos de monitoreo, a través del ELK, el cual ahora permite:

- Monitoreo en tiempo real y una mejor estructuración de los logs.
- Automatización de la recopilación de logs con almacenamiento optimizado.
- Escalabilidad, permitiendo la integración de nuevos servicios sin dificultades.

Este cambio ha mejorado la productividad y estabilidad del sistema, siendo una herramienta esencial para la operación y mantenimiento de los servicios en producción.

5. Aprendizaje: Con Stack ELK se ha vuelto más eficiente buscar y analizar logs, y se puede dar respuesta más rápido a incidentes que con CloudWatch. La capacidad de visualizarlos en Kibana ha ayudado a identificar problemas en los servicios y a organizarlos mejor en Elasticsearch. Además, la infraestructura ahora es más escalable, lo que permite agregar nuevos servicios sin impactar el rendimiento del sistema.

Pero el tipo de juego de apuestas o de servicios ha mostrado otros problemas. Cada apuesta crea varios logs (ganancia, pérdida, empate, etc.), aumentando la cantidad de datos y dificultando el rastreo de una acción específica del usuario. Además, la ausencia de un Request ID impide agrupar todos los eventos asociados a una misma apuesta y dificulta el análisis y correlación de eventos.

Para hacer más eficiente el sistema, lo próximo será asignar un Request ID único a cada apuesta y así poder hacer un mejor seguimiento de las transacciones. Además, se mejorará el formato de los logs para disminuir la cantidad de datos sin perder información importante y así mejorar la gestión y el seguimiento del sistema.

4.2. Investigación – Acción (Fase 2)

1. Diagnóstico: A pesar de la mejora en la capacidad de búsqueda y análisis de logs con Stack ELK, se ha detectado la falta de trazabilidad en los eventos que se generan en el sistema de juego de apuestas. Hoy en día una sola apuesta genera varios registros (ganancia, pérdida, empate, etc.), lo que fragmenta la información y dificulta darle seguimiento a una apuesta en particular.

El problema es que no existe un identificador único (Request ID) que agrupe todos los logs de una misma apuesta. Sin este elemento, los registros quedan dispersos, lo que complica la correlación de eventos y afecta la capacidad de diagnóstico y respuesta ante fallos o anomalías en el sistema.

2. Planificación: Se decidióimplementar el identificador por cada accion del usuario, para otimizar aun mas la bsuqueda o trazabilidad de logs, asi mismo se decicio registrar un dato agregado, y es que tambien tiene que decir si es un GET, POST.

3. Ejecución: Se creó un script para asignar un Request-ID a los logs, lo cual es útil cuando un usuario activa alguna función del proyecto, especialmente si esta función llama a otras funciones. Este identificador único permite garantizar la trazabilidad de las acciones dentro del sistema. A veces los logs se disparan tan rápido que no se puede saber si son logs continuos, por lo que el Request-ID permite saber hasta dónde llegó una función en específico. En este caso, los dos archivos crearon un interceptor. Este interceptor añade metadatos a cada petición HTTP, como por ejemplo un identificador único (Request-ID) y el método HTTP (GET, POST, etc.). Estos datos son útiles para logs y tracing en sistemas distribuidos, mejorando la observabilidad de los eventos de la aplicación.Ver **ANEXO D**.

4. Evaluación: La introducción del Request-ID ha mejorado enormemente la trazabilidad y la auditoría de logs en el sistema de juego apuestas. Ahora cada llamada HTTP lleva un ID único, lo que permite seguirle la pista a todas las acciones que provienen de una misma apuesta. Esto ha hecho posible correlacionar eventos en Kibana y disminuir el tiempo para encontrar y resolver problemas en producción. Antes, el tiempo de diagnóstico de errores podía llegar hasta 2 horas desde que se recibía la alerta por correo, e incluso en algunos casos hasta 6 horas. Con la adición del Request-ID, este proceso ahora toma solo 30 minutos desde que se informa un incidente hasta que se identifica el problema

El monitoreo en tiempo real también ha mejorado, ya que los registros están estructurados y organizados. Con el Request-ID se pueden rastrear transacciones enteras y encontrar errores en un abrir y cerrar de ojos. Esto ha mejorado la forma de supervisar el sistema y de gestionar logs en entornos de alta concurrencia.

5. Aprendizaje: Aunque la búsqueda de logs ha mejorado bastante respecto a la primera fase, hacerse al Stack ELK tiene su curva de aprendizaje. La herramienta necesita conocimiento de criterios para hacer búsquedas precisas, filtros y comandos. Para sacarle el máximo provecho y explotar todo su potencial, se debe capacitar de manera general a todo el personal para que entiendan cómo funciona y puedan manipularla correctamente.

4.3. Investigación – Acción (Fase 3)

1. Diagnóstico: Se observó una mejora empírica en la manera de buscar logs y, por ende, se pudo solucionar errores en los servicios. Pero aunque los datos en Kibana sean más

legibles (como vimos en la fase 1) y la búsqueda sea más rápida (como vimos en la fase 2), el equipo seguía teniendo problemas para usar comodines de búsqueda y atajos en la plataforma. Como resultado, los desarrolladores tenían que consultar la documentación con frecuencia para hacer consultas complejas de manera eficiente.

2. Planificación: Se organizó una capacitación general donde cada integrante del equipo debía exponer un caso real con el que solía tener problemas para encontrar logs. La idea era que estos ejemplos se utilizaran como guía en la capacitación para demostrar cómo usar las herramientas de búsqueda de Kibana y mejorar su uso diario.

3. Ejecución: Se llevó a cabo la capacitación con ejemplos prácticos de situaciones comunes. Durante aproximadamente una hora, se proyectó Kibana en un televisor, y se trabajó en conjunto con el equipo para resolver los casos presentados. La sesión permitió aclarar dudas y proporcionar estrategias efectivas para mejorar la búsqueda de logs.

4. Evaluación: La capacitación fue altamente productiva, ya que, si bien inicialmente existían dudas sobre la herramienta, el debate generado permitió que surgieran preguntas adicionales basadas en problemas cotidianos del equipo. Estas dudas fueron abordadas en la misma sesión, lo que mejoró la comprensión y aplicación de Kibana.

Para cerrar el tema de la madurez se volvio a medir el grado de madurez. Despues de habler implementado y capacitado la Stack ELK. El modelo de madurez cuenta con sus propias dimensiones, métricas y fórmulas para determinar el grado de madurez (Zarour et al., 2020). Sin embargo, para esta investigación solo se consideró la mediana estadística para las dimensiones.

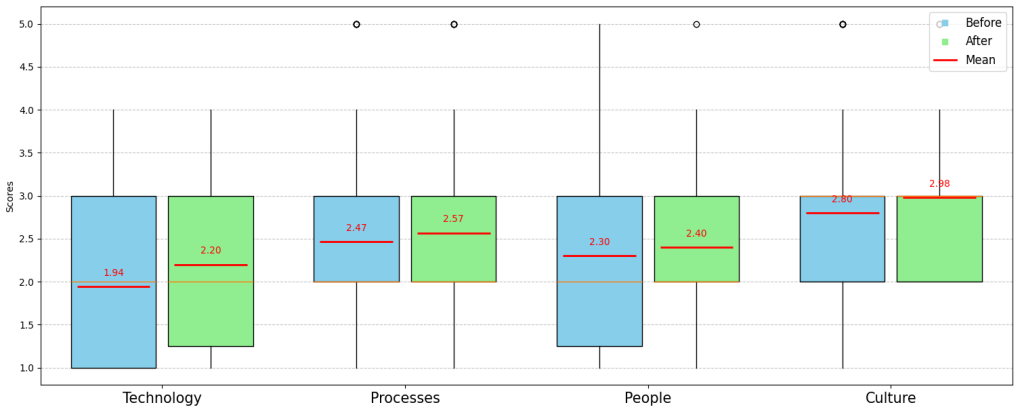


Figura 9. Resultados de la Madurez por dimensión
Fuente: Elaboración propia

En la **Figura 9** se muestra como aumentó el grado de cada dimensión, siendo la dimensión de cultura la que más aumentó en comparación con las demás. Estos resultados se obtuvieron estadísticamente luego de aplicar los cuestionarios en dos momentos de la investigación (antes y después de la implementación de la herramienta ELK).

Los datos del cuestionario fueron analizados de manera general siguiendo el método de cálculo de madurez propuesto por Zarour , se identificó un aumento. Este hallazgo indica que, mediante un diagnóstico y la atención a los problemas más críticos, y considerando la madurez DevOps, es posible mejorar su nivel de madurez, como se muestra en la **Figura 10**.

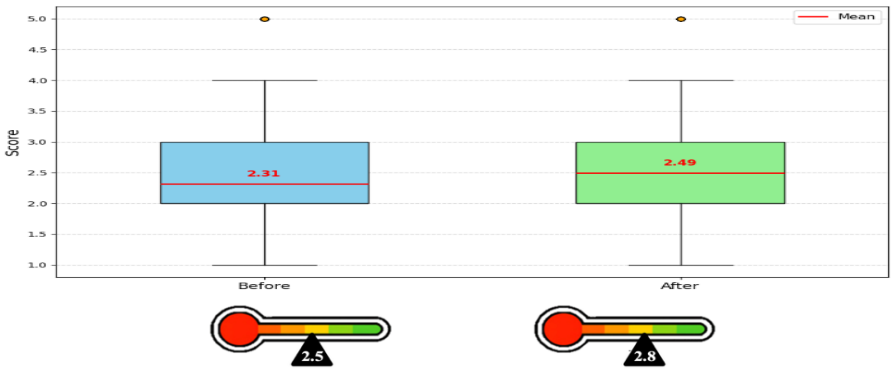


Figura 10. Redultado de la madurez
Fuente: Elaboración Propia

En la **Figura 10** se muestra que el nivel de madurez organizacional calculado aumentó de 2.5 a 2.8 tras la implementación del stack ELK según la formula del cuestionario propuesto por Zarour. Sin embargo si analizamos estadisticamente el valor medio se elevó de 2.31 a 2.49. En términos de dispersión, el rango intercuartílico permaneció sin cambios (Q1 = 2, Q3 = 3), y los valores mínimo y máximo continuaron en 1 y 5, respectivamente. Un valor de $p < 0.001$ indica una diferencia estadísticamente significativa entre las dos mediciones, lo que sugiere un cambio consistente en la distribución de las respuestas.

5. Aprendizaje: Como resultado de la capacitación, el equipo mejoró significativamente su adaptación y aceptación de la herramienta, lo que llevó a un uso más eficiente de Kibana en el entorno de trabajo. No obstante, durante la capacitación surgió un punto crítico a considerar: los logs presentan diferentes niveles de clasificación (error, advertencia, informativo), pero en algunos casos, los logs de error estaban llegando como informativos.

Para abordar este problema, se propuso realizar una revisión y depuración del código fuente, con el objetivo de reestructurar la clasificación de logs. Esto permitiría mejorar la precisión en la categorización de eventos y, a su vez, agilizar aún más la búsqueda y resolución de errores dentro del sistema.

V. RESULTADOS

Esta investigacion al ser descriptivo – preexperimental (Caso de estudio) se limita a presentar los resultados utlizando estadistica especificamente necesaria como descriptiva, tablas de frecuencia, y graficos (Roberto Hernández Sampieri et al., 2014).

5.1. Resultados de Utilidad Percibida(PU).

En la **Tabla 9** se muestra los valores utilizados para construir la tabla de frecuencias de la dimensión Utilidad Percibida a partir de las respuestas (Ver **ANEXO E**). Para determinar la distribución de los puntajes obtenidos, fue necesario calcular los parámetros estadísticos básicos que permiten construir una tabla de frecuencias.

Tabla 9. Datos para la tabla de Frecuencias de Utilidad Percibida.

Datos		Valores
Valor Minimo	=	4
Valor Maximo	=	28
Rango (R)	=	24
Número de Clase (m)	=	3
Amplitud (c)	=	8

Fuente: Elaboración propia.

A partir de los valores presentados en la **Tabla 9** valor mínimo, valor máximo, rango, número de clases y amplitud, se procedió a estructurar los intervalos que permiten agrupar los datos de manera ordenada. Con estos parámetros se elaboró la tabla de frecuencias, donde cada intervalo refleja un nivel categórico de utilidad percibida (bajo, medio y alto).

Tabla 10. Tabla de Frecuencias de Utilidad Percibida.

Niveles	Intervalos	N	%
Bajo	[4 – 12 >	0	0.0
Medio	[12 - 20 >	1	16.7
Alto	[20 – 28]	5	83.3
Total		6	100

Fuente: Elaboracion propia.

Los resultados muestran que ningún participante se ubicó en el nivel bajo de utilidad percibida, mientras que un 16.7% se encuentra en el nivel medio. La tendencia predominante señala que la mayoría de los usuarios (83.3%) se concentra en el nivel alto, lo que evidencia que la herramienta ELK es percibida como altamente útil para mejorar el desempeño en las tareas de monitoreo y análisis. Esta distribución confirma una valoración favorable de la herramienta dentro de la organización evaluada.

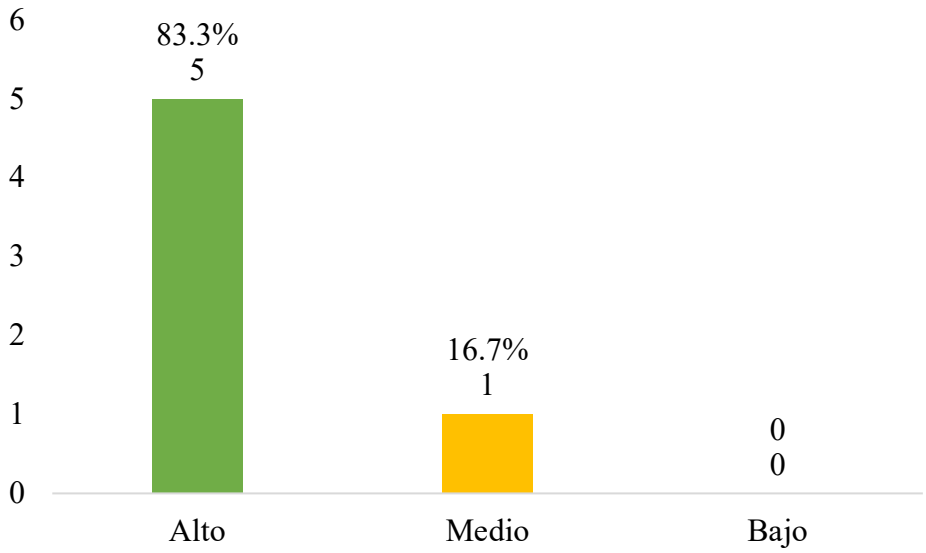


Figura 11. Gráfico de barras de la distribución de los niveles de Utilidad Percibida
Fuente: Elaboración propia.

En la **Figura 11** se confirma visualmente la marcada concentración de respuestas en el nivel alto de utilidad percibida, donde se ubica el 83.3% de los participantes. Solo un usuario (16.7%) se sitúa en el nivel medio, mientras que no se registran casos en el nivel bajo. Esta distribución gráfica refuerza la tendencia observada en la tabla de frecuencias y evidencia que la mayoría del equipo considera que la herramienta ELK aporta un beneficio significativo en su desempeño operativo.

La **Figura 11** muestra que la mayoría de usuarios se ubica en el nivel alto de utilidad percibida, sin respuestas en el nivel bajo. Esto confirma que la utilidad percibida es **alta**, validando la hipótesis.

5.2. Resultados de Facilidad de Uso (PEOU)

Para analizar la dimensión Facilidad de Uso Percibida (PEOU), primero se calcularon los parámetros estadísticos necesarios para elaborar la tabla de frecuencias en base a las respuestas del cuestionario (Ver **ANEXO E**). Estos valores permiten organizar los datos en

intervalos claros y facilitan la identificación del nivel de facilidad con el que los usuarios perciben la herramienta ELK.

Tabla 11. Datos para la tabla de frecuencias de Facilidad de Uso.

Datos		Valores
Valor Minimo	=	4
Valor Maximo	=	28
Rango (R)	=	24
Número de Clase (m)	=	3
Amplitud (c)	=	8

Fuente: Elaboración propia

Con los valores obtenidos; mínimo, máximo, rango, número de clases y amplitud, se definieron los intervalos que permiten clasificar los puntajes en los niveles bajo, medio y alto. A partir de esta estructura se construyó la tabla de frecuencias correspondiente a la dimensión de Facilidad de Uso, permitiendo observar la distribución de las percepciones de los usuarios.

Tabla 12. Tabla de Frecuencias de Facilidad de Uso

Niveles	Intervalos	N	%
Bajo	[4 – 12 >	0	0.0
Medio	[12 - 20 >	1	16.7
Alto	[20 – 28]	5	83.3
Total		6	100

Fuente: Elaboracion propia.

En la **Tabla 12** muestran que ningún usuario se ubicó en el nivel bajo de facilidad de uso, lo que indica que nadie percibe la herramienta como difícil de manejar. Solo un 16.7% se sitúa en el nivel medio, evidenciando una percepción moderada respecto a su usabilidad. En contraste, el 83.3% de los participantes se encuentra en el nivel alto, lo que refleja que la mayoría considera que ELK es una herramienta fácil de utilizar dentro de su flujo de trabajo. En conjunto, la distribución sugiere una experiencia de uso mayormente positiva y accesible para el equipo evaluado.

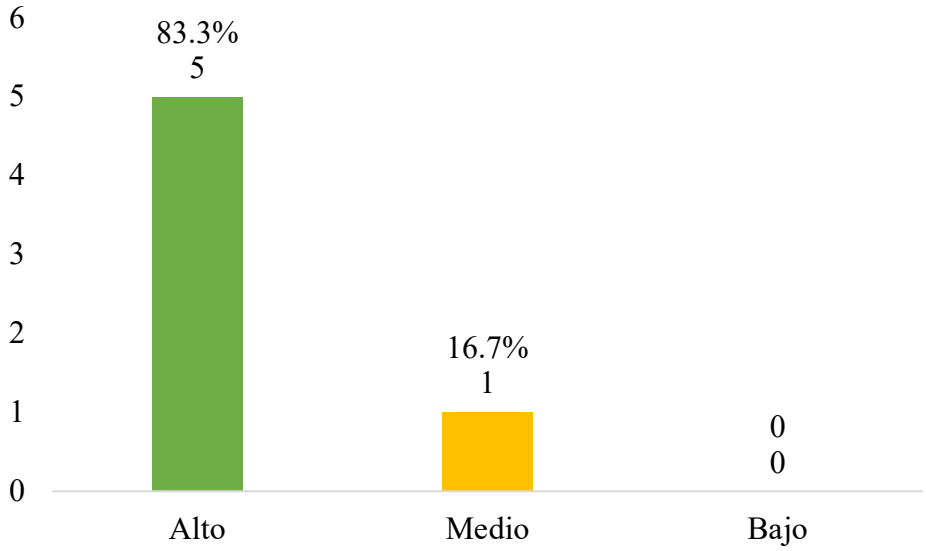


Figura 12. Gráfico de barras de la distribución de los niveles de Facilidad de Uso
Fuente: Elaboración propia.

El la **Figura 12** refuerza la tendencia observada en la tabla, mostrando una clara concentración de respuestas en el nivel alto de facilidad de uso. La presencia de un único caso en el nivel medio indica que existen percepciones ligeramente menos favorables, aunque sin llegar a catalogar la herramienta como difícil. La ausencia total de respuestas en el nivel bajo confirma que ELK es considerada, de manera general, como una herramienta intuitiva y manejable por los usuarios.

La concentración de respuestas en el nivel alto y la ausencia de niveles bajos confirman que la herramienta es considerada fácil de usar. Por ello, la hipótesis de facilidad de uso **se cumple**

5.3. Resultados Intención de Uso(BI).

Para evaluar la dimensión Intención de Uso, se calcularon los parámetros estadísticos en base a los resultados del cuestionario (Ver **ANEXO E**) que permiten organizar los puntajes obtenidos en intervalos claramente definidos. Estos valores constituyen la base para construir la tabla de frecuencias y facilitar la interpretación del nivel de disposición que tienen los usuarios para continuar utilizando la herramienta ELK.

Tabla 13. Datos para la tabla de frecuencias de Intención de Uso.

Datos		Valores
Valor Minimo	=	3
Valor Maximo	=	21
Rango (R)	=	18
Número de Clase (m)	=	3
Amplitud (c)	=	6

Fuente: Elaboración propia.

Con los valores obtenidos; mínimo, máximo, rango, número de clases y amplitud, se establecieron los intervalos correspondientes a los niveles bajo, medio y alto. Esta clasificación permitió elaborar la tabla de frecuencias que refleja cómo se distribuyen las percepciones de intención de uso entre los participantes.

Tabla 14. Tabla de Frecuencias de Intención de Uso.

Niveles	Intervalos	N	%
Bajo	[3 – 9 >	0	0
Medio	[9 - 15 >	0	0
Alto	[15 – 21]	6	100
Total		6	100

Fuente: Elaboración propia.

Los resultados de la **Tabla 14** evidencian una tendencia absoluta: el 100% de los participantes se ubica en el nivel alto de intención de uso. No se registraron casos en los niveles bajo ni medio, lo que indica que todos los usuarios muestran una disposición firme y consistente a seguir utilizando la herramienta ELK en sus actividades de monitoreo y análisis. Esta uniformidad refleja no solo satisfacción, sino también una aceptación consolidada hacia su uso continuo.

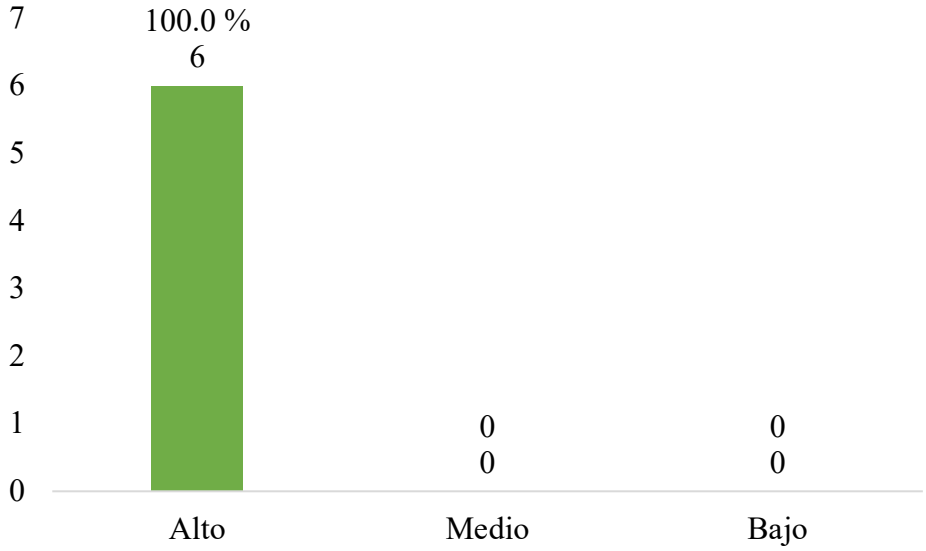


Figura 13. Gráfico de barras de la distribución de los niveles de Intención de Uso.
Fuente: Elaboración propia.

La **Figura 13** de barras refuerza esta tendencia al mostrar que la totalidad de los participantes se concentra en el nivel alto de intención de uso. La ausencia total de respuestas en los niveles bajo y medio confirma que la herramienta no genera resistencia ni dudas respecto a su continuidad. Visualmente, el gráfico evidencia una aceptación unánime, consolidando la percepción de que ELK es una herramienta que los usuarios están dispuestos a seguir utilizando. Dado que el 100% de los participantes se ubica en el nivel alto, la hipótesis se valida plenamente: la intención de uso es **alta**.

5.4. Resumen de resultados de la dimensiones.

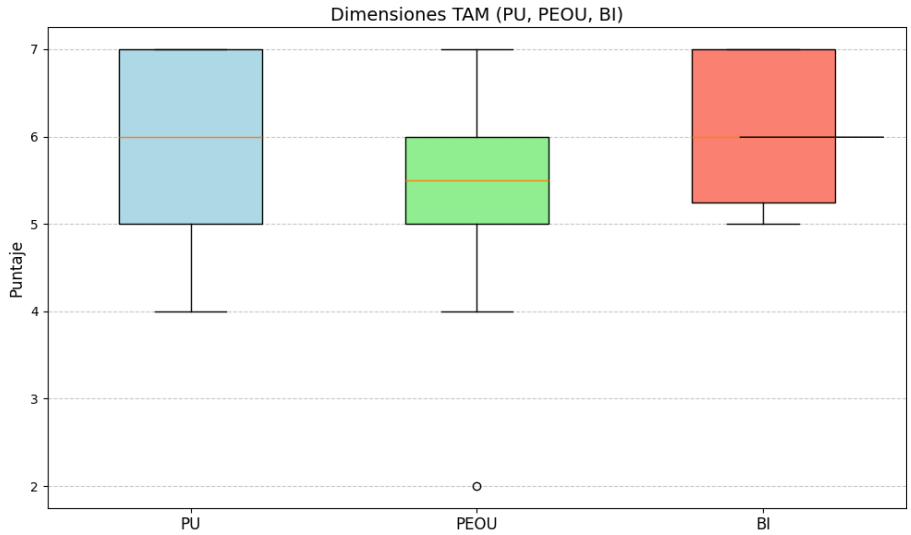


Figura 14. Resultados de las dimensiones de TAM.
Fuente: Elaboración Propia.

En la **Figura 14**, el gráfico de cajas muestra que las tres dimensiones del TAM presentan valores concentrados en la zona superior de la escala, lo que indica evaluaciones consistentemente favorables. Utilidad Percibida y Intención de Uso muestran rangos compactos sin valores extremos notorios, lo que refleja percepciones estables entre los participantes. Aunque Facilidad de Uso presenta un valor atípico hacia la parte baja, la mediana y la mayor parte de los datos permanecen en niveles altos, por lo que la tendencia general sigue siendo positiva. En conjunto, el gráfico confirma que las percepciones hacia la herramienta ELK son mayoritariamente altas en todas las dimensiones evaluadas.

El análisis estadístico de las respuestas del cuestionario TAM aplicado al equipo permitió determinar las medidas de tendencia central y dispersión para las tres dimensiones evaluadas: Utilidad Percibida (PU), Facilidad de Uso Percibida (PEOU) e Intención de Uso (BI).

Tabla 15. Resultados estadisticos de las dimensiones del Modelo de Aceptación Tegnologica.

Dimensiones	Media	Mediana	Moda	Desviación Estandar
Utilidad Percivida	5.92	6	7	0.997
Facilidad de Uso	5.38	5.5	6	1.184
Intención de Uso	6.06	6	6	0.780

Fuente: Elaboración propia.

1. Utilidad Percibida (PU)
   - Media = 5.92: Indica que los usuarios perciben que el uso del stack ELK mejora significativamente su rendimiento y productividad, posicionándose muy cerca del valor máximo (7).
   - Mediana = 6: Confirma que al menos la mitad de los encuestados asignó puntuaciones altas, lo que sugiere una percepción estable y homogénea de la utilidad.
   - Moda = 7: Refleja que el valor más frecuente fue el más alto posible, evidenciando que la mayoría de los participantes considera a ELK altamente útil dentro de su flujo de trabajo.
   - Desviación estándar = 0.997: Indica poca dispersión, es decir, que los encuestados coinciden en que la herramienta es útil y no hay grandes diferencias entre ellos.

- La utilidad percibida es claramente positiva. Los usuarios valoran que ELK facilita la identificación de fallos, acelera el diagnóstico de errores y optimiza el proceso de monitoreo, lo que demuestra una fuerte relación entre la implementación de la herramienta y el aumento de la eficiencia operativa.

2. Facilidad de Uso Percibida (PEOU)

- Media = 5.38: Señala que los usuarios perciben la herramienta como medianamente fácil de usar, pero con cierta complejidad técnica.
- Mediana = 5.5: Muestra que la mitad de los encuestados calificó alrededor del punto medio alto, lo que indica que la experiencia de aprendizaje fue satisfactoria para la mayoría, pero no para todos.
- Moda = 6: El valor más frecuente fue alto, lo que indica que muchos usuarios sí la consideran accesible y fácil de usar después de un tiempo de aprendizaje.
- Desviación estándar = 1.184: Es la más alta de las tres dimensiones, lo que sugiere mayor variabilidad en las opiniones de los participantes. Esto indica que algunos usuarios aprendieron a usar ELK rápidamente, mientras que otros tuvieron dificultades en funciones avanzadas, como crear filtros o búsquedas personalizadas.

3. Intención de Uso (BI)

- Media = 6.06: Es el valor más alto de las tres dimensiones, lo que indica una disposición a seguir utilizando la herramienta.
- Mediana = 6: Ratifica la tendencia media-alta de las valoraciones, evidenciando aceptación y satisfacción continua.
- Moda = 6: Es el valor más común, lo que indica acuerdo en la intención de uso por parte de los trabajadores.
- Desviación estándar = 0.780: Es la menor de todas, señal de alta consistencia en las respuestas; prácticamente todos los encuestados tienen la misma intención favorable de seguir utilizando ELK.

Los resultados indican que la utilidad percibida (PU) fue valorada favorablemente, ya que las respuestas se agrupan en los valores altos de la escala, lo que indica que los participantes piensan que ELK mejora su desempeño laboral. Por otro lado, la intención de uso (BI) también obtuvo una mediana alta, lo que significa que los usuarios tienen la intención de continuar

utilizando la herramienta en el futuro cercano.
Por el contrario, la facilidad de uso percibida (PEOU) presentó mayor dispersión, incluso con un valor atípico, lo que indica que algunos usuarios encontraron dificultades en la usabilidad. Esta dispersión indica que se pueden hacer mejoras en la interfaz o en las capacitaciones para reforzar la adopción completa.

Pero los gráficos muestran una opinión generalmente positiva de la herramienta, sobre todo en cuanto a lo útil que es y si la usarían. Si bien hay algunas diferencias individuales en cuanto a la facilidad de uso, éstas no afectan la aceptación tecnológica general del sistema.

5.5. Resultados del Modelo de Aceptación Tegnológica (TAM)

Para determinar el nivel global de aceptación tecnológica (TAM), se calcularon los parámetros estadísticos necesarios en base a las respuestas del cuestionario (Ver **ANEXO F**) para elaborar la tabla de frecuencias. Estos valores permiten organizar la sumatoria total de las dimensiones TAM en intervalos definidos y facilitan identificar el grado general de aceptación de la herramienta ELK en la empresa evaluada.

Tabla 16. Datos para la tabla de frecuencias del Modelo de Aceptación Tegnológica.

Datos		Valores
Valor Minimo	=	11
Valor Maximo	=	77
Rango (R)	=	66
Número de Clase (m)	=	3
Amplitud (c)	=	22

Fuente: Elaboración propia.

Con los valores obtenidos; mínimo, máximo, rango, número de clases y amplitud, se estructuraron los intervalos correspondientes a los niveles bajo, medio y alto. A partir de ello se construyó la tabla de frecuencias del TAM general, lo que permite observar de manera clara cómo se distribuyen las puntuaciones totales de aceptación tecnológica entre los participantes.

Tabla 17. Tabla de frecuencias del Modelo de Aceptación Tegnológica.

Niveles	Intervalos	N	%

Bajo	[11 - 33>	0	0.0
Medio	[33 - 55 >	1	16.7
Alto	[55 - 77]	5	83.3
Total		6	100

Fuente: Elaboración propia.

En la **Tabla 17** muestra una tendencia claramente favorable: el 83.3% de los participantes se ubica en el nivel **alto** de aceptación tecnológica, mientras que el 16.7% se encuentra en el nivel **medio**. No se registran casos en el nivel bajo, lo que indica que ningún usuario percibe una aceptación limitada. Este patrón evidencia una valoración mayoritariamente positiva respecto al uso de la herramienta ELK dentro del enfoque DevOps Reforzado.

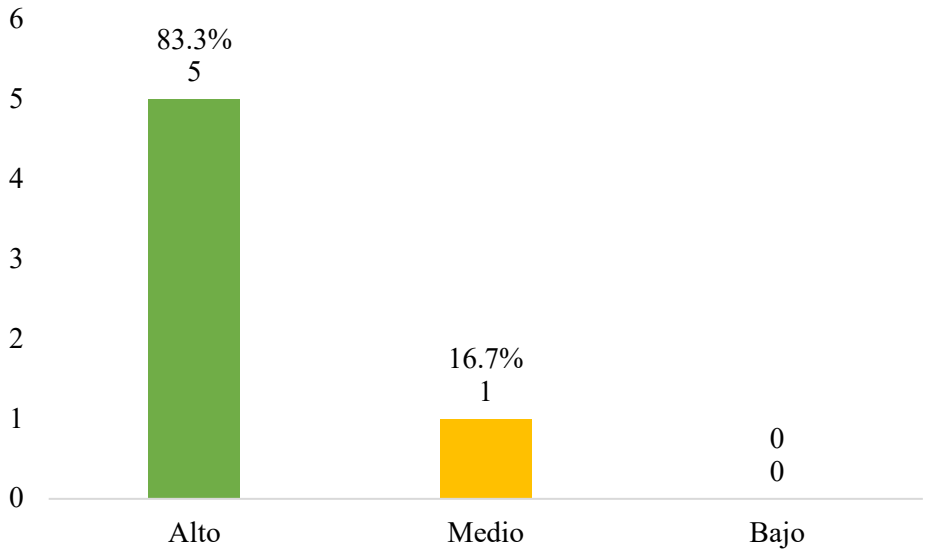


Figura 15. Gráfico de barras de la distribución de los niveles del Modelo de Aceptación Tegnológica.
Fuente: Elaboracion propia

En la **Figura 16** se confirma la predominancia del nivel alto, evidenciando que la aceptación tecnológica es ampliamente favorable entre los usuarios. La presencia de un solo caso en el nivel medio no altera la tendencia general, ya que no existen valoraciones en el nivel bajo. Visualmente, el gráfico refuerza que la mayoría del equipo demuestra una aceptación sólida del uso de la herramienta en su trabajo.

Los resultados obtenidos confirman la hipótesis general planteada, ya que el Modelo de Aceptación Tecnológica (TAM) muestra una clara predominancia de valores en el nivel alto en la mayoría de los participantes.

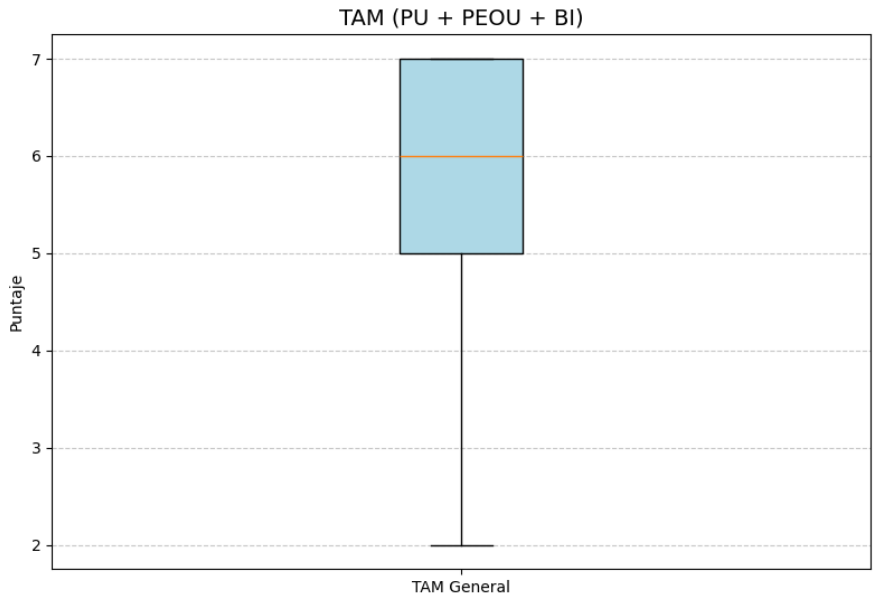


Figura 16. Resultado de TAM.
Fuente: Elaboración propia.

Asimismo, el análisis consolidado de las tres dimensiones TAM permitió generar un grafico general que resume la percepción global de los usuarios respecto al uso de ELK. Este gráfico evidencia que la mediana de los puntajes se encuentra en un nivel alto (≈6), y que la mayoría de respuestas están concentradas entre valores de 5 y 7. Si bien se observa un valor mínimo aislado, los resultados reflejan una tendencia general favorable hacia la aceptación tecnológica de la herramienta. Esta consolidación sugiere que, en términos globales, los usuarios perciben que ELK es útil, razonablemente fácil de usar y que tienen disposición a continuar utilizándolo, lo cual valida su implementación dentro del entorno operativo evaluado.

Tabla 18. Resultados estadisticos del Modelo de Aceptación Tegnologica.

Datos	TAM
Media	5.75
Mediana	6
Moda	6
Desviación Estandar	1.06

Fuente: Elaboración propia.

Los valores obtenidos muestran una aceptación tecnológica favorable, con una media de 5.75 y una mediana de 6, lo que indica que la mayoría de las valoraciones se concentran en niveles altos. La moda igual a 6 refuerza que este puntaje es el más frecuente entre los usuarios.

La desviación estándar de 1.06 evidencia una variabilidad baja a moderada, lo que significa que las percepciones son relativamente homogéneas. En conjunto, estos resultados confirman un nivel sólido y consistente de aceptación tecnológica.

5.6. Regresión de TAM

En la **Tabla 19** se presentan los promedios por dimensión. A partir de estos valores, se realizó posteriormente un análisis de regresión lineal para poder estimar en qué medida la utilidad percibida (PU) y la facilidad de uso percibida (PEOU) explican la intención de uso (BI).

Tabla 19. Promedio de la Dimensiones de TAM

PU	PEOU	->	BI
7	5.75	->	7
5.75	5.25	->	5.67
5.25	5.5	->	5.67
6	5	->	6
7	7	->	7
4.5	3.75	->	5

Fuente: Elaboración propia

Como se plantea en el modelo TAM, se realizó el siguiente análisis de regresión, considerando como variable dependiente a BI y como variables independientes a PU y PEOU.

Tabla 20. Estadísticos del modelo de regresión lineal múltiple (PU y PEOU → BI)

Estadísticas de la regresión	
Coeficiente de correlación múltiple	0.983
Coeficiente de determinación R^2	0.9660
R^2 ajustado	0.943
Error típico	0.191
Observaciones	6.000

Fuente: Elaboración propia

En la **Tabla 20** se observa que el coeficiente de correlación múltiple (R) es 0.983, lo cual implica que existe una relación muy fuerte entre los valores reales de la intención de uso (BI) y los valores de BI estimados por el modelo usando la utilidad percibida (PU) y la facilidad de uso percibida (PEOU).

Asimismo, el coeficiente de determinación ($R^2 = 0.966$) indica que el 96.6% de la intención de uso (BI) se explica a partir de PU (qué tan útil se percibe ELK) y PEOU (qué tan fácil se percibe ELK) de manera conjunta. Además, el R^2 ajustado (0.943) confirma que el modelo mantiene un ajuste alto incluso considerando el tamaño muestral. Finalmente, el error típico (0.191) sugiere que el modelo presenta un nivel bajo de error al estimar BI.

Tabla 21. Análisis de varianza (ANOVA) del modelo de regresión TAM (PU y PEOU → BI)

Análisis de Varianza					
	Grados de libertad	Suma de cuadrados	Promedio de los cuadrados	F	Valor crítico de F
Regresión	2.000	3.095	1.547	42.627	0.006
Residuos	3.000	0.109	0.036		
Total	5.000	3.204			

Fuente: Elaboración propia

En la **Tabla 21** se observa que el estadístico F es 42.627 y que la significancia del modelo ($p = 0.006$) es menor a 0.05. Esto implica que el modelo de regresión es globalmente significativo, es decir, que la utilidad percibida (PU) y la facilidad de uso percibida (PEOU), consideradas en conjunto, explican la intención de uso (BI) de manera estadísticamente detectable en la muestra evaluada.

Tabla 22. Coeficientes e inferencia del modelo de regresión (PU y PEOU → BI

	Coeficientes	Error típico	Estadístico t	Probabilidad	Inferior 95%	Superior 95%
Intercepción	1.327	0.519	2.560	0.083	-0.323	2.978
PU	0.727	0.160	4.532	0.020	0.216	1.237
PEOU	0.080	0.149	0.534	0.630	-0.395	0.554

Fuente: Elaboración propia

En la **Tabla 22** se observa que la utilidad percibida (PU) presenta un coeficiente de 0.727 con un valor de significancia $p = 0.020$, lo cual implica que PU tiene un efecto positivo y significativo sobre la intención de uso (BI). En términos prácticos, esto significa que por cada punto que aumenta PU, la intención de uso BI aumenta aproximadamente 0.73 puntos, manteniendo constante la facilidad de uso percibida.

En contraste, la facilidad de uso percibida (PEOU) presenta un coeficiente de 0.080 con un valor p = 0.630, lo cual implica que no se evidencia un efecto directo significativo de PEOU sobre BI en esta muestra. Por tanto, la intención de uso se encuentra principalmente asociada a la percepción de utilidad.

VI. DISCUSIONES

Los hallazgos de esta investigación corroboran lo que señalan Muñoz (Muñoz et al., 2021), en que fortalecer DevOps con herramientas enmarcadas en ISO/IEC 29110 mejora la estabilidad operativa y la trazabilidad. En este caso, el 83.3% de los usuarios calificó la utilidad percibida como alta, confirmando que ELK sí contribuye en la práctica al monitoreo, como proponen estos autores en su plataforma para fortalecer DevOps.

Además, los resultados confirman lo que demostró Miguel (Miguel et al., 2024), que ELK mejora significativamente la detección y el análisis de errores en escenarios reales. En la empresa intervenida, la adopción del Stack ELK permitió reducir el tiempo de diagnóstico en aproximadamente 75%, lo que concuerda con la evidencia empírica sobre la eficiencia de ELK en la centralización de logs y la observabilidad operativa.

Para el Modelo TAM, los resultados concuerdan con lo que plantea Reyes (Reyes & Castañeda, 2020), ya que la utilidad percibida es lo que más influye en la intención de uso. En este estudio, la utilidad percibida alcanzó una **media de 5.92**, mientras que la intención de uso obtuvo una **media de 6.06** y un **100% de respuestas en nivel alto**, confirmando el patrón teórico del TAM y reforzando el papel de la utilidad como principal predictor del uso futuro de la herramienta.

Respecto a la facilidad de uso, nuestros resultados **coinciden parcialmente** con los de Kamal (Kamal et al., 2020), quienes identificaron que la facilidad de uso influye de forma significativa, pero puede presentar variabilidad dependiendo del nivel técnico del usuario. En esta investigación se observó una mayor dispersión en PEOU (desviación estándar = 1.184), incluyendo un valor atípico hacia la zona baja. Esta variabilidad **corrobora** la necesidad de capacitación inicial, tal como lo indican estos autores para contextos donde la curva de aprendizaje puede ser pronunciada.

Finalmente, la aceptación general del TAM con una **media global de 5.75** y **83.3% de usuarios en nivel alto coincide** con los resultados de Wang (Wang et al., 2023), quienes evidenciaron que herramientas que generan valor directo y ayudan a mejorar el desempeño diario presentan altas tasas de aceptación conductual. Esto confirma que la adopción de ELK no solo mejora la madurez DevOps, sino que también facilita una aceptación tecnológica sólida y sostenida, tal como lo predice la teoría.

Estudios como los de Muñoz muestran que la implementación de ISO/IEC 29110 refuerza procesos sin alterar la forma de trabajo, incrementa la calidad y reduce retrabajos en equipos ágiles (Muñoz et al., 2020). Del mismo modo, Díaz reportan que las organizaciones certificadas mejoran la gestión de tiempos, control de versiones e indicadores (Díaz & Muñoz, 2017).

Nuestro estudio coincide con estos resultados, puesto que la integración de herramientas de monitoreo (ELK) permitió fortalecer las actividades de trazabilidad, control y verificación dentro del proceso DevOps Reforzado. Los participantes percibieron mejoras en visibilidad, control operativo y estandarización, lo cual es coherente con los beneficios observados en las implementaciones previas a nivel internacional.

VII. CONCLUSIONES

La investigación logró demostrar que el uso del stack ELK en el marco de DevOps Reforzado logró obtener alta aceptación tecnológica, demostrando que este tipo de herramientas son adecuadas para pequeñas empresas con bajos recursos. La replicabilidad de los resultados en las tres variables del modelo TAM prueba que ELK fue aceptado y que satisface las condiciones teóricas que predicen la adopción tecnológica.

La utilidad percibida (media = 5.92; 83.3% alto) muestra que los usuarios son conscientes de que ELK mejora su rendimiento, disminuye el tiempo de diagnóstico y ayuda en la toma de decisiones. Tras estos resultados, se puede inferir que la herramienta influye directamente en la eficiencia de los equipos, convirtiéndose en un elemento para fortalecer las prácticas de observabilidad y monitoreo en el entorno DevOps.

La intención de uso arrojó una tendencia totalmente positiva (M = 6.06; 100% en nivel alto). Según el modelo TAM, esta variable representa la intención futura del individuo de adoptar una tecnología. Sobre la base de estos resultados, se puede inferir que la aceptación obtenida no es efímera ni ligada a la intervención, sino que los usuarios están dispuestos a integrar ELK en su práctica laboral de forma permanente. Esto certifica su continuidad en el mundo DevOps.

Aunque la facilidad de uso presentó mayor variabilidad (desviación estándar = 1.184) y algunos usuarios reportaron dificultad inicial en funciones avanzadas de Kibana, la valoración general fue positiva. En función de esto, se concluye que la curva de aprendizaje inicial no representa una barrera crítica para la adopción, pero sí evidencia la necesidad de acompañar la implementación con capacitación práctica para garantizar una experiencia uniforme entre los miembros del equipo.

De acuerdo con el modelo TAM, la intención de uso (BI) puede explicarse a partir de la utilidad percibida (PU) y la facilidad de uso percibida (PEOU). En esta investigación se observó que la utilidad percibida ejerce una mayor influencia sobre la intención de uso, lo cual se entiende porque ELK es valorado principalmente por mejorar el desempeño y apoyar el trabajo. En contraste, la facilidad de uso tendría menor impacto directo debido a que la herramienta requiere una curva de aprendizaje para su uso eficiente.

Las tres dimensiones del modelo TAM son internamente consistentes y muestran una aceptación tecnológica global positiva. La utilidad percibida resultó ser el principal predictor, seguido de la intención de uso y la facilidad de uso percibida. De este comportamiento se infiere que la adopción de ELK se basa en los beneficios prácticos que aporta al día a día, en línea con las bases teóricas del modelo y corroborando su uso en entornos DevOps.

Se determina que al aplicar DevOps Reforzado puede impactar positivamente en la calidad del software, estandarizando los procesos de desarrollo y despliegue a través de la automatización, el control de cambios y la retroalimentación continua para reducir errores, mejorar la mantenibilidad y aumentar la confianza en producción. Sin embargo, este enfoque por sí solo no es una garantía de la calidad del producto; asegurar la calidad implica otros factores adicionales que deben ser tomados en cuenta.

Por otro lado, la integración del Stack ELK fortalece la calidad desde la capa operativa, proporcionando observabilidad y monitoreo (centralización, análisis y visualización de logs) para identificar fallas tempranas y acelerar el diagnóstico y resolución de problemas. Como resultado, los tiempos de recuperación se acortan y el servicio se vuelve más estable. Así, ambos enfoques contribuyen a una entrega más estable y controlada, generando productos más confiables, rastreables y sostenibles en el tiempo.

Esta investigación proporciona evidencia empírica sobre los impactos de la adopción de instrumentos de seguimiento en microempresas tecnológicas. A diferencia de estudios anteriores que proponen modelos teóricos, aquí se evidencia que la formalización del proceso, al ser implementada, genera mejoras reales en la madurez, trazabilidad y adopción tecnológica. Esta evidencia apoya la literatura de DevOps Reforzado en entornos de recursos escasos.

Se determina que el marco DevOps Reforzado, al combinar las prácticas DevOps con una guía de proceso ajustada a pequeñas organizaciones, es un marco potencial para fortalecer la implementación de fases cruciales como el monitoreo y la mejora continua. Además, el uso del Modelo de Aceptación Tecnológica (TAM) demostró que la adopción de las herramientas planteadas está supeditada al valor que perciban que añaden al trabajo del equipo; en este caso, la aceptación tecnológica encontrada sugiere que, al acompañar DevOps Reforzado con herramientas que generen beneficios operativos reales, el equipo estará más dispuesto a adoptarlas y mantenerlas en el tiempo.

La metodología de Investigación-Acción permitió la mejora iterativa, la detección temprana de problemas y el aprendizaje organizacional. Tras su implementación, se determina que esta metodología es idónea para las pymes, ya que permite realizar mejoras sin detener la producción y fomenta una adopción progresiva y controlada.

Si bien los resultados fueron positivos, el estudio se llevó a cabo en una única empresa, lo que limita su generalización. Es aconsejable replicar el estudio en otros contextos e incluir otros constructos del TAM (como confianza o riesgo percibido) para profundizar en los factores que influyen en la adopción tecnológica en entornos DevOps.

Se determina que en el DevOps Reforzado se cumplieron las tareas definidas para el proceso de Monitoreo (Ver **ANEXO A**). En específico, se eligieron e instalaron las herramientas de monitoreo, las cuales permitieron establecer una supervisión más controlada y continua del entorno.

Dado que el alcance de esta investigación se limitó a evaluar la aceptación de la herramienta implementada, no se analizó su sostenibilidad o continuidad de uso en el largo plazo. Sin embargo, los resultados obtenidos, en los que la utilidad percibida explica la intención de uso, permiten inferir que existe una alta probabilidad de que la herramienta continúe utilizándose a lo largo del tiempo, debido al valor que aporta al trabajo del equipo.

VIII. TRABAJOS A FUTURO

Con respecto a los trabajos futuros, se podría realizar la certificación ISO/IEC 29110 para determinar en qué medida la empresa está preparada para dicha certificación y evaluar si la implementación ha sido rigurosa o no.

Otro trabajo futuro podría consistir en replicar el mismo experimento, pero utilizando otro conjunto de herramientas, para analizar si la edad de los trabajadores influye en la aceptación de las mismas. Asimismo, sería pertinente investigar los años de experiencia de los empleados y cómo estos podrían afectar su receptividad hacia las herramientas.

La replicación de esta investigación dependerá, en primer lugar, del nivel de madurez que se mida al inicio, ya que dichos resultados permitirán identificar qué procesos la empresa necesita fortalecer y, en consecuencia, automatizar. No obstante, también es posible que otras organizaciones presenten falencias específicamente en la etapa de monitoreo; en ese escenario, la replicabilidad sería más amplia. Pero habría que analizar si la herramienta utilizada en esta investigación (Stack ELK) se adapta igual de bien al contexto, necesidades y características técnicas de cada empresa.

También se podría realizar una investigación de tipo experimental para control total y administrar todas las tareas DevOps con herramientas propuestas por el investigador. Esto haría posible conocer cómo interactúan las prácticas DevOps con la adopción de herramientas específicas en el entorno de trabajo.

Como el investigador formó parte del equipo de implementación, puede existir un sesgo en las respuestas de percepción de aceptación tecnológica. Por lo cual, es recomendable replicar la investigación incluyendo un agente externo que sea el encargado de implementar la herramienta, y el investigador se encargue de medir la variable y analizar los resultados, para hacer más objetivo el estudio.

IX. REFERENCIAS

7. SO/IEC TR 29110-1:2016(es), Ingeniería de Software y Sistemas — Perfiles de ciclo de vida para Pequeñas Organizaciones (VSEs) — Parte 1: Visión general. (s. f.). Recuperado 2 de julio de 2023, de https://www.iso.org/obp/ui#iso:std:iso-iec:tr:29110:-1:ed-2:v1:es

Abarca, M., Arisaca, R., & Dávila, A. (2015). Implementación del Perfil Básico de la ISO/IEC 29110 de una Pequeña Empresa Desarrolladora de Software: Lecciones Aprendidas. En Ibero-American Conference on Software Engineering (Ed.), *18th Ibero-American Conference on Software Engineering, CIBSE 2015* (pp. 776-787).

Acevedo, D., Muñoz, M., Galvan, S., Mejía, J., & Robles, B. (2025). *Toward the Implementation of DevOps: A Guide of Tools, Practices and Activities* (pp. 106-123). https://doi.org/10.1007/978-3-032-04288-0_7

Alamin, Z., Dahlan, Khaeruddin, & Sahrul Ramadhan. (2025). Evolving DevOps Practices in Modern Software Engineering: Trends, Challenges, and Impacts on Quality and Delivery Performance. *Journix: Journal of Informatics and Computing*, *1*(1), 21-29. https://doi.org/10.63866/journix.v1i1.4

Amazon. (2016, febrero 26). *¿Qué es la entrega continua? – Amazon Web Services*. https://aws.amazon.com/es/devops/continuous-delivery/

Amazon. (2023, agosto 24). *¿Qué es la infraestructura como código? - Explicación de IaC - AWS*. https://aws.amazon.com/es/what-is/iac/

Atlassian. (2021, agosto 31). *¿En qué consiste la integración continua? | Atlassian*. https://www.atlassian.com/es/continuous-delivery/continuous-integration

Atlassian. (2023, mayo 10). *Continuous integration vs. delivery vs. deployment*. https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment

Atlassian. (2024). *Automatización de pruebas de DevOps | Atlassian*. https://www.atlassian.com/es/devops/devops-tools/test-automation

Davis, F. (1985). *A Technology Acceptance Model for Empirically Testing New End-User Information Systems*.

Díaz, O., & Muñoz, M. (2017). Reinforcing DevOps approach with security and risk management: An experience of implementing it in a data center of a mexican organization. *2017 6th International Conference on Software Process Improvement (CIMPS)*, 1-7. https://doi.org/10.1109/CIMPS.2017.8169957

Felipe Redondo, A. M., & Núñez Cárdenas, F. de J. (2022). DevOps: un vistazo rápido. *Ciencia Huasteca Boletín Científico de la Escuela Superior de Huejutla*, *10*(19), 35-40. https://doi.org/10.29057/esh.v10i19.8121

García, J., Minero, J. J., & Lara, E. (2023). Automatización de los procesos del estándar ISO/IEC 29110 a través de la Adopción de DevOps. *RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação*, *2023*(49), 37-51. https://doi.org/10.17013/risti.49.37-51

Hamza, U., Abdullah, N. L., & Syed-Mohamad, S. M. (2024). DevOps Adoption Guidelines, Challenges and Benefits: A Systematic Literature Review. *Journal of Advanced Research in Applied Sciences and Engineering Technology*, *61*, 114-136. https://doi.org/10.37934/araset.63.2.114136

IBM. (2022, mayo 15). *¿Qué es el despliegue continuo? | IBM*. https://www.ibm.com/es-es/topics/continuous-deployment

IBM. (2022, septiembre 29). *Infrastructure as Code | IBM*. https://www.ibm.com/topics/infrastructure-as-code

IBM. (2023, noviembre 4). *What is continuous delivery? | IBM*. https://www.ibm.com/topics/continuous-delivery#What+is+continuous+delivery%3F

Kamal, S. A., Shafiq, M., & Kakria, P. (2020). Investigating acceptance of telemedicine services through an extended technology acceptance model (TAM). *Technology in Society*, *60*. https://doi.org/10.1016/j.techsoc.2019.101212

Lwakatare, L. E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., Kuvaja, P., Mikkonen, T., Oivo, M., & Lassenius, C. (2019). DevOps in practice: A multiple case study of five companies. *Information and Software Technology*, *114*, 217-230. https://doi.org/10.1016/j.infsof.2019.06.010

Macarthy, R. W., & Bass, J. M. (2020). An Empirical Taxonomy of DevOps in Practice. *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 221-228. https://doi.org/10.1109/SEAA51224.2020.00046

Microsoft. (2022, septiembre 27). *What is infrastructure as code (IaC)? - Azure DevOps | Microsoft Learn*. https://learn.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code

Microsoft. (2023, mayo 11). *Seven best practices for Continuous Monitoring with Azure Monitor | Microsoft Azure Blog*. https://azure.microsoft.com/en-us/blog/7-best-practices-for-continuous-monitoring-with-azure-monitor/

Microsoft Learn. (2025, octubre 27). *¿Qué es DevOps? - Azure DevOps | Microsoft Learn*. https://learn.microsoft.com/es-es/devops/what-is-devops

Miguel, J., Mondragón, R., De Luis, E., Jordi, G., & Ruiz, S. (s. f.). *SOC-ELK*.

Minero, J. J., Garcia, J., & Lara, E. (2020). Evaluation of the Implementation of the ISO/IEC 29110 Standard at the Software Development Center from the Institute Technological Superior of Nochistlán. *2020 9th International Conference On Software Process Improvement (CIMPS)*, 12-18. https://doi.org/10.1109/CIMPS52057.2020.9390105

Munoz, M., & Montoya-Mendez, P. (2021). Identification of issues in the implementation of the ISO/IEC 29110 standard: comparison between the state of the art and the state of practice. *2021 10th International Conference On Software Process Improvement (CIMPS)*, 90-97. https://doi.org/10.1109/CIMPS54606.2021.9652708

Muñoz, M., Mejia, J., & Laporte, C. Y. (2020). Implementing ISO/IEC 29110 to reinforce four very small entities of Mexico under an agile approach. *IET Software*, *14*(2), 75-81. https://doi.org/10.1049/iet-sen.2019.0040

Muñoz, M., & Negrete, M. (2020). Reinforcing DevOps Generic Process with a Guidance Based on the Basic Profile of ISO/IEC 29110. En *Advances in Intelligent Systems and Computing* (Vol. 1071, pp. 65-79). Springer. https://doi.org/10.1007/978-3-030-33547-2_6

Muñoz, M., Negrete, M., & Arcilla-Cobián, M. (2021). Using a platform based on the Basic profile of ISO/IEC 29110 to reinforce DevOps environments. *JUCS - Journal of Universal Computer Science*, *27*(2), 91-110. https://doi.org/10.3897/jucs.65080

Muñoz, M., & Rodríguez, M. N. (2021). A guidance to implement or reinforce a DevOps approach in organizations: A case study. *Journal of Software: Evolution and Process*. https://doi.org/10.1002/smr.2342

Paez, N. (2018). Versioning Strategy for DevOps Implementations. *2018 Congreso Argentino de Ciencias de la Informática y Desarrollos de Investigación (CACIDI)*, 1-6. https://doi.org/10.1109/CACIDI.2018.8584362

Red Hat. (2023). *La automatización de DevOps*. https://www.redhat.com/es/topics/automation/what-is-devops-automation#lo-que-se-puede-automatizar

Reyes, M., & Castañeda, P. (2020). Aplicación del Modelo de Aceptación Tecnológica en Sistemas de Información de la Administración Pública del Perú. *Revista peruana de computación y sistemas*, *3*(1), 15-22. https://doi.org/10.15381/rpcs.v3i1.18350

Roberto Hernández Sampieri, Carlos Fernández Collado, & María del Pilar Baptista Lucio. (2014). *Metodologia de la Investigacion* (Vol. 6).

Sentrio. (2022). *Herramientas de automatización en DevOps*. https://sentrio.io/blog/herramientas-de-automatizacion-en-devops/

Terron, V., & Mejía, J. (2023). Propuesta de mejoras para la implementación del estándar ISO/IEC 29110 utilizando lógica matemática. *RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação*, *2023*(49), 52-66. https://doi.org/10.17013/risti.49.52-66

TRBL-SERVICES. (2021, marzo 21). *Introducción a DevOps. Qué es y cómo implementarlo*. https://trbl-services.eu/blog-introduccion-devops/

Wang, C., Ahmad, S. F., Bani Ahmad Ayassrah, A. Y. A., Awwad, E. M., Irshad, M., Ali, Y. A., Al-Razgan, M., Khan, Y., & Han, H. (2023). An empirical evaluation of technology acceptance model for Artificial Intelligence in E-commerce. *Heliyon*, *9*(8), e18349. https://doi.org/10.1016/j.heliyon.2023.e18349

Zarour, M., Alhammad, N., Alenezi, M., & Alsarayrah, K. (2020a). DEVOPS PROCESS MODEL ADOPTION IN SAUDI ARABIA: AN EMPIRICAL STUDY. *Jordanian Journal of Computers and Information Technology*, *0*, 1. https://doi.org/10.5455/jjcit.71-1580581874

X. ANEXOS

ANEXO A. Tareas del DevOps Reforzado según procesos
Tareas para el proceso de Requisitos

Actividad	Rol	ID	Tareas
I1. REQUISITOS	PM, AN	I.1.1	Revisar la Declaración de Trabajo o el contrato que especifica la comprensión del proyecto
	PO, PM	I.1.2	Definir con las partes interesadas las instrucciones de entrega de cada uno de los entregables especificados en la Declaración de Trabajo
	PO, PM, AN	I.1.3	Documentar o actualizar las historias de usuario. Identificar y consultar fuentes de información como partes interesadas (entrevistas), sistemas anteriores, documentos, etc. Analizar las historias de usuario identificadas para determinar su viabilidad y el alcance del proyecto. Generar o actualizar el backlog del producto
	PM, AN	I.1.4	Identificar las tareas específicas y configurar las tareas a realizar para producir los entregables y los componentes de software identificados en la Declaración de Trabajo o contrato. Identificar las tareas para realizar las instrucciones de entrega y documentar las tareas
	PO, AN	I.1.5	Realizar la sesión de refinamiento del backlog si es necesario para definir qué historias de usuario se esperan en el próximo sprint y comunicar las incertidumbres sobre las historias de usuario que no están claras. El propietario del producto puede mejorar las historias de usuario según sea necesario
	PM, AN	I.1.6	Identificar y documentar los recursos: humanos, materiales, equipos y herramientas, estándares, incluida la capacitación requerida del equipo de trabajo para llevar a cabo el proyecto. Incluir en el cronograma las fechas en las que se necesitarán los recursos y la capacitación.
	PM, AN	I.1.7	Establecer la composición del equipo de trabajo asignando roles y responsabilidades de acuerdo con los recursos para definir la organización jerárquica orientada al flujo de valor en el mercado
	PM, DT, QA, IS, GK, AN,RE, DE	I.1.8	Definir la complejidad y el tiempo de las historias de usuario utilizando una técnica de estimación entre el equipo. Se recomienda utilizar técnicas como el Planning Poker para estimar

	PM, AN	I.1.9	Asignar fechas estimadas de inicio y finalización a cada una de las tareas del backlog del producto con el fin de crear el cronograma de las tareas del proyecto teniendo en cuenta los recursos asignados, el trabajo no planificado, la deuda técnica, la secuencia y dependencia de las tareas. Además, se recomienda acordar un objetivo compartido entre todo el equipo para alcanzar en el sprint
	PO, PM	I.1.10	Calcular y documentar el esfuerzo estimado y el costo del proyecto utilizando puntos de función
	PM, AN	I.1.11	Verificar y obtener la aprobación del backlog del producto. Verificar la corrección y la testabilidad del backlog del producto y su consistencia con la descripción del producto. Además, revisar que las historias de usuario sean completas, no ambiguas y no contradictorias con los criterios de aceptación. Los resultados encontrados se documentan en los resultados de verificación y se realizan correcciones hasta que el backlog del producto sea aprobado por el analista. Si se necesitan cambios significativos, se inicia una solicitud de cambio

Fuente : (Muñoz & Negrete, 2020)

Tareas para el proceso de Planificación

Actividad	Rol	ID	Tareas
I2. PLANIFICACIÓN	PM, DT, QA, OT, IS, GK, AN, RE, DE	I.2.1	Reducir la deuda técnica y el trabajo no planificado teniendo en cuenta las tareas del sprint actual. Si hay tickets pendientes en el tablero de impedimentos
	PM, AN	I.2.2	Identificar y documentar los riesgos que pueden afectar al proyecto
	PM, AN	I.2.3	Definir el control de versiones entre el equipo en cada componente y entregable utilizando técnicas de control de versiones. Se recomienda encarecidamente aplicar una nomenclatura de cambios de fase, mayores y menores durante el proyecto en el plan del proyecto
	PM, AN	I.2.4	Generar el plan del proyecto integrando los elementos previamente identificados y documentados
	PM, AN	I.2.5	Incluir la descripción del producto, alcance, objetivos y entregables en el plan del proyecto

	PO. PM, DT, QA, OT, IS, GK, AN, RE, DE	I.2.6	Verificar el plan del proyecto actual con los miembros del equipo con el fin de lograr una comprensión común y obtener su compromiso con el proyecto para obtener la aprobación del plan del proyecto. Los resultados encontrados se documentan en un informe de verificación y se realizan correcciones hasta que el documento sea aprobado por el responsable del proceso
	PM, AN	I.2.7	Documentar la versión preliminar de la documentación de usuario del software o actualizar la existente, si corresponde
	PM, AN	I.2.8	Verificar y obtener la aprobación de la documentación de usuario del software, si corresponde. Verificar la consistencia de la documentación de usuario del software con el backlog del producto. Los resultados encontrados se documentan en los resultados de verificación y se realizan correcciones hasta que la documentación sea aprobada

Fuente : (Muñoz & Negrete, 2020)

Tareas para el proceso de Desarrollo

Actividad	Rol	ID	Tareas
C1. DESARROLLO	DE	C.1.1	Generar archivos de configuración si es necesario aplicarlos en servidores, contenedores y herramientas. Se recomienda encarecidamente crear plantillas de acuerdo con el proyecto para utilizar plantillas definidas y ahorrar tiempo en la configuración
	DE	C.1.2	Crear cualquier otro script o información de configuración requerida para crear la infraestructura de software
	DE	C.1.3	Crear los scripts de migraciones y seeders de la base de datos necesarios para crear la base de datos o actualizarla si es necesario
	DE	C.1.4	Ejecutar los scripts para mantener la base de datos actualizada
	DT, OP	C.1.5	Definir el desarrollo basado en tronco (trunk-based development) utilizando diferentes entornos y trabajando en pequeños lotes. Se recomienda encarecidamente utilizar diferentes ramas (branches) para tener un mejor control, como mínimo, las ramas master, develop y test.
	DT, OT, DE	C.1.6	Establecer un entorno que admita la integración continua. Se recomienda encarecidamente utilizar un servidor de integración continua para definir la configuración en un archivo específico. Añadir una forma de prevenir problemas y recuperar el servicio en caso de fallos sin demoras

	DT, OT, DE	C.1.7	Establecer un entorno que admita la automatización del despliegue. Se recomienda encarecidamente utilizar un servidor de integración continua para definir la configuración en un archivo específico
	DT, OT, DE	C.1.8	Establecer un entorno que admita la automatización de la entrega. Se recomienda encarecidamente utilizar un servidor de integración continua para definir la configuración en un archivo específico
	DT, OT, DE	C.1.9	Establecer un entorno que admita la automatización de la liberación. Se recomienda encarecidamente utilizar un servidor de integración continua para definir la configuración en un archivo específico

Fuente : (Muñoz & Negrete, 2020)

Tareas para el proceso de Construcción

Actividad	Rol	ID	Tareas
C.2 CONSTRUCCIÓN	DE.OP	C.2.1	Crear imágenes de máquinas virtuales o contenedores preconfigurados utilizando los archivos de configuración si existen. Se recomienda encarecidamente utilizar contenedores para mantener la misma configuración en todos los entornos, en cualquier dispositivo, y no cambiar o introducir variables desconocidas en la configuración actual
	PM	C.2.2	Asegurarse de que cada entorno tenga la misma configuración. Se recomienda encarecidamente tener una máquina virtual para reducir el tiempo de configuración.
	DE,OP	C.2.3	Definir la estrategia de seguridad a aplicar en contenedores o en el entorno necesario. Se recomienda utilizar técnicas como middleware, autenticación de dos factores o cualquier técnica necesaria.
	DE,OP	C.2.4	Establecer o actualizar la estrategia de seguridad en el entorno a utilizar en el proyecto.
	DE, OT	C.2.5	Construct or update software components based on the detailed part of the software design and following the acceptance criteria
	PM, AN	C.2.6	Monitor the project plan execution and record actual data in progress status record

	PO, PM, AN	C.2.7	Realizar reuniones de revisión con las partes interesadas, registrar los acuerdos y darles seguimiento hasta su finalización. Las solicitudes de cambio iniciadas por las partes interesadas o por el equipo de trabajo, que afecten a las partes interesadas, deben ser negociadas para llegar a un acuerdo aceptado por ambas partes. Si es necesario, actualizar el plan del proyecto de acuerdo con el nuevo acuerdo con las partes interesadas
	PO, PM, AN	C.2.8	Definir un proceso de aprobación de cambios que analice y evalúe la solicitud de cambio en cuanto a su impacto en costos, plazos e impacto técnico. La solicitud de cambio puede ser iniciada tanto por las partes interesadas externas como internamente por el equipo. Actualizar el plan del proyecto si el cambio aceptado no afecta los acuerdos con las partes interesadas. Si la solicitud de cambio afecta dichos acuerdos, será necesario negociarla entre ambas partes y se agregará al backlog del producto para ser considerada en el próximo sprint
	PM	C.2.9	Realizar reuniones de revisión con el equipo de control de calidad, seguridad de la información, analistas, ingenieros de confiabilidad y el equipo de trabajo. Identificar problemas, revisar el estado de los riesgos, registrar acuerdos y darles seguimiento hasta su finalización.

Fuente : (Muñoz & Negrete, 2020)

Tareas para el proceso de Pruebas

Actividad	Rol	ID	Tareas
C.3 PRUEBAS	DT	C.3.1	Establecer o actualizar casos de prueba y procedimientos de prueba para las pruebas de integración y agregarlos a la automatización de pruebas basada en el backlog del producto y el diseño del software. El stakeholder proporciona datos de prueba si es necesario.
	PM, AN	C.3.2	Verificar y obtener la aprobación de los casos de prueba y los procedimientos de prueba. Verificar la consistencia entre el backlog del producto, el diseño del software y los casos de prueba y los procedimientos de prueba. Los resultados encontrados se documentan en los resultados de verificación y se realizan correcciones hasta que el documento sea aprobado por el analista
	DT	C.3.3	Comprender los casos de prueba y los procedimientos de prueba. Establecer o actualizar el entorno de pruebas.
	AN	C.3.4	Actualizar el registro de trazabilidad incorporando los casos de prueba y los procedimientos de prueba

	PM	C.3.5	Incorporar el diseño de software y el registro de trazabilidad en la configuración del software como parte de la línea base. Incorporar los casos de prueba y los procedimientos de prueba al repositorio del proyecto
	DT	C.3.6	Diseñar o actualizar casos de prueba unitarios y aplicarlos para verificar que los componentes de software implementen la parte detallada del diseño de software.
	DT.DE	C.3.7	Establecer un entorno que admita la automatización de pruebas. Se recomienda encarecidamente utilizar un servidor de integración continua para definir la configuración en un archivo específico
	DT, OT	C.3.8	Integrar el software utilizando componentes de software y actualizar los casos de prueba y los procedimientos de prueba para las pruebas de integración, según sea necesario.
	DT, AN	C.3.9	Realizar pruebas de software utilizando casos de prueba y procedimientos de prueba para las pruebas de integración y documentar los resultados en el informe de pruebas
	DT	C.3.10	Corregir los defectos o ejecutar la configuración de pruebas encontrada hasta lograr una prueba unitaria exitosa (alcanzando los criterios de salida). Se recomienda encarecidamente obtener un registro de las pruebas realizadas y los resultados, incluso si se trata de un resultado fallido
	DT, QA, OT, IS, GK, DE	C.3.11	Actualizar el registro de trazabilidad incorporando los componentes de software construidos o modificados
	DT	C.3.12	Incorporar los componentes de software y el registro de trazabilidad en la configuración del software como parte de la línea base
	DT, AN	C.3.13	Actualizar el registro de trazabilidad, si es necesario.
	AN	C.3.14	Incorporate the test cases and test procedures, software, traceability record, test report, product operation guide and software user documentation to the software configuration as part of the baseline

Fuente : (Muñoz & Negrete, 2020)

Tareas para el proceso de Despliegue

Actividad	Rol	ID	Tareas

T.1 DESPLIEGUE	RE	T.1.1	Realizar copias de seguridad según la estrategia de control de versiones. Si es necesario, considerar cambiar a proveedores externos para asegurar las copias de seguridad.
	RE	T.1.2	Restaurar la rama anterior de producción. Si es necesario, utilizar una reversión (rollback) si se detecta un problema
	RE	T.1.3	Configurar en el archivo de integración continua una forma de recuperarse de fallos en el servicio sin demora, si es necesario
	PM	T.1.4	Evaluar el progreso del proyecto con respecto al plan del proyecto, comparando: -Tareas reales frente a tareas planificadas. -Resultados reales frente a los objetivos establecidos del proyecto. -Asignación real de recursos frente a los recursos planificados. -Costo real frente al presupuesto estimado. -Revisar el gráfico de avance (burndown chart) con la previsión. -Riesgo real frente a los riesgos previamente identificados
	PM	T.1.5	Establecer acciones para corregir desviaciones, problemas, trabajo no planificado y deuda técnica, así como los riesgos identificados que afecten al cumplimiento del plan, según sea necesario. Documentar estas acciones en un registro de correcciones y darles seguimiento hasta su finalización
	PM	T.1.6	Identificar los cambios en las historias de usuario y/o en el plan del proyecto para abordar desviaciones importantes, riesgos potenciales o problemas que afecten al cumplimiento del plan. Documentar estos cambios en una solicitud de cambio y darles seguimiento hasta su resolución.
	DT, DE	T.1.7	Actualizar el repositorio del proyecto y los servicios en la nube.
	AN	T.1.8	Documentar la guía de operación del producto o actualizar la guía actual, si corresponde
	PM, AN	T.1.9	Verificar y obtener la aprobación de la guía de operación del producto, si corresponde. Verificar la consistencia de la guía de operación del producto con el software. Los resultados encontrados se documentan en los resultados de verificación y se realizan correcciones hasta que el documento sea aprobado por el diseñador
	AN	T.1.10	Documentar la documentación de usuario del software o actualizar la existente, si corresponde

	PM, AN	T.1.11	Verificar y obtener la aprobación de la documentación de usuario del software, si corresponde. Verificar la consistencia de la documentación de usuario del software con el software. Los resultados encontrados se documentan en los resultados de verificación y se realizan correcciones hasta que el documento sea aprobado por las partes interesadas
	AN	T.1.12	Documentar la documentación de mantenimiento o actualizar la existente.
	PM, AN	T.1.13	Verificar y obtener la aprobación de la documentación de mantenimiento. Verificar la consistencia de la documentación de mantenimiento con la configuración del software. Los resultados encontrados se documentan en los resultados de verificación y se realizan correcciones hasta que el documento sea aprobado por el ingeniero de DevOps

Fuente : (Muñoz & Negrete, 2020)

Tareas para el proceso de Ejecución

Actividad	Rol	ID	Tareas
T.2 EJECUCIÓN	PM, AN	T.2.1	Incorporar la documentación de mantenimiento como línea base para la configuración del software
	PO, PM	T.2.2	Realizar la entrega según las instrucciones de entrega en el archivo de integración continua
	DT, DE	T.2.3	Empaquetar el código de formas adecuadas para el despliegue
	DE, RE	T.2.4	Desplegar el código en el servidor adecuado para verificar las funciones. Se recomienda encarecidamente aplicar el mismo proceso.
	DE, RE	T.2.5	Copiar paquetes o archivos en los servidores de producción o en el lugar necesario según el objetivo del proyecto. Se recomienda encarecidamente tener un archivo de automatización de despliegue para evitar esta tarea

Fuente : (Muñoz & Negrete, 2020)

Tareas para el proceso de Monitoreo

Actividad	Rol	ID	Tareas
T.3 MONITOREO	DE	T.3.1	Decidir las herramientas de monitoreo a aplicar en el servidor. Se recomienda encarecidamente utilizar una herramienta de monitoreo para obtener gráficos sobre el despliegue del proyecto

	GK	T.3.2	Asegurarse del rendimiento del despliegue verificando los gráficos de monitoreo.
	GK	T.3.3	Medir las estadísticas de la herramienta de monitoreo del servidor
	GK	T.3.4	Configurar notificaciones proactivas según sea necesario para tener un registro de problemas. Se recomienda encarecidamente obtener información sobre características como seguridad, disponibilidad, rendimiento, etc

Fuente : (Muñoz & Negrete, 2020)

Tareas para el proceso de Retroalimentación

Actividad	Rol	ID	Tareas
F.1 RETROALIMENTACIÓN	PM	F.3.1	Evaluar al equipo al finalizar el sprint para conocer lo que funcionó bien, lo que no funcionó tan bien y lo que se podría mejorar.
	PM	F.3.2	Analizar el nivel organizacional de la organización según el modelo organizativo de Westrum, que consiste en clasificarlo en patológico, burocrático y generativo, a través de una encuesta de Westrum, para conocer la satisfacción laboral
	PM	F.3.3	Obtener el comportamiento continuo de los usuarios en la aplicación utilizando el monitoreo para obtener retroalimentación y así obtener mejoras futuras
	PM	F.3.4	Publicar informes posteriores con la información sobre los problemas encontrados y las áreas de mejora de manera amplia y visible para todo el equipo

Fuente : (Muñoz & Negrete, 2020)

ANEXO B. Cuestionario para la evaluación de la madurez

Encuesta para evaluar la madurez de DevOps

HOLA ESTIMADO! :D

Muchas gracias por participar, esta información que brindes será de mucha ayuda para futuras investigaciones , de esta manera facilitaremos mucho la realización de los procesos en el desarrollo de software

* Obligatoria

Conozcámonos un poco...

1. Bajo el contexto DevOps, ¿Usted se considera parte del equipo de Desarrollo o del equipo de Operaciones? *

Equipo de Desarrollo (Dev):

- Diseña, desarrolla y prueba el software.
- Se enfoca en la creación de nuevas funcionalidades y mejoras.
- Trabaja en la automatización de procesos de construcción y pruebas.
- Responsable de la entrega continua y la calidad del código.

Equipo de Operaciones (Ops):

- Se encarga de la infraestructura, la implementación y la gestión de sistemas.
- Asegura la estabilidad, seguridad y rendimiento de los entornos de producción.
- Implementa y gestiona la infraestructura como código.
- Monitorea y responde a problemas en tiempo real.
- Colabora en la automatización de tareas operativas.

◯ Dev

◯ Ops

◯ Ambos

2. Nombre Completo *

Apellido y Nombre :D

Vamos a lo nuestro ...

Si tienes alguna duda, estaré presente para responder

3. ¿Cómo se crean y preparan los ambientes para el desarrollo y operación de software? *

Se refiere al proceso de crear, configurar y gestionar los entornos necesarios para el desarrollo, pruebas y despliegue de aplicaciones. Esto implica la automatización de la infraestructura y configuración, permitiendo la creación rápida y consistente de entornos.

- ◯ Crear entornos de forma manual.
- ◯ Todas las configuraciones se guardan y actualizan externamente.
- ◯ Usar virtualización cuando sea posible.
- ◯ Gestionar todos los entornos de manera eficiente.
- ◯ Automatizar completamente la creación de entornos.

4. ¿Qué tan seguro y confiable consideras que son las versiones finales de las entregas antes de su implementación? *

Se refiere a la evaluación y confirmación de que las nuevas versiones o entregas de software cumplen con los requisitos solicitados. La validación es esencial para garantizar la calidad y el rendimiento de las aplicaciones antes de ser implementadas en entornos de producción.

- ◯ Pruebas hechas a mano o con automatización mínima.
- ◯ Automatización de pruebas funcionales.
- ◯ Pruebas automáticas activadas según necesidad.
- ◯ Pruebas iniciales y visualización en un tablero compartido con el equipo operativo.
- ◯ Uso de herramientas para pruebas de resiliencia y robustez o pruebas de estrés .

5. ¿Cómo manejan y organizan la gestión de datos en el proceso? *

Se refiere a cómo se manejan, almacenan, gestionan y aseguran los datos a lo largo del ciclo de vida del desarrollo y despliegue de aplicaciones. La gestión de datos es crucial para garantizar la consistencia, la integridad y la disponibilidad de la información utilizada por una aplicación.

- ◯ Migración manual y no versionada de datos.
- ◯ Cambios en la base de datos hechos con scripts automáticos versionados.
- ◯ Automatización de cambios mínimos en la base de datos durante la implementación.
- ◯ Pruebas de actualizaciones y retrocesos de la base de datos en cada implementación.
- ◯ Evaluación del rendimiento de la base de datos después de cada lanzamiento.

6. ¿En qué medida se automatiza la implementación en tu organización? *

Se refiere al nivel en la cual los procesos de implementación y despliegue de software han sido automatizados.

- ◯ Implementación y despliegue manual.
- ◯ Automatización de los builds.
- ◯ Automatización de la implementación y despliegue en entornos previos a producción.
- ◯ Automatización de la implementación en entornos de producción.
- ◯ Colaboración entre equipos operación y desarrollo para gestionar riesgos y acelerar el proceso implementación y despliegue.

7. ¿Cómo gestionan la construcción (build) en tu organización? *

Se refiere al proceso de compilación del código fuente de una aplicación para crear un ejecutable o paquete que pueda ser desplegado y ejecutado en un entorno determinado. La gestión de los builds es un componente esencial en el ciclo de vida del desarrollo de software y tiene un impacto directo en la calidad y eficiencia de las entregas de software

- ◯ Procesos manuales para construir software / Sin versionado de artefactos.
- ◯ Builds automatizada regular con pruebas; cualquier build se puede recrear desde la fuente.
- ◯ Ciclo automatizado de build y prueba con cada cambio.
- ◯ Métricas de build recopiladas, visibles y consideradas.
- ◯ Mejora continua del proceso, con mayor visibilidad y retroalimentación más rápida.

8. ¿Cómo es la colaboración y el intercambio de conocimientos entre los miembros del equipo? *

Se refiere sobre la efectividad de la comunicación y la interacción entre los diversos roles que componen un equipo, especialmente entre los equipos de desarrollo y operaciones.

- ◯ Sin herramientas de colaboración.
- ◯ Herramienta de planificación de proyectos.(Trello, Jira, Monday, etc)
- ◯ Integración de herramientas y colaboración de equipo.
- ◯ Herramienta de gestión del conocimiento.(Concluence, Notion, Slacl, etc)
- ◯ Otras

9. ¿Utiliza su organización la gestión de configuración de software (SCM)? *

Gestión de la Configuración del Software o Software Configuration Management (SCM) es un conjunto de prácticas, procesos y herramientas que se utilizan en el desarrollo de software para gestionar y **controlar los cambios** en este tipo de artefactos a lo largo de su ciclo de vida. Un ejemplo de SCM seria la utilización de GIT.

- ◯ Sin SCM.
- ◯ SCM estandarizado.
- ◯ La configuración se entrega con el código.
- ◯ Herramientas de auto recuperación.(Chaos Monkey,Kubernetes Auto-Healing, Puppet, Chef, Ansible).

10. ¿Cómo monitorean procesos y datos en tu organización? *

Se refiere a la capacidad de la organización para supervisar y recopilar información en tiempo real sobre el rendimiento de los procesos, la infraestructura , los servidores y los datos utilizados en el desarrollo y despliegue de aplicaciones.

- ◯ No utiliza herramientas de monitoreo y solo reacciona a problemas cuando los usuarios informan de ellos.
- ◯ Utiliza una herramienta básica de monitoreo de servidores.
- ◯ Implementa una suite de monitoreo que integra múltiples herramientas para supervisar no solo la infraestructura, sino también las aplicaciones y los servicios.
- ◯ Utiliza herramientas avanzadas de analítica de datos para prever posibles problemas.

11. ¿De qué manera se rastrean los problemas/errores? *

Se refiere los procesos y herramientas utilizados para identificar, registrar, priorizar y gestionar los problemas o bugs que surgen durante el desarrollo y despliegue de aplicaciones.

- ◯ Los problemas se comunican principalmente a través de correos electrónicos, llamadas, WhatsApp o mensajes.
- ◯ La organización tiene un sistema formal para rastrear y documentar todos los problemas y reportes de errores.
- ◯ Puede haber integración con herramientas de monitoreo para la detección automática de problemas.
- ◯ Después de cada despliegue, el equipo revisa los informes de problemas de las herramientas de monitoreo, si es que reportaron alguno.

12. ¿Cómo organizan la implementación de nuevas entregas en producción en tu organización? *

Se refiere al el proceso y las prácticas utilizadas para llevar a cabo el despliegue de nuevas versiones de software en entornos de producción. Este proceso es esencial para garantizar la entrega continua y confiable de nuevas funcionalidades, mejoras y correcciones a los usuarios finales.

- ◯ Las implementaciones y despliegue se realizan de forma ad-hoc(improvisada).
- ◯ La implementación y despliegue sigue un programa predefinido y planificado. Las entregas se realizan en fechas específicas, aunque no necesariamente de manera automatizada.
- ◯ La implementación y despliegue se realiza de manera automática mediante scripts o herramientas(Utilización de herramientas como Jenkins o GitLab CI, etc).
- ◯ La implementación y despliegue se realizan con regularidad, con un enfoque en reducir el tiempo entre versiones.
- ◯ Los cambios se despliegan automáticamente en producción tan pronto como están listos(Implementación continua con herramientas como Travis CI o GitHub Actions, etc).

13. ¿Cuál es el enfoque del proceso de desarrollo en su organización? *

Se refiere al enfoque específico que adopta la organización para llevar a cabo el desarrollo de software. Este puede variar según las prácticas, metodologías y marcos de trabajo que la organización haya decidido seguir.

- ◯ Desarrollo ad-hoc (improvisado).
- ◯ Desarrollo Scrum.
- ◯ Desarrollo ágil.
- ◯ Desarrollo Lean.
- ◯ Proceso de desarrollo integrado con Six Sigma.
- ◯ Otras

14. ¿Cómo se lleva a cabo el proceso de pruebas de software en tu organización? *

Se refiere a las prácticas que la organización utiliza para planificar, ejecutar y gestionar las pruebas de software. Esto incluye aspectos como la estrategia de pruebas, la automatización de pruebas, la integración con el desarrollo, y la gestión de defectos identificados durante las pruebas.

- ◯ Pruebas ad-hoc. Explorar una nueva característica sin un plan específico.
- ◯ Pruebas basadas en requisitos. Asegurarse de que todas las funciones mencionadas en los documentos de requisitos funcionen correctamente.
- ◯ Pruebas integradas. Verificación de la interacción adecuada entre diferentes componentes del software.
- ◯ Pruebas cualitativas. Evaluación subjetiva de la calidad del software en términos de experiencia del usuario.
- ◯ Pruebas continuas. Automatización de pruebas que se ejecutan de manera regular durante el desarrollo.
- ◯ Otras

15. ¿Cómo se organiza el proceso de gestión de proyectos? *

Se refiere a las prácticas utilizadas para planificar, coordinar y gestionar el desarrollo y despliegue de software en el marco de la cultura DevOps. Este enfoque implica una integración estrecha entre los equipos de desarrollo y operaciones para lograr una entrega continua y eficiente.

- ◯ Gestión de proyectos inconsistente. Diferentes equipos utilizan métodos diversos sin estandarización.
- ◯ Gestión de proyectos y requisitos. Asegurarse de que las metas del proyecto estén alineadas con los requisitos del cliente.
- ◯ Gestión de proyectos integrada. Uso de herramientas colaborativas para seguir el progreso y comunicarse eficientemente.
- ◯ Gestión cuantitativa de proyectos. Utilización de métricas para evaluar el tiempo y los recursos dedicados en comparación con los resultados esperados.
- ◯ Gestión organizada del rendimiento. Uso de cronogramas y planes detallados para optimizar el rendimiento del equipo y cumplir con los plazos establecidos.

16. ¿Cuál es el estado de la documentación de desarrollo? *

Se refiere a la disponibilidad, la calidad y la actualización de la documentación relacionada con el proceso de desarrollo de software. La documentación desempeña un papel crucial en la gestión del conocimiento, la colaboración entre equipos y la facilitación de procesos dentro de un entorno DevOps.

- ◯ La documentación de implementación y desarrollo no está disponible o está desactualizada.
- ◯ La documentación de desarrollo y los archivos de configuración, solo los mas importantes, están actualizados.
- ◯ Se proporciona validación regular de la documentación y descripciones de configuración relacionadas.
- ◯ Se ajusta y se mejora la documentación continuamente basándose en la experiencia acumulada y los estándares.
- ◯ Otras

17. ¿Cómo se manejan los procesos en tu organización? *

Se refiere a cómo se planifican, ejecutan, supervisan y mejoran los diferentes pasos y actividades involucrados en el ciclo de vida del desarrollo y despliegue de software.

- ◯ Procesos no controlados o reactivos (sin gestión).
- ◯ Procesos gestionados, pero no estandarizados.
- ◯ Procesos estandarizados en toda la organización.
- ◯ Todos los aspectos del proceso son claros y se pueden prever.
- ◯ Los procesos se han mejorado al máximo y están completamente integrados en la operación.

18. ¿Cómo se organizan los equipos en su organización? *

Se refiere a la forma en que los equipos de trabajo están organizados para colaborar en el desarrollo, despliegue y mantenimiento de software.

- ◯ Organizado según habilidades.
- ◯ Los equipos se forman para completar entregas específicas.
- ◯ La formación de equipos centrados en proyectos particulares y sus objetivos.
- ◯ Los equipos se centran en productos particulares o áreas de negocio.
- ◯ Equipos con miembros de diversas disciplinas trabajan juntos para lograr objetivos clave.

19. ¿Cómo se organiza el proceso de aprendizaje en su organización? *

Se refiere a cómo la organización fomenta y gestiona la adquisición continua de conocimientos y habilidades por parte de sus miembros.

- ◯ Aprendizaje no estructurado, informal y basado en necesidades individuales.
- ◯ Miembros del equipo aprenden juntos, compartiendo conocimientos y experiencias.
- ◯ El aprendizaje se adapta a las necesidades específicas en diferentes etapas del proceso organizativo. Un programa de formación diseñado para abordar desafíos específicos en cada etapa de producción.
- ◯ Aprendizaje que ocurre a través de la colaboración entre diferentes áreas o procesos.
- ◯ Adquisición de conocimientos a través de fuentes externas a la organización.

20. ¿Cómo se mejoran las habilidades y capacidades en tu organización? *

Se refiere a cómo la organización promueve el crecimiento y la mejora de las habilidades y conocimientos de sus miembros en áreas relevantes para el desarrollo de software y la implementación de prácticas DevOps.

- ◯ Enfoque ad-hoc (improvisado) para el desarrollo de competencias.
- ◯ A través de programas de capacitación y desarrollo.
- ◯ Evaluación y desarrollo específico de habilidades necesarias, luego de la evaluación se hacen programas para mejor debilidades.
- ◯ Aprendizaje guiado por expertos en la materia.
- ◯ La empresa establece ciclos regulares de evaluación de habilidades y proporciona oportunidades de formación para mantenerse al día con las demandas del mercado.

21. ¿Cuál es el principal tipo de comunicación en su organización? *

Se refiere al método predominante mediante el cual los miembros del equipo se comunican entre sí para colaborar en el desarrollo, despliegue y operación del software.

- ◯ Limitado intercambio de información.
- ◯ Flujo eficiente de información dentro de un equipo.
- ◯ Intercambio eficaz de información entre diferentes equipos.
- ◯ Intercambio constante y cooperativo de información.
- ◯ Después de cada iteración de desarrollo, se realizan reuniones rápidas para proporcionar retroalimentación.

22. ¿Cuál es el nivel de comprensión de los requisitos? *

Se refiere a la medida en que los equipos involucrados en el desarrollo y despliegue de software comprenden los requisitos del proyecto.

- ◯ El propósito del proyecto, no se comunica eficientemente.
- ◯ Los requisitos específicos para entregar un producto o servicio son comprensibles.
- ◯ Requisitos específicos relacionados con el alcance y los objetivos del proyecto.
- ◯ Especificaciones detalladas para los productos o servicios ofrecidos.
- ◯ Especificaciones y expectativas claras en el funcionamiento interno de la organización.

23. ¿Cuál es la comprensión y uso de la cultura en la organización? *

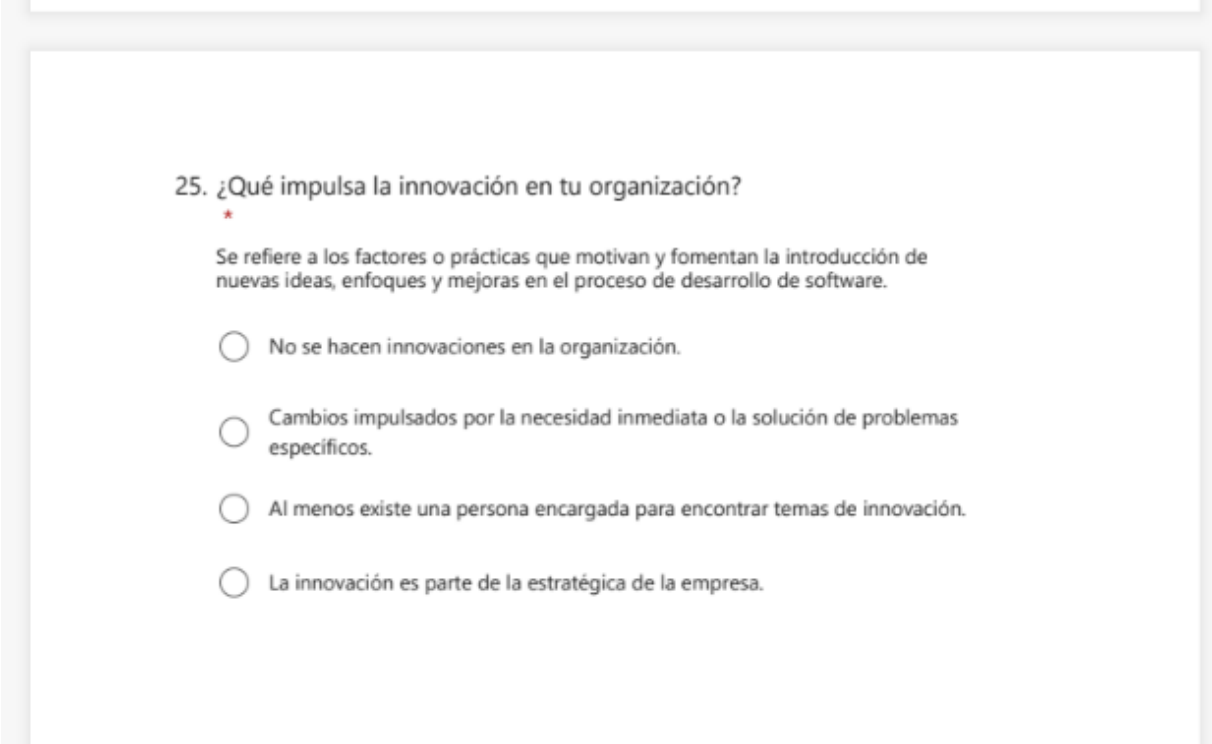
Se refiere a cómo la organización entiende y aplica los principios culturales asociados con DevOps. La cultura en DevOps es un componente crítico que impulsa la colaboración, la transparencia, la automatización y la entrega continua. La pregunta busca evaluar en qué medida la organización ha internalizado y está aplicando estos valores culturales.

- ◯ No reconocen la influencia de la cultura organizacional en la toma de decisiones cotidiana.
- ◯ Reconocimiento de cómo ciertos aspectos culturales pueden facilitar o dificultar el éxito.
- ◯ La empresa identifica la colaboración y la adaptabilidad cultural como impulsores clave de su estrategia ágil.
- ◯ La empresa invierte en programas de desarrollo cultural para fortalecer valores clave.
- ◯ Se establece y fomenta activamente una cultura de colaboración y se incorpora en todas las facetas del trabajo diario.

24. ¿Cómo se evalúa el nivel de comunicación y colaboración en tu organización? *

Se refiere a la eficacia y la intensidad de la interacción entre los equipos de desarrollo, operaciones y otros actores relevantes en el ciclo de vida del desarrollo de software. La comunicación y colaboración son fundamentales en un entorno DevOps para superar las barreras tradicionales entre equipos y lograr una entrega de software más rápida, confiable y continua.

- ◯ Equipos trabajan de forma aislada, sin compartir información.
- ◯ La empresa utiliza herramientas de gestión de proyectos para seguir y mejorar la comunicación entre los miembros del equipo.
- ◯ Equipos trabajan juntos en sesiones de lluvia de ideas para resolver problemas y generar nuevas ideas.
- ◯ Se implementa un sistema que rastrea el tiempo de respuesta en las interacciones del equipo para identificar áreas que necesitan mejoras en la eficiencia y comunicación.



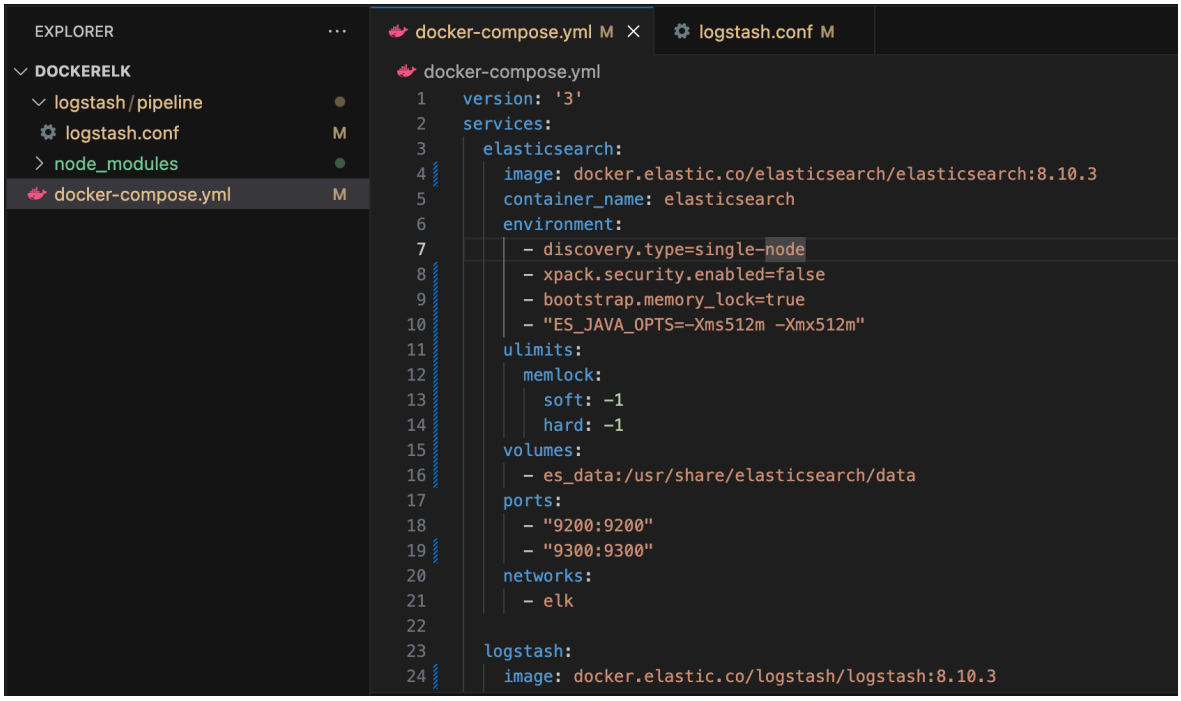
Fuente : (Zarour et al., 2020)

ANEXO C. Implementación del ELK

a) Creción de la Rama : Se creo una rama especifica apartir de la rama de produccion por cada servicio, para poder configuarar el ELK

```
~/api_usoft > prod !1
git checkout prod
D       "src/docs/BetsW3/Integracio\314\201n Plataforma de parley V3.0 (6).pdf"
Already on 'prod'
Your branch is up to date with 'origin/prod'.

~/api_usoft > prod !1
git checkout -b prod-elk
Switched to a new branch 'prod-elk'

~/api_usoft > prod-elk !1
git branch
```

b) Crear una imagen dokerizada del ELK : Se creo una imagen docker con el Stak ELK en puertos locales como se aprecia en la imagen



Se evidencia la **imagen Docker** del stack ELK. Como parte del proceso, la documentación oficial indicaba la creación de una carpeta denominada logstash, dentro de la cual debía incluirse otra subcarpeta llamada pipeline. En esta última se debía alojar el archivo logstash.conf, que contiene las configuraciones necesarias para el procesamiento y envío de los logs

```yaml
version: '3'
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:8.10.3
    container_name: elasticsearch
    environment:
      - discovery.type=single-node
      - xpack.security.enabled=false
      - bootstrap.memory_lock=true
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
    ulimits:
      memlock:
        soft: -1
        hard: -1
    volumes:
      - es_data:/usr/share/elasticsearch/data
    ports:
      - "9200:9200"
      - "9300:9300"
    networks:
      - elk

  logstash:
    image: docker.elastic.co/logstash/logstash:8.10.3
    container_name: logstash
    ports:
      - "5044:5044"
    volumes:
      - ./logstash/pipeline:/usr/share/logstash/pipeline
    networks:
      - elk

  kibana:
    image: docker.elastic.co/kibana/kibana:8.10.3
    container_name: kibana
    environment:
      - ELASTICSEARCH_HOSTS=http://elasticsearch:9200
    ports:
      - "5601:5601"
    networks:
      - elk

networks:
  elk:
    driver: bridge
```

Se puede observar el archivo docker-compose.yml, que constituye la configuración esencial para levantar la imagen Docker del stack ELK. Este archivo define, entre otras cosas, los puertos locales en los que estaría alojada la imagen, permitiendo que el sistema interactúe correctamente con el entorno local durante las pruebas. se muestra la configuración local del

stack ELK dockerizado, donde se evidencian los puertos asignados para cada uno de sus componentes principales: Elasticsearch, Logstash y Kibana. Esta configuración fue diseñada para asegurar una comunicación eficiente entre los servicios en el entorno local.

Configuración del logstash.conf y los puertos

```
input {
  tcp {
    port => 5044
    codec => json {
      target => "data"
    }
  }
}


output {
  if [@metadata][target_index] {
    elasticsearch {
      hosts => ["http://elasticsearch:9200"]
      index => "%{[@metadata][target_index]}-%{+YYYY.MM.dd}"
    }
  } else {
    stdout { codec => rubydebug }
    # Mensaje de advertencia o acción alternativa si `@metadata.target_index` no
está presente
  }
}
```

Se evidencia la configuración del Logstash y Elasticsearch dentro del stack ELK. El Logstash fue configurado para escuchar en el puerto 5044, donde recibía los logs enviados desde el sistema. Posteriormente, estos logs eran procesados y enviados a Elasticsearch a través del puerto 9200.

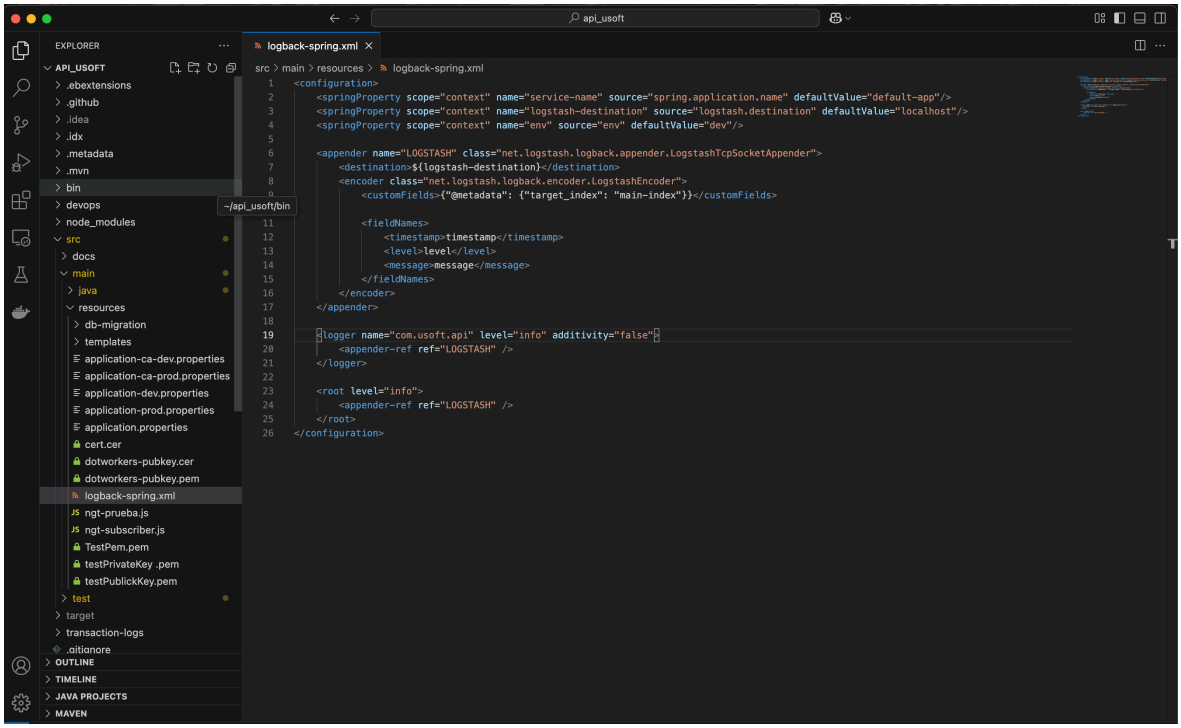
Es importante aclarar que cada log llega en formato JSON, el cual incluye un título conocido como “index”. Este índice requiere ciertos parámetros específicos, como un nombre definido y la fecha correspondiente. Estos parámetros fueron configurados en esta etapa, asegurando que los logs se almacenen de manera organizada y puedan ser consultados eficientemente en Kibana.

Esta configuración permitió no solo el flujo adecuado de los logs entre Logstash y Elasticsearch, sino también el establecimiento de un esquema de almacenamiento que facilita la identificación y análisis de los datos en el entorno local.

c) Configuraciòn del codigo fuente en los servicios locales

En esta actividad, se realizaron las configuraciones necesarias para la generación y gestión de logs. Para ello, se creó el archivo logback-spring.xml, encargado de enviar los registros de eventos.

Es importante destacar que, inicialmente, la configuración se implementó en un entorno local con el propósito de validar el flujo completo de los logs. Este proceso permitió verificar cómo los registros eran generados y enviados a Logstash, posteriormente almacenados en Elasticsearch, y finalmente visualizados en Kibana. Esta prueba en local garantizó que la integración funcionara correctamente antes de su despliegue en producción.



En la figura X refleja el Logback, un marco de registro (logging) utilizado en aplicaciones Java. Está configurado para enviar logs a Logstash, que es parte del stack ELK (Elastic, Logstash, Kibana).

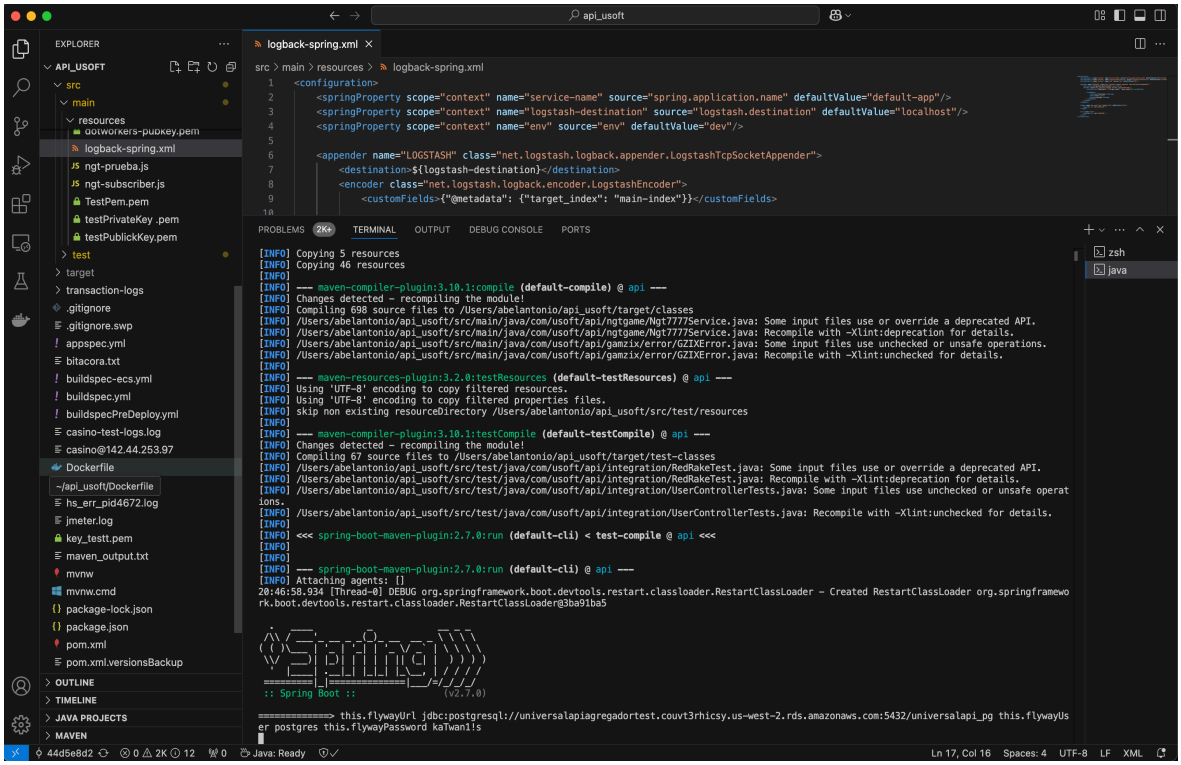
Desglose del código:

1. Propiedades dinámicas:

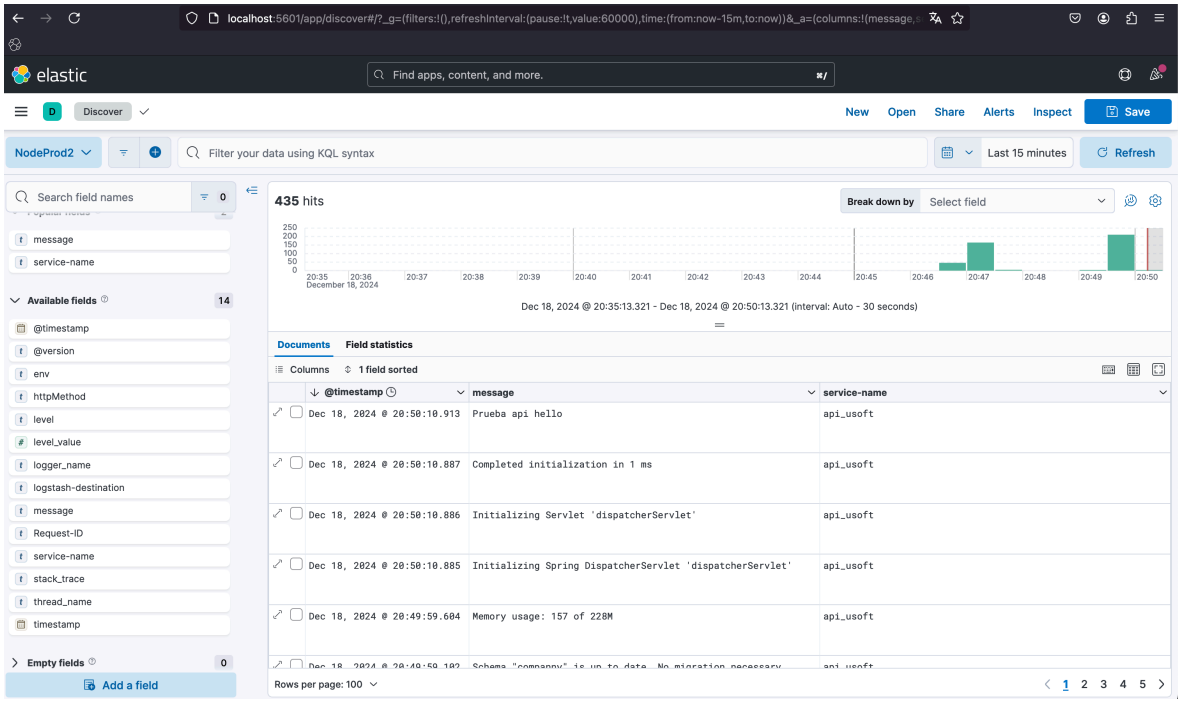
- <springProperty> define variables que se obtienen de la configuración de Spring:
- service-name: Nombre de la aplicación, tomado de spring.application.name.
- logstash-destination: Dirección del servidor Logstash, predeterminada en localhost.
- env: Entorno de la aplicación, predeterminado en dev.

2. Appender:

- El appender LOGSTASH envía los logs a Logstash usando un socket TCP.
- <destination> especifica la dirección de Logstash, tomada de la propiedad logstash-destination.
- <encoder> formatea los logs en formato JSON compatible con Logstash.
- <customFields> añade metadatos personalizados, como @metadata con el índice de destino (main-index).
- <fieldNames> define cómo se nombran los campos en el log JSON (por ejemplo, timestamp, level, message



En la figura anterior se observa que el proyecto inició su ejecución localmente, y desde el momento en que se inicia, se generan únicamente registros relacionados con la etapa de inicio



En la figura anterior se muestra la interfaz de Kibana, donde, como se indicó anteriormente, se pueden observar los registros de inicio junto con un log de la del “hello”, lo cual confirma que la aplicación está funcionando correctamente. Además, se visualiza un service name que indica que los registros pertenecen al proyecto “api-usof”. Lo cual de igual manera se hizo con los demas servicios, si bien es cierto cambiaba el lenduaje de programación pero la logica seguia siendo la misma. Enviar logs con los nombres de que servicio esta llegando

d) Configuración en produccion

Una vez que comprendí cómo funciona, decidimos adaptarlo a producción. En producción, la configuración se gestiona de manera un poco diferente. Notarán que el archivo logback es más sencillo, ya que todo está predefinido en otro archivo.

```
<configuration>
    <springProperty scope="context" name="service-name"
source="spring.application.name" defaultValue="default-app"/>
    <springProperty scope="context" name="logstash-destination"
source="logstash.destination" defaultValue="192.168.1.46:5044"/>
    <springProperty scope="context" name="env" source="spring.application.env"
defaultValue="dev"/>
    <appender name="LOGSTASH"
class="net.logstash.logback.appender.LogstashTcpSocketAppender">
        <destination>${logstash-destination}</destination>
        <encoder class="net.logstash.logback.encoder.LogstashEncoder">
<!--            <customFields>{"@metadata": {"target_index": "upi-prod-
index"}}</customFields>-->
            <fieldNames>
                <timestamp>timestamp</timestamp>
                <level>level</level>
                <message>message</message>
            </fieldNames>
        </encoder>
    </appender>

    <logger name="com.usoft.api" level="info" additivity="false">
        <appender-ref ref="LOGSTASH" />
    </logger>

    <root level="info">
        <appender-ref ref="LOGSTASH" />
    </root>
</configuration>
```

La figura anterior muestra un archivo de configuración en formato XML para Logback. Este archivo está configurado para enviar los registros de una aplicación Java a un servidor Logstash mediante el protocolo TCP. A continuación, se detallan sus características y configuraciones principales.

1. Propiedades dinámicas (<springProperty>):

Estas propiedades obtienen valores de las configuraciones de Spring.

service-name: Toma su valor de spring.application.name (nombre de la aplicación en Spring).

logstash-destination:Dirección y puerto del servidor Logstash donde se enviarán los logs.

env: Define el entorno de la aplicación (e.g., desarrollo o producción).

2. Appender LOGSTASH:

<destination>: Indica la dirección de Logstash para recibir los logs. Usa la propiedad logstash-destination.

<encoder>: Formatea los logs en formato JSON compatible con Logstash.

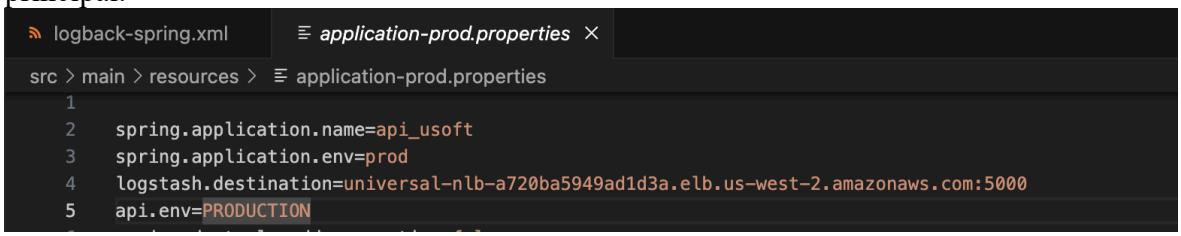
<fieldNames>: Personaliza los nombres de los campos en los logs:

timestamp: Fecha y hora.

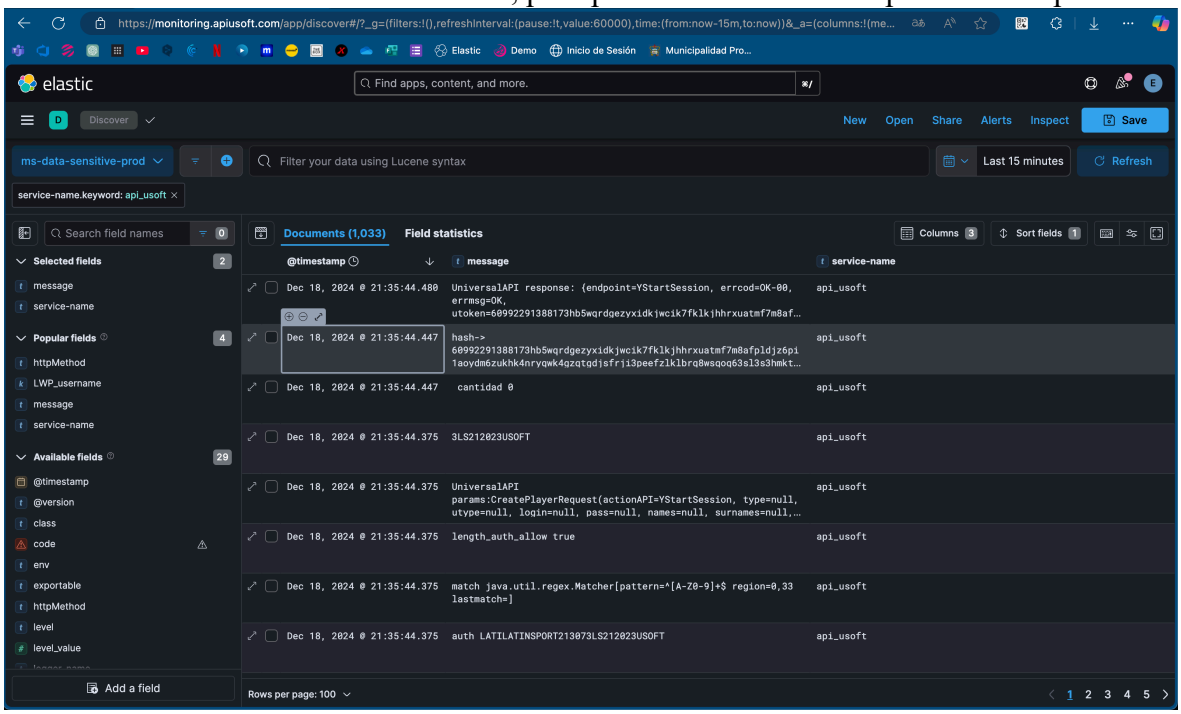
level: Nivel del log (e.g., INFO, ERROR).

message: Mensaje del log.

El archivo de propiedades contiene los valores predefinidos necesarios para las variables dinámicas utilizadas en el archivo XML. Estas propiedades permiten parametrizar la configuración de manera flexible, facilitando ajustes sin modificar directamente el archivo principal.



- spring.application.name: Define el nombre de la aplicación: api_usoft. Se asigna al <springProperty> service-name.
- spring.application.env: Define el entorno de la aplicación: prod. Se asigna al <springProperty> env.
- logstash.destination:Dirección y puerto del servidor Logstash: universal-nlb-a720ba5949ad1d3a.elb.us-west-2.amazonaws.com:5000.
- api.env: Define una variable adicional (PRODUCTION) que no se utiliza directamente en el XML, pero podría ser útil en otras partes de la aplicación.



En producción, se ve como en la figura anterior, donde los logs están llegando correctamente junto con su service name. Esto permite realizar consultas y monitorear lo que está ocurriendo en tiempo real. Además, es posible automatizar el sistema para que se actualice de manera periódica o cada vez que el usuario lo requiera.

ANEXO D. Implementación del Request ID

```java
package com.usoft.api.config;

import org.slf4j.MDC;
import org.springframework.stereotype.Component;
import org.springframework.web.servlet.HandlerInterceptor;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.UUID;

@Component
public class RequestInterceptor implements HandlerInterceptor {

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse
response, Object handler) throws Exception {
        // Cambia "requestId" a "Request-ID"
        MDC.put("Request-ID", UUID.randomUUID().toString());
        MDC.put("httpMethod", request.getMethod());
        return true;
    }

    @Override
    public void afterCompletion(HttpServletRequest request, HttpServletResponse
response, Object handler, Exception ex) throws Exception {
        MDC.clear();
    }
}
```

RequestInterceptor:

Define la lógica que se ejecutará antes y después de procesar una solicitud HTTP.

Método preHandle:

- Se ejecuta antes de que la solicitud llegue al controlador.
- Genera un identificador único (Request-ID) para la solicitud usando UUID.
- Extrae el método HTTP (GET, POST, etc.) de la solicitud.

Agrega ambos datos (Request-ID y httpMethod) al MDC (Mapped Diagnostic Context), una estructura que asocia información con el hilo de ejecución actual. Esto permite que estos datos sean accesibles para los logs.

Método afterCompletion:

- Se ejecuta después de que se completa la solicitud.
- Limpia el MDC para evitar que los datos persistan en otros contextos o hilos.

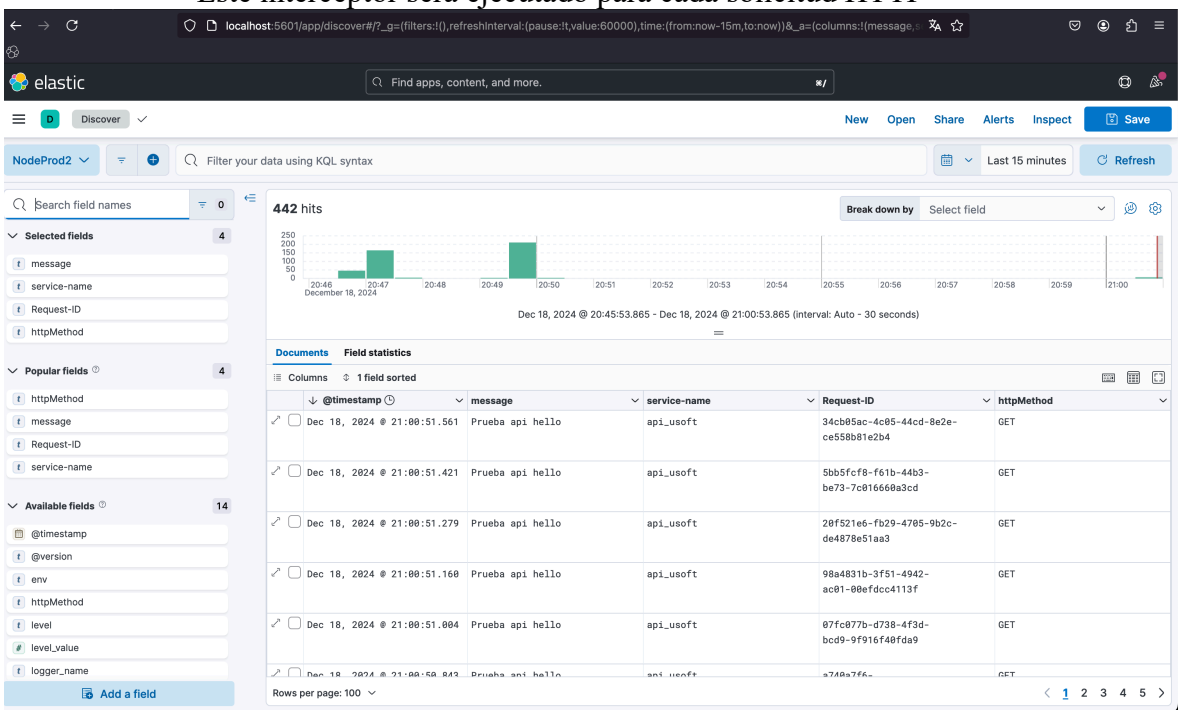
```java
package com.usoft.api.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class WebConfig implements WebMvcConfigurer {

    @Bean
    public RequestInterceptor requestInterceptor() {
        return new RequestInterceptor();
    }

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(requestInterceptor());
    }
}
```

WebConfig:

- Configura la aplicación para usar un interceptor personalizado.
- Método requestInterceptor():
- Declara un bean de tipo RequestInterceptor.

Método addInterceptors:

- Registra el interceptor definido (RequestInterceptor) en la aplicación.
- Este interceptor será ejecutado para cada solicitud HTTP



En la figura anterior se evidencian los logs generados localmente, donde se incluye el respectivo Request-ID y el método HTTP capturado. En este caso, solo se está llamando a una función específica. Sin embargo, en el entorno de producción se podrá observar la

funcionalidad completa, incluyendo cómo el Request-ID permite rastrear de manera efectiva todas las llamadas relacionadas a una misma solicitud, asegurando una trazabilidad adecuada

PRODUCCION

Cuando migramos a producción, la configuración se mantuvo casi igual. No hubo cambios sustanciales en su estructura, lo que facilitó el proceso de implementación. Esto demuestra que el diseño inicial fue lo suficientemente robusto para adaptarse al entorno productivo con mínimas modificaciones.

```java
package com.usoft.api.config;

import org.slf4j.MDC;
import org.springframework.stereotype.Component;

import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;
import java.io.IOException;
import java.util.UUID;

@Component
public class RequestWebFilter implements Filter {

    private final ApplicationProperties applicationProperties;
    private final LoggerUtil loggerUtil;

    public RequestWebFilter(ApplicationProperties applicationProperties,
LoggerUtil loggerUtil) {
        this.applicationProperties = applicationProperties;
        this.loggerUtil = loggerUtil;
    }

    @Override
    public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain) throws IOException, ServletException {
        HttpServletRequest httpRequest = (HttpServletRequest) request;

        // Extract or generate the Request ID
        String requestId = httpRequest.getHeader(PropertiesConstants.REQUEST_ID);
        if (requestId == null) {
            requestId = UUID.randomUUID().toString();
        }

        try {
            // Añade `requestId`, `httpMethod`, y `serviceName` al MDC
            MDC.put("Request-ID", requestId); // Request ID único
            MDC.put("httpMethod", httpRequest.getMethod()); // Método HTTP
            //MDC.put("serviceName", applicationProperties.getApplicationName());
// Nombre del servicio


            // Log de detalles de la solicitud usando LoggerUtil
            loggerUtil.showLogRequest(httpRequest);

            // Continuar con el filtro
            chain.doFilter(request, response);
        } finally {
            // Limpiar MDC para evitar fugas de memoria
            MDC.clear();
        }
```

```
    }

    @Override
    public void init(FilterConfig filterConfig) {
        // Lógica de inicialización si es necesario
    }

    @Override
    public void destroy() {
        // Lógica de limpieza si es necesario
    }
}
```

Este script implementa un filtro de servlet, que intercepta todas las solicitudes HTTP antes de que lleguen al controlador y realiza las siguientes acciones:
Componentes principales:

- RequestWebFilter: Clase que implementa la interfaz Filter.
- Es un componente Spring, lo que permite que se inyecte y registre automáticamente en el contexto de Spring.

Atributos y dependencias:

- applicationProperties: Objeto para acceder a propiedades configuradas en la aplicación (como el nombre del servicio).
- loggerUtil: Clase de utilidad para manejar el registro de logs (loggers).

Métodos principales:

- doFilter: Intercepta cada solicitud HTTP.
- Obtención del Request-ID:
- Extrae el Request-ID del encabezado HTTP.
- Si no está presente, genera un nuevo ID único usando UUID.

Contexto de diagnóstico (MDC):

- Añade al MDC información útil como:
- Request-ID: Identificador único para rastrear la solicitud.
- httpMethod: Método HTTP (GET, POST, etc.).
- (Opcionalmente) serviceName: Nombre del servicio.

Registro de solicitud:

- Usa loggerUtil para registrar detalles de la solicitud.
- Continúa el procesamiento:
- Llama a chain.doFilter para pasar la solicitud al siguiente componente en la cadena.

Limpieza del MDC:

- Elimina las entradas del MDC al finalizar, evitando fugas de memoria.

```java
package com.usoft.api.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.boot.web.servlet.FilterRegistrationBean;
import org.springframework.context.annotation.Bean;

@Configuration
public class WebConfig implements WebMvcConfigurer {

    @Bean
    public FilterRegistrationBean<RequestWebFilter>
requestWebFilterRegistration(RequestWebFilter requestWebFilter) {
        FilterRegistrationBean<RequestWebFilter> registrationBean = new
FilterRegistrationBean<>();
        registrationBean.setFilter(requestWebFilter);
        registrationBean.setOrder(1); // Establece el orden de ejecución del
filtro, si tienes más filtros
        return registrationBean;
    }
}
```

Este script configura el filtro RequestWebFilter para que se registre y se ejecute en todas las solicitudes HTTP.

Componentes principales:

- Web Config: Clase marcada como @Configuration, lo que indica que proporciona configuraciones de Spring.

Métodos principales:

- requestWebFilterRegistration:Registra el filtro RequestWebFilter en el servidor web embebido de Spring Boot (como Tomcat).
- FilterRegistrationBean: Es un contenedor que permite personalizar cómo y cuándo se ejecuta un filtro.

Configuración clave:

- Filtro (setFilter): Registra el filtro RequestWebFilter.
- Orden (setOrder): Especifica el orden de ejecución del filtro en relación con otros filtros.

¿Cómo funcionan juntos?

RequestWebFilter:

- Define la lógica para interceptar solicitudes HTTP.
- Extrae o genera un Request-ID único, registra detalles de la solicitud y limpia el contexto al finalizar.

WebConfig:

- Registra RequestWebFilter para que sea reconocido como un filtro válido en el servidor embebido.
- Define el orden de ejecución del filtro en la cadena de filtros.

Flujo de funcionamiento conjunto:

Inicio de una solicitud HTTP:

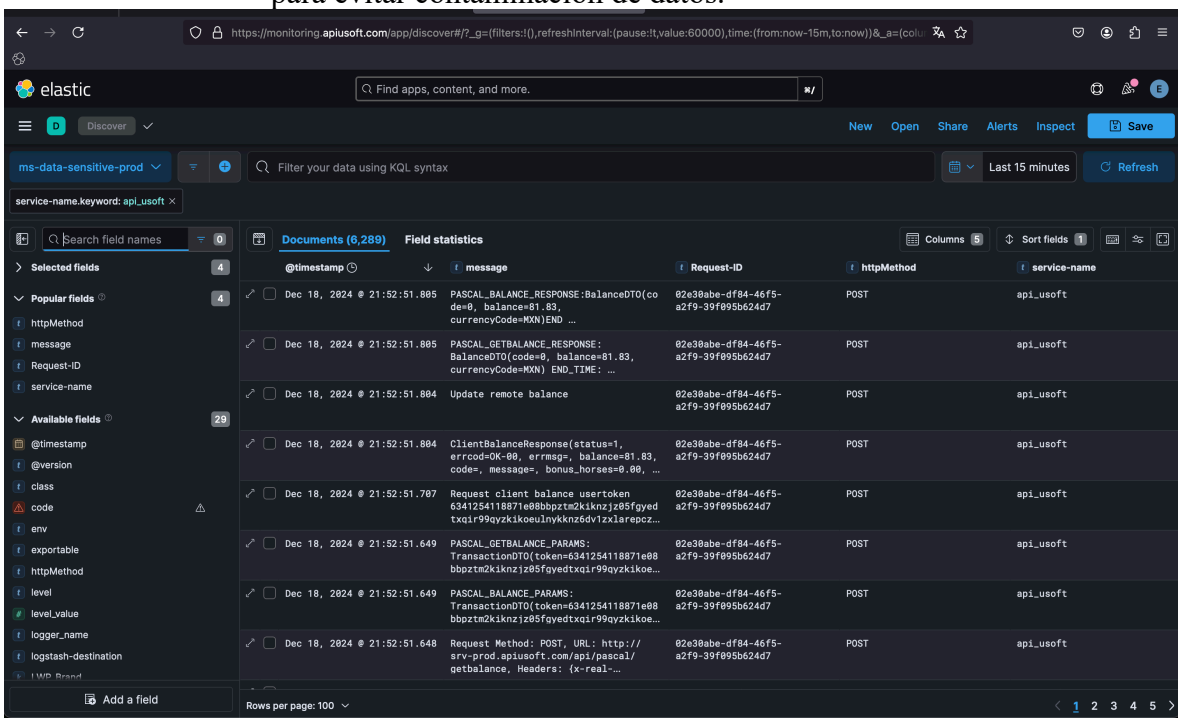
- Una solicitud HTTP entrante pasa a través de todos los filtros registrados.

Interceptación por RequestWebFilter:

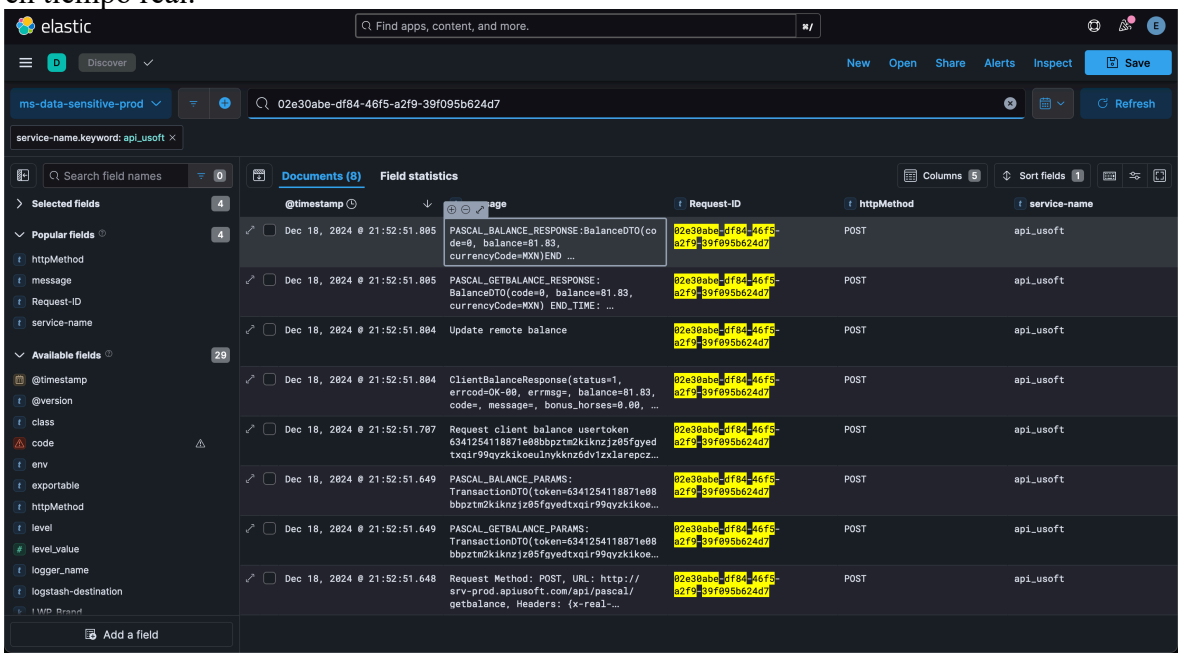
- Se ejecuta debido a su registro en WebConfig.
- Extrae o genera un Request-ID.
- Registra detalles relevantes de la solicitud en los logs (por ejemplo, método HTTP).
- Pasa la solicitud al siguiente componente en la cadena.

Limpieza:

- Al finalizar la solicitud, el filtro elimina cualquier información del MDC para evitar contaminación de datos.



En la figura anterior se evidencia el funcionamiento en producción, mostrando los mensajes con información, incluyendo el método respectivo, su Request-ID, y un identificador que confirma que pertenecen al proyecto api-usof. A partir de este punto, es posible realizar búsquedas reales en los registros, lo que facilita el análisis y la trazabilidad de las solicitudes en tiempo real.



En la figura anterior, se presenta un caso en el que se busca un Request-ID específico previamente generado, y el sistema muestra todos los logs asociados a esa operación, incluyendo las funciones llamadas como parte del mismo flujo. Esto proporciona una trazabilidad completa de la operación, lo que facilita tanto el análisis como la solución de problemas. En este caso específico, el tiempo necesario para identificar y resolver el problema se redujo de 2 horas a tan solo 30 minutos.

ANEXO E. Resultados de la Evaluación del cuestionario TAM

Utilidad Percibida (PU)				Total	ESTADISTICA DESCRIPTIVA	Promedio Ind.	Promedio Gnr.	Mediana	Moda	Desciación estandar	MAX	MIN	ESTADISTICA PREDICTIVA
Usar ELK mejora mi rendimiento en el trabajo	Usar ELK incrementa mi productividad	Usar ELK mejora mi eficacia en el trabajo.	Encuentro útil el uso de ELK en mi trabajo										
7	7	7	7	28		7							
5	6	6	6	23		5.8							
5	5	5	6	21		5.3							
6	5	7	6	24		6	5.917	6.00	7.00	1.00	7.00	4.00	
7	7	7	7	28		7							
5	5	4	4	18		4.5							

Facilidad de Uso(PEOU)				Total	ESTADISTICA DESCRIPTIVA								ESTADISTICA PREDICTIVA
Mi interacción con ELK es clara y comprensible.	Usar ELK no requiere mucho esfuerzo mental.	Encuentro que ELK es fácil de usar.	Me resulta fácil lograr que ELK haga lo que necesito.										
6	5	6	6	23		5.8							
6	5	6	4	21		5.3							
6	6	5	5	22		5.5							
4	6	5	5	20		5	5.38	5.50	6.00	1.18	7.00	2.00	
7	7	7	7	28		7							
5	2	4	4	15		3.8							

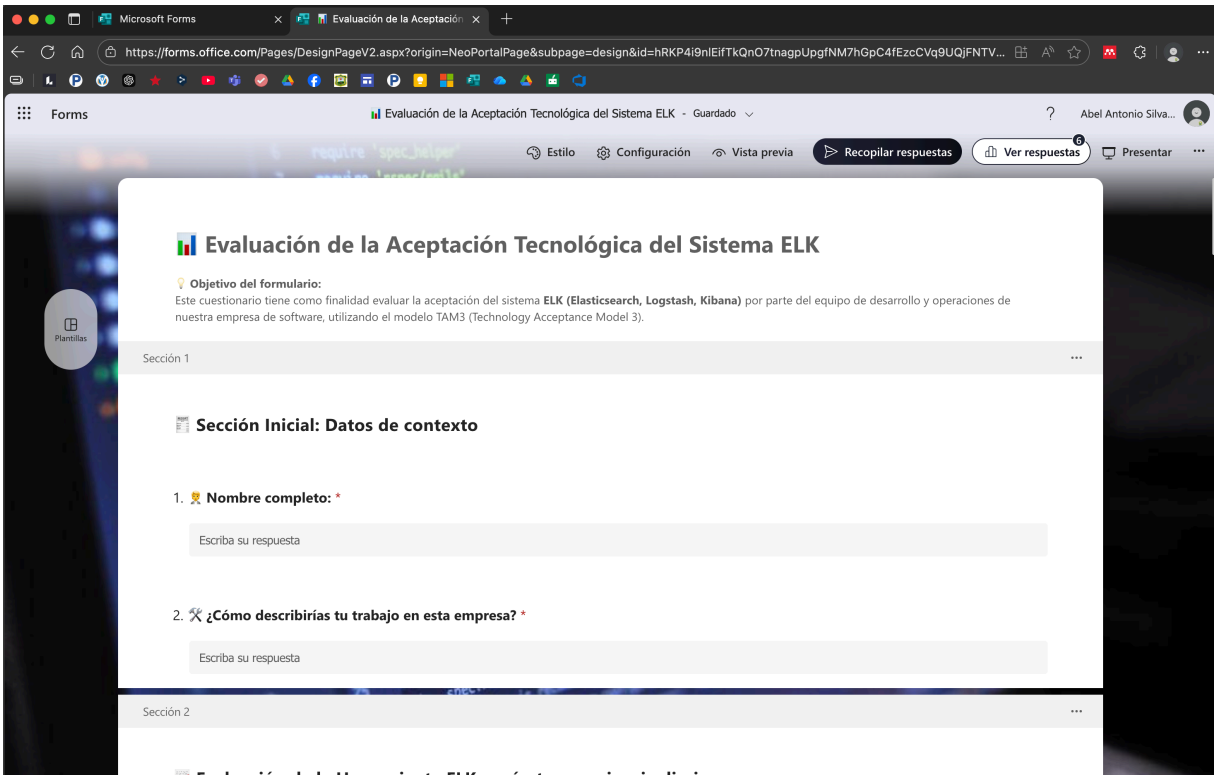
Intención de Uso(BI)			Total	ESTADISTICA DESCRIPTIVA							ESTADISTICA PREDICTIVA
Si tengo acceso a ELK, tengo la intención de usarlo.	Dado que tengo acceso a ELK, predigo que lo usaré.	Planeo usar ELK en los próximos meses.									
7	7	7	21		7						
5	6	6	17		5.7						
6	6	5	17		5.7	6.06	6.00	6.00	0.78	7.00	5.00
6	6	6	18		6						
7	7	7	21		7						
5	5	5	15		5						

ANEXO F. Resultado General del Cuestionario TAM

Utilidad Percibida				Facilidad de Uso				Intención de Uso			Total	Promedio Ind.
7	7	7	7	6	5	6	6	7	7	7	72	6.5
5	6	6	6	6	5	6	4	5	6	6	61	5.6
5	5	5	6	6	6	5	5	6	6	5	60	5.5
6	5	7	6	4	6	5	5	6	6	6	62	5.6
7	7	7	7	7	7	7	7	7	7	7	77	7
5	5	4	4	5	2	4	4	5	5	5	48	4.3

Promedio Ind.	Promedio Gnr.	Mediana	Moda	Desciación estanda	MAX	MIN
6.5	5.750	6.00	6.00	1.06	7.00	2.00
5.6						
5.5						
5.6						
7						
4.3						

ANEXO G. Cuestionario TAM



Evaluación de la Herramienta ELK según tu experiencia diaria

A continuación encontrarás una serie de afirmaciones relacionadas con el uso de **ELK (Elasticsearch, Logstash, Kibana)** en tu trabajo. Por favor, responde con sinceridad usando la escala de **1 a 7**, donde:

- **1** = Totalmente en desacuerdo
- **4** = Ni de acuerdo ni en desacuerdo
- **7** = Totalmente de acuerdo

Confidencialidad: Tus respuestas serán utilizadas solo con fines internos y de mejora continua.

3. **Utilidad Percibida** *

Perceived Usefulness (PU)

	Totalmente en desacuerdo	En desacuerdo	Ligeramente en desacuerdo	Neutral / Ni de acuerdo ni en desacuerdo	Ligeramente de acuerdo	De acuerdo	Tota
Usar ELK mejora mi rendimiento en el trabajo	○	○	○	○	○	○	
Usar ELK incrementa mi productividad	○	○	○	○	○	○	
Usar ELK mejora mi eficacia en el trabajo.	○	○	○	○	○	○	
Encuentro útil el uso de ELK en mi trabajo	○	○	○	○	○	○	

4. **Facilidad de Uso Percibida** *

Perceived Ease of Use (PEOU)

	Totalmente en desacuerdo	En desacuerdo	Ligeramente en desacuerdo	Neutral / Ni de acuerdo ni en desacuerdo	Ligeramente de acuerdo	De acuerdo	Tota
Mi interacción con ELK es clara y comprensible.	◯	◯	◯	◯	◯	◯	
Usar ELK no requiere mucho esfuerzo mental.	◯	◯	◯	◯	◯	◯	
Encuentro que ELK es fácil de usar.	◯	◯	◯	◯	◯	◯	
Me resulta fácil lograr que ELK haga lo que necesito.	◯	◯	◯	◯	◯	◯	

5. **Intención de Uso** *

Behavioral Intention (BI)

	Totalmente en desacuerdo	En desacuerdo	Ligeramente en desacuerdo	Neutral / Ni de acuerdo ni en desacuerdo	Ligeramente de acuerdo	De acuerdo	Tota
Si tengo acceso a ELK, tengo la intención de usarlo.	◯	◯	◯	◯	◯	◯	
Dado que tengo acceso a ELK, predigo que lo usaré.	◯	◯	◯	◯	◯	◯	
Planeo usar ELK en los próximos meses.	◯	◯	◯	◯	◯	◯	

ANEXO H. Matriz de Consistencia

Problemas	Objetivos	Hipótesis	Variable	Dimensiones	Indicadores	Metología
General			Aceptación tecnológica de las herramientas(TAM)	1. Utilidad percibida 2. Facilidad de Uso 3. Intención de Uso	**1.1** Incremento del rendimiento laboral percibido **1.2** Aumento de la productividad **1.3** Mejora de la eficacia en las tarea **1.4** Percepción de utilidad general de la herramienta **2.1** Claridad y comprensión en la interacción con la herramienta **2.2**	**Tipo de Investigación** Aplicada **Alcance de la Investigación** Descriptivo (Caso de estudio) Diseño de la Investigación Preexperimental **Población** Indeterminada, Conformada por miembros de la empresas de software **Muestra** No probabilística, seleccionada por factibilidad
Cuál es el grado de la aceptación tecnológica de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas	Determinar el grado la aceptación tecnológica de las herramientas propuestas s para implementar DevOps Reforzado en las pequeñas empresas	La aceptación tecnológica de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas es alta				
Especifico						
Cuál es el grado de Utilidad percibida de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas de software	Determinar el grado de Utilidad percibida de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas	La Utilidad percibida de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas es alta				
Cuál es el grado de Intención de uso de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas de software	Determinar el grado de Intención de uso de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas	la Intención de uso de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas es alta				

Cuál es el grado de Facilidad de uso de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas de software	Determinar el grado de Facilidad de uso de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas	la Facilidad de uso de las herramientas propuestas para implementar DevOps Reforzado en las pequeñas empresas es alta			Nivel de esfuerzo mental requerido **2.3** Percepción de facilidad de uso general **2.4** Capacidad para lograr objetivos con la herramienta **3.1** Intención declarada de uso futuro **3.2** Predicción de uso basada en disponibilidad. **3.3** Planificación de uso sostenido en el tiempo	**Instrumento** La técnica fue una encuesta, mientras el instrumento corresponde a un cuestionario TAM

Fuente : Elaboración Propia