



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ingeniería Electrónica y Eléctrica

Escuela Profesional de Ingeniería Electrónica

**Implementación de un sistema de monitoreo Open
Source en una empresa de telecomunicaciones**

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el Título Profesional de Ingeniero Electrónico

AUTOR

Alvaro Mauricio MENA FLORES

ASESOR

Mg. Wilbert CHAVEZ IRAZABAL

Lima, Perú

2026



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Mena, A. (2026). *Implementación de un sistema de monitoreo Open Source en una empresa de telecomunicaciones*. [Trabajo de suficiencia profesional de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería Electrónica y Eléctrica, Escuela Profesional de Ingeniería Electrónica]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Alvaro Mauricio Mena Flores
Tipo de documento de identidad	DNI
Número de documento de identidad	47164056
URL de ORCID	No Aplica
Datos de asesor	
Nombres y apellidos	Wilbert Chavez Irazabal
Tipo de documento de identidad	DNI
Número de documento de identidad	08121733
URL de ORCID	https://orcid.org/0000-0002-7978-7031
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Jorge Elías Moscoso Sánchez
Tipo de documento	DNI
Número de documento de identidad	07206008
Miembro del jurado 1	
Nombres y apellidos	Daniel Díaz Ataucuri
Tipo de documento	DNI
Número de documento de identidad	07139361
Miembro del jurado 2	
Nombres y apellidos	Arlich Joel Portillo Allende
Tipo de documento	DNI
Número de documento de identidad	10125495
Datos de investigación	

Línea de investigación	C.0.3.20 Gestión de los Sistemas informáticos y de información
Grupo de investigación	No Aplica
Agencia de financiamiento	No Aplica
Ubicación geográfica de la investigación	Edificio: Time Empresarial País: Perú Departamento: Lima Provincia: Lima Distrito: Magdalena del Mar Latitud: -12.1016911 Longitud: -77.0597816
Año o rango de años en que se realizó la investigación	Enero 2024 – Diciembre 2024
URL de disciplinas OCDE	Ingeniería eléctrica, Ingeniería Electrónica https://purl.org/pe-repo/ocde/ford#2.02.01 Ingeniería de sistemas y comunicaciones https://purl.org/pe-repo/ocde/ford#2.02.00



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
(Universidad del Perú, DECANA DE AMÉRICA)
FACULTAD DE INGENIERÍA ELECTRÓNICA Y ELÉCTRICA
Teléfono 619-7000 Anexo 4226
Calle Germán Amezaga 375 – Lima 1 – Perú



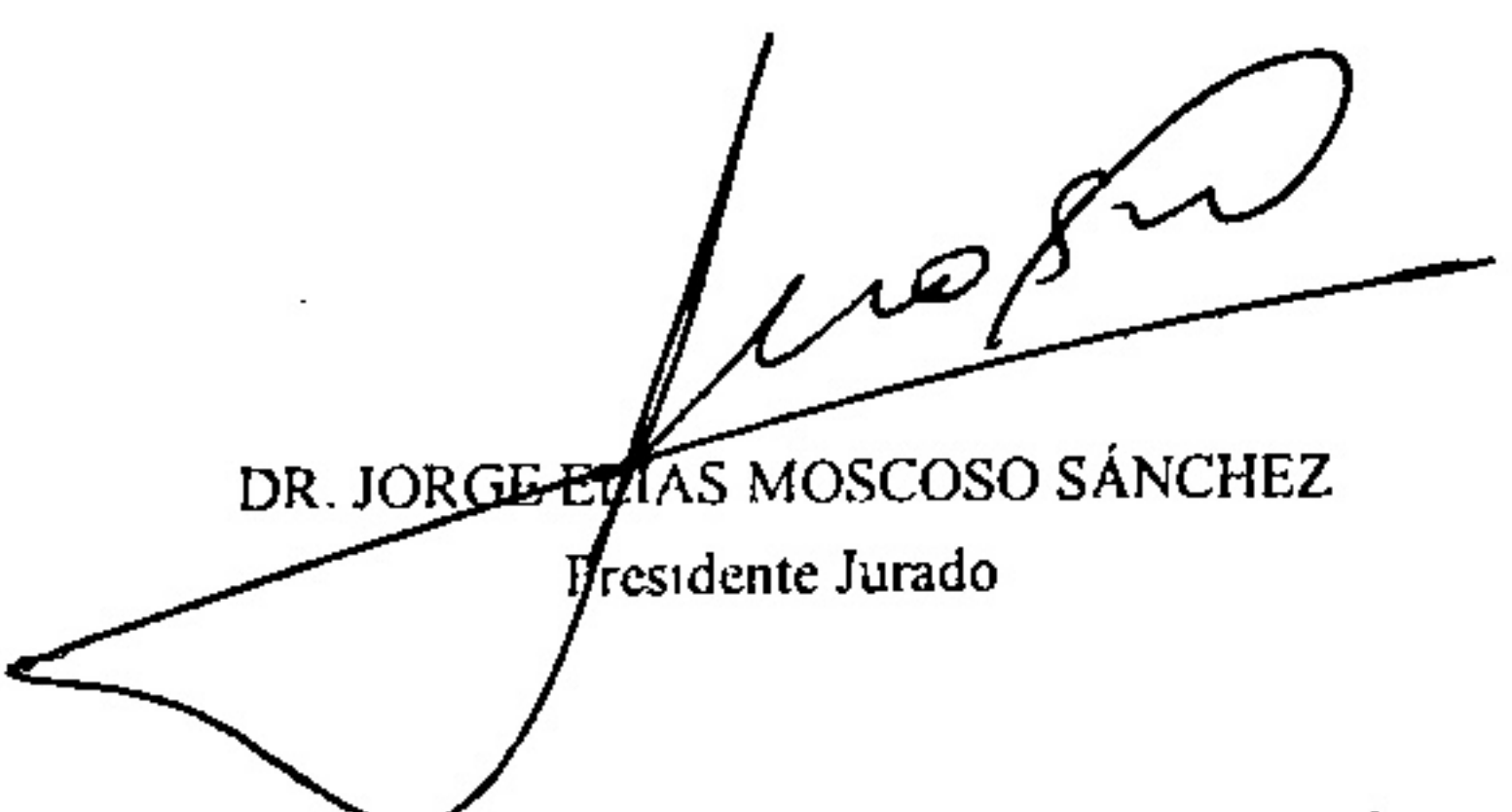
ACTA DE SUSTENTACIÓN
TRABAJO DE
SUFICIENCIA PROFESIONAL
ACTA N°55/FIEE-PPP/2026

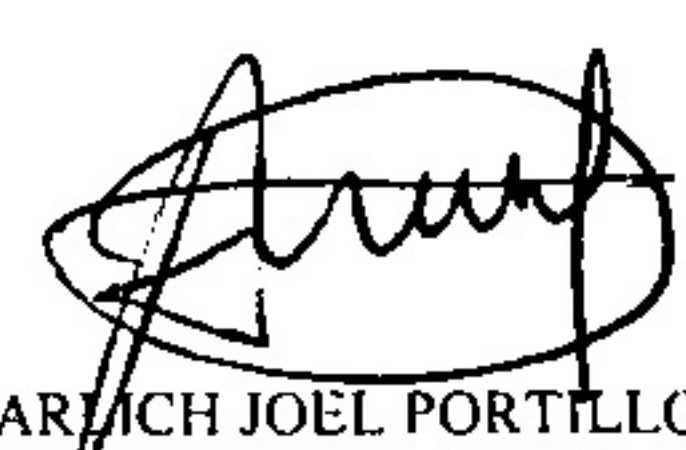
Los suscritos Miembros del Jurado, docentes permanentes de las Escuelas Profesionales de la Facultad de Ingeniería Electrónica y Eléctrica, reunidos en la fecha 13 de Agosto del 2026, como Presidente de Jurado el **DR. JORGE ELÍAS MOSCOSO SÁNCHEZ**, integrado por el Miembro de Jurado el **MG. DANIEL DÍAZ ATAUCURI**, el Miembro de Jurado el **MG. ARLICH JOEL PORTILLO ALLENDE** y Miembro Asesor el **MG. WILBERT CHAVEZ IRAZABAL**.

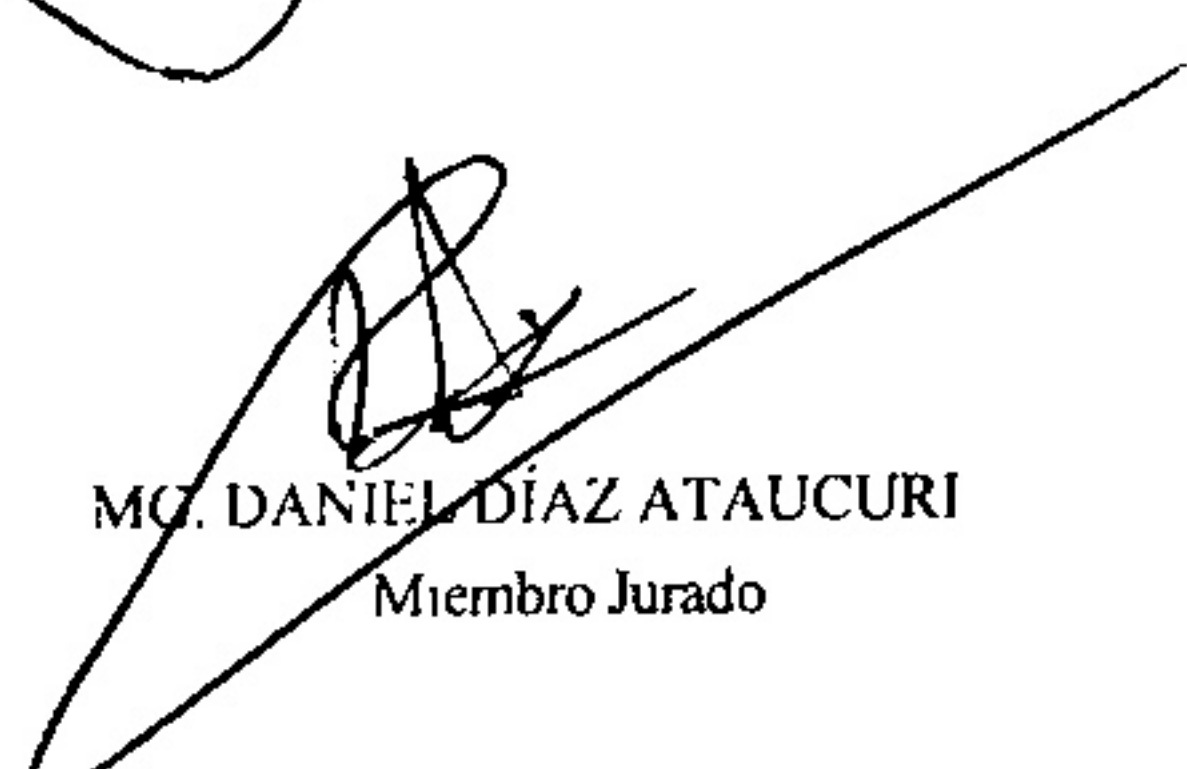
Después de escuchar la Sustentación Presencial del Trabajo de Suficiencia Profesional del Bachiller **ALVARO MAURICIO MENA FLORES** con código 11190097 que para optar el Título Profesional de Ingeniero Electrónico sustentó el Trabajo de Suficiencia Profesional titulado **IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO OPEN SOURCE EN UNA EMPRESA DE TELECOMUNICACIONES**.

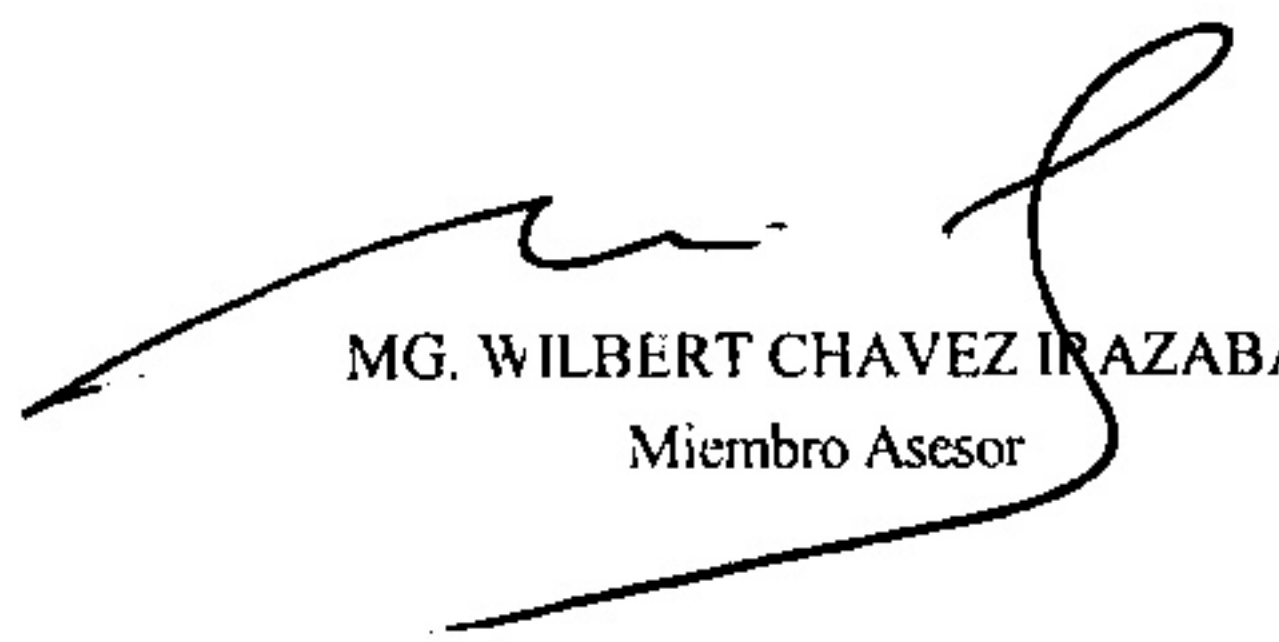
El jurado examinador procedió a formular las preguntas reglamentarias y, luego de una deliberación en privado, decidió otorgarle el calificativo de 16 (Dieciséis).

Ciudad Universitaria, 13 de Agosto del 2026


DR. JORGE ELÍAS MOSCOSO SÁNCHEZ
Presidente Jurado


MG. ARLICH JOEL PORTILLO ALLENDE
Miembro Jurado


MG. DANIEL DÍAZ ATAUCURI
Miembro Jurado


MG. WILBERT CHAVEZ IRAZABAL
Miembro Asesor



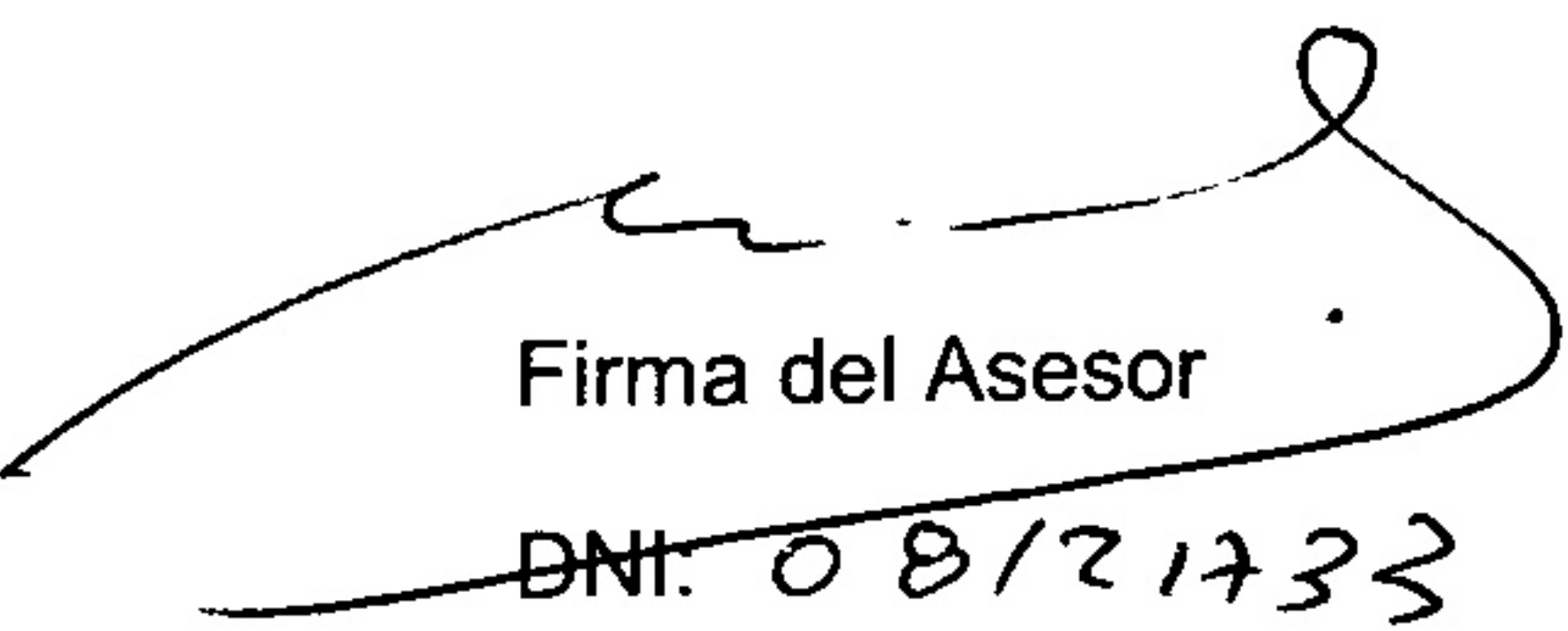
UNIVERSIDAD NACIONAL MAYOR DE
SAN MARCOS
Universidad del Perú, Decana de América

CERTIFICADO DE SIMILITUD

Yo, **WILBERT CHAVEZ IRAZABAL** en mi condición de asesor acreditado con el Dictamen N°57/FIEE-PPP/2026 del Trabajo de Suficiencia Profesional titulado **IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO OPEN SOURCE EN UNA EMPRESA DE TELECOMUNICACIONES**, presentado por el bachiller **ALVARO MAURICIO MENA FLORES**, para optar el título profesional de Ingeniero Electrónico. CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante la plataforma PIDS, el documento evaluado cuenta con el 4% de similitud, nivel PERMITIDO para continuar con los trámites correspondientes y para su publicación en el repositorio institucional.

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del título profesional de Ingeniero Electrónico.

Lima, 13 de Agosto del 2026



Firma del Asesor

DNI: 0 8121733

Nombres y apellidos del asesor:
WILBERT CHAVEZ IRAZABAL

DEDICATORIA

Este trabajo está dedicado a mi madre y a mi abuela, por darme su amor incondicional, sus cuidados y sus fuerzas para salir adelante. Me lo dedico también a mí, por mi esfuerzo, mis noches sin dormir, las celebraciones perdidas, por todo lo que sacrifiqué y sembré por un objetivo mayor, del cual hoy cosecho sus frutos.

AGRADECIMIENTOS

Agradezco a mi familia, por haber sido mi soporte continuo en este viaje llamado vida, y a mi querida y gloriosa Universidad Nacional Mayor de San Marcos por haberme formado y darme la oportunidad de ser ingeniero.

RESUMEN

El presente trabajo describe la implementación de una plataforma de monitoreo *open source* basada en Zabbix en una empresa líder del sector telecomunicaciones en el Perú. Esta empresa utilizaba distintas herramientas licenciadas, como CA UIM para el monitoreo de componentes de infraestructura TI, SolarWinds para el monitoreo de equipos de red y Humio para la gestión de *logs*. Sin embargo, estas plataformas se encontraban en versiones antiguas y generaban silos de información. Esto dificultaba tener una visión integral del estado de la infraestructura tecnológica y la correlación de eventos entre componentes. Como resultado, la detección de fallas no era oportuna e impactaba la gestión de incidentes y la continuidad de los servicios.

Debido al presupuesto limitado, se decidió utilizar una plataforma de monitoreo *open source* que cumpla con los estándares de seguridad, innovación y funcionalidad requeridos. Este proyecto, desarrollado a lo largo de un año, tuvo como objetivo incrementar la disponibilidad de los servicios que soportan al negocio. Para ello, se implementaron indicadores y mecanismos de detección proactiva de fallos, alertamiento automático y acciones de auto remediación. Esta implementación no solo incrementó la disponibilidad de los servicios que soportan al negocio, sino que también estableció una respuesta proactiva ante incidentes, mejoró significativamente la experiencia de los usuarios finales y evitó altos costos de licenciamiento asociados.

Palabras clave: monitoreo, *open source*, Zabbix, equipos de red, infraestructura TI, *logs*.

ABSTRACT

This paper describes the implementation of an open-source monitoring solution based on Zabbix in a leading telecommunications company in Peru. The company had been using different licensed tools, such as CA UIM for monitoring IT infrastructure components, SolarWinds for monitoring network devices, and Humio for log management. However, these platforms were running on outdated versions and created information silos. This made it difficult to obtain a comprehensive view of the technological infrastructure status and to correlate events across components. As a result, fault detection was not timely, negatively impacting incident management and service continuity.

Due to budget constraints, it was decided to implement an open-source monitoring solution that met the required standards of security, innovation, and functionality. This project, developed over the course of one year, aimed to increase the availability of business-supporting services. To achieve this, proactive failure detection indicators and mechanisms were implemented, along with automatic alerting and self-remediation actions. This implementation not only increased the availability of the services supporting the business, but also established a proactive incident response approach, significantly improved the end-user experience, and avoided high associated licensing costs.

Keywords: monitoring, open source, Zabbix, network devices, IT infrastructure, logs.

ÍNDICE GENERAL

CARATULA	i
RESUMEN	iv
ABSTRACT	v
ÍNDICE GENERAL	vi
LISTA DE FIGURAS	ix
LISTA DE TABLAS	xi
CAPÍTULO I: INTRODUCCIÓN	1
1.1. Objetivo del informe	1
1.2. Estructura del informe	1
CAPITULO II: INFORMACIÓN DEL LUGAR DONDE SE DESARROLLÓ LA ACTIVIDAD	3
2.1. Institución – Actividad que desarrolla	3
2.2. Periodo de duración de la actividad.	3
2.3. Finalidad y objetivos de la entidad	3
2.4. Razón social	4
2.5. Dirección postal	4
2.6. Nombre y correo electrónico del profesional a cargo.	4
CAPITULO III: DESCRIPCIÓN DE LA ACTIVIDAD	5
3.1. Organización de la actividad	5
3.2. Finalidad y objetivos de la Actividad	5
3.2.1. Finalidad	5
3.2.2. Objetivos	6

3.3. Problemática	6
3.3.1. Problema General	8
3.3.2. Problemas Específicos	8
3.3.3. Justificación e importancia de la investigación	8
3.4. Metodología	10
3.4.1. Bases teóricas	11
3.4.2. Marco conceptual	16
3.5. Procedimiento	22
3.5.1. Fase de análisis	22
3.5.2. Fase de diseño	26
3.5.3. Fase de implementación	31
3.5.4. Fase de pruebas	39
3.5.5. Fase de marcha blanca	44
3.5.6. Fase de operación	49
3.6. Resultados de la actividad	53
CAPITULO IV: DISCUSIÓN DE RESULTADOS	55
4.1. Descripción de la implementación	55
4.2. Evaluación Económica	57
CAPITULO V: CONCLUSIONES	59
CAPITULO VI: RECOMENDACIONES	60
CAPITULO VII: REFERENCIAS BIBLIOGRÁFICAS	61
CAPITULO VIII: ANEXOS	64
ANEXO A: Glosario de términos	64
ANEXO B: Contenido del archivo de configuración de Zabbix Server	65
ANEXO C: Contenido del archivo de configuración de Zabbix FrontEnd	90

ANEXO D: Severidad de alertas configuradas	91
ANEXO E: Configuración de notificación por correo electrónico	91

LISTA DE FIGURAS

Figura 1 Organigrama de la empresa en la cual se ejecutó el trabajo	5
Figura 2 Arquitectura de la plataforma de monitoreo implementada	27
Figura 3 Arquitectura del clúster de bases de datos	28
Figura 4 Estado, replicación y roles del clúster de bases de datos	29
Figura 5 Plataforma Zabbix funcionando	34
Figura 6 Segmentación del monitoreo según el entorno.	35
Figura 7 Recepción de métricas en la plataforma	40
Figura 8 Alertas generadas por la plataforma de monitoreo	41
Figura 9 Dashboard principal	42
Figura 10 Métricas de salud del clúster de base de datos	43
Figura 11 Alertas para el monitoreo del clúster de base de datos	43
Figura 12 Recalibración de lógica de alertamiento	45
Figura 13 Configuración de dependencia de alertas	46
Figura 14 Acción de auto remediación configurada	47
Figura 15 Notificación por correo debido a escalamiento	48
Figura 16 Monitoreo de salud de la plataforma	48
Figura 17 Segmentación de permisos	51

LISTA DE TABLAS

Tabla 1 Inventario de recursos tecnológicos a monitorear	24
Tabla 2 Detalle de infraestructura de los servidores de la plataforma de monitoreo	30
Tabla 3 Tipo de dispositivo o recurso y protocolo de monitoreo utilizado	36
Tabla 4 Tipo de dispositivo o recurso y métricas claves obtenidas	37
Tabla 5 Tipo de dispositivo o recurso y alertas importantes	38
Tabla 6 Resumen de resultados antes y después de la implementación	54
Tabla 7 Análisis económico, costo de licenciamiento anual en dólares americanos	58

CAPÍTULO I: INTRODUCCIÓN

1.1. Objetivo del informe

Una de las empresas de telecomunicaciones líderes en el Perú necesitaba contar con un sistema de monitoreo que le permita tener visibilidad completa del estado de salud y uso de su infraestructura TI de servidores, bases de datos, redes y servicios que soportan su operación y negocio. Esto debido a la afectación de la disponibilidad de sus servicios que sufría continuamente, lo cual afectaba directamente a los ingresos y la reputación de la empresa.

Siendo una limitante importante el presupuesto designado por la empresa para la implementación de este servicio de monitoreo, se realizó un análisis interno por parte de las áreas de TI involucradas y se decidió que el sistema de monitoreo sea *open source*.

El presente trabajo explica la manera en la cual se implementó este sistema de monitoreo *open source* llamado Zabbix, las consideraciones que se tuvieron, los requerimientos necesarios, el alcance del monitoreo, las mejoras aplicadas a los procesos asociados, la mejora continua aplicada al sistema de monitoreo y los resultados obtenidos.

1.2. Estructura del informe

El presente trabajo se desarrolla en nueve capítulos. El Capítulo 1 presenta la introducción al trabajo, señalando su propósito y alcance. En el Capítulo 2 se detalla la información de la entidad en la que se llevó a cabo la actividad. El Capítulo 3 describe la actividad realizada, incluyendo los objetivos, los

problemas identificados, la finalidad del trabajo, la metodología empleada, los sustentos teóricos, los conceptos y definiciones, así como el desarrollo completo de la actividad. El Capítulo 4 presenta los resultados obtenidos tras la implementación, mientras que el Capítulo 5 expone las conclusiones del autor derivadas del análisis del trabajo realizado. En el Capítulo 6 se incluyen las recomendaciones basadas en la experiencia adquirida durante el desarrollo del trabajo. El Capítulo 7 recopila las referencias bibliográficas que respaldan los conceptos y definiciones utilizadas, y el Capítulo 8 incluye los anexos asociados, que presentan información complementaria y soporte documental.

CAPITULO II: INFORMACIÓN DEL LUGAR DONDE SE DESARROLLÓ LA ACTIVIDAD

2.1. Institución – Actividad que desarrolla

Empresa de telecomunicaciones líder en el Perú que brinda servicios móviles, internet fijo y servicios de valor agregado. No se puede especificar el nombre por motivos de confidencialidad.

2.2. Periodo de duración de la actividad.

Enero 2024 – diciembre 2024.

2.3. Finalidad y objetivos de la entidad

Esta empresa líder en el Perú tiene como finalidad principal brindar servicios de telecomunicaciones que permitan la conectividad de personas, empresas e instituciones, contribuyendo al desarrollo digital del país. A través de la provisión de servicios de telefonía, internet y transmisión de datos, la organización busca garantizar la disponibilidad, calidad y continuidad de sus servicios, asegurando una experiencia confiable para sus usuarios y fomentando la transformación digital en los diferentes sectores económicos.

Esta organización tiene como objetivos principales:

- Proveer servicios de telecomunicaciones con altos niveles de disponibilidad, calidad y seguridad.

- Mejorar la experiencia del cliente mediante la mejora continua de los servicios ofrecidos, garantizando estabilidad, velocidad y confiabilidad.
- Expandir y modernizar la infraestructura de red y servicios digitales, con el fin de mejorar la cobertura y capacidad de atención a los usuarios.
- Optimizar la gestión operativa mediante el uso de tecnologías que permitan una supervisión eficiente, detección temprana de fallas y reducción de tiempos de respuesta ante incidentes.
- Impulsar la transformación digital mediante la adopción de soluciones tecnológicas innovadoras que contribuyan al desarrollo del negocio y del entorno digital del país.

2.4. Razón social

No se puede brindar la razón social por motivos de confidencialidad.

2.5. Dirección postal

Lima 15001, Perú.

2.6. Nombre y correo electrónico del profesional a cargo.

Martin Carranza, martin.carranza@tdilatam.com. Jefe de servicios.

CAPITULO III: DESCRIPCIÓN DE LA ACTIVIDAD

3.1. Organización de la actividad

El presente trabajo se realizó dentro de la gerencia de operaciones de TI. El organigrama representa las 3 direcciones generales con las que cuenta la empresa y ubica la Gerencia de Operaciones TI dentro de su dirección correspondiente (Ver Figura 1).

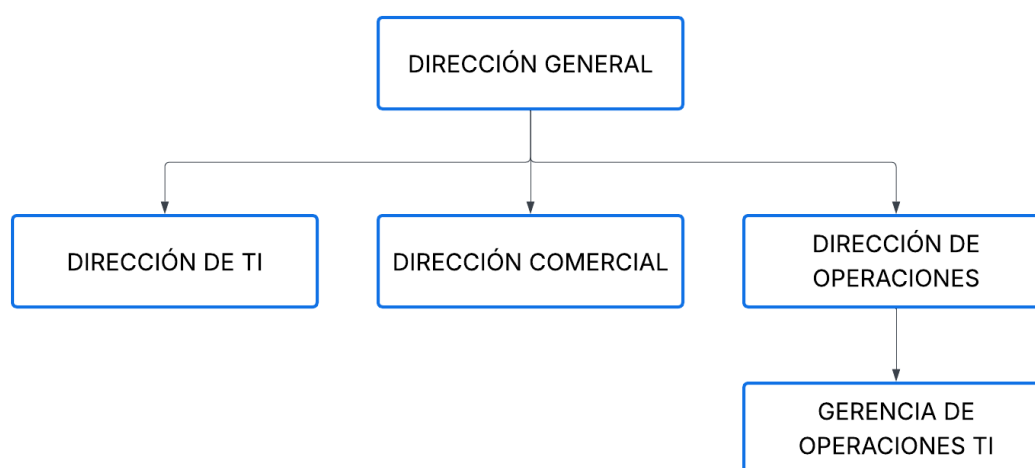


FIGURA 1 Organigrama de la empresa en la cual se ejecutó el trabajo. Fuente. Elaboración propia (2026).

3.2. Finalidad y objetivos de la Actividad

3.2.1. Finalidad

Implementar un sistema de monitoreo *open source* que brinde visibilidad completa del estado de salud y uso de su infraestructura TI de servidores,

bases de datos, redes y servicios que soportan los procesos de negocio. Así mismo debe centralizar e incrementar la cantidad de dispositivos monitoreados, reducir el MTTR y aumentar la disponibilidad de los servicios por encima del 90%.

3.2.2. Objetivos

Objetivo General

Implementar un sistema de monitoreo *open source* que permita mejorar la visibilidad, el alertamiento y la capacidad de respuesta ante incidentes en la infraestructura, los servicios y componentes de TI.

Objetivos específicos:

1. Implementar una plataforma de monitoreo centralizada con mecanismos de supervisión multicomponente y multiprotocolo para la infraestructura tecnológica.
2. Identificar e implementar indicadores claves y alertas proactivas para los servicios y componentes de TI.
3. Configurar un mecanismo de notificación automática por correo electrónico ante eventos, y la generación automática de tickets para alertas críticas.
4. Implementar mecanismos de auto escalamiento y auto remediación para la gestión de incidentes.

3.3. Problemática

Asegurar y mantener la continuidad de los servicios tecnológicos que soportan al negocio de las empresas se ha vuelto un factor clave dentro de sus operaciones. Una de las empresas líderes del sector telecomunicaciones en el Perú ha venido enfrentando constantes dificultades para mantener la disponibilidad de sus servicios debido a problemas en su infraestructura

tecnológica. Los servidores, las bases de datos, dispositivos de red y servicios que soportan su operación sufren caídas frecuentes y en muchos casos las fallas no se detectan a tiempo, lo que termina afectando el servicio que ofrece a sus clientes, generando pérdidas económicas y dañando la reputación de la organización.

La compañía operaba con herramientas de monitoreo licenciadas en versiones antiguas, sin soporte oficial y con limitaciones importantes. Estas plataformas presentaban fallos recurrentes, perdían datos críticos de supervisión y trabajaban de manera aislada, sin ofrecer una visión consolidada de toda la infraestructura y servicios. Esta situación impedía tomar decisiones rápidas y precisas frente a incidentes, pues la información estaba fragmentada en diferentes sistemas.

Otro aspecto relevante es que no se disponía de personal con la capacitación adecuada para mantener y evolucionar las soluciones existentes. Los equipos de TI no podían desarrollar monitores específicos para servicios propios del negocio ni garantizar la continuidad de las herramientas. Como consecuencia, los tiempos de respuesta ante incidentes eran elevados y la recuperación de los servicios demoraba más de lo aceptable para una empresa que depende de su infraestructura TI las 24 horas del día.

A todo ello se sumaba una restricción presupuestaria importante. La compañía no consideraba viable invertir en soluciones de monitoreo licenciadas debido a los altos costos que les representaba, por lo que las áreas técnicas tuvieron que evaluar alternativas diferentes. Tras un análisis interno, se concluyó que la mejor opción era implementar un sistema de monitoreo basado en software *open source*, que ofreciera flexibilidad, escalabilidad y, sobre todo, la posibilidad de centralizar la visibilidad de toda la infraestructura en una sola plataforma con un costo de implementación poco significativo.

En este contexto, surge la necesidad de implementar un sistema de monitoreo integral que permita detectar fallas en tiempo real, emitir alertas automáticas, reducir los tiempos de respuesta mediante acciones de auto remediación y consolidar la supervisión de toda la infraestructura tecnológica en un entorno

unificado. Todo ello bajo un enfoque que no solo atienda la parte técnica, sino que también se alinee a buenas prácticas de gestión como ITIL v4 y contemple aspectos de seguridad de la información.

3.3.1. Problema General

¿Cómo mejorar la visibilidad, el alertamiento y la capacidad de respuesta ante incidentes en la infraestructura tecnológica y los servicios de TI en una empresa de telecomunicaciones del Perú, con un presupuesto reducido?

3.3.2. Problemas Específicos

- ¿Cómo centralizar la supervisión de los diferentes componentes de la infraestructura tecnológica tales como servidores, bases de datos, dispositivos de red y servicios?
- ¿Cómo reducir el tiempo de detección de fallas o caídas en los servicios de TI?
- ¿Cómo mejorar la notificación de fallas en los servicios de TI?
- ¿Cómo reducir el tiempo de respuesta ante incidentes técnicos?

3.3.3. Justificación e importancia de la investigación

La continuidad operativa de los servicios y de la infraestructura tecnológica que los soporta se ha convertido en un aspecto crítico para las organizaciones, especialmente en empresas del sector telecomunicaciones, donde la disponibilidad de los sistemas impacta directamente en la calidad del servicio y en la satisfacción de los clientes. Para garantizar dicha continuidad, resulta indispensable contar con herramientas que permitan monitorear la infraestructura tecnológica de manera integral, oportuna y confiable, facilitando la detección temprana de fallas y una adecuada gestión de incidentes.

En este contexto, el desarrollo del presente trabajo se justifica desde el punto de vista práctico debido a la necesidad real que tenía la empresa de telecomunicaciones de mejorar la visibilidad, el monitoreo y la capacidad de respuesta ante incidentes en su infraestructura tecnológica. Antes de la implementación del sistema de monitoreo *open source*, la organización enfrentaba constantes caídas de servicios, detección tardía de fallas y una gestión reactiva ante incidentes, lo cual impactaba negativamente en la continuidad operativa y en la calidad del servicio brindado a los clientes.

La implementación de una plataforma de monitoreo centralizada basada en Zabbix permitió supervisar de manera integral servidores, bases de datos, dispositivos de red y servicios críticos del negocio, facilitando la detección temprana de fallos y la generación de alertas proactivas en tiempo real. Asimismo, la configuración de notificaciones automáticas y acciones de auto remediación contribuyeron a reducir los tiempos de respuesta ante incidentes, fortaleciendo la gestión de TI y mejorando la disponibilidad de los servicios tecnológicos. Esta implementación ayudó a resolver un problema práctico concreto de la empresa, ya que proporcionó una solución viable y eficiente, alineándose con buenas prácticas de gestión de servicios de TI y permitiendo una operación más estable y confiable.

Desde el punto de vista económico, el desarrollo del presente TSP se justifica debido a que la plataforma de monitoreo implementada permitió suprimir los costos asociados al uso de herramientas de monitoreo licenciadas, las cuales eran costosas, obsoletas y carecían de soporte adecuado. Al optar por una plataforma *open source* de clase *enterprise* como Zabbix, la empresa evitó altos gastos por licencias y renovaciones, logrando una optimización del presupuesto destinado a tecnología.

Además, la mejora en la detección oportuna de incidentes y la reducción de tiempos de indisponibilidad de los servicios contribuyeron indirectamente a disminuir pérdidas económicas asociadas a interrupciones del servicio y reclamos de clientes. La plataforma de monitoreo implementada demostró ser

sostenible en el tiempo, ya que puede ser mantenida y escalada con recursos internos, sin requerir inversiones significativas adicionales.

Por lo tanto, el trabajo realizado no solo fue técnicamente viable, sino también económicamente rentable, ya que permitió maximizar el uso de los recursos existentes y generar un impacto positivo en la eficiencia operativa y financiera de la empresa.

3.4. Metodología

El presente trabajo se desarrolló bajo un enfoque aplicado, debido a que solucionó un problema real identificado en una empresa de telecomunicaciones en la cual se tenía limitada visibilidad y capacidad de respuesta ante incidentes en su infraestructura tecnológica. Asimismo, el trabajo tuvo un alcance descriptivo y no experimental, ya que se analizaron las condiciones existentes basándose en información obtenida del entorno operativo real.

La recopilación de información se realizó a través de técnicas de observación directa, analizando registros históricos de incidencias gestionadas por el área de TI, así como revisando y evaluando las herramientas de monitoreo utilizadas por la organización. Estas técnicas permitieron identificar las principales deficiencias en la supervisión de servidores, bases de datos, redes y servicios críticos para el negocio.

La ejecución del trabajo se desarrolló en diversas etapas. En primer lugar, se efectuó un diagnóstico de la situación actual de la infraestructura tecnológica y de los procesos de monitoreo existentes, con la finalidad de identificar los componentes críticos, los requerimientos operativos y objetivos del negocio. Posteriormente, se diseñó la arquitectura del sistema de monitoreo *open source*, y se definieron los indicadores, métricas y mecanismos de alertamiento necesarios para una supervisión efectiva.

En la siguiente etapa, se realizó la instalación y configuración de la plataforma de monitoreo Zabbix, integrando los distintos componentes de la infraestructura y configurando alertas automáticas, notificaciones y acciones de auto remediación. Finalmente, se llevaron a cabo pruebas operativas y un proceso de mejora continua, evaluando y afinando el comportamiento del sistema y su impacto en la reducción de incidentes y en los tiempos de respuesta.

El análisis de la información se efectuó mediante la comparación de indicadores clave antes y después de la implementación del sistema de monitoreo, tales como la disponibilidad de los servicios, el tiempo medio de detección de fallas (MTTD) y el tiempo medio de solución de incidentes (MTTR). Los resultados obtenidos permitieron demostrar la efectividad de la plataforma de monitoreo implementada y extraer conclusiones respecto a su contribución a la mejora de la gestión de los servicios de TI y a la continuidad operativa de la organización.

3.4.1. Bases teóricas

3.4.1.1. Marco Teórico

Zabbix

Zabbix es una plataforma de monitoreo *open source* de nivel empresarial que permite supervisar el estado y rendimiento de servidores, redes, bases de datos y aplicaciones mediante el uso de agentes, protocolos estándar y mecanismos de recolección de métricas. Además, ofrece funcionalidades de alertamiento, visualización y automatización, lo que la convierte en una herramienta adecuada para la gestión integral de la infraestructura de TI (Zabbix, 2026a).

Zabbix es una solución de monitoreo distribuida, que tiene como componente central de la arquitectura de monitoreo al Zabbix Server, el cual es responsable de la recolección, procesamiento y análisis de la información

proveniente de los distintos elementos monitoreados. Este componente ejecuta mecanismos para la obtención de datos, evalúa las condiciones definidas para el alertamiento y genera notificaciones dirigidas a los usuarios ante la detección de eventos o incidentes. De manera complementaria, también tiene la capacidad de realizar verificaciones remotas sobre servicios de red, páginas web o dispositivos IP, mediante comprobaciones simples.

Desde el punto de vista arquitectónico, la plataforma de monitoreo se compone de tres elementos fundamentales: el Zabbix Server, la interfaz web (Zabbix Frontend) y la base de datos. Esta última desempeña un rol crítico, ya que almacena toda la información de configuración, así como los datos operativos y estadísticos del sistema. Tanto el Zabbix Server como la interfaz web interactúan constantemente con la base de datos; por ejemplo, cuando se crea una nueva métrica desde la interfaz web, este es registrado en la base de datos y posteriormente consultado por el Zabbix Server, el cual almacena temporalmente esta información en memoria caché para optimizar su procesamiento. Este mecanismo puede generar un ligero retraso en la visualización de cambios recientes dentro de la plataforma (Zabbix, 2026b).

Otros componentes importantes para el correcto monitoreo de los elementos supervisados son el agente de Zabbix (Zabbix Agent) y el Zabbix Proxy.

El agente de Zabbix se instala en los equipos o dispositivos que se desean supervisar. Su función principal es recopilar información local sobre el estado de los recursos del sistema, tales como el uso de CPU, memoria, almacenamiento y otros parámetros operativos relevantes.

El agente obtiene estos datos directamente desde el sistema monitoreado mediante el uso de llamadas nativas, lo que le permite operar de manera eficiente y con bajo consumo de recursos. Posteriormente, esta información es enviada al Zabbix Server para su procesamiento y análisis.

En cuanto a su funcionamiento, el agente soporta dos tipos principales de verificación: chequeos pasivos y activos. En los chequeos pasivos, el agente responde a solicitudes realizadas por el Zabbix Server o Proxy, devolviendo la información requerida en el momento en que es consultada. Por otro lado,

en los chequeos activos, el agente recopila información de manera autónoma y la envía periódicamente al Zabbix Server o Proxy, previa obtención de la lista de ítems que debe monitorear. Esta flexibilidad permite adaptar el monitoreo según las necesidades de la infraestructura y optimizar el uso de recursos de red. El agente de Zabbix es compatible con múltiples plataformas, incluyendo sistemas Windows y Unix, donde se ejecuta directamente sobre los servidores monitoreados, permitiendo una supervisión continua y detallada de los componentes del sistema (Zabbix, 2026c).

Por otro lado, el Zabbix Proxy es un componente que actúa como intermediario entre los dispositivos monitoreados y el Zabbix Server, permitiendo la recolección de datos desde uno o varios elementos de la infraestructura. Este componente opera en representación del Zabbix Server, recopilando información de los dispositivos y almacenándola temporalmente de forma local antes de enviarla al Zabbix Server para su procesamiento.

La implementación de Zabbix Proxy dentro de la arquitectura es opcional; sin embargo, resulta altamente recomendable en entornos de gran escala o distribuidos, ya que permite balancear la carga del Zabbix Server. Al delegar la recolección de datos al Zabbix Proxy, se reduce el consumo de recursos del Zabbix Server, especialmente en términos de uso de CPU y operaciones de entrada/salida en disco, optimizando así el rendimiento general de la plataforma. Asimismo, el uso de Zabbix Proxy es especialmente útil en escenarios donde se requiere monitorear ubicaciones remotas, sucursales o redes geográficamente distribuidas. En estos casos, el Zabbix Proxy facilita la centralización del monitoreo, asegurando la continuidad en la recolección de datos incluso ante posibles interrupciones en la comunicación con el Zabbix Server (Zabbix, 2026d).

3.4.1.2. Antecedentes

El desarrollo del presente trabajo tiene como antecedentes experiencias y estudios realizados a nivel nacional e internacional.

Tal como evidenció Guevara (2023) en los resultados de su tesis, la implementación de Zabbix como plataforma de monitoreo permitió mejoras significativas en la detección de anomalías de rendimiento dentro de la infraestructura de TI. Como consecuencia, se observó un incremento en la satisfacción de los usuarios de los servicios y en la eficiencia de la gestión operativa. Estos hallazgos corroboran que la adopción de mecanismos de alertamiento automatizado y monitoreo centralizado no solo fortalece el soporte técnico, sino que también establece una base sólida para una gestión eficiente de la infraestructura de TI, afirmándose como una solución eficaz frente a las necesidades críticas de disponibilidad en organizaciones de alta demanda.

Paredes (2025) en su estudio desarrollado en la Escuela de Postgrado CAEN-Tech, implementó una solución de monitoreo basada en Zabbix con el propósito de fortalecer la gestión operativa y garantizar la continuidad de los servicios tecnológicos críticos. Los resultados evidenciaron mejoras significativas en el control y supervisión de los recursos de TI, permitiendo reducir los tiempos de inactividad de los sistemas, optimizar la capacidad de respuesta ante incidentes y aumentar la visibilidad del entorno tecnológico. Asimismo, la implementación contribuyó al fortalecimiento de las competencias técnicas del personal encargado de las labores de monitoreo y administración de sistemas.

Desde el punto de vista económico, esta investigación concluyó que la adopción de Zabbix representó una alternativa viable frente a soluciones propietarias, generando ahorros anuales aproximados de S/ 10,000. Así mismo, la evaluación financiera del proyecto mostró indicadores favorables de rentabilidad y sostenibilidad, demostrando que la implementación de una plataforma de monitoreo basada en software libre puede aportar beneficios tanto operativos como económicos para la organización que la implementa.

Ccorimanya (2024), evidenció en su trabajo de suficiencia profesional que la implementación del sistema de monitoreo automatizado con Zabbix en la empresa Alo Call SAC generó mejoras sustanciales en la gestión de

incidentes y en la eficiencia operativa del área de TI. Antes de su adopción, los tiempos de atención superaban las 4 horas y el 70% de las veces las caídas de servicio alcanzaban más de 4 por semana; sin embargo, tras la implementación los tiempos de atención mayores a 1 hora se redujeron a cero y el número máximo de caídas semanales disminuyó a 2, con el 70% de las veces limitado a solo una caída. Asimismo, el tiempo de identificación de problemas pasó de depender del reporte de usuarios a resolverse en menos de 30 minutos. La percepción de los usuarios también cambió radicalmente, pasaron de considerar el sistema de monitoreo, antes de la implementación, como “nada efectivo” a calificarlo, tras la implementación, como “muy efectivo” en la prevención de fallos. Finalmente, los indicadores financieros reflejaron la rentabilidad del proyecto, con un incremento del ROE de 57% en el primer año a 83% en el tercero, y del ROA de 42% a 72% en el mismo periodo, confirmando que la inversión en Zabbix no solo optimizó la infraestructura tecnológica, sino que también fortaleció la solidez financiera de la organización.

Gómez (2025), en su tesis, muestra que el desempeño de Zabbix dentro de una operación de TI, reporta resultados altamente favorables en diversos indicadores clave. La claridad visual alcanzó una puntuación promedio de 4.5 sobre 5, evidenciando una amplia aceptación por parte de los usuarios técnicos, quienes destacaron la utilidad de los dashboards para interpretar de manera rápida y precisa el estado de los dispositivos críticos. Asimismo, los tiempos de alerta obtuvieron la valoración más alta (4.7 sobre 5), lo que resalta la capacidad del sistema para generar notificaciones inmediatas y precisas, contribuyendo a una mejora significativa en la capacidad de respuesta ante incidentes. De igual manera, la adaptabilidad presentó un promedio de 4.6 sobre 5, demostrando una integración eficiente con infraestructuras heterogéneas, incluyendo servidores, redes, sistemas de energía y sensores ambientales. Por otro lado, la facilidad de uso registró una puntuación ligeramente menor (4.2 sobre 5), lo que sugiere la existencia de ciertas dificultades iniciales en el manejo de la plataforma, posiblemente asociadas a la necesidad de mayor capacitación o documentación técnica. En cuanto a la integración de protocolos, se obtuvo un promedio de 4.4 sobre 5, destacando

su compatibilidad con tecnologías como SNMP e IPMI, aunque con oportunidades de mejora en su configuración inicial. Finalmente, la reducción de falsos positivos alcanzó una puntuación de 4.5 sobre 5, validando la eficacia del sistema para filtrar eventos irrelevantes y proporcionar alertas oportunas, optimizando así la gestión operativa.

Ramírez (2023), en su trabajo realizado, para una empresa de telecomunicaciones líder en Colombia, optimizó la gestión de incidentes de la amplia red de fibra óptica que esta empresa poseía. Antes de la implementación de la plataforma de monitoreo basada en Zabbix, los procesos de identificación y solución de fallas eran prolongados, ya que el descubrimiento de los problemas dependía de múltiples herramientas y generaba retrasos significativos en el escalamiento, ocasionando pérdidas económicas por la interrupción de servicios. Con el diseño e implementación de la plataforma de monitoreo basada en Zabbix, se logró reducir el tiempo de detección y reacción a aproximadamente 15 minutos, lo que representa una disminución de más del 50 % respecto a los tiempos antes de la implementación. La integración de información visual en listas y mapas en un solo entorno permitió al equipo de TI mantener una visión completa de la red, filtrar datos relevantes y diagnosticar fallas con mayor precisión, mejorando la continuidad del servicio y la eficiencia operativa de la organización.

3.4.2. Marco conceptual

Alertas

Las alertas son el mecanismo mediante el cual un sistema de monitoreo notifica de manera automática la ocurrencia de eventos o condiciones anómalas de los elementos monitoreados. Un sistema de alertamiento efectivo permite que los equipos técnicos actúen de forma proactiva, reduciendo los tiempos de detección y respuesta ante incidentes.

API

Una API (*Application Programming Interface*) es un conjunto de reglas, protocolos y herramientas que permite la comunicación e interacción entre diferentes aplicaciones o sistemas, facilitando el intercambio de datos y funcionalidades sin necesidad de conocer la implementación interna de cada sistema (Richardson & Amundsen, 2013).

Auto remediación

La auto remediación es un mecanismo mediante el cual un sistema de monitoreo ejecuta acciones automáticas predefinidas para corregir o mitigar una falla sin intervención manual. Estas acciones pueden incluir el reinicio de servicios, la ejecución de scripts o la aplicación de configuraciones correctivas.

Bases de datos

Una base de datos es un sistema informático que proporciona servicios de almacenamiento, gestión y acceso a los datos almacenados, permitiendo que múltiples usuarios o aplicaciones consulten y manipulen la información de manera eficiente y segura (Elmasri & Navathe, 2016).

BMC Remedy

BMC Remedy es una plataforma de gestión de servicios de TI (ITSM) que permite la administración integral de procesos como la gestión de incidentes, problemas, cambios y solicitudes de servicio. Esta solución facilita la automatización de flujos de trabajo, la centralización de la información y la mejora en la atención de eventos críticos dentro de la infraestructura tecnológica. Asimismo, se encuentra alineada con las buenas prácticas

propuestas por ITIL, contribuyendo a una gestión más eficiente y estructurada de los servicios de TI (BMC Software, 2026).

Disponibilidad de servicios

La disponibilidad de los servicios se refiere a la capacidad de un sistema o servicio tecnológico para mantenerse operativo y accesible durante un periodo de tiempo determinado. En el contexto de las empresas de telecomunicaciones, la disponibilidad es un factor crítico, ya que las interrupciones del servicio impactan directamente en la operación del negocio y en la experiencia del cliente. La gestión de la disponibilidad es una práctica fundamental dentro de la gestión de servicios de TI (Axelos, 2019).

Dispositivos de red

Los dispositivos de red son equipos que permiten la interconexión y comunicación entre sistemas informáticos dentro de una red, facilitando el envío, recepción y gestión del tráfico de datos. Entre ellos, los switches operan en redes locales para conmutar paquetes entre dispositivos, los routers enrutan la información entre diferentes redes, y los balanceadores de carga distribuyen el tráfico entre múltiples servidores para optimizar el rendimiento y la disponibilidad del servicio (Tanenbaum & Wetherall, 2011).

Gestión de incidentes

La gestión de incidentes es el proceso orientado a restaurar la operación normal de un servicio lo más rápido posible ante la ocurrencia de una interrupción o degradación del mismo. Este proceso busca minimizar el impacto negativo en el negocio y está estrechamente relacionado con el

monitoreo y el alertamiento oportuno de fallas en la infraestructura tecnológica (Axelos, 2019).

Hipervisor

Un hipervisor es un software o capa de virtualización que permite crear y gestionar múltiples máquinas virtuales sobre un mismo hardware físico, asignando recursos como CPU, memoria y almacenamiento a cada máquina virtual, de manera eficiente y aislando cada sistema operativo para su funcionamiento independiente (Hwang, Fox, & Dongarra, 2012).

Integración de sistemas

Para efectos del presente trabajo, la integración de sistemas se define como el proceso mediante el cual dos o más plataformas tecnológicas independientes se conectan e intercambian información de manera automática, con el objetivo de optimizar procesos operativos y mejorar la eficiencia en la gestión de eventos. En este mismo contexto, la integración permite que herramientas como plataformas de supervisión y sistemas de gestión de servicios trabajen de manera conjunta, facilitando la automatización de acciones como la generación de tickets ante incidentes críticos (Hohpe & Woolf, 2003).

Plataforma de monitoreo

Una plataforma de monitoreo es un conjunto integrado de herramientas que permite supervisar de forma continua el estado, rendimiento y disponibilidad de la infraestructura y servicios de tecnologías de la información. Estas plataformas facilitan la recolección, procesamiento y visualización de métricas, así como la generación de alertas ante eventos anómalos,

contribuyendo a una gestión más eficiente de los servicios de TI y a la toma oportuna de decisiones (Axelos, 2019).

Proceso de negocio

Los procesos de negocio son un conjunto estructurado de actividades o tareas relacionadas que, al ejecutarse de manera secuencial o coordinada, generan valor económico o funcional para una organización, como por ejemplo la gestión de una compra a través de un portal web en un entorno de comercio electrónico (Davenport, 1993).

Servidores físicos y virtuales

Los servidores físicos son equipos de hardware dedicados que proporcionan servicios informáticos y ejecutan aplicaciones directamente sobre sus recursos, mientras que los servidores virtuales son instancias creadas mediante virtualización que operan sobre un servidor físico, permitiendo ejecutar múltiples sistemas operativos de forma independiente y optimizar el uso de los recursos (Erl, Puttini, & Mahmood, 2013).

Silo de información

En el ámbito de la gestión de infraestructuras tecnológicas, un silo de información representa una segmentación crítica de la información operativa. Se define como un repositorio de datos fijos que permanece bajo el control de un departamento o unidad específica, y que se encuentra aislado del resto de la organización debido a la falta de integración técnica o compatibilidad de sistemas. Según Hoffer et al. (2019), los silos de datos son el resultado de un desarrollo fragmentado de sistemas de información, donde cada aplicación es diseñada para resolver un problema específico sin considerar la

interoperabilidad, lo que genera una visión parcial e inconsistente de la salud de los servicios de TI.

Software de Código Abierto (Open Source)

El software de código abierto se define como un programa informático cuyo código fuente se distribuye bajo una licencia que permite a los usuarios estudiarlo, modificarlo y redistribuirlo libremente. Según Raymond (2001), este modelo se diferencia del desarrollo tradicional por su naturaleza colaborativa y descentralizada, lo que permite una corrección de errores más ágil y una evolución constante impulsada por la comunidad. En el contexto de la infraestructura de TI, el *open source* representa una alternativa estratégica frente a las soluciones propietarias, ya que elimina los costos de licenciamiento y evita la dependencia tecnológica de un único proveedor (*vendor lock-in*).

TI (Tecnologías de la información)

Las Tecnologías de la Información (TI) comprenden el conjunto de recursos tecnológicos, incluyendo hardware, software, redes y servicios, utilizados para la creación, almacenamiento, procesamiento y transmisión de información dentro de una organización (Laudon & Laudon, 2016).

Tiempo medio de detección (MTTD)

El tiempo medio de detección (*Mean Time to Detect*, MTTD) es un indicador que mide el tiempo promedio que transcurre desde que ocurre una falla o incidente en un sistema hasta que este es detectado por los mecanismos de monitoreo. Un MTTD bajo indica que la organización cuenta con herramientas y procesos eficaces para identificar de manera temprana las anomalías en su

infraestructura tecnológica, lo cual resulta fundamental para reducir el impacto de los incidentes y mejorar la continuidad de los servicios (Axelos, 2019).

Tiempo medio de recuperación (MTTR)

El tiempo medio de recuperación (*Mean Time to Repair o Mean Time to Resolve*, MTTR) es una métrica que representa el tiempo promedio necesario para restaurar un servicio o componente tecnológico a su estado operativo normal después de la ocurrencia de un incidente. Este indicador es clave en la gestión de servicios de TI, ya que permite evaluar la eficiencia de los procesos de atención de incidentes y la capacidad de respuesta del equipo técnico ante fallas que afectan la operación del negocio (Axelos, 2019).

3.5. Procedimiento

El procedimiento de ejecución del presente trabajo se desarrolló siguiendo un enfoque secuencial y estructurado, alineado a las buenas prácticas de gestión de proyectos bajo el modelo *waterfall*, tal como lo establece el *Project Management Institute* (PMI), y a las buenas prácticas de gestión de servicios de TI propuestas por ITIL v4. Este enfoque permitió asegurar un control adecuado de cada etapa del proyecto, minimizar riesgos y garantizar la estabilidad de la plataforma implementada.

El procedimiento se organizó en las siguientes fases: análisis, diseño, implementación, pruebas, marcha blanca y operación, las cuales se describen a continuación.

3.5.1. Fase de análisis

Esta fase, que tuvo una duración de 2 semanas, fue el punto de partida para la implementación del sistema de monitoreo, permitiendo comprender de

manera integral la situación inicial de la organización y establecer las bases técnicas y estratégicas del proyecto. En esta etapa se realizó un levantamiento detallado de información con el objetivo de identificar las limitaciones de las herramientas de monitoreo existentes, así como los riesgos operativos derivados de su uso.

En primer lugar, se identificó que la empresa utilizaba múltiples herramientas licenciadas que operaban de manera independiente, generando silos de información. Esta fragmentación impedía contar con una vista consolidada de la infraestructura tecnológica, dificultando la correlación de eventos y el análisis integral de incidentes. Cada plataforma supervisaba únicamente un componente específico (por ejemplo, servidores o red), lo que obligaba al equipo de operaciones a revisar manualmente distintos paneles para determinar el origen de una falla, lo cual incrementaba el tiempo promedio de detección y afectaba directamente la continuidad de los servicios.

Asimismo, se evidenció una supervisión incompleta en las diferentes capas de los servicios. Si bien algunos servidores contaban con monitoreo básico de recursos (CPU, memoria y almacenamiento), no existía una supervisión de los procesos, bases de datos ni puertos. Tampoco se monitoreaban adecuadamente los tiempos de respuesta de aplicaciones expuestas mediante URLs y APIs, lo que generaba una brecha entre la percepción del usuario final y la información disponible para el área de TI.

Otra limitación relevante fue la falta de un esquema de alertamiento oportuno y estructurado. Las herramientas existentes generaban notificaciones aisladas sin criterios de priorización ni escalamiento automático. En varios casos, las alertas se enviaban únicamente por correo electrónico sin integración con mecanismos de gestión de incidentes, lo que retrasaba la atención. Esta situación impactaba directamente en indicadores como el tiempo promedio de detección (MTTD) y el tiempo promedio de recuperación (MTTR), afectando la disponibilidad de los servicios críticos del negocio.

Adicionalmente, se identificó la ausencia de mecanismos de auto remediación. El entorno dependía exclusivamente de la intervención manual

del equipo técnico para la ejecución de acciones correctivas, incluso en incidentes repetitivos y previsibles, como reinicios de servicios o liberación de recursos. Esta dependencia incrementaba la carga operativa y reducía la eficiencia del área de infraestructura.

Como parte del levantamiento de información, se realizó también un inventario detallado de los componentes críticos de la infraestructura tecnológica. Entre ellos se identificaron servidores físicos y virtuales, hipervisores, bases de datos, dispositivos de red (switches y routers), balanceadores de carga, sistemas de almacenamiento y servicios expuestos al exterior mediante URLs y APIs. Este inventario permitió clasificar los activos según su nivel de criticidad para el negocio, priorizando aquellos que soportaban procesos de negocio críticos, la tabla siguiente muestra la cantidad de dispositivos que forman parte del alcance del monitoreo, agrupados según su criticidad para la empresa, su tipo y ubicación (ver Tabla 1).

Tabla 1 Inventario de recursos tecnológicos a monitorear

Tipo de recurso	Cantidad	Criticidad	Ubicación
Servidores virtuales	1200	Media	Cluster VMware / Hyper-V
Hipervisores y nodos	40	Crítica	Data Center principal y secundario
Bases de datos	50	Crítica	Servidores virtuales
Equipos de red	60	Alta	Sede central
Balanceadores	10	Crítica	Data Center principal y secundario
Servicios web	20	Crítica	Red interna y externa
Procesos de Sistema Operativo	100	Crítica	Servidores virtuales
Puertos y Servicios	50	Crítica	Servidores virtuales

Fuente. Elaboración propia (2026).

En paralelo, se analizaron los requerimientos funcionales y no funcionales que debía cumplir la nueva plataforma de monitoreo. En cuanto a los requerimientos funcionales, se estableció la necesidad de contar con monitoreo unificado, generación de alertas configurables, visualización centralizada mediante *dashboards*, monitoreo de disponibilidad de servicios web y capacidad de supervisión de distintas tecnologías. También se consideró indispensable la posibilidad de implementar acciones automatizadas ante determinados eventos conocidos.

Respecto a los requerimientos no funcionales, se evaluaron criterios de alta disponibilidad, escalabilidad, seguridad y optimización. La solución debía garantizar continuidad operativa incluso ante fallas parciales del sistema de monitoreo, permitir crecimiento progresivo conforme aumentara la infraestructura y cumplir con estándares de seguridad en la gestión de accesos y comunicaciones. Además, considerando el contexto de reducción presupuestal de la empresa, se priorizó una alternativa que evitara altos costos de licenciamiento sin sacrificar capacidades de nivel empresarial.

Como resultado del análisis comparativo de alternativas disponibles en el mercado, se determinó la adopción de Zabbix versión 6.0 LTS como plataforma de monitoreo *open source* de clase *enterprise*, debido a su madurez tecnológica, arquitectura flexible y amplia compatibilidad con diversos entornos (Zabbix, 2026a). Esta versión ofrece soporte a largo plazo *Long Time Support* (LTS), lo que garantiza estabilidad y actualizaciones de seguridad durante un periodo extendido, aspecto fundamental para entornos productivos.

La elección de Zabbix como plataforma de monitoreo no se basó únicamente en el factor económico, sino en su alineación con los objetivos estratégicos del proyecto: mejorar la visibilidad integral de la infraestructura, reducir los tiempos de detección y recuperación ante incidentes, e incorporar capacidades de auto remediación. Asimismo, su modelo modular permitió

proyectar un crecimiento ordenado del sistema de monitoreo conforme evolucionaran las necesidades del negocio.

En síntesis, la fase de análisis permitió identificar las brechas existentes en la gestión de monitoreo, definir los requerimientos técnicos y estratégicos del nuevo sistema, y sustentar la selección de la plataforma tecnológica que serviría como base para las fases posteriores del proyecto. Esta etapa resultó fundamental para asegurar que la implementación no respondiera únicamente a una necesidad técnica o económica, sino a una mejora sustancial orientada a fortalecer la disponibilidad y continuidad operativa de la organización.

3.5.2. Fase de diseño

La fase de diseño, con una duración de 1 semana, tuvo como objetivo definir la arquitectura técnica de la plataforma de monitoreo, asegurando que esta cumpliera con los requerimientos identificados en la etapa de análisis y estuviera alineada con principios de alta disponibilidad, escalabilidad, seguridad y buenas prácticas de arquitectura de tecnologías de información. En esta etapa no solo se definieron los componentes que conformarían la plataforma de monitoreo, sino también la forma en que interactuarían entre sí para garantizar continuidad operativa y rendimiento adecuado ante el crecimiento proyectado de la infraestructura.

Se eliminaron los puntos únicos de falla en los componentes críticos del sistema, especialmente en la capa de base de datos, considerando que el sistema de almacenamiento constituye el núcleo del funcionamiento de la plataforma de monitoreo. Asimismo, se adoptó un enfoque de distribución de funciones, separando roles para evitar sobrecarga en un único servidor y facilite la escalabilidad a futuro.

La arquitectura diseñada quedó compuesta por los siguientes elementos:

- Un servidor de base de datos activo.
- Un servidor de base de datos pasivo.

- Un servidor que aloja Zabbix Server y Zabbix Frontend, en el cual también se instaló el servicio de MariaDB MaxScale para la gestión del clúster de bases de datos.
- Un servidor Zabbix Proxy para el monitoreo distribuido.

La siguiente figura representa la arquitectura de servidores utilizados y el sentido del flujo de los datos (Ver Figura 2):

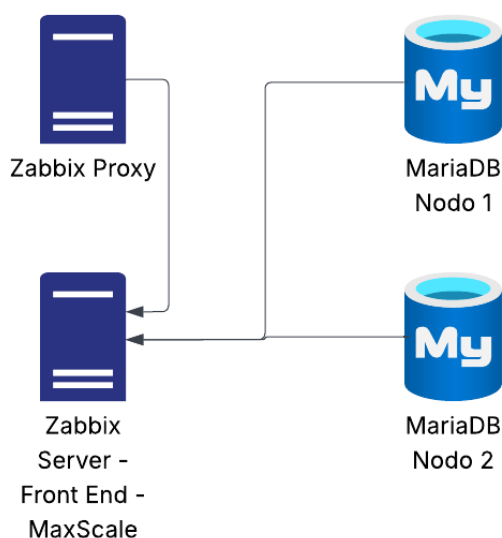


Figura 2 Arquitectura de la plataforma de monitoreo implementada.

Fuente. Elaboración propia (2026).

Las bases de datos fueron implementadas sobre MariaDB versión 10.6, desplegadas en un clúster de dos nodos configurados bajo un esquema activo-pasivo. Esta configuración permite que el nodo activo procese las operaciones de escritura, mientras que el nodo pasivo mantiene una réplica sincronizada de la información y ejecuta las operaciones de lectura, quedando disponible para asumir el rol principal ante una eventual falla.

La gestión del clúster se realizó mediante MariaDB MaxScale, componente que actúa como proxy inteligente entre la aplicación y los nodos de base de datos, permitiendo enrutar las conexiones de manera transparente según el estado de cada servidor (MariaDB, 2023). El diseño contempló que el servicio de Zabbix Server no se conectara directamente a un nodo específico de base

de datos, sino a través del servicio MaxScale que gestiona el clúster de base de datos y se comporta como una puerta que desacopla el acceso a la base de datos de los nodos que componen el clúster. La siguiente figura muestra la arquitectura de servidores que forman parte del clúster de base de datos y el sentido de conexión (ver Figura 3).

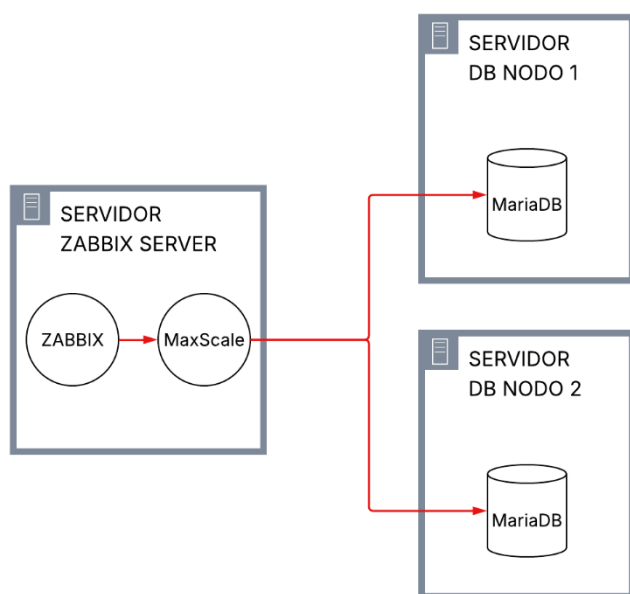


Figura 3 Arquitectura del clúster de bases de datos. Fuente. Elaboración propia (2026).

De esta forma, ante un cambio de rol entre nodos, la aplicación continuaría operando sin requerir reconfiguración manual. Esta decisión arquitectónica responde al principio de abstracción de la capa de datos y mejora la resiliencia del sistema.

MaxScale además proporciona capacidades de monitoreo del estado del clúster, validación de replicación y conmutación automática de roles entre los nodos, lo que reduce significativamente el riesgo de indisponibilidad prolongada. Estos datos de monitoreo del clúster pueden observarse con la ejecución del comando `"maxctrl list servers"`. La siguiente figura muestra la salida del comando y la información importante que brinda sobre la salud del clúster de base de datos y el estado de los nodos que la componen (ver Figura 4).

```
[root@zabbix-server ~]# maxctl list servers
```

Server	Address	Port	Connections	State	GTID
MariaDB-Node-1	192.168.1.10	3306	10	Master, Running	0-1-10521
MariaDB-Node-2	192.168.1.11	3306	5	Slave, Running	0-1-10521

Figura 4 Estado, replicación y roles del clúster de bases de datos.

Fuente. Elaboración propia (2026).

El servidor principal se configuró para alojar el Zabbix Server y Zabbix Frontend. En este mismo servidor se implementó MariaDB MaxScale, debido a que su consumo de recursos es relativamente bajo y su proximidad lógica al Zabbix Server facilita la gestión centralizada de conexiones hacia la base de datos. No obstante, el diseño consideró que, en caso de crecimiento futuro significativo, este componente podría migrarse a un servidor independiente sin afectar la arquitectura general.

Se definió además la habilitación de cifrado en las comunicaciones internas cuando fuera posible para reforzar la seguridad.

Con el fin de mejorar la eficiencia del monitoreo y reducir la carga directa sobre el Zabbix Server, se incorporó un Zabbix Proxy dentro de la arquitectura ya que permite recolectar datos de múltiples dispositivos en entornos remotos o segmentos específicos de red, enviándolos posteriormente al servidor central de manera consolidada. Esta decisión responde al principio de escalabilidad horizontal, permitiendo que el sistema crezca mediante la incorporación de nuevos Zabbix Proxies según sea necesario (Zabbix, 2026d).

El dimensionamiento de los servidores se realizó tomando como referencia la documentación oficial de Zabbix y la experiencia previa en implementaciones similares en entornos empresariales.

Se consideraron factores como:

- Cantidad proyectada de dispositivos a monitorear.
- Número estimado de métricas por dispositivo.
- Frecuencia de recolección de métricas.

- Retención histórica de datos.
- Crecimiento anual estimado.

A continuación, se detalla el dimensionamiento de los servidores de la plataforma de monitoreo, indicando su rol, recursos de CPU, memoria, disco, y el sistema operativo instalado en cada uno de ellos (Ver Tabla 2):

Tabla 2 Detalle de infraestructura de los servidores de la plataforma de monitoreo

Rol	CPU	RAM	Disco	Sistema Operativo
Zabbix Server, Zabbix Frontend, MaxScale	8	32 GB	50 GB	Red Hat Enterprise 8.0
Zabbix Proxy	8	32 GB	50 GB	Red Hat Enterprise 8.0
Base de datos Activo	8	32 GB	1000 GB	Red Hat Enterprise 8.0
Base de datos Pasivo	8	32 GB	1000 GB	Red Hat Enterprise 8.0

Fuente. Elaboración propia (2026).

Todos los servidores fueron dimensionados y configurados con recursos de hardware adecuados, asegurando la capacidad necesaria para el procesamiento eficiente de datos y la estabilidad operativa del sistema. En el caso de los servidores de base de datos, se asignó espacio de disco adicional debido a la proyección de almacenamiento de métricas históricas durante un periodo mínimo de un año. Esta retención prolongada de datos responde a la necesidad de realizar análisis de tendencias, evaluación de capacidad y generación de reportes históricos para la toma de decisiones, lo cual fue considerado crítico para aportar valor estratégico más allá del monitoreo operativo diario.

Además del diseño físico de la arquitectura, se definió la estructura lógica del sistema de monitoreo. Esta organización es fundamental para garantizar orden administrativo, escalabilidad y facilidad de mantenimiento.

Se creó una estructura de grupos de elementos monitoreados, basado en tres criterios principales:

- Entorno: producción y no producción.
- Función: servidores, bases de datos, hipervisores y dispositivos de red.
- Sistema operativo: Linux, Windows, Solaris y AIX.

Esta clasificación permite segmentar responsabilidades, aplicar plantillas específicas según tipo de sistema y generar reportes diferenciados por criticidad. Asimismo, facilita la asignación de permisos a distintos perfiles de usuario dentro del sistema, reforzando la gobernanza operativa.

Estas decisiones contribuyen a una administración ordenada y coherente del entorno, evitando configuraciones inconsistentes a medida que la plataforma crece.

En resumen, la fase de diseño fue clave ya que permitió establecer una arquitectura robusta, resiliente y preparada para el crecimiento. Asegurando que la solución no solo responda a las necesidades actuales, sino que también pueda adaptarse a la evolución futura de la infraestructura tecnológica monitoreada.

3.5.3. Fase de implementación

Durante esta fase, durante 2 semanas, se llevó a cabo la instalación, configuración e integración de todos los componentes que conforman la plataforma de monitoreo definida en la etapa de diseño. El objetivo principal fue materializar la arquitectura propuesta, garantizando la correcta comunicación entre los distintos elementos y asegurando su funcionamiento estable antes de pasar a la etapa de pruebas.

La instalación de los componentes se realizó sobre servidores con sistema operativo Red Hat Enterprise Linux 8, previamente preparados con las actualizaciones correspondientes y configuraciones básicas de red. Todas las actividades fueron ejecutadas utilizando el usuario root, considerando que se trataba de una implementación controlada dentro de un entorno corporativo y que era necesario contar con privilegios administrativos para la instalación de paquetes, configuración de servicios y ajuste de parámetros del sistema operativo. Los procedimientos seguidos se basaron en la documentación oficial de Zabbix 6.0 LTS, respetando las dependencias y recomendaciones del fabricante.

Debido a que todos los servidores de la plataforma de monitoreo y los equipos a monitorear se encontraban dentro de la misma red interna de la organización, protegida por controles de seguridad perimetral, se optó por deshabilitar el servicio de firewall local del sistema operativo en los servidores que componían la plataforma de monitoreo. Esta decisión se tomó para evitar bloqueos de puertos necesarios para la comunicación entre el Zabbix Server, el Proxy, los agentes y la base de datos.

No obstante, esta configuración se realizó bajo la confirmación de que el entorno estaba segmentado y protegido por dispositivos de seguridad externos, tales como firewalls perimetrales y políticas de acceso restringidas. Asimismo, se mantuvo un control de accesos basado en credenciales y roles dentro de la plataforma de monitoreo, con el fin de preservar la seguridad lógica del sistema.

3.5.3.1. Instalación

El proceso de instalación comenzó con la configuración de los nodos de base de datos, desplegando en ambos nodos del clúster el motor de base de datos MariaDB versión 10.6. Se configuró un esquema de replicación activo–pasivo, donde uno de los nodos asumía el rol principal para operaciones de escritura

y el segundo mantenía una réplica sincronizada para operaciones de lectura y contingencia.

Posteriormente, se instaló y configuró MariaDB MaxScale en el servidor principal donde posteriormente se instalaría el Zabbix Server y Frontend. MaxScale tiene como funciones monitorear el estado de los nodos del clúster, gestionar la replicación y actuar como gateway para las conexiones provenientes del Zabbix Server. Así mismo, se configuraron los servicios del Zabbix Server (ver Anexo A) y Frontend (ver Anexo B) para que las conexiones hacia la base de datos se realizaran a través de MaxScale, evitando dependencias directas con un nodo de base de datos específico y permitiendo conmutación automática ante fallas.

Para asegurar la disponibilidad del servicio de base de datos, se validó la correcta replicación de datos entre ambos nodos y el comportamiento del sistema ante simulaciones de indisponibilidad de uno de los nodos, asegurando que el nodo pasivo pudiera asumir el rol principal sin pérdida de información ni interrupciones prolongadas del servicio.

Una vez operativa la base de datos, se procedió con la instalación del Zabbix Server en el servidor principal, siguiendo el procedimiento de instalación documentado en el manual oficial (Zabbix, 2026e). Se configuraron los parámetros necesarios para la conexión con la base de datos a través de MaxScale, así como los ajustes relacionados con el almacenamiento de datos históricos, cachés internas y procesos de recolección.

Posteriormente, se instaló el Zabbix Frontend, el cual proporciona la interfaz web para la administración y visualización del sistema. Se configuró el servidor web y los parámetros de conexión con el Zabbix Server, verificando la correcta carga de la interfaz gráfica y el acceso mediante credenciales administrativas.

Durante esta etapa se realizaron pruebas básicas de funcionamiento, como la creación de un dispositivo monitoreado de prueba y la validación de la

recepción de métricas, con el fin de confirmar que la comunicación entre el Zabbix Frontend, el Zabbix Server y la base de datos se encontraba operativa.

De forma paralela, se instaló y configuró el Zabbix Proxy en el servidor destinado a la recolección distribuida de métricas. Se estableció la comunicación segura entre el Zabbix Proxy y el Zabbix Server, definiendo los parámetros de sincronización y almacenamiento temporal de datos.

Culminada esta etapa se validó el acceso a la plataforma y se verificó la correcta navegabilidad dentro de la misma. La siguiente figura muestra la interfaz principal de la plataforma Zabbix, luego de validar su correcta instalación. (ver Figura 5).

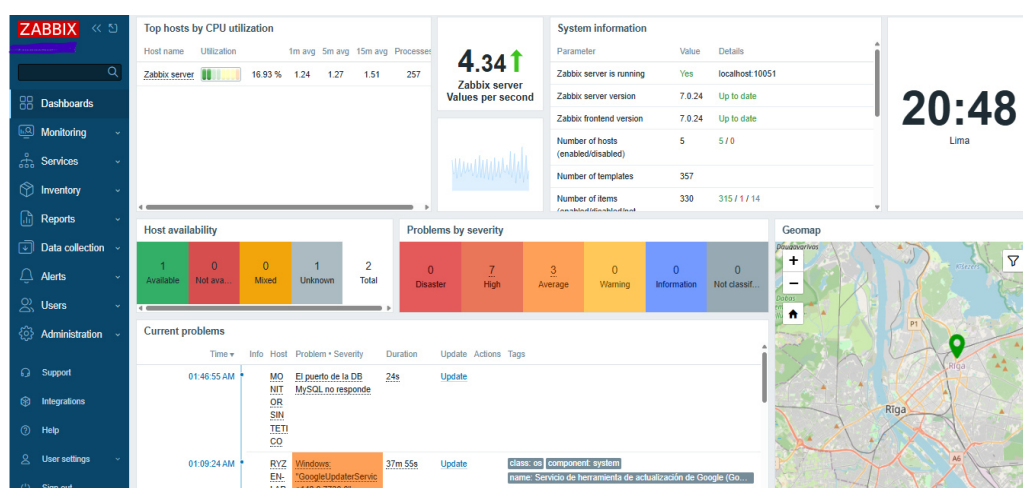


Figura 5 Plataforma Zabbix funcionando. Fuente. Elaboración propia (2026).

3.5.3.2. Configuración

El Zabbix Proxy fue configurado para monitorear específicamente los dispositivos correspondientes al entorno de no producción, permitiendo segmentar la carga y optimizar el rendimiento del Zabbix Server. La siguiente figura muestra cómo se dividió la carga del monitoreo entre el Zabbix Server y el Zabbix Proxy. (ver Figura 6).

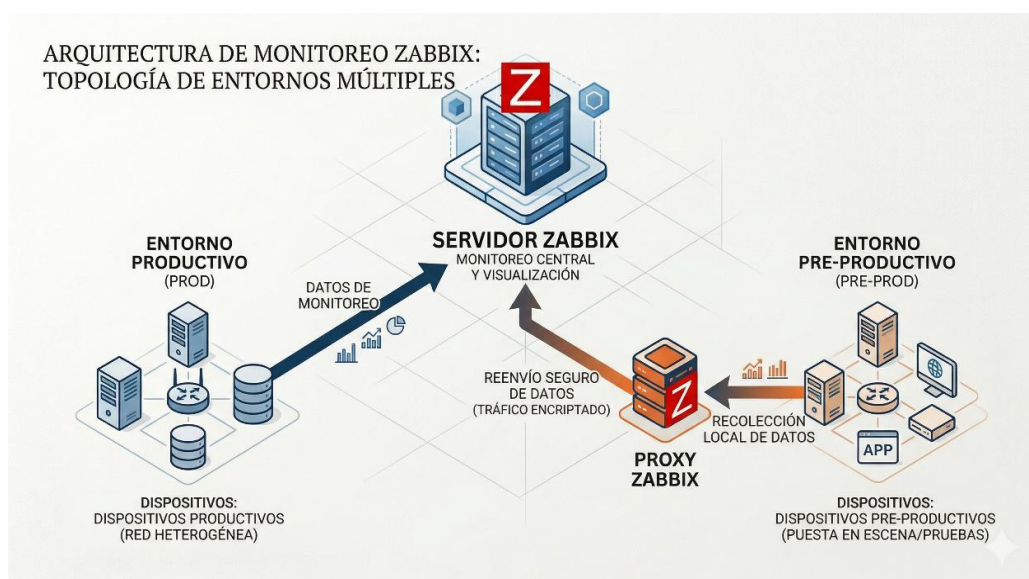


Figura 6 Segmentación del monitoreo según el entorno. Fuente. Elaboración propia (2026).

El monitoreo de los sistemas operativos se configuró mediante Zabbix Agent Active, permitiendo que los agentes instalados en los servidores envíen métricas de manera activa hacia el Zabbix Server o Proxy correspondiente. Este enfoque reduce la dependencia de consultas periódicas desde el Zabbix Server y mejora la eficiencia en la transmisión de datos. Adicionalmente, se revisaron y ajustaron las plantillas de monitoreo estándar provistas por Zabbix para sistemas operativos Linux, Windows, Solaris y AIX, adecuándolas a los requerimientos internos de la organización y a los umbrales definidos como aceptables, para lo cual se modificaron disparadores para alinearlos con los niveles de criticidad establecidos durante la fase de análisis.

Para el monitoreo de los equipos de como *routers*, *switches*, *access points* y balanceadores de carga se utilizó el protocolo SNMPv2 y se desarrollaron plantillas personalizadas para la recolección periódica de métricas como utilización de interfaces, estadística de paquetes, estado de enlaces, carga de CPU, memoria y métricas específicas requeridas por el área de redes.

En aquellos casos donde los activos monitoreados no soportaban o permitían la instalación de agentes Zabbix ni supervisión por SNMPv2, se implementaron mecanismos alternativos. Estos incluyeron:

- Monitoreo basado en logs mediante la lectura y análisis de archivos específicos.
- Pruebas sintéticas para validar disponibilidad y tiempos de respuesta de servicios.
- Consumo de APIs para la obtención de métricas expuestas por estos activos.

Dependiendo del tipo de dispositivo y del protocolo de monitoreo que este soportaba, se definió y configuró el mecanismo de supervisión correspondiente en la plataforma Zabbix (véase Tabla 3).

Tabla 3 Tipo de dispositivo o recurso y protocolo de monitoreo utilizado

Tipo de dispositivo o recurso monitoreado	Protocolo de monitoreo
Servidores (Windows, Linux, AIX, Solaris)	Zabbix Agent Active
Dispositivos de red (<i>routers, switches, access point</i> , balanceadores de carga, <i>firewalls</i>)	SNMPv2
APIs	HTTP
URLs y páginas web	HTTP
Otros	ICMP, TCP

Fuente. Elaboración propia (2026).

De manera consecuente, el uso de plantillas personalizadas permitió estandarizar la obtención de métricas clave con granularidad de 1 minuto. La tabla siguiente muestra el tipo de dispositivo o recurso monitoreado y las métricas más importantes que sirven para supervisar su estado y salud (véase Tabla 4). Así como la lógica de gatilladores, según los umbrales configurados, para la generación de alertas importantes. La tabla siguiente muestra el tipo de dispositivo monitoreado y las alertas configuradas que se generan cuando las métricas clave salen de su comportamiento normal (véase Tabla 5) según su severidad (ver Anexo C) que aseguren la supervisión de los dispositivos o recursos monitoreados por la plataforma de monitoreo implementada.

Tabla 4 Tipo de dispositivo o recurso y métricas claves obtenidas

Tipo de dispositivo o recurso monitoreado	Métricas clave
Servidores	Respuesta del servidor (<i>up/down</i>). Consumo de CPU, memoria, disco. Tráfico de interfaces: paquetes enviados, recibidos, perdidos, errores, descartados. Estado de servicios y procesos. Estado y tiempo de respuesta de puertos y protocolos. Mensajes de error y alerta en registros del sistema operativo y/o aplicaciones.
Dispositivos de red	Respuesta del dispositivo (<i>up/down</i>). Consumo de CPU, memoria, disco. Tráfico de interfaces: paquetes enviados, recibidos, perdidos, errores, descartados. Estado (<i>up/down</i>) de componentes (<i>fan, power supply, blades</i>). Estado y tiempo de respuesta de puertos y protocolos.
APIs	Respuesta del API. Métricas específicas expuestas por el API.
URLs y páginas web	Tiempo y velocidad de respuesta de la URL o página web. Código de respuesta HTTP. Validación de parámetros de respuesta en la cabecera y cuerpo de la petición HTTP.
Otros	Estado y tiempo de respuesta ICMP. Estado y tiempo de respuesta de puertos TCP.

Fuente. Elaboración propia (2026).

Tabla 5 Tipo de dispositivo o recurso y alertas importantes

Tipo de dispositivo o recurso monitoreado	Alertas importantes
Servidores	Servidor no responde (severidad crítica). Alto consumo de CPU, memoria, disco. (severidad media, alta y crítica). Alta cantidad de paquetes perdidos, con errores o descartados (severidad media). Proceso o servicio caído (severidad alta). Puerto o protocolo no responde (severidad crítica). Errores detectados en el sistema operativo y/o aplicaciones (severidad alta y crítica).
Dispositivos de red	Dispositivo no responde (severidad crítica). Alto consumo de CPU, memoria, disco. (severidad media, alta y crítica). Alta cantidad de paquetes perdidos, con errores o descartados (severidad media). Componente caído (severidad alta). Puerto o protocolo no responde (severidad crítica).
APIs	API no responde (severidad crítica). Alerta personalizada según la métrica obtenida.
URLs y páginas web	Página web o URL no responde (severidad crítica). Lentitud de respuesta de página web o URL (severidad alta). Página web o URL presenta error de respuesta HTTP (severidad crítica).
Otros	Dispositivo no responde (severidad crítica). Puerto no responde (severidad crítica).

Fuente. Elaboración propia (2026).

En resumen, la fase de implementación permitió desplegar una solución integral de monitoreo alineada con el diseño previamente definido. Se aseguraron la correcta integración entre componentes, la configuración adecuada de mecanismos de recolección de datos y la personalización de

plantillas según las necesidades de la organización. Esta etapa sentó las bases para las fases siguientes, garantizando que el sistema estuviera listo para operar antes de su puesta en producción definitiva.

3.5.4. Fase de pruebas

Durante la fase de pruebas, desarrollada a lo largo de dos semanas, se validó el correcto funcionamiento de la plataforma de monitoreo Zabbix, verificando de manera integral la recolección de métricas, la generación de alertas y la adecuada visualización de los datos mediante paneles de control y reportes. Esta etapa tuvo como objetivo principal comprobar que la plataforma de monitoreo implementada cumpliera con los requerimientos técnicos y funcionales definidos en la fase de diseño, garantizando además un funcionamiento estable, confiable y continuo dentro del entorno productivo. Asimismo, se buscó asegurar que los mecanismos de detección temprana de incidentes operaran de acuerdo con los umbrales y niveles de severidad previamente establecidos, permitiendo identificar oportunamente posibles fallas o degradaciones en el rendimiento de la infraestructura monitoreada.

Inicialmente, se realizaron pruebas funcionales orientadas a validar la correcta integración de los agentes con el sistema operativo y el servidor Zabbix al cual reportan así como los métodos de monitoreo configurados en la plataforma. Se verificó que los servidores, dispositivos de red y servicios críticos enviaran información de forma periódica, según su granularidad configurada, al Zabbix Server, comprobando la disponibilidad y consistencia de métricas como el uso de CPU, consumo de memoria, espacio en disco, tráfico de red, estado de procesos y respuesta de servicios. Además, se confirmó que la información recopilada fuera almacenada y actualizada correctamente, permitiendo su visualización en tiempo real a través de los paneles de monitoreo, gatillando las alertas y facilitando el seguimiento del estado operativo de la infraestructura (ver Figura 7).



Figura 7 Recepción de métricas en la plataforma. Fuente. Elaboración propia (2026).

De igual manera, se evaluó el correcto funcionamiento de las métricas, las alertas y las acciones configuradas en la plataforma, asegurando que los eventos generados correspondieran efectivamente a condiciones reales de operación y se activaran de acuerdo con la lógica y los umbrales previamente establecidos. Asimismo, se comprobó que las acciones asociadas a dichos eventos se ejecutaran correctamente, garantizando una respuesta adecuada ante las diferentes condiciones detectadas. La figura siguiente muestra las diversas alertas que se generan dentro de la plataforma, basadas en el comportamiento de las métricas clave monitoreadas y en las reglas de evaluación configuradas para cada elemento supervisado (ver Figura 8).

Severity	Info	Host	Problem
Alta		PROD-APP-SERVER	↓ La web "https://prod-app-server/mis-consumos" presenta gran cantidad de errores HTTP 500 ?
Crítica		PROD-APP-SERVER	↓ La pagina web "https://prod-app-server/recargas" no responde ?
Crítica		PROD-DB-SERVER	La particion "/root/app" se encuentra al 90% de uso
Crítica		PROD-DB-SERVER	El servicio DNS no responde
Alta		PROD-APP-SERVER	↓ El API "https://prod-app-server/consulta-saldo" no responde ?
Alta		PROD-APP-SERVER	↓ Alta cantidad de paquetes perdidos en la interfaz ETH0 ?
Alta		PROD-DB-SERVER	El FAN #2 del servidor está fallando
Crítica		PROD-DB-SERVER	El servidor demonlab.duckdns.org no responde
Alta		PROD-APP-SERVER	↓ Alto consumo de memoria durante 10m) ?
Crítica		PROD-APP-SERVER	↓ Alto consumo de CPU durante 10m) ?
Crítica		PROD-DB-SERVER	El puerto de la DB MySQL no responde

Figura 8 Alertas generadas por la plataforma de monitoreo. Fuente. Elaboración propia (2026).

Posteriormente, se ejecutaron pruebas de alertamiento, las cuales consistieron en simular escenarios controlados de falla sobre los distintos componentes monitoreados. Entre las pruebas realizadas se incluyeron la detención intencional de servicios, la desconexión temporal de interfaces de red, la saturación controlada de recursos del sistema y la indisponibilidad simulada de elementos monitoreados. Estas acciones permitieron verificar la capacidad de la plataforma para detectar anomalías dentro de los tiempos esperados y generar notificaciones automáticas a través de los canales configurados, tales como correo electrónico (ver Anexo D) y mensajería en línea. Durante estas pruebas se midieron indicadores clave como el tiempo medio de detección (MTTD) y el tiempo medio de recuperación (MTTR), con

el fin de evaluar la eficacia del sistema de monitoreo en la reducción del impacto operativo ante incidentes.

En paralelo, se validó la correcta visualización de la información en los dashboards diseñados para los distintos perfiles de usuario. Se comprobó que los paneles mostraran datos actualizados en tiempo real, gráficos históricos consistentes y estados claramente diferenciados por niveles de severidad. La figura siguiente muestra el dashboard principal utilizado por los usuarios que realizan la supervisión de alertas (ver Figura 9). Asimismo, se revisó la generación de reportes periódicos, verificando la integridad y coherencia de la información presentada, lo cual resulta fundamental para la toma de decisiones y el análisis de tendencias de capacidad.

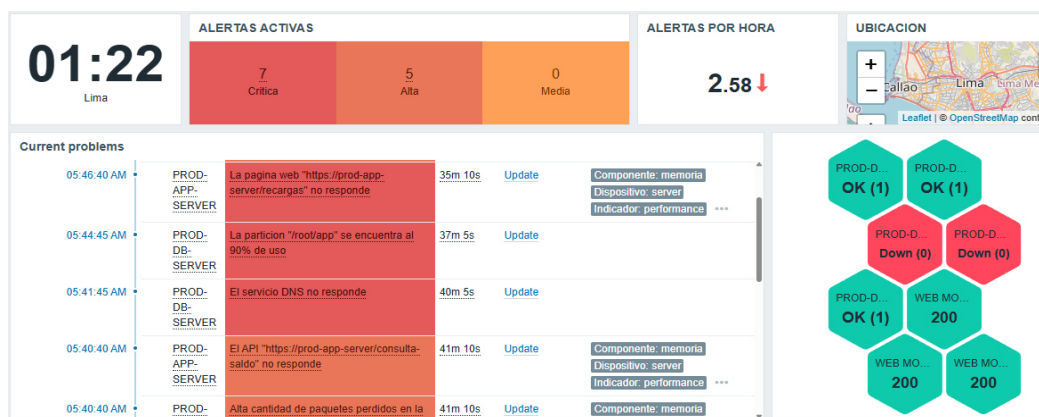


Figura 9 Dashboard principal. Fuente. Elaboración propia (2026).

De manera adicional, se implementó un monitoreo personalizado para el clúster de base de datos utilizado por la plataforma de monitoreo. Debido a la criticidad de este componente dentro de la arquitectura, se desarrollaron métricas específicas orientadas a supervisar el estado de cada nodo, el rol desempeñado (maestro o replica), el estado del proceso de replicación y los eventos de conmutación por falla (*failover*). Este monitoreo especializado permitió contar con visibilidad detallada sobre la capa de persistencia de datos, reduciendo el riesgo de inconsistencias o pérdida de disponibilidad.

Para la obtención de estas métricas se utilizó el componente MaxScale, encargado de la gestión y balanceo de conexiones hacia el clúster. En

particular, se empleó el comando *maxscale list servers*, el cual expone información relevante sobre el estado de los nodos, su rol dentro del clúster y los identificadores asociados a la replicación (MariaDB, 2023). La salida de este comando fue procesada mediante scripts personalizados e integrada como métricas dependientes dentro de la plataforma de monitoreo. La figura siguiente muestra las métricas de salud del clúster de base de datos enviadas a Zabbix que se utilizarán para su monitoreo. (ver Figura 10). Estas métricas permitieron generar disparadores específicos ante eventos como pérdida de sincronización, cambio inesperado de rol o indisponibilidad de un nodo (ver Figura 11).

Host	Name ▲	Last check	Last value
DB CLUSTER	Database Cluster Information	29s	{ "cluster_state": "...
DB CLUSTER	DB Cluster Replication ID: Node 1	29s	800
DB CLUSTER	DB Cluster Replication ID: Node 2	29s	900
DB CLUSTER	DB Cluster Role: Node 1	29s	slave (2)
DB CLUSTER	DB Cluster Role: Node 2	29s	master (1)
DB CLUSTER	DB Cluster State	29s	DOWN (0)
DB CLUSTER	DB Replication Status: Node 1	29s	DOWN (0)
DB CLUSTER	DB Replication Status: Node 2	29s	DOWN (0)
DB CLUSTER	DB State: Node 1	29s	DOWN (0)
DB CLUSTER	DB State: Node 2	29s	DOWN (0)

Figura 10 Métricas de salud del clúster de base de datos. Fuente. Elaboración propia (2026).

Severity	Recovery time	Status	Info	Host	Problem
High	01:32:39 AM	RESOLVED		DB CLUSTER	El Nodo 2 ha cambiado de rol. Nuevo Rol: master (1)
High	01:32:39 AM	RESOLVED		DB CLUSTER	El Nodo 1 ha cambiado de rol. Nuevo Rol: slave (2)
High		PROBLEM		DB CLUSTER	Desfase en la replicacion de la data del Cluster
High		PROBLEM		DB CLUSTER	El Nodo 2 del Cluster no está replicando la data
High		PROBLEM		DB CLUSTER	El Nodo 1 del Cluster no está replicando la data
High		PROBLEM		DB CLUSTER	El Nodo 1 del Cluster no está en Running
High		PROBLEM		DB CLUSTER	El Nodo 2 del Cluster no está en Running
High		PROBLEM		DB CLUSTER	El Cluster DB no está en Running

Figura 11 Alertas para el monitoreo del clúster de base de datos. Elaboración propia (2026).

Esto se realizó mediante pruebas controladas, forzando la caída del nodo primario para verificar el comportamiento del clúster y la correcta reasignación de roles. Durante estas pruebas se evaluó no solo la estabilidad del servicio de base de datos, sino también la capacidad de la plataforma de monitoreo para reflejar en tiempo real el cambio de estado y generar las alertas correspondientes. Los resultados obtenidos evidenciaron que la plataforma de monitoreo era capaz de identificar oportunamente los eventos de conmutación de roles entre los nodos y notificar al equipo de operaciones dentro de los tiempos establecidos.

Finalmente, se documentaron todos los escenarios de prueba, registrando condiciones iniciales, acciones ejecutadas, resultados obtenidos y tiempos de respuesta medidos. Esta documentación permitió validar formalmente la aceptación técnica de la solución y dejó evidencia para futuras auditorías o procesos de mejora continua. En conjunto, la fase de pruebas confirmó que la plataforma de monitoreo cumplía con los objetivos establecidos, garantizando visibilidad integral de la infraestructura, detección temprana de incidentes, soporte efectivo a la continuidad operativa del servicio y resiliencia ante fallos.

3.5.5. Fase de marcha blanca

Durante la fase de marcha blanca, cuya duración fue de 30 días calendario, la plataforma de monitoreo Zabbix operó en paralelo con las herramientas de supervisión previamente utilizadas por la organización, como CA UIM para el monitoreo de infraestructura TI, SolarWinds para el monitoreo de dispositivos de red y Humio para la gestión de *logs*, esto con el objetivo de validar su desempeño en un entorno real sin afectar la continuidad operativa. Esta etapa permitió comparar resultados, verificar la consistencia de los eventos detectados y ajustar progresivamente la configuración antes de su adopción definitiva como herramienta principal de monitoreo. El funcionamiento simultáneo facilitó la identificación de brechas, diferencias en criterios de alertamiento, tiempos de respuesta y oportunidades de mejora en la parametrización del sistema.

Durante este periodo, se realizó un análisis detallado de los umbrales definidos para cada métrica crítica, tales como uso de CPU, consumo de memoria, latencia de red, estado de procesos, respuesta de puertos, disponibilidad de servicios y uso de almacenamiento. En función del comportamiento histórico y de las condiciones reales de operación, se recalibraron las lógicas de los disparadores para asegurar que las alertas representaran situaciones efectivamente anómalas y no variaciones normales del entorno. La siguiente figura muestra la lógica de alertamiento recalibrada para que considere la tendencia del comportamiento de la métrica en los últimos 30 días y se active solo cuando exceda en un 30% este comportamiento y su valor sea mayor a 10 (ver Figura 12). Este proceso fue clave para reducir la generación de falsos positivos, los cuales pueden provocar fatiga e ineficiencia operativa en los equipos de soporte y disminuir la efectividad del sistema de monitoreo.

The image shows a configuration form for an alert named 'Alto consumo de CPU'. The form includes fields for Name, Event name, Operational data, Severity, and Expression. The 'Expression' field is highlighted with a red box and contains the following logic:

```
last(/HOST/ITEM) > (trendavg(/HOST/ITEM,30d) * 1.3)
and trendavg(/HOST/ITEM,30d) > 10
```

The 'Severity' section shows a row of buttons: 'Not classified', 'Information', 'Precaucion', 'Media', 'Alta' (which is selected and highlighted in red), and 'Critica'.

Figura 12 Recalibración de lógica de alertamiento. Fuente. Elaboración propia (2026).

Asimismo, se optimizaron las reglas de alertamiento considerando ventanas horarias, niveles de criticidad, dependencias entre servicios y personal responsable de las atenciones. Por ejemplo, se configuraron dependencias lógicas que evitaban la generación masiva de alertas cuando la causa raíz correspondía a la indisponibilidad de un componente principal. De esta manera, si un dispositivo monitoreado presentaba una falla, el sistema evitaba notificar individualmente cada servicio dependiente. La figura siguiente

muestra como luego de la configuración de dependencias se suprimen las alertas que se activan como consecuencia de la caída del dispositivo monitoreado (ver Figura 13), priorizando la alerta del elemento de causa raíz. Esta optimización permitió mejorar la claridad de los eventos y facilitar el análisis por parte del personal técnico.

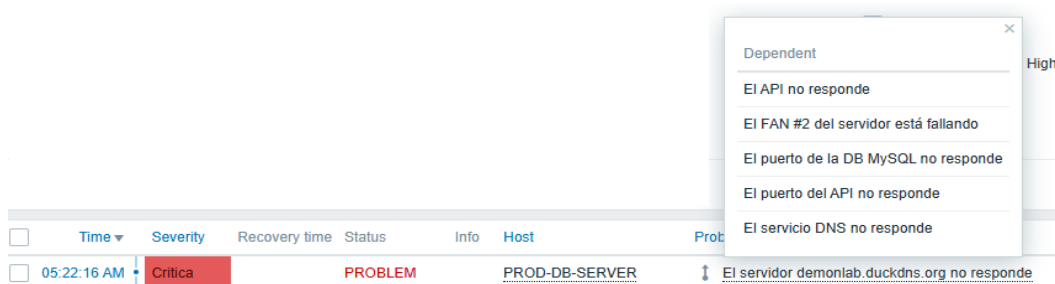


Figura 13 Configuración de dependencia de alertas. Fuente. Elaboración propia (2026).

En paralelo, se revisaron y estandarizaron los mensajes de notificación enviados a través de los distintos canales de comunicación al personal responsable de las atenciones. Las alertas fueron adaptadas al idioma español, utilizando una redacción clara, estructurada y orientada a la acción, incluyendo información relevante como nombre del dispositivo, servicio afectado, severidad del evento, hora de ocurrencia y posible causa raíz. Además, se incorporaron recomendaciones básicas de verificación en los mensajes, con el fin de agilizar la respuesta inicial del equipo de soporte. Esta mejora en la calidad de las notificaciones contribuyó a una gestión más eficiente y rápida de los incidentes.

Un componente fundamental de la marcha blanca fue la implementación de acciones de auto remediación para incidentes previamente identificados como recurrentes y de resolución estandarizada. Estas acciones incluyeron el reinicio automático de servicios, la ejecución de scripts correctivos y la validación de conectividad o disponibilidad mediante comandos específicos. Por ejemplo, ante la detección de la caída de un servicio no crítico, el sistema ejecutaba de forma automática un intento de reinicio controlado y verificaba posteriormente su estado antes de escalar el incidente al equipo

correspondiente. La figura siguiente muestra la configuración de una acción de auto remediación que reinicia el servicio web en caso el servicio falle (ver Figura 14). Este enfoque permitió reducir significativamente los tiempos de resolución (MTTR) y minimizar la intervención manual en eventos de bajo impacto.

☐ Reinicio servicio web	Action operation	Script Agent	<code>systemctl restart httpd</code>
Reinicio servicio web	Trigger equals <i>PROD-DB-SERVER: El servicio web no responde</i>	Run script "Reinicio servicio web" on current host	

Figura 14 Acción de auto remediación configurada. Fuente. Elaboración propia (2026).

En el caso de servicios que interactuaban con bases de datos, se configuraron consultas automáticas para validar la disponibilidad de instancias y el correcto desempeño de las bases de datos. Cuando se detectaban inconsistencias leves o bloqueos temporales, se ejecutaban scripts de verificación que permitían determinar si la condición era transitoria o requería intervención humana. Esta estrategia fortaleció la estabilidad operativa y redujo interrupciones innecesarias.

Adicionalmente, se establecieron mecanismos de escalamiento automático alineados con la estructura organizacional. Se definieron flujos de notificación diferenciados según la criticidad del evento, asegurando que los incidentes de alta severidad fueran comunicados de manera inmediata al Centro de Operaciones de Red (NOC), al equipo de operaciones y, de ser necesario, a las jefaturas responsables. La siguiente figura muestra una notificación enviada por correo electrónico a un buzón de un jefe responsable debido a que la alerta de severidad crítica aún no se encuentra resuelta (ver Figura 15). Estos escalamientos consideraron tiempos máximos de atención; si un incidente no era reconocido o resuelto dentro de un intervalo determinado, el sistema procedía a notificar a un nivel superior de responsabilidad. Este

esquema garantizó trazabilidad, control y cumplimiento de los acuerdos de nivel de servicio (SLA).

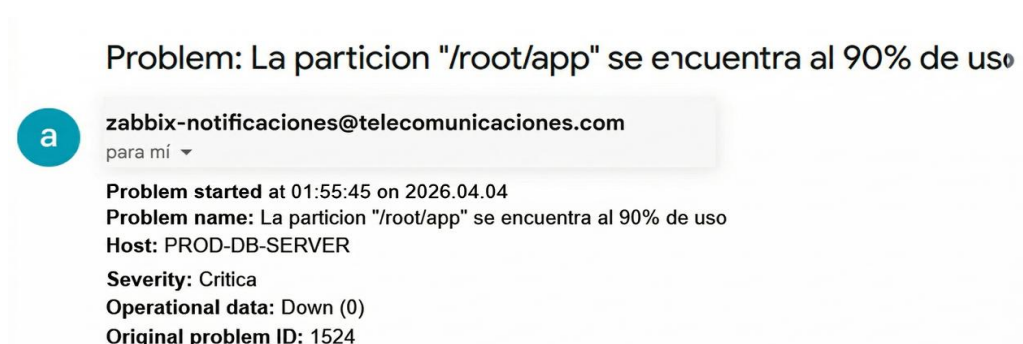


Figura 15 Notificación por correo debido a escalamiento. Fuente. Elaboración propia (2026).

Durante la marcha blanca también se realizó un monitoreo continuo del desempeño general de la plataforma, evaluando el consumo de recursos del Zabbix Server y Proxy, la capacidad de procesamiento de eventos y la estabilidad de la base de datos histórica (ver Figura 16). Se verificó que la plataforma de monitoreo pudiera sostener la carga generada por el número total de dispositivos monitoreados y métricas configuradas, asegurando que no se produjeran retrasos en la actualización de datos ni en la generación de alertas. Este seguimiento permitió realizar ajustes menores de rendimiento antes de la puesta en producción definitiva.



Figura 16 Monitoreo de salud de la plataforma. Fuente. Elaboración propia (2026).

Para lograr estos ajustes en las configuraciones, fue fundamental contar con la retroalimentación continua de los equipos responsables de la supervisión de alertas, la resolución de incidentes y la gestión de cambios. Su experiencia operativa permitió identificar alertas redundantes, umbrales inadecuados y oportunidades de mejora en las lógicas de disparo, contribuyendo a una parametrización más precisa y alineada con la realidad del entorno productivo. Asimismo, el análisis conjunto de eventos reales facilitó la depuración de falsos positivos y la priorización adecuada de incidentes según su impacto en el negocio. Esta colaboración no solo optimizó el desempeño de la plataforma de monitoreo, sino que también fortaleció la adopción de la herramienta por parte de las áreas involucradas, asegurando que la solución respondiera efectivamente a las necesidades operativas y a los acuerdos de nivel de servicio establecidos.

En conclusión, la fase de marcha blanca cumplió un rol estratégico en la transición hacia el uso formal de la nueva plataforma de monitoreo. Al operar en paralelo con las herramientas existentes, ajustar parámetros críticos, optimizar el alertamiento, implementar mecanismos de auto remediación y establecer escalamientos automáticos, se logró consolidar una solución madura, confiable y alineada con las necesidades operativas de la organización. Esta etapa permitió minimizar riesgos antes de la adopción definitiva y aseguró que la plataforma estuviera plenamente preparada para soportar el entorno productivo de manera eficiente y sostenible.

3.5.6. Fase de operación

En la fase de operación se formalizó el uso de la plataforma de monitoreo Zabbix como herramienta oficial de supervisión de la infraestructura tecnológica de la organización. Esta formalización implicó no solo la validación técnica de su funcionamiento, sino también su incorporación dentro de los

procesos operativos y normativos vigentes dentro de la organización. A partir de este momento y durante aproximadamente 9 meses, la herramienta pasó de un estado de implementación y ajuste a formar parte integral del ecosistema de gestión de servicios de TI, convirtiéndose en el punto central para la detección, registro y seguimiento de eventos e incidentes.

Como parte de esta etapa, se realizó la estructuración formal de usuarios y perfiles de acceso, definiendo grupos en función de las áreas responsables de cada componente de la infraestructura. Se crearon grupos específicos tales como administrador de servidores (adm-server), administrador de bases de datos (adm-db), administrador de redes (adm-redes) y operadores del Centro de Operaciones de Red (NOC). Esta segmentación permitió organizar la plataforma acorde a la estructura organizacional existente, asegurando que cada equipo tuviera visibilidad y control sobre los recursos bajo su competencia.

La asignación de permisos se realizó bajo el principio de mínimo privilegio, otorgando únicamente los accesos estrictamente necesarios para el desempeño de las funciones de cada rol, con el fin de reducir riesgos de seguridad y evitar modificaciones no autorizadas en la plataforma. Los administradores de servidores, bases de datos y redes contaban con permisos de escritura sobre los dispositivos, grupos y plantillas correspondientes a su ámbito de responsabilidad, lo que les permitía realizar ajustes en métricas, disparadores, umbrales y acciones cuando fuera necesario, así como administrar la configuración de los elementos bajo su gestión. Por su parte, los operadores del NOC disponían de permisos de lectura sobre la totalidad de los dispositivos monitoreados, con el objetivo de supervisar el estado general de la infraestructura, gestionar eventos y escalar incidentes según los procedimientos establecidos, sin que pudieran intervenir en la configuración técnica asociada ni modificar los parámetros de monitoreo definidos por los administradores. Esta distribución de privilegios permitió mantener un adecuado control de acceso y una clara separación de responsabilidades dentro de la plataforma (ver Figura 17).



Figura 17 Segmentación de permisos. Fuente. Elaboración propia (2026).

Esta segregación de accesos se alineó con buenas prácticas de seguridad de la información y control interno, reduciendo el riesgo de modificaciones no autorizadas o cambios accidentales en la configuración del sistema. Además, permitió mantener trazabilidad sobre las acciones realizadas dentro de la plataforma, ya que cada modificación quedaba registrada con el usuario correspondiente. De esta manera, se fortaleció el control operativo y se garantizó la integridad de la configuración del monitoreo, aspecto fundamental en entornos productivos que soportan procesos críticos dentro de la organización.

Durante la fase de operación también se establecieron lineamientos formales para la administración y mantenimiento continuo de la plataforma. Se definieron procedimientos para la incorporación de nuevos dispositivos, actualización de plantillas, revisión periódica de umbrales y depuración de elementos obsoletos. Asimismo, se estableció un calendario de revisión de capacidad, desempeño y salud del Zabbix Server y Proxy y sus procesos, considerando el crecimiento progresivo de la infraestructura tecnológica. Estas actividades aseguraron que la herramienta se mantuviera actualizada, eficiente y alineada con la evolución del entorno.

Otro aspecto clave fue la integración del sistema de monitoreo en el proceso de gestión de incidentes. A partir de su adopción oficial, todas las alertas críticas detectadas por la plataforma generaban de manera automática un ticket de incidente en la plataforma de gestión de servicios BMC Remedy para su asignación a las áreas resolutoras y su posterior seguimiento. Las notificaciones automáticas permitieron una detección temprana de fallas, reduciendo los tiempos de reacción y mejorando la coordinación entre las áreas técnicas. De esta manera, el monitoreo dejó de ser un mecanismo reactivo basado en reportes de usuarios y pasó a constituirse en un modelo proactivo orientado a la prevención y mitigación temprana de impactos. La figura siguiente muestra el flujo de integración entre Zabbix y BMC Remedy para la creación automática de tickets generados por alertas críticas (ver Figura 18).



Figura 18 Integración Zabbix – Remedy. Fuente. Elaboración propia (2026).

Asimismo, se promovió el uso de dashboards ejecutivos para la supervisión de indicadores clave de desempeño (KPIs), tales como disponibilidad de servicios, cumplimiento de acuerdos de nivel de servicio (SLA), tendencias de consumo de recursos y frecuencia de incidentes por categoría. Esta información permitió a las jefaturas y áreas de gestión contar con una visión

consolidada del estado de la infraestructura, facilitando la toma de decisiones basada en datos y el planeamiento de mejoras o ampliaciones de capacidad.

La fase de operación también contempló actividades de capacitación y transferencia de conocimiento hacia los equipos involucrados. Se realizaron sesiones de entrenamiento para el uso adecuado de la herramienta, interpretación de eventos, gestión de alertas y aplicación de buenas prácticas en monitoreo. Esta capacitación contribuyó a fortalecer la autonomía de los equipos técnicos y a consolidar la adopción cultural de la plataforma como parte esencial de la operación diaria.

Finalmente, la incorporación formal del sistema de monitoreo dentro del marco operativo de la organización representó un avance significativo hacia una gestión de servicios más estructurada y alineada con estándares internacionales. La visibilidad integral de la infraestructura, la automatización de alertas, la trazabilidad de eventos y la integración con la gestión de incidentes permitieron establecer un modelo de operación más proactivo, controlado y orientado a la continuidad del negocio. En conjunto, la fase de operación consolidó la plataforma de monitoreo implementada como un componente estratégico para la estabilidad, eficiencia y mejora continua de los servicios tecnológicos de la empresa.

3.6. Resultados de la actividad

Luego de la puesta en producción de la plataforma de monitoreo basada en Zabbix, se evidenciaron mejoras significativas y cuantificables en los ámbitos técnico, operativo y económico de la organización. La implementación de la nueva solución permitió optimizar los procesos de supervisión de la infraestructura tecnológica, incrementar la eficiencia en la detección y gestión de incidentes, así como reducir los costos asociados al licenciamiento y mantenimiento de las herramientas de monitoreo. A continuación, se presenta una comparación entre la situación inicial, representada por los indicadores registrados antes de la implementación, y los resultados obtenidos tras la

adopción de la nueva plataforma. Esta comparación permite evidenciar de manera objetiva el impacto generado por la solución implementada (ver Tabla 6).

Tabla 6 Resumen de resultados después de la implementación

Indicador	Antes de la implementación	Después de la implementación
Plataformas de monitoreo utilizadas	3	1
Cantidad de dispositivos monitoreados	400	1400
Porcentaje de dispositivos monitoreados	20%	70%
Porcentaje de dispositivos críticos monitoreados	50%	100%
Correlación de eventos	0	+100 por día
Tiempo promedio de detección (MTTD)	1 hora	10 minutos
Tiempo promedio de resolución (MTTR)	3 horas	1 hora
Tiempo promedio de registro de incidentes críticos	30 minutos	1 minuto
Disponibilidad promedio de los servicios	75%	96%

Fuente. Elaboración propia en base a los datos recopilados (2026).

CAPITULO IV: DISCUSIÓN DE RESULTADOS

4.1. Descripción de la implementación

- Antes de la implementación, la organización operaba con tres plataformas de monitoreo distintas, lo que generaba silos de la información, duplicidad de alertas, nula correlación de eventos y mayor complejidad operativa. Tras la implementación, se consolidó el monitoreo en una única plataforma centralizada, simplificando la gestión operativa y mejorando la visibilidad global del entorno tecnológico.
- Inicialmente, solo 400 dispositivos se encontraban bajo monitoreo, lo que representaba una cobertura limitada de la infraestructura tecnológica. Esta situación implicaba zonas sin supervisión y riesgos ante fallas no detectadas. Luego de la implementación, el número de dispositivos monitoreados aumentó a 1400, ampliando significativamente la cobertura y fortaleciendo el control sobre servidores, servicios y equipos de red.
- Antes del proyecto, apenas el 20% del total de dispositivos de la organización contaba con algún tipo de monitoreo activo, lo que evidenciaba una baja visibilidad operativa. Esta limitación reducía la capacidad de detección temprana de incidentes y afectaba la gestión preventiva. Posteriormente, se alcanzó un 70% de cobertura, incrementando sustancialmente el nivel de supervisión y mejorando la capacidad de respuesta ante eventos anómalos.
- Antes de la implementación, solo el 50% de los dispositivos considerados críticos para el negocio estaba monitoreado, lo que representaba un riesgo significativo para la continuidad operativa. La falta de supervisión total sobre estos componentes podía derivar en impactos directos al negocio. Con la implementación, se logró monitorear el 100% de los

dispositivos críticos, asegurando una vigilancia permanente de los elementos clave para el negocio.

- Previo a la implementación, debido a los silos de información no existía un mecanismo de correlación de eventos, lo que obligaba a analizar y correlacionar manualmente cada alerta generada, lo cual incrementaba los tiempos de diagnóstico y dificultaba la identificación de la causa raíz. Después de la implementación, se lograron correlacionar más de 100 eventos diarios, permitiendo agrupar alertas relacionadas, reducir ruido y agilizar el análisis de incidentes, lo cual optimizó la gestión operativa de los equipos.
- Inicialmente, el tiempo promedio de detección de incidentes era de aproximadamente una hora, debido a la falta de monitoreo centralizado y alertas optimizadas. Esta demora incrementaba el impacto potencial de las fallas sobre los servicios. Tras la implementación, el MTTD se redujo a 10 minutos, mejorando considerablemente la capacidad de identificación temprana y permitiendo una reacción más oportuna ante incidentes.
- Antes del proyecto, el tiempo promedio de resolución de incidentes era de aproximadamente tres horas, debido principalmente a la limitada visibilidad de la infraestructura y a la ausencia de automatización en los procesos de atención, lo que impactaba negativamente en los niveles de servicio y en la percepción de disponibilidad. Tras la implementación, el MTTR se redujo a una hora, como resultado de una detección más oportuna, una mejor correlación de los eventos y la incorporación de mecanismos de auto remediación y auto escalamiento, los cuales permitieron ejecutar acciones correctivas de manera automática y escalar los incidentes de forma eficiente hacia los equipos responsables.
- Antes de la implementación, el tiempo promedio para registrar incidentes críticos era de 30 minutos, debido a que el proceso dependía en gran medida de la detección manual y del reporte por parte de usuarios o personal operativo. Esta demora impactaba directamente en el inicio de las acciones correctivas. Con la implementación del sistema de monitoreo y la generación automática de alertas, el tiempo de registro se redujo a 1

minuto, permitiendo una reacción prácticamente inmediata ante eventos de alta criticidad.

- Inicialmente, la disponibilidad promedio de los servicios alcanzaba el 75%, reflejando interrupciones frecuentes y tiempos prolongados de recuperación ante fallas. Tras la implementación, la disponibilidad se incrementó al 96%, evidenciando una mejora significativa en la continuidad operativa y en la estabilidad de los servicios críticos para el negocio, lo cual incrementó significativamente la calidad de los servicios ofrecidos a los usuarios.

Todos estos datos fueron obtenidos a partir de la información relevada y documentada durante la fase de análisis, desarrollada al inicio del proceso de implementación, así como de los registros históricos de indicadores proporcionados por la mesa de ayuda BMC Remedy. Esta información permitió establecer una línea base objetiva para comparar la situación previa y los resultados alcanzados tras la puesta en marcha de la plataforma de monitoreo.

4.2. Evaluación Económica

Antes de la ejecución del proyecto, la organización destinaba aproximadamente 350 mil dólares anuales al licenciamiento de diversas herramientas de monitoreo y a la contratación de servicios profesionales asociados. Esta situación implicaba un costo operativo elevado, producto de la coexistencia de múltiples plataformas, lo que dificultaba la consolidación tecnológica y reducía la eficiencia en la gestión de la infraestructura de monitoreo.

Tras la implementación de la nueva solución, basada en una plataforma de monitoreo open source con costo de licenciamiento nulo, el gasto anual se redujo a 100 mil dólares, correspondiente únicamente a los servicios profesionales requeridos para su operación y soporte. Como resultado, la organización obtuvo un ahorro anual de aproximadamente 250 mil dólares. Este cambio no solo generó un impacto económico significativo, sino que

también permitió optimizar la inversión en herramientas de supervisión mediante la estandarización de la plataforma tecnológica y la disminución de los costos recurrentes de licenciamiento (ver Tabla 7).

Tabla 7 Análisis económico, costo de licenciamiento anual en dólares americanos

Antes de la implementación	Después de la implementación	Ahorro
350 mil dólares	100 mil dólares	250 mil dólares (71.43%)

Fuente. Elaboración propia (2026).

CAPITULO V: CONCLUSIONES

- La implementación de una plataforma de monitoreo centralizada permitió consolidar la supervisión de la infraestructura tecnológica en un único entorno, eliminando la fragmentación de información existente y reduciendo la complejidad operativa. Esto se reflejó en la unificación del monitoreo y en el incremento significativo de dispositivos supervisados, mejorando la visibilidad global y su tiempo de disponibilidad.
- La definición e implementación de indicadores clave junto con mecanismos de alertamiento proactivo permitió mejorar significativamente la capacidad de detección temprana de incidentes. Esto se evidenció en la reducción del tiempo promedio de detección (MTTD) y en la capacidad de correlacionar eventos, facilitando un análisis más eficiente y oportuno de las fallas en los servicios de TI.
- La configuración de un mecanismo de notificación automática, junto con la generación de tickets automáticos para incidentes críticos, optimizó el proceso de gestión de incidentes al reducir los tiempos de registro y asegurar una comunicación oportuna hacia los equipos responsables. Esto permitió una atención más ágil y estructurada de los eventos, fortaleciendo la trazabilidad y el control operativo.
- La implementación de mecanismos de auto escalamiento y auto remediación contribuyó a optimizar la capacidad de respuesta ante incidentes, permitiendo la ejecución automática de acciones correctivas y el escalamiento oportuno de eventos críticos. Como resultado, se logró una reducción significativa en el tiempo promedio de resolución (MTTR), mejorando la continuidad operativa y los niveles de servicio.

CAPITULO VI: RECOMENDACIONES

- Se sugiere profundizar en la implementación de mecanismos de auto remediación mediante el desarrollo de scripts más avanzados y la automatización de escenarios adicionales, lo que permitiría reducir aún más los tiempos de resolución y minimizar la intervención manual del personal técnico.
- Se recomienda crear nuevas lógicas de alertamiento en función del comportamiento histórico de la infraestructura y recomendación de los administradores de los dispositivos de TI, con el objetivo de reducir falsos positivos y mejorar la precisión en la detección de eventos relevantes.
- Se recomienda continuar con el despliegue del monitoreo hasta alcanzar el 100% de los dispositivos de la infraestructura tecnológica, incluyendo aquellos de menor criticidad, con el fin de lograr una visibilidad completamente integral y fortalecer la detección temprana de posibles incidentes.

CAPITULO VII: REFERENCIAS BIBLIOGRÁFICAS

Axelos. (2019). *ITIL® foundation: ITIL 4 edition*. TSO.

BMC Software. (2026). *BMC Helix ITSM overview*. <https://www.bmc.com/it-solutions/bmc-helix-itsm.html>

Ccorimanya, J. H. (2024). *Implementación de un sistema de monitoreo para controlar incidentes en los servidores de una empresa de servicios de software* [Trabajo de suficiencia profesional para optar el título profesional de Ingeniero Empresarial y de Sistemas]. Universidad San Ignacio de Loyola. <https://repositorio.usil.edu.pe/server/api/core/bitstreams/a2d9f19d-de61-4fb9-b154-afceca01b33d/content>

Davenport, T. H. (1993). *Process innovation: Reengineering work through information technology*. Harvard Business School Press.

Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of database systems* (7th ed.). Pearson.

Erl, T., Puttini, R., & Mahmood, Z. (2013). *Cloud computing: Concepts, technology & architecture*. Prentice Hall.

Gómez López, J. (2014). *Gestión de incidentes de seguridad informática*. Editorial Vértice.

Gómez, W. (2025). *Evaluación de una herramienta open source de monitoreo para toma de decisiones en un datacenter* [Proyecto de titulación]. <https://www.dspace.espol.edu.ec/handle/123456789/67428>

Guevara, N. (2023). *Sistema de monitoreo de infraestructura de TI para la gestión de recursos en la Oficina de Tecnologías de la Información de la*

Universidad Nacional de Cajamarca [Tesis de titulación, Universidad Nacional de Cajamarca].

<https://repositorio.unc.edu.pe/bitstream/handle/20.500.14074/6712/TESIS%20-%20GUEVARA%20SANDOVAL%20NILDA.pdf?sequence=1&isAllowed=y>

Hoffer, J. A., Venkataraman, R., & Topi, H. (2019). *Modern database management* (13th ed.). Pearson Education.

Hwang, K., Fox, G. C., & Dongarra, J. J. (2012). *Distributed and cloud computing: From parallel processing to the Internet of things*. Morgan Kaufmann.

Laudon, K. C., & Laudon, J. P. (2016). *Management information systems: Managing the digital firm* (14th ed.). Pearson.

MariaDB. (2026). *MariaDB MaxScale tutorials*.
<https://mariadb.com/docs/maxscale/mariadb-maxscale-tutorials>

Paredes, M. A. (2025). *Implementación de Zabbix para optimizar el monitoreo de infraestructura tecnológica en el CAEN – EPG, Lima 2022* [Trabajo de suficiencia profesional para optar el título profesional de Ingeniero Empresarial y de Sistemas]. Universidad Tecnológica del Perú .
<https://repositorio.usil.edu.pe/entities/publication/000208f5-6ac6-492b-b656-6a5fc2e02efa>

Project Management Institute. (2021). *A guide to the project management body of knowledge (PMBOK® guide)* (7th ed.). PMI.

Ramírez, A. X. (2023). *Diseño de una plataforma de monitoreo basado en los estados de operación de los activos de red de una empresa de telecomunicaciones* [Trabajo de grado en Ingeniería Electrónica]. Universidad Distrital Francisco José de Caldas .
<https://repository.udistrital.edu.co/server/api/core/bitstreams/dba10bec-42f3-4721-b5fa-bd1212051636/content>

Raymond, E. S. (2001). *The cathedral and the bazaar: Musings on Linux and open source by an accidental revolutionary*. O'Reilly Media.

Richardson, L., & Amundsen, M. (2013). *RESTful web APIs*. O'Reilly Media.

Tanenbaum, A. S., & Wetherall, D. J. (2011). *Computer networks* (5th ed.). Pearson.

Zabbix. (2026a). *What is Zabbix*.
<https://www.zabbix.com/documentation/7.0/en/manual/introduction/about>

Zabbix. (2026b). *Zabbix server*.
<https://www.zabbix.com/documentation/7.0/en/manual/concepts/server>

Zabbix. (2026c). *Zabbix agent*.
<https://www.zabbix.com/documentation/7.0/en/manual/concepts/agent>

Zabbix. (2026d). *Zabbix proxy*.
<https://www.zabbix.com/documentation/7.0/en/manual/concepts/proxy>

Zabbix. (2026e). *Get Zabbix*.
https://www.zabbix.com/download?zabbix=7.0&os_distribution=red_hat_enterprise_linux&os_version=8&components=server_frontend_agent&db=mysql&ws=apache

CAPITULO VIII: ANEXOS

ANEXO A: Glosario de términos

Dispositivo monitoreado: equipo o sistema cuya disponibilidad, rendimiento y funcionamiento son supervisados mediante herramientas de monitoreo, con el fin de detectar fallas, optimizar recursos y asegurar la continuidad de los servicios.

DNS (Domain Name System): Sistema de nombres de dominio que traduce nombres legibles por humanos (como www.ejemplo.com) en direcciones IP numéricas, permitiendo la localización de dispositivos y servicios dentro de una red o en internet.

MariaDB: sistema de gestión de bases de datos relacional de código abierto, compatible con MySQL, utilizado para almacenar y gestionar la información de aplicaciones como Zabbix.

Puerto: Punto lógico de comunicación identificado por un número que permite diferenciar y dirigir el tráfico de red hacia servicios o aplicaciones específicas dentro de un dispositivo. En el contexto de redes basadas en TCP/IP, los puertos facilitan que múltiples servicios operen simultáneamente en un mismo servidor sin interferencias.

ROA: Significa *Return on Assets*, en español rentabilidad sobre los activos. Es un indicador financiero que mide cuánta ganancia genera una empresa con todos sus activos, es decir, con todo lo que posee (dinero, maquinaria, inventarios, edificios, etc.).

ROE: Significa *Return on Equity*, en español rentabilidad sobre el patrimonio, o capital Contable. Es un indicador financiero que mide cuánta ganancia genera una empresa con el dinero aportado por sus accionistas.

SNMPv2 (Simple Network Management Protocol versión 2): Protocolo de red que permite supervisar y administrar dispositivos de red, como routers, switches, balanceadores de carga y otros equipos, mediante la recolección de información sobre su estado y rendimiento.

URL (Uniform Resource Locator): Dirección única que permite localizar un recurso en internet.

ANEXO B: Contenido del archivo de configuración de Zabbix Server

```
# This is a configuration file for Zabbix server daemon
# To get more information about Zabbix, visit https://www.zabbix.com
```

```
##### GENERAL PARAMETERS #####
```

```
### Option: ListenPort
#   Listen port for trapper.
#
# Mandatory: no
# Range: 1024-32767
# Default:
# ListenPort=10051
```

```

#### Option: SourceIP
#   Source IP address for outgoing connections.
#
# Mandatory: no
# Default:
# SourceIP=

```

```

#### Option: LogType
#   Specifies where log messages are written to:
#       system - syslog
#       file   - file specified with LogFile parameter
#       console - standard output
#
# Mandatory: no
# Default:
# LogType=file

```

```

#### Option: LogFile
#   Log file name for LogType 'file' parameter.
#
# Mandatory: yes, if LogType is set to file, otherwise no
# Default:
# LogFile=

```

LogFile=/var/log/zabbix/zabbix_server.log

```

#### Option: LogFileSize
#   Maximum size of log file in MB.
#   0 - disable automatic log rotation.
#
# Mandatory: no
# Range: 0-1024
# Default:
# LogFileSize=1

```

LogFileSize=0

```

#### Option: DebugLevel
#   Specifies debug level:
#   0 - basic information about starting and stopping of Zabbix processes
#   1 - critical information
#   2 - error information
#   3 - warnings
#   4 - for debugging (produces lots of information)
#   5 - extended debugging (produces even more information)
#
# Mandatory: no
# Range: 0-5
# Default:
# DebugLevel=3

```

```

#### Option: PidFile
#   Name of PID file.
#
# Mandatory: no
# Default:
# PidFile=/tmp/zabbix_server.pid

```

```
PidFile=/run/zabbix/zabbix_server.pid
```

```

#### Option: SocketDir
#   IPC socket directory.
#       Directory to store IPC sockets used by internal Zabbix services.
#
# Mandatory: no
# Default:
# SocketDir=/tmp

```

NOTE: Support for Oracle DB is deprecated since Zabbix 7.0 and will be removed in future versions.

```
SocketDir=/run/zabbix
```

```

#### Option: DBHost
#   Database host name.
#   If set to localhost, socket is used for MySQL.
#   If set to empty string, socket is used for PostgreSQL.
#   If set to empty string, the Net Service Name connection method is used
to connect to Oracle database; also see
#   the TNS_ADMIN environment variable to specify the directory where the
tnsnames.ora file is located.
#
# Mandatory: no
# Default:
# DBHost=localhost

```

```

#### Option: DBName
#   Database name.
#   If the Net Service Name connection method is used to connect to Oracle
database, specify the service name from
#   the tnsnames.ora file or set to empty string; also see the TWO_TASK
environment variable if DBName is set to
#   empty string.
#
# Mandatory: yes
# Default:
# DBName=

```

```
DBName=zabbix
```

```

#### Option: DBSchema
#   Schema name. Used for PostgreSQL.
#
# Mandatory: no
# Default:
# DBSchema=

```

```

#### Option: DBUser
#   Database user.
#
# Mandatory: no
# Default:
# DBUser=

```

```
DBUser=zabbix
```

```

#### Option: DBPassword
#   Database password.
#   Comment this line if no password is used.
#
# Mandatory: no
# Default:
DBPassword=m41ns0ft

```

```

#### Option: DBSocket
#   Path to MySQL socket.
#
# Mandatory: no
# Default:
# DBSocket=

```

```

#### Option: DBPort
#   Database port when not using local socket.
#   If the Net Service Name connection method is used to connect to Oracle
#   database, the port number from the
#   tnsnames.ora file will be used. The port number set here will be ignored.
#
# Mandatory: no
# Range: 1024-65535
# Default for MySQL: 3306
# Default for PostgreSQL: 5432
# DBPort=

```

```

#### Option: AllowUnsupportedDBVersions
#   Allow server to work with unsupported database versions.
#   0 - do not allow
#   1 - allow
#
# Mandatory: no
# Default:

```

AllowUnsupportedDBVersions=1

Option: HistoryStorageURL

History storage HTTP[S] URL.

#

Mandatory: no

Default:

HistoryStorageURL=

Option: HistoryStorageTypes

Comma separated list of value types to be sent to the history storage.

#

Mandatory: no

Default:

HistoryStorageTypes=uint,dbl,str,log,text

Option: HistoryStorageDateIndex

Enable preprocessing of history values in history storage to store values in different indices based on date.

0 - disable

1 - enable

#

Mandatory: no

Default:

HistoryStorageDateIndex=0

Option: ExportDir

Directory for real time export of events, history and trends in newline delimited JSON format.

If set, enables real time export.

#

Mandatory: no

Default:

ExportDir=

Option: ExportFileSize

Maximum size per export file in bytes.

Only used for rotation if ExportDir is set.

#

Mandatory: no

Range: 1M-1G

Default:

ExportFileSize=1G

Option: ExportType

List of comma delimited types of real time export - allows to control export entities by their

type (events, history, trends) individually.

Valid only if ExportDir is set.

#

```
# Mandatory: no
# Default:
# ExportType=events,history,trends
```

ADVANCED PARAMETERS

```
### Option: StartPollers
#   Number of pre-forked instances of pollers.
#
# Mandatory: no
# Range: 0-1000
# Default:
# StartPollers=5
```

```
### Option: StartAgentPollers
#   Number of pre-forked instances of asynchronous Zabbix agent pollers.
#   Also see MaxConcurrentChecksPerPoller.
#
# Mandatory: no
# Range: 0-1000
# Default:
# StartAgentPollers=1
```

```
### Option: StartHTTPAgentPollers
#   Number of pre-forked instances of asynchronous HTTP agent pollers.
#   Also see MaxConcurrentChecksPerPoller.
#
# Mandatory: no
# Range: 0-1000
# Default:
# StartHTTPAgentPollers=1
```

```
### Option: StartSNMPPollers
#   Number of pre-forked instances of asynchronous SNMP pollers. Also see
#   MaxConcurrentChecksPerPoller.
#
# Mandatory: no
# Range: 0-1000
# Default:
# StartSNMPPollers=1
```

```
### Option: MaxConcurrentChecksPerPoller
#   Maximum number of asynchronous checks that can be executed at once
#   by each HTTP agent poller or agent poller.
#
# Mandatory: no
# Range: 1-1000
# Default:
# MaxConcurrentChecksPerPoller=1000
```

Option: StartIPMIPollers

Number of pre-forked instances of IPMI pollers.
 # The IPMI manager process is automatically started when at least one IPMI poller is started.
 #
 # Mandatory: no
 # Range: 0-1000
 # Default:
 # StartIPMIPollers=0

Option: StartPreprocessors

Number of pre-started instances of preprocessing worker threads should be set to no less than
 # the available CPU core count. More workers should be set if preprocessing is not CPU-bound and has
 # lots of network requests.
 # Mandatory: no
 # Range: 1-1000
 # Default:
 # StartPreprocessors=16

Option: StartConnectors

Number of pre-forked instances of connector workers.
 # The connector manager process is automatically started when connector worker is started.
 #
 # Mandatory: no
 # Range: 0-1000
 # Default:
 # StartConnectors=0

Option: StartPollersUnreachable

Number of pre-forked instances of pollers for unreachable hosts (including IPMI and Java).
 # At least one poller for unreachable hosts must be running if regular, IPMI or Java pollers
 # are started.
 #
 # Mandatory: no
 # Range: 0-1000
 # Default:
 # StartPollersUnreachable=1

Option: StartHistoryPollers

Number of pre-forked instances of history pollers.
 # Only required for calculated checks.
 # A database connection is required for each history poller instance.
 #
 # Mandatory: no

Range: 0-1000
Default:
StartHistoryPollers=5

Option: StartTrappers

Number of pre-forked instances of trappers.
Trappers accept incoming connections from Zabbix sender, active agents and active proxies.
At least one trapper process must be running to display server availability and view queue
in the frontend.

Mandatory: no
Range: 0-1000
Default:
StartTrappers=5

Option: StartPingers

Number of pre-forked instances of ICMP pingers.

Mandatory: no
Range: 0-1000
Default:
StartPingers=1

Option: StartDiscoverers

Number of pre-started instances of discovery workers.

Mandatory: no
Range: 0-1000
Default:
StartDiscoverers=5

Option: StartHTTTPollers

Number of pre-forked instances of HTTP pollers.

Mandatory: no
Range: 0-1000
Default:
StartHTTTPollers=1

Option: StartTimers

Number of pre-forked instances of timers.
Timers process maintenance periods.
Only the first timer process handles host maintenance updates. Problem suppression updates are shared
between all timers.

Mandatory: no
Range: 1-1000

```
# Default:
# StartTimers=1

#### Option: StartEscalators
#   Number of pre-forked instances of escalators.
#
# Mandatory: no
# Range: 1-100
# Default:
# StartEscalators=1

#### Option: StartAlerters
#   Number of pre-forked instances of alerters.
#   Alerters send the notifications created by action operations.
#
# Mandatory: no
# Range: 1-100
# Default:
# StartAlerters=3

#### Option: JavaGateway
#   IP address (or hostname) of Zabbix Java gateway.
#   Only required if Java pollers are started.
#
# Mandatory: no
# Default:
# JavaGateway=

#### Option: JavaGatewayPort
#   Port that Zabbix Java gateway listens on.
#
# Mandatory: no
# Range: 1024-32767
# Default:
# JavaGatewayPort=10052

#### Option: StartJavaPollers
#   Number of pre-forked instances of Java pollers.
#
# Mandatory: no
# Range: 0-1000
# Default:
# StartJavaPollers=0

#### Option: StartVMwareCollectors
#   Number of pre-forked vmware collector instances.
#
# Mandatory: no
# Range: 0-250
# Default:
```

```
# StartVMwareCollectors=0
```

```
#### Option: VMwareFrequency
```

```
#   How often Zabbix will connect to VMware service to obtain a new data.
```

```
#
```

```
# Mandatory: no
```

```
# Range: 10-86400
```

```
# Default:
```

```
# VMwareFrequency=60
```

```
#### Option: VMwarePerfFrequency
```

```
#   How often Zabbix will connect to VMware service to obtain performance data.
```

```
#
```

```
# Mandatory: no
```

```
# Range: 10-86400
```

```
# Default:
```

```
# VMwarePerfFrequency=60
```

```
#### Option: VMwareCacheSize
```

```
#   Size of VMware cache, in bytes.
```

```
#   Shared memory size for storing VMware data.
```

```
#   Only used if VMware collectors are started.
```

```
#
```

```
# Mandatory: no
```

```
# Range: 256K-2G
```

```
# Default:
```

```
# VMwareCacheSize=8M
```

```
#### Option: VMwareTimeout
```

```
#   Specifies how many seconds vmware collector waits for response from VMware service.
```

```
#
```

```
# Mandatory: no
```

```
# Range: 1-300
```

```
# Default:
```

```
# VMwareTimeout=10
```

```
#### Option: SNMPTrapperFile
```

```
#   Temporary file used for passing data from SNMP trap daemon to the server.
```

```
#   Must be the same as in zabbix_trap_receiver.pl or SNMPTT configuration file.
```

```
#
```

```
# Mandatory: no
```

```
# Default:
```

```
# SNMPTrapperFile=/tmp/zabbix_traps.tmp
```

```
SNMPTrapperFile=/var/log/snmptrap/snmptrap.log
```

```

#### Option: StartSNMPTrapper
#   If 1, SNMP trapper process is started.
#
# Mandatory: no
# Range: 0-1
# Default:
# StartSNMPTrapper=0

```

```

#### Option: ListenIP
#   List of comma delimited IP addresses that the trapper should listen on.
#   Trapper will listen on all network interfaces if this parameter is missing.
#
# Mandatory: no
# Default:
# ListenIP=0.0.0.0

```

```

#### Option: HousekeepingFrequency
#   How often Zabbix will perform housekeeping procedure (in hours).
#   Housekeeping is removing outdated information from the database.
#   To prevent Housekeeper from being overloaded, no more than 4 times
HousekeepingFrequency
#   hours of outdated information are deleted in one housekeeping cycle, for
each item.
#   To lower load on server startup housekeeping is postponed for 30 minutes
after server start.
#   With HousekeepingFrequency=0 the housekeeper can be only executed
using the runtime control option.
#       In this case the period of outdated information deleted in one
housekeeping cycle is 4 times the
#   period since the last housekeeping cycle, but not less than 4 hours and
not greater than 4 days.
#
# Mandatory: no
# Range: 0-24
# Default:
# HousekeepingFrequency=1

```

```

#### Option: MaxHousekeeperDelete
#   The table "housekeeper" contains "tasks" for housekeeping procedure in
the format:
#   [housekeeperid], [tablename], [field], [value].
#       No more than 'MaxHousekeeperDelete' rows (corresponding to
[tablename], [field], [value])
#   will be deleted per one task in one housekeeping cycle.
#   If set to 0 then no limit is used at all. In this case you must know what
you are doing!
#
# Mandatory: no
# Range: 0-1000000
# Default:

```

```
# MaxHousekeeperDelete=5000
```

```
#### Option: CacheSize
```

```
#   Size of configuration cache, in bytes.
```

```
#   Shared memory size for storing host, item and trigger data.
```

```
#
```

```
# Mandatory: no
```

```
# Range: 128K-64G
```

```
# Default:
```

```
# CacheSize=32M
```

```
#### Option: CacheUpdateFrequency
```

```
#   How often Zabbix will perform update of configuration cache, in seconds.
```

```
#
```

```
# Mandatory: no
```

```
# Range: 1-3600
```

```
# Default:
```

```
# CacheUpdateFrequency=10
```

```
#### Option: StartDBSyncers
```

```
#   Number of pre-forked instances of DB Syncers.
```

```
#
```

```
# Mandatory: no
```

```
# Range: 1-100
```

```
# Default:
```

```
# StartDBSyncers=4
```

```
#### Option: HistoryCacheSize
```

```
#   Size of history cache, in bytes.
```

```
#   Shared memory size for storing history data.
```

```
#
```

```
# Mandatory: no
```

```
# Range: 128K-2G
```

```
# Default:
```

```
# HistoryCacheSize=16M
```

```
#### Option: HistoryIndexCacheSize
```

```
#   Size of history index cache, in bytes.
```

```
#   Shared memory size for indexing history cache.
```

```
#
```

```
# Mandatory: no
```

```
# Range: 128K-2G
```

```
# Default:
```

```
# HistoryIndexCacheSize=4M
```

```
#### Option: TrendCacheSize
```

```
#   Size of trend write cache, in bytes.
```

```
#   Shared memory size for storing trends data.
```

```
#
```

```
# Mandatory: no
```

```
# Range: 128K-2G
# Default:
# TrendCacheSize=4M
```

```
#### Option: TrendFunctionCacheSize
#   Size of trend function cache, in bytes.
#   Shared memory size for caching calculated trend function data.
#
# Mandatory: no
# Range: 128K-2G
# Default:
# TrendFunctionCacheSize=4M
```

```
#### Option: ValueCacheSize
#   Size of history value cache, in bytes.
#   Shared memory size for caching item history data requests.
#   Setting to 0 disables value cache.
#
# Mandatory: no
# Range: 0,128K-64G
# Default:
# ValueCacheSize=8M
```

```
#### Option: Timeout
#   Specifies how long to wait (in seconds) for establishing connection and
#   exchanging data with Zabbix proxy, agent, web service, and for SNMP checks
#   (except SNMP `walk[OID]` and `get[OID]` items) and `icmping[*]` item.
#
# Mandatory: no
# Range: 1-30
# Default:
# Timeout=3
```

```
Timeout=4
```

```
#### Option: TrapperTimeout
#   Specifies timeout in seconds for:
#       retrieval of historical data from Zabbix proxy
#       sending configuration data to Zabbix proxy
#       Global script / remote command execution on Zabbix proxy / server
#
# Mandatory: no
# Range: 1-300
# Default:
# TrapperTimeout=300
```

```
#### Option: UnreachablePeriod
#   After how many seconds of unreachability treat a host as unavailable.
#
# Mandatory: no
```

```
# Range: 1-3600
# Default:
# UnreachablePeriod=45

#### Option: UnavailableDelay
#   How often host is checked for availability during the unavailability period,
#   in seconds.
#
# Mandatory: no
# Range: 1-3600
# Default:
# UnavailableDelay=60

#### Option: UnreachableDelay
#   How often host is checked for availability during the unreachability period,
#   in seconds.
#
# Mandatory: no
# Range: 1-3600
# Default:
# UnreachableDelay=15

#### Option: AlertScriptsPath
#   Full path to location of custom alert scripts.
#   Default depends on compilation options.
#   To see the default path run command "zabbix_server --help".
#
# Mandatory: no
# Default:
# AlertScriptsPath=/usr/lib/zabbix/alertscripts

#### Option: ExternalScripts
#   Full path to location of external scripts.
#   Default depends on compilation options.
#   To see the default path run command "zabbix_server --help".
#
# Mandatory: no
# Default:
# ExternalScripts=/usr/lib/zabbix/externalscripts

#### Option: FpingLocation
#   Location of fping.
#   Make sure that fping binary has root ownership and SUID flag set.
#
# Mandatory: no
# Default:
# FpingLocation=/usr/sbin/fping

#### Option: Fping6Location
#   Location of fping6.
```

```

# Make sure that fping6 binary has root ownership and SUID flag set.
# Make empty if your fping utility is capable to process IPv6 addresses.
#
# Mandatory: no
# Default:
# Fping6Location=/usr/sbin/fping6

### Option: SSHKeyLocation
# Location of public and private keys for SSH checks and actions.
#
# Mandatory: no
# Default:
# SSHKeyLocation=

### Option: LogSlowQueries
# How long a database query may take before being logged (in
# milliseconds).
# Only works if DebugLevel set to 3, 4 or 5.
# 0 - don't log slow queries.
#
# Mandatory: no
# Range: 1-3600000
# Default:
# LogSlowQueries=0

LogSlowQueries=3000

### Option: TmpDir
# Temporary directory.
#
# Mandatory: no
# Default:
# TmpDir=/tmp

### Option: StartProxyPollers
# Number of pre-forked instances of pollers for passive proxies.
#
# Mandatory: no
# Range: 0-250
# Default:
# StartProxyPollers=1

### Option: ProxyConfigFrequency
# How often Zabbix Server sends configuration data to a Zabbix Proxy in
# seconds.
# This parameter is used only for proxies in the passive mode.
#
# Mandatory: no
# Range: 1-3600*24*7
# Default:

```

```
# ProxyConfigFrequency=10
```

```
#### Option: ProxyDataFrequency
```

```
#   How often Zabbix Server requests history data from a Zabbix Proxy in seconds.
```

```
#   This parameter is used only for proxies in the passive mode.
```

```
#
```

```
# Mandatory: no
```

```
# Range: 1-3600
```

```
# Default:
```

```
# ProxyDataFrequency=1
```

```
#### Option: StartLLDProcessors
```

```
#   Number of pre-forked instances of low level discovery processors.
```

```
#
```

```
# Mandatory: no
```

```
# Range: 1-100
```

```
# Default:
```

```
# StartLLDProcessors=2
```

```
#### Option: AllowRoot
```

```
#   Allow the server to run as 'root'. If disabled and the server is started by 'root', the server
```

```
#   will try to switch to the user specified by the User configuration option instead.
```

```
#   Has no effect if started under a regular user.
```

```
#   0 - do not allow
```

```
#   1 - allow
```

```
#
```

```
# Mandatory: no
```

```
# Default:
```

```
# AllowRoot=0
```

```
#### Option: User
```

```
#   Drop privileges to a specific, existing user on the system.
```

```
#   Only has effect if run as 'root' and AllowRoot is disabled.
```

```
#
```

```
# Mandatory: no
```

```
# Default:
```

```
# User=zabbix
```

```
#### Option: SSLCertLocation
```

```
#   Location of SSL client certificates.
```

```
#   This parameter is used in web monitoring and for communication with Vault.
```

```
#   Default depends on compilation options.
```

```
#   To see the default path run command "zabbix_server --help".
```

```
#
```

```
# Mandatory: no
```

```
# Default:
```

```
# SSLCertLocation=${datadir}/zabbix/ssl/certs
```

```
#### Option: SSLKeyLocation
```

```
# Location of private keys for SSL client certificates.
# This parameter is used in web monitoring and for communication with Vault.
```

```
# Default depends on compilation options.
# To see the default path run command "zabbix_server --help".
```

```
#
# Mandatory: no
# Default:
```

```
# SSLKeyLocation=${datadir}/zabbix/ssl/keys
```

```
#### Option: SSLCALocation
```

```
# Override the location of certificate authority (CA) files for SSL server certificate verification.
```

```
# If not set, system-wide directory will be used.
```

```
# This parameter is used in web monitoring, SMTP authentication, HTTP agent items and for communication with Vault.
```

```
#
# Mandatory: no
# Default:
```

```
# SSLCALocation=
```

```
#### Option: StatsAllowedIP
```

```
# List of comma delimited IP addresses, optionally in CIDR notation, or DNS names of external Zabbix instances.
```

```
# Stats request will be accepted only from the addresses listed here. If this parameter is not set no stats requests
```

```
# will be accepted.
```

```
# If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally
```

```
# and ':::0' will allow any IPv4 or IPv6 address.
```

```
# '0.0.0.0/0' can be used to allow any IPv4 address.
```

```
# Example:
```

```
StatsAllowedIP=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
```

```
#
# Mandatory: no
# Default:
```

```
# StatsAllowedIP=
```

```
StatsAllowedIP=127.0.0.1
```

```
##### LOADABLE MODULES #####
```

```
#### Option: LoadModulePath
```

```
# Full path to location of server modules.
```

```
# Default depends on compilation options.
```

```
# To see the default path run command "zabbix_server --help".
```

```
#
```

```
# Mandatory: no
# Default:
# LoadModulePath=${libdir}/modules
```

Option: LoadModule

```
# Module to load at server startup. Modules are used to extend functionality
# of the server.
# Formats:
#     LoadModule=<module.so>
#     LoadModule=<path/module.so>
#     LoadModule=</abs_path/module.so>
#     Either the module must be located in directory specified by
#     LoadModulePath or the path must precede the module name.
#     If the preceding path is absolute (starts with '/') then LoadModulePath is
#     ignored.
#     It is allowed to include multiple LoadModule parameters.
#
# Mandatory: no
# Default:
# LoadModule=
```

TLS-RELATED PARAMETERS

Option: TLSCAFile

```
# Full pathname of a file containing the top-level CA(s) certificates for
# peer certificate verification.
#
# Mandatory: no
# Default:
# TLSCAFile=
```

Option: TLSCRLFile

```
# Full pathname of a file containing revoked certificates.
#
# Mandatory: no
# Default:
# TLSCRLFile=
```

Option: TLSCertFile

```
# Full pathname of a file containing the server certificate or certificate chain.
#
# Mandatory: no
# Default:
# TLSCertFile=
```

Option: TLSKeyFile

```
# Full pathname of a file containing the server private key.
#
# Mandatory: no
# Default:
```

```
# TLSKeyFile=
```

```
##### For advanced users - TLS ciphersuite selection criteria #####
```

```
### Option: TLSCipherCert13
```

```
# Cipher string for OpenSSL 1.1.1 or newer in TLS 1.3.
```

```
# Override the default ciphersuite selection criteria for certificate-based encryption.
```

```
#
```

```
# Mandatory: no
```

```
# Default:
```

```
# TLSCipherCert13=
```

```
### Option: TLSCipherCert
```

```
# GnuTLS priority string or OpenSSL (TLS 1.2) cipher string.
```

```
# Override the default ciphersuite selection criteria for certificate-based encryption.
```

```
# Example for GnuTLS:
```

```
# NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL:+CTYPE-X.509
```

```
# Example for OpenSSL:
```

```
# EECDH+aRSA+AES128:RSA+aRSA+AES128
```

```
#
```

```
# Mandatory: no
```

```
# Default:
```

```
# TLSCipherCert=
```

```
### Option: TLSCipherPSK13
```

```
# Cipher string for OpenSSL 1.1.1 or newer in TLS 1.3.
```

```
# Override the default ciphersuite selection criteria for PSK-based encryption.
```

```
# Example:
```

```
#
```

```
TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

```
#
```

```
# Mandatory: no
```

```
# Default:
```

```
# TLSCipherPSK13=
```

```
### Option: TLSCipherPSK
```

```
# GnuTLS priority string or OpenSSL (TLS 1.2) cipher string.
```

```
# Override the default ciphersuite selection criteria for PSK-based encryption.
```

```
# Example for GnuTLS:
```

```
# NONE:+VERS-TLS1.2:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL
```

```
# Example for OpenSSL:
```

```
# kECDHEPSK+AES128:kPSK+AES128
```

```

#
# Mandatory: no
# Default:
# TLSCipherPSK=

#### Option: TLSCipherAll13
#   Cipher string for OpenSSL 1.1.1 or newer in TLS 1.3.
#   Override the default ciphersuite selection criteria for certificate- and PSK-
#   based encryption.
#   Example:
#
TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_
AES_128_GCM_SHA256
#
# Mandatory: no
# Default:
# TLSCipherAll13=

#### Option: TLSCipherAll
#   GnuTLS priority string or OpenSSL (TLS 1.2) cipher string.
#   Override the default ciphersuite selection criteria for certificate- and PSK-
#   based encryption.
#   Example for GnuTLS:
#
NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+ECDHE-
PSK:+PSK:+AES-128-GCM:+AES-128-
CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP=NULL:+SIGN-
ALL:+CTYPE-X.509
#   Example for OpenSSL:
#
EECDH+aRSA+AES128:RSA+aRSA+AES128:kECDHEPSK+AES128:kPSK
+AES128
#
# Mandatory: no
# Default:
# TLSCipherAll=

#### Option: DBTLSConnect
#   Setting this option enforces to use TLS connection to database.
#   required   - connect using TLS
#   verify_ca  - connect using TLS and verify certificate
#   verify_full - connect using TLS, verify certificate and verify that database
#               identity specified by DBHost
#               matches its certificate
#   On MySQL starting from 5.7.11 and PostgreSQL following values are
#   supported: "required", "verify_ca" and
#   "verify_full".
#   On MariaDB starting from version 10.2.6 "required" and "verify_full"
#   values are supported.
#   Default is not to set any option and behavior depends on database
#   configuration

```

```

#
# Mandatory: no
# Default:
# DBTLSConnect=

#### Option: DBTLSCAFile
#     Full pathname of a file containing the top-level CA(s) certificates for
#     database certificate verification.
#     Supported only for MySQL and PostgreSQL
#
# Mandatory: no
#     (yes, if DBTLSConnect set to one of: verify_ca, verify_full)
# Default:
# DBTLSCAFile=

#### Option: DBTLSCertFile
#     Full pathname of file containing Zabbix server certificate for authenticating
#     to database.
#     Supported only for MySQL and PostgreSQL
#
# Mandatory: no
# Default:
# DBTLSCertFile=

#### Option: DBTLSKeyFile
#     Full pathname of file containing the private key for authenticating to
#     database.
#     Supported only for MySQL and PostgreSQL
#
# Mandatory: no
# Default:
# DBTLSKeyFile=

#### Option: DBTLSCipher
#     The list of encryption ciphers that Zabbix server permits for TLS protocols
#     up through TLSv1.2
#     Supported only for MySQL
#
# Mandatory no
# Default:
# DBTLSCipher=

#### Option: DBTLSCipher13
#     The list of encryption ciphersuites that Zabbix server permits for TLSv1.3
#     protocol
#     Supported only for MySQL, starting from version 8.0.16
#
# Mandatory no
# Default:
# DBTLSCipher13=

```

Option: Vault

```
# Specifies vault:
#     HashiCorp - HashiCorp KV Secrets Engine - Version 2
#     CyberArk - CyberArk Central Credential Provider
#
# Mandatory: no
# Default:
# Vault=HashiCorp
```

Option: VaultToken

```
# Vault authentication token that should have been generated exclusively
# for Zabbix server with read only permission
# to paths specified in Vault macros and read only permission to path
# specified in optional VaultDBPath
# configuration parameter.
# It is an error if VaultToken and VAULT_TOKEN environment variable are
# defined at the same time.
#
# Mandatory: no
# (yes, if Vault is explicitly set to HashiCorp)
# Default:
# VaultToken=
```

Option: VaultURL

```
# Vault server HTTP[S] URL. System-wide CA certificates directory will be
# used if SSLCALocation is not specified.
#
# Mandatory: no
# Default:
# VaultURL=https://127.0.0.1:8200
```

Option: VaultPrefix

```
# Custom prefix for Vault path or query depending on the Vault.
# Most suitable defaults will be used if not specified.
# Note that 'data' is automatically appended after mountpoint for HashiCorp
# if VaultPrefix is not specified.
# Example prefix for HashiCorp:
#     /v1/secret/data/zabbix/
# Example prefix for CyberArk:
#     /AIMWebService/api/Accounts?
# Mandatory: no
# Default:
# VaultPrefix=
```

Option: VaultDBPath

```
# Vault path or query depending on the Vault from where credentials for
# database will be retrieved by keys.
# Keys used for HashiCorp are 'password' and 'username'.
# Example path with VaultPrefix=/v1/secret/data/zabbix/:
```

```

#         database
#         Example path without VaultPrefix:
#             secret/zabbix/database
#     Keys used for CyberArk are 'Content' and 'UserName'.
#     Example query:
#
AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix_server_da
tabase
#     This option can only be used if DBUser and DBPassword are not
specified.
#
# Mandatory: no
# Default:
# VaultDBPath=

### Option: VaultTLSCertFile
#     Name of the SSL certificate file used for client authentication. The
certificate file must be in PEM1 format.
#     If the certificate file contains also the private key, leave the SSL key file
field empty. The directory
#         containing this file is specified by configuration parameter
SSLCertLocation.
#
# Mandatory: no
# Default:
# VaultTLSCertFile=

### Option: VaultTLSKeyFile
#     Name of the SSL private key file used for client authentication. The private
key file must be in PEM1 format.
#     The directory containing this file is specified by configuration parameter
SSLKeyLocation.
#
# Mandatory: no
# Default:
# VaultTLSKeyFile=

### Option: StartReportWriters
#     Number of pre-forked report writer instances.
#
# Mandatory: no
# Range: 0-100
# Default:
# StartReportWriters=0

### Option: WebServiceURL
#     URL to Zabbix web service, used to perform web related tasks.
#     Example: http://localhost:10053/report
#
# Mandatory: no

```

```

# Default:
# WebServiceURL=

#### Option: ServiceManagerSyncFrequency
#   How often Zabbix will synchronize configuration of a service manager (in
seconds).
#
# Mandatory: no
# Range: 1-3600
# Default:
# ServiceManagerSyncFrequency=60

#### Option: ProblemHousekeepingFrequency
#   How often Zabbix will delete problems for deleted triggers (in seconds).
#
# Mandatory: no
# Range: 1-3600
# Default:
# ProblemHousekeepingFrequency=60

## Option: StartODBCPollers
#   Number of pre-forked ODBC poller instances.
#
# Mandatory: no
# Range: 0-1000
# Default:
# StartODBCPollers=1

#### Option: EnableGlobalScripts
#   Enable global scripts on Zabbix server.
#   0 - disable
#   1 - enable
#
# Mandatory: no
# Default:
# EnableGlobalScripts=1
EnableGlobalScripts=0

# Option: AllowSoftwareUpdateCheck
#   Allow Zabbix UI to receive information about software updates from
zabbix.com
#   0 - disable software update checks
#   1 - enable software update checks
#
# Mandatory: no
# Default:
# AllowSoftwareUpdateCheck=1

#### Option: SMSDevices
#   List of comma delimited modem files allowed to use Zabbix server

```

```
# SMS sending not possible if this parameter is not set
# Example: SMSDevices=/dev/ttyUSB0,/dev/ttyUSB1
#
# Mandatory: no
# Default:
# SMSDevices=
```

```
##### For advanced users - TCP-related fine-tuning parameters #####
```

```
## Option: ListenBacklog
# The maximum number of pending connections in the queue. This
# parameter is passed to
# listen() function as argument 'backlog' (see "man listen").
#
# Mandatory: no
# Range: 0 - INT_MAX (depends on system, too large values may be silently
# truncated to implementation-specified maximum)
# Default: SOMAXCONN (hard-coded constant, depends on system)
# ListenBacklog=
```

```
##### High availability cluster parameters #####
```

```
## Option: HANodeName
# The high availability cluster node name.
# When empty, server is working in standalone mode; a node with empty
# name is registered with address for the frontend to connect to.
#
# Mandatory: no
# Default:
# HANodeName=
```

```
## Option: NodeAddress
# IP or hostname with optional port to specify how frontend should connect
# to the server.
# Format: <address>[:<port>]
#
# If IP or hostname is not set, then ListenIP value will be used. In case
# ListenIP is not set, localhost will be used.
# If port is not set, then ListenPort value will be used. In case ListenPort is
# not set, 10051 will be used.
# This option can be overridden by address specified in frontend
# configuration.
#
# Mandatory: no
# Default:
# NodeAddress=localhost:10051
```

```
##### Browser monitoring #####
```

```

#### Option: WebDriverURL
#   WebDriver interface HTTP[S] URL. For example http://localhost:4444
#   used with Selenium WebDriver standalone server.
#
# Mandatory: no
# Default:
# WebDriverURL=

#### Option: StartBrowserPollers
#   Number of pre-forked instances of browser item pollers.
#
# Mandatory: no
# Range: 0-1000
# Default:
# StartBrowserPollers=1

##### Additional configuration files #####

#### Option: Include
#   You may include individual files or all files in a directory in the configuration
#   file.
#   Installing Zabbix will create include directory in /usr/local/etc, unless
#   modified during the compile time.
#
# Mandatory: no
# Default:
# Include=

Include=/etc/zabbix/zabbix_server.d/*.conf

# Include=/usr/local/etc/zabbix_server.general.conf
# Include=/usr/local/etc/zabbix_server.conf.d/
# Include=/usr/local/etc/zabbix_server.conf.d/*.conf

```

ANEXO C: Contenido del archivo de configuración de Zabbix FrontEnd

```

$DB['TYPE']           = 'MYSQL';
$DB['SERVER']         = 'localhost';
$DB['PORT']           = '0';
$DB['DATABASE']       = 'zabbix';
$DB['USER']           = 'zabbix';
$DB['PASSWORD']       = 'm41ns0ft';
$DB['SCHEMA']         = "";
$DB['ENCRYPTION']     = false;
$DB['KEY_FILE']       = "";
$DB['CERT_FILE']      = "";
$DB['CA_FILE']        = "";

```

```

$DB['VERIFY_HOST']      = false;
$DB['CIPHER_LIST']      = "";
$DB['VAULT']            = "";
$DB['VAULT_URL']        = "";
$DB['VAULT_PREFIX']     = "";
$DB['VAULT_DB_PATH']    = "";
$DB['VAULT_TOKEN']      = "";
$DB['VAULT_CERT_FILE']  = "";
$DB['VAULT_KEY_FILE']   = "";
$ZBX_SERVER_NAME        = 'zabbix-mainsoft';
$IMAGE_FORMAT_DEFAULT   = IMAGE_FORMAT_PNG;
$SSO['CERT_STORAGE']    = 'database';
$ZBX_SERVER_TLS['ACTIVE'] = '0';
$ZBX_SERVER_TLS['CA_FILE'] = "";
$ZBX_SERVER_TLS['KEY_FILE'] = "";
$ZBX_SERVER_TLS['CERT_FILE'] = "";
$ZBX_SERVER_TLS['CERTIFICATE_ISSUER'] = "";
$ZBX_SERVER_TLS['CERTIFICATE_SUBJECT'] = "";

```

ANEXO D: Severidad de alertas configuradas

Las severidades de las alertas configuradas, ordenadas de menor a mayor severidad son:

- No clasificada.
- Informativa.
- Precaución.
- Media.
- Alta.
- Crítica.

ANEXO E: Configuración de notificación por correo electrónico

Parámetros de configuración:

- Name: Gmail.
- Type: Email.
- Email provider: Generic SMTP.
- SMTP server: smtp.gmail.com.

- SMTP server port: 587.
- Email: zabbix@notifications.com
- SMTP helo: example.com.
- Connection security: STARTTLS.