Automatización y mejora continúa en los procesos de la jefatura Core Voz y Cico de Telefónica

Colombia Telecomunicaciones

Autor
DAVID SANTIAGO AVILAN LOPEZ

UNIVERSIDAD SANTO TOMAS
Facultad de ingeniería de telecomunicaciones

Bogotá D.C. 2024

Automatización y mejora continúa en los procesos de la jefatura Core Voz y Cico de Telefónica

Colombia Telecomunicaciones

Autor
David Santiago Avilan Lopez

INFORME FINAL DE PASANTÍA

Director Interno
Fernando Prieto Bustamante

Director Externo
Jorge Eduardo Triana

UNIVERSIDAD SANTO TOMAS
Facultad de ingeniería de Telecomunicaciones
Bogotá D.C. Mayo, 2024

Dedicatoria

Dedico este trabajo a todas las personas y organizaciones que me acompañaron en este proceso: a mis profesores, mentores, compañeros, y especialmente a mi familia, por creer en mí y apoyarme en cada paso del camino.

Contenido

Indicé de Figuras

Indicé de Gráficos

Indicé de Tablas

Resumen

El presente trabajo de monografía describe el desarrollo de una automatización y mejora continua en los procesos de la jefatura Core Voz y Cico de Telefónica. El enfoque principal es desarrollar un sistema automatizado que permita la gestión y supervisión eficiente de servidores remotos, implementando buenas prácticas de ITIL 4 y el uso de metodologías ágiles.

La integración de herramientas como Zabbix, por parte de la empresa, es crucial para mejorar la eficiencia operativa y la calidad del servicio, proporcionando soluciones innovadoras que optimizan los procesos y reducen el trabajo manual. Este documento detalla el desarrollo del proyecto, la implementación de los scripts automatizados y la integración de estos con las plataformas existentes, así como los resultados obtenidos y las mejoras logradas.

Abstract

This monograph describes the development of an automation and continuous improvement in the processes of the Core Voice and Cico department of Telefónica. The main focus is to develop an automated system that allows the efficient management and monitoring of remote servers, implementing ITIL 4 best practices and the use of agile methodologies.

The integration of tools such as Zabbix, by the company, is crucial to improve operational efficiency and service quality, providing innovative solutions that optimize processes and reduce manual work. This document details the development of the project, the implementation of the automated scripts and their integration with existing platforms, as well as the results obtained and the improvements achieved.

Introducción

En el presente documento evidencia las actividades realizadas como practicante en la empresa Colombia Telecomunicaciones, específicamente en el área de Core Voz y CICO, la cual se encarga de apoyar la operación de las redes de telecomunicaciones Core y las plataformas de la red móvil.

La propuesta busca una solución innovadora que permita mejorar el flujo de trabajo y reducir los procesos manuales internos, así como optimizar los proyectos en curso y futuros, promoviendo un mejoramiento continuo mediante el uso de herramientas y metodologías ágiles para un adecuado orden y planificación en el desarrollo de los objetivos establecidos por la jefatura.

El proyecto de mejoramiento y desarrollo se reflejó en el área de Core Voz y CICO a través de un proyecto interno de desarrollo de scripts y la implementación de metodologías ágiles. Las actividades propuestas se llevaron a cabo en un periodo de seis meses de manera híbrida, incluyendo orientaciones, clases, comunicación con otras empresas de apoyo y recomendaciones por parte de los miembros de la jefatura.

Gracias a los cursos proporcionados por la empresa, el análisis y la estrategia de metodologías ágiles fueron más sencillos de implementar en el proyecto vigente, destacando el uso de estas metodologías en las TI para mejorar la experiencia en el desarrollo de proyectos.

El documento incluye una descripción del problema identificado en la empresa junto con su justificación, los objetivos planteados para resolver dicho problema, un marco referencial sobre la empresa, un marco teórico de los temas a tratar, las actividades desarrolladas, y finalmente, las conclusiones.

1. MARCO GENERAL DEL PROYECTO

1.1. Planteamiento del problema

En Colombia Telecomunicaciones, la gestión y supervisión de servidores remotos se enfrenta a desafíos significativos debido a la naturaleza manual y repetitiva de las tareas involucradas. Desde la recolección de datos sobre sistemas de archivos hasta la generación de alertas por posibles problemas, estas actividades consumen recursos valiosos y tiempo, lo que puede provocar retrasos en la resolución de incidentes y una disminución en la eficiencia operativa. Este impacto negativo repercute directamente en la calidad del servicio, especialmente en áreas críticas como Core Voz y Cico.

La jefatura de Core Voz y Cico ha identificado un problema de alto uso del disco en dos equipos vMTAS (Cali y Barranquilla). Este uso excesivo ha superado el umbral crítico del 80% en un archivo del sistema (trace.log), lo que está causando problemas en el sistema y generando alarmas relacionadas con la capacidad del disco. Como resultado, el equipo de ingeniería debe programar ventanas de mantenimiento constantes para eliminar este archivo, el cual está ocupando una cantidad significativa de espacio en disco.

El principal reto radica en la necesidad de desarrollar un sistema automatizado que simplifique estas tareas y mejore la eficiencia en la gestión de los servidores remotos. Este sistema debe integrarse de manera efectiva con herramientas de monitoreo existentes, como Zabbix, y adoptar metodologías ágiles como ITIL4, Scrum y Kanban para garantizar una administración más eficaz y una respuesta ágil a los cambios en el entorno de TI.

La consolidación del monitoreo y supervisión en herramientas como Zabbix se presenta como una solución crucial para mejorar la eficacia y uniformidad en los procesos. Esta integración permitiría un aumento significativo en el nivel de automatización dentro de la empresa, al tiempo que facilitaría una gestión más eficiente y centralizada de los recursos de TI. Además, esta mejora en la eficiencia operativa y la calidad del servicio ofrecido se verían respaldada por una identificación temprana de problemas y una toma de decisiones más informada.

1.2. Justificación

La automatización y mejora continua en los procesos de la jefatura Core Voz y Cico de Telefónica es crucial para garantizar la eficiencia operativa y la calidad del servicio. En un entorno tan dinámico y competitivo como el de las telecomunicaciones, es esencial contar con sistemas que no solo optimicen los recursos y procesos, sino que también aseguren una respuesta rápida y eficaz ante cualquier incidencia.

La implementación de metodologías ágiles y buenas prácticas de ITIL4 permite establecer un marco de trabajo estructurado y adaptable, promoviendo la colaboración entre equipos y facilitando la gestión del cambio. Estas metodologías ayudan a mantener un enfoque constante en la mejora continua, permitiendo a la organización anticiparse a problemas y responder de manera proactiva a las necesidades del mercado.

La justificación de este proyecto radica en la necesidad de incrementar la eficiencia operativa, reducir tiempos de respuesta ante incidentes y optimizar el uso de recursos. Al automatizar procesos críticos y aplicar principios de mejora continua, se espera no solo mejorar la calidad del servicio ofrecido a los clientes, sino también generar un impacto positivo en la productividad y satisfacción del equipo de trabajo.

Además, la integración de sistemas de monitoreo avanzados como Zabbix, permite una supervisión en tiempo real de los recursos y procesos, facilitando la detección temprana de problemas y la implementación de soluciones rápidas y efectivas. Esta capacidad de monitoreo y respuesta es fundamental para mantener la continuidad del servicio y asegurar que los estándares de calidad se mantengan elevados. Esto es esencial para la evolución y fortalecimiento de la jefatura Core Voz y Cico de Telefónica, permitiendo a la organización mantenerse competitiva y asegurar la satisfacción de sus clientes en un mercado en constante cambio.

1.3. Objetivos

1.3.1. Objetivo General

Generar un sistema automatizado para la gestión y supervisión eficiente de servidores remotos, implementando las mejores prácticas de ITIL 4 para garantizar el mejoramiento continuo en la jefatura Core Voz y Cico de Telefónica.

1.3.2. Objetivos específicos

- Desarrollar un script para identificar y eliminar automáticamente archivos pesados en servidores remotos, optimizando el almacenamiento y la eficiencia operativa.

- Integrar el script con Zabbix para monitorizar y controlar el proceso de eliminación de archivos mediante la configuración de alarmas, garantizando una supervisión efectiva de la ejecución y los resultados

- Adaptar el marco de referencia ITIL4 con proyección a las buenas prácticas para la optimización y organización de los procesos del área.

1.4. Marco referencial

1.4.1. Perfil empresarial

Colombia Telecomunicaciones S.A. E.S.P., conocida como Telefónica Colombia, es una empresa que opera principalmente en el sector de las Tecnologías de la Información y Comunicación (TIC). Ofrece servicios de telefonía fija y móvil, Internet de banda ancha, fibra óptica al hogar, televisión de pago y servicios digitales.

Telefónica tiene presencia en Colombia desde 2004, comenzando sus actividades en el mercado de telefonía móvil tras la adquisición de la operación celular de Bellsouth en el país. En 2006, Telefónica expandió su presencia al mercado de telefonía fija al adquirir una participación de control del 52% en Colombia Telecomunicaciones S.A. E.S.P.Actualmente, Telefónica está presente en 69 ciudades y municipios del país con fibra óptica, en 240 con banda ancha fija, y en más de 1,120 con telefonía móvil 4G LTE (*Telefónica Colombia | Innovación En Telecomunicaciones*, n.d.).

1.4.2. Misión

Ofrecer conexiones que unan a las personas, en lugar de aislarlas; conexiones que inviten a las personas a ser ellas mismas, a expresarse, a compartir (*Misión - Telefónica*, n.d.).

1.4.3. Visión

Ser parte activa de una sociedad más justa en la que las personas puedan desarrollar todo su potencial, utilizando la fuerza transformadora de lo digital.

1.4.4. Core Voz y Cico

La jefatura de Core Voz y Cico se encarga de apoyar la operación de las redes de telecomunicaciones Core y las plataformas de la red móvil. Su responsabilidad incluye la administración, monitoreo y gestión de todas las funciones relacionadas con el tráfico de voz. El equipo de ingenieros atiende los casos de fallos de llamadas y trabaja en el mejoramiento continuo mediante el desarrollo de nuevas herramientas y aplicaciones que faciliten la ejecución automática de planes y actividades en la red. Todo esto se realiza basándose en los estándares internacionales de calidad y en el análisis histórico del comportamiento de la red, con el objetivo de mejorar continuamente y asegurar la calidad y disponibilidad del servicio.

2. MARCO TEÓRICO

2.1. Zabbix

Zabbix es una herramienta de software de supervisión y control diseñada para monitorizar una amplia variedad de elementos críticos en redes, servidores, máquinas virtuales, servicios, aplicaciones, bases de datos y más. Su sistema se basa en alarmas y notificaciones configurables, lo que permite establecer respuestas rápidas y eficaces ante cualquier evento o anomalía, facilitando una gestión proactiva de la infraestructura.(*Zabbix Features Overview*, n.d.)

Este software de monitorización es de código abierto y se accede a través de una interfaz web accesible desde cualquier ubicación sin costo, lo que garantiza una evaluación constante del estado de la red y sus componentes. Zabbix es ampliamente adoptado por muchas organizaciones debido a su robustez, escalabilidad y a las funciones avanzadas que ofrece.(*Zabbix Features Overview*, n.d.)

Es utilizado por las principales empresas de diversos sectores, incluyendo telecomunicaciones, finanzas, educación, comercio minorista y salud, gracias a su capacidad para proporcionar información detallada sobre el rendimiento y la disponibilidad de sistemas críticos. Zabbix se destaca por su flexibilidad y por permitir una adecuada planificación y respuesta ante problemas, lo que lo convierte en una herramienta indispensable para asegurar la estabilidad y eficiencia de las infraestructuras tecnológicas empresariales.(*Monitoreo de TI Empresarial Con Zabbix*, n.d.)

Zabbix tiene los siguientes procesos importantes:

- Servidor: el servidor zabbix es el proceso central del software del zabbix Este servidor recibe datos de los Agentes Zabbix, almacena la información recopilada en una base de datos, y realiza acciones de procesamiento, análisis y notificación basadas en las reglas configuradas por el usuario (1 Servidor, n.d.).

- Agente: El agente Zabbix se implementa en un objetivo de monitoreo r activamente recursos locales y aplicaciones (estadísticas de discos duros, memoria, procesador, etc.) (*2 Agente*, n.d.).

  El agente reúne información operativa localmente y reporta datos al Servidor Zabbix para su posterior procesamiento. En caso de fallas, el servidor Zabbix puede alertar activamente a los administradores de la máquina en particular que informó el fracaso.

  El agente de Zabbix tiene 2 modos de comprobación, el activo y el pasivo:

Modo pasivo: El Agente Zabbix espera a que el Servidor Zabbix se comunique con él para solicitar y recopilar datos de monitoreo. Siempre a la espera de instrucciones del Servidor Zabbix.

Modo Activo: El Agente Zabbix inicia la comunicación con el Servidor Zabbix de manera proactiva. En este modo, el Agente Zabbix envía activamente datos de monitoreo al Servidor Zabbix en intervalos programados, sin esperar una solicitud explícita del servidor.

Estos componentes de Zabbix usan protocolo TCP, el agente de Zabbix usa el puerto 10050 y el servidor usa el puerto 10051.

Zabbix es utilizado en la empresa para la supervisión y control de elementos críticos en redes y servidores. Trabaja en colaboración con la empresa Imagunet, que brinda asesorías en los procesos automáticos para ayudar a escalar el servicio. Esta herramienta es fundamental en Colombia Telecomunicaciones debido a su capacidad para proporcionar información detallada sobre el rendimiento y la disponibilidad de sistemas críticos, facilitando una gestión proactiva de la infraestructura. Telefónica está iniciando un proceso de migración a Zabbix, por lo que se decidió realizar el desarrollo de los proyectos utilizando este software.

2.2. Virtual Mobile Telephony Application Service (VMTAS)

Un Virtual Mobile Telephony Application Server (VMTAS) es un servidor de aplicación diseñado para proporcionar y gestionar distintos servicios de comunicaciones móviles. el VMTAS se encarga de administrar servicios de mensajería voz, voz sobre ip (VoIP), mensajería multimedia (MMS) y otros servicios de valor.

Actúa como una plataforma de despliegue de aplicaciones móviles, por lo que se pueden implementar y ejecutar aplicaciones dentro del VMTAS para ofrecer nuevas funcionalidades y un mejor servicio.

2.3. SCRUM

Scrum es un marco de trabajo ágil que promueve el desarrollo de proyectos mediante un enfoque estructurado basado en iteraciones cortas y regulares de trabajo, conocidas como "sprints". En este marco, los equipos autónomos y multifuncionales colaboran en ciclos rápidos de desarrollo (SBOKTM Guide, 2013).

Scrum se basa en un modelo de proceso empírico, fomentando el respeto por las personas y la auto organización de los equipos para abordar situaciones repentinas y resolver

problemas complejos. La metodología se centra en la inspección y adaptación continua para generar más valor.

Los principales roles en Scrum son:

Product Owner: Persona responsable de maximizar el valor del proyecto, actuando como la voz del cliente. El Product Owner articula los requisitos del cliente y mantiene la justificación del proyecto.

Scrum Máster: Líder servicial que facilita la creación de entregables. El Scrum Máster se caracteriza por su papel de facilitador, eliminando obstáculos y fomentando un ambiente productivo para el equipo.

Equipo Scrum: Grupo que contribuye a la generación de valor, trabajando en la creación de los entregables del proyecto.

Por lo general, un equipo Scrum consta de entre 1 y 9 personas, cabe resaltar que todos los miembros del equipo son parte integral de la solución.

Scrum utiliza un "Scrum Board" (tablero Scrum) para visualizar el progreso de las tareas, lo que permite una gestión efectiva del trabajo durante el desarrollo del proyecto.

2.4. ITIL4

ITIL (Information Technology Infrastructure Library) es un marco de referencia que promueve las mejores prácticas para la gestión de servicios de TI, proporcionando un enfoque coherente que satisface las necesidades del negocio y de los usuarios al centrarse en la generación de valor (Prieto Fernando, n.d.).

Está constituido por cinco componentes que generar valor al producto. Sus cinco componentes son:

- Principios guía
- Gobierno
- Cadena de valor del servicio
- Practicas
- Mejora continua

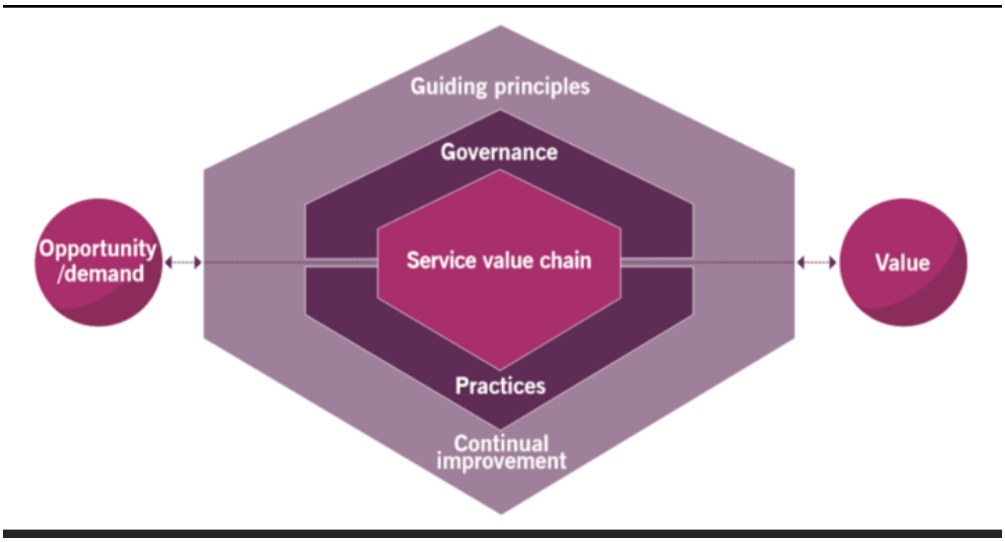


Figura 1. Sistema de valor del servicio de ITIL4 *(*Componentes de ITIL 4*, n.d.)*

Como se representa en la figura 1 el valor es la salida que se obtiene cuando las oportunidades y demandas de los clientes son debidamente gestionadas a través del sistema de valor del servicio (SVS), a continuación, se explicará cada una de estas partes:

Oportunidad y demanda: Las oportunidades son posibilidades de añadir valor a las partes interesadas (stakeholders). La de manda es la necesidad de productos y servicios que tienen los clientes.

Principios guía: El propósito fundamental de ITIL 4 es proporcionar un marco de trabajo práctico y adaptable que ayude a las organizaciones a mejorar la prestación de servicios de TI. Sus principios básicos son los siguientes:

- **Enfoque en el Valor**: Reconoce que el valor se genera tanto internamente como externamente. Es crucial mantener un enfoque constante en el valor durante todas las actividades realizadas.

- **Colaborar y promover Visibilidad:** La colaboración implica comunicación efectiva y toma de decisiones basadas en datos visibles y coherentes, no necesariamente en consenso.
- **Pensar y trabajar de forma Holística:** Reconoce la complejidad de los sistemas y promueve el enfoque integral en la gestión de servicios de TI.

- **Mantener lo simple y lo práctico:** La simplicidad maximiza la satisfacción y la adopción. Es más fácil entender y adoptar soluciones simples.

- **Optimizar y Automatizar:** Se debe simplificar antes de automatizar. Definir métricas claras es esencial para medir y mejorar continuamente.

Gobierno: estructuras organizativas y los procesos que aseguran que las tecnologías de la información apoyan y permiten lograr las estrategias y los objetivos de la organización.

Cadena de valor del servicio: La cadena de valor del servicio es un modelo operativo que define un conjunto de actividades interconectadas que una organización lleva a cabo para entregar productos o servicios con valor a sus clientes y facilitar la obtención de valor.

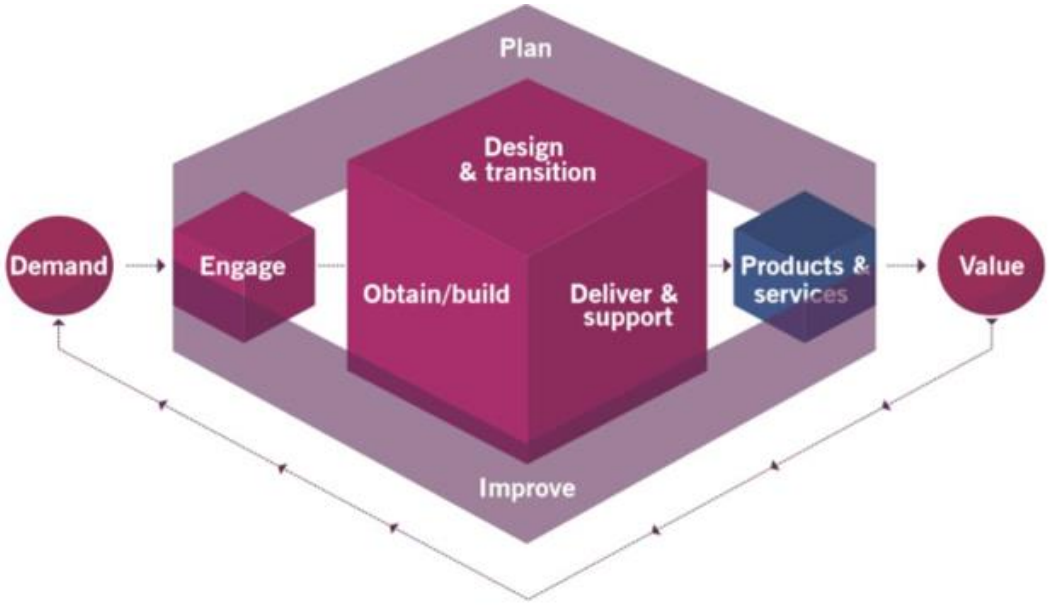


Figura 2. Actividades de la cadena de valor del servicio *(*Componentes de ITIL 4*, n.d.)*

La figura 2 representa las 6 actividades consideradas para generar un posible flujo de valor especificado de cada organización, a continuación, se explican las actividades propuestas por ITIL 4 para la cadena de valor:

- Mejora: Asegura una mejora continua a nivel de productos y servicios.
- Diseño y transición: productos y servicios que cumplan con las expectativas.
- Obtener y construir: Garantizar que el servicio esté disponible y cumpla con las especificaciones acordadas.
- Entrega y soporte: Asegurar que los servicios son entregados y reciben el apoyo de acuerdo con las especificaciones y las expectativas de las partes acordadas
- Plan: Asegurar que existe una comprensión compartida de la visión, del estado actual y de la dirección de mejora.
- Participación: comprensión de las necesidades de los interesados, transparencia y compromiso continuo y buenas relaciones con los interesados

Practicas: conjunto de recursos organizativos diseñados para llevar a cabo los trabajos necesarios y lograr los objetivos propuestos.

Prácticas generales de gestión	Prácticas de gestión de servicios	Prácticas de gestión técnica
• Gestión de la arquitectura • Mejora continua • Gestión de la seguridad de la información. • Conocimiento administrativo • Medición y reporte • Gestión del cambio organizacional • Gestión del portafolios • Gestión de proyectos • Gestión de relaciones • Gestión de riesgos • Servicio de gestión financiera • Gestión de la estrategia • Gestión de proveedores • Mano de obra y gestión del talento	• Gestión de la disponibilidad • Análisis de negocios • Gestión de capacidad y rendimiento • Control de cambios • Gestión de incidentes • Gestión de activos de TI • Seguimiento y gestión de eventos • Manejo de problemas • Gestión de lanzamiento • Gestión del catálogo de servicios • Gestión de la configuración del servicio • Gestión de la continuidad del servicio • Diseño del servicio • Mesa de servicio • Gestión del nivel de servicio • Gestión de la solicitud del servicio • Servicio de validación y pruebas	• Gestión de la implementación • Gestión de infraestructuras y plataformas • Desarrollo y gestión de software

Figura 3. Practicas ITIL 4 (Capacitación ITIL4, 2023)

En la figura 3, se observa la estructura de las prácticas de ITIL 4, las cuales se dividen en tres categorías p: prácticas generales de gestión, prácticas de gestión de servicios y prácticas de gestión técnica. En total, ITIL 4 incluye 34 prácticas que abarcan diversas áreas de gestión de servicios de TI.

Las prácticas están diseñadas para proporcionar orientación y estructura a las organizaciones que buscan implementar procesos efectivos de gestión de servicios de TI basados en las mejores prácticas establecidas por ITIL.

2.5. KANBAN

Kanban es un sistema visual utilizado para la gestión efectiva de procesos y el seguimiento del trabajo a medida que avanza a través de un flujo de trabajo. Se trata de una metodología que tiene un enfoque organizacional, diseñada para optimizar la eficiencia y la productividad en diversas áreas.

La palabra "Kanban" es de origen japonés y se traduce como "tarjeta visual" o "tarjeta que se puede ver". Este sistema fue desarrollado por Toyota en la década de 1950 como parte de su sistema de producción Just-in-Time (*December 2020-Https://Prokanban.Org/the-Kanban-Guide*, n.d.).

Al igual que Scrum, Kanban utiliza tableros visuales, conocidos como "boards", para visualizar el flujo de trabajo. Se centra en tres pilares fundamentales: limitar el trabajo en curso solo a lo necesario, mantener un flujo eliminando los problemas que interrumpen la producción, y reducir el tiempo necesario para producir cada componente del sistema (*December 2020-Https://Prokanban.Org/the-Kanban-Guide*, n.d.).

En los tableros Kanban, la prioridad varía según la tipología de las tareas, sus niveles de prioridad y las políticas establecidas explícitamente.

Los tres principios más importantes de Kanban son:

1. **Empezar desde donde esta:** Kanban promueve cambios progresivos y evolutivos, comenzando desde la situación actual y avanzando de manera incremental.
2. **Cambio incremental y evolutivo:** No es necesario implementar cambios radicales; Kanban favorece la mejora continua a través de pequeños ajustes.

3. **Respeto por los roles actuales:** Kanban reconoce y respeta los roles y responsabilidades existentes en el equipo, fomentando la colaboración y la autonomía en el proceso de mejora.

Este sistema es altamente beneficioso para el desarrollo de software y la gestión de proyectos, permitiendo a los equipos visualizar su trabajo, optimizar la eficiencia y mejorar continuamente sus procesos.

Existen diversas herramientas disponibles para crear estos tableros Kanban. Incluso plataformas como Microsoft Teams y Azure tienen funcionalidades implementadas para gestionar tableros Kanban de manera efectiva.

3. DESARROLLO DEL SCRIPT

3.1. Análisis DOFA

Se realizó un análisis DOFA (Fortalezas, Oportunidades, Debilidades, Amenazas), presentado en la tabla 1, que resume los aspectos clave identificados en el proceso. Todo esto con el objetivo de establecer las oportunidades para desarrollar un plan de mejoramiento continúo.

FORTALEZAS (+)	DEBILIDADES (-)
• La jefatura de Core Voz y CICO cuenta con personal capacitado para atender las incidencias diarias. • Se pone un gran enfoque en el uso de metodologías ágiles, lo que mejora la adaptabilidad y la eficiencia • Se promueve a los empleados a realizar cursos para su mejoramiento personal y laboral. • Las fallas se discuten con todo el equipo, lo que fomenta la colaboración y la aportación de soluciones	• Hay falencias en la información recolectada de cada orden de trabajo. • No hay documentación clara de los incidentes que se presentan en cada orden de trabajo • Los procesos carecen de automatización, lo que reduce la eficiencia • La jefatura carece de permisos de firewall, lo que limita la realización de algunos procesos. • Existe documentación que no ha sido actualizada, lo que puede conducir a errores y malentendidos
FACTORES EXTERNOS	
OPORTUNIDADES (+)	**AMENAZAS (-)**
• La migración de plataformas a Zabbix permite un mejor control y monitoreo de los sistemas. • Hay disponibilidad de recursos para realizar automatizaciones y desarrollos de software, mejorando la eficiencia operativa. • La incorporación de nuevos servicios permite involucrar más al cliente y satisfacer mejor sus necesidades.	• Hay problemas con los servidores y las plataformas que se manejan, lo que puede afectar la continuidad del servicio • La falta de procedimientos claros en las órdenes de trabajo puede llevar a errores en las centrales, afectando a muchos usuarios. • La mala documentación de las órdenes de trabajo puede provocar que se queden sin clasificar adecuadamente • El mercado es muy competitivo, lo que requiere una mejora continua para mantenerse relevante

Tabla 1. Análisis DOFA de la jefatura. Fuente: elaboración propia

3.2. Recopilación de información vMTAS

Durante la pasantía, se recopiló información sobre el uso del disco en los servidores vMTAS de Cali y Barranquilla. Como se muestra en la Figura 4, cada Vmtas tiene asignado un número determinado de hosts (PL) con sus respectivas direcciones IP.

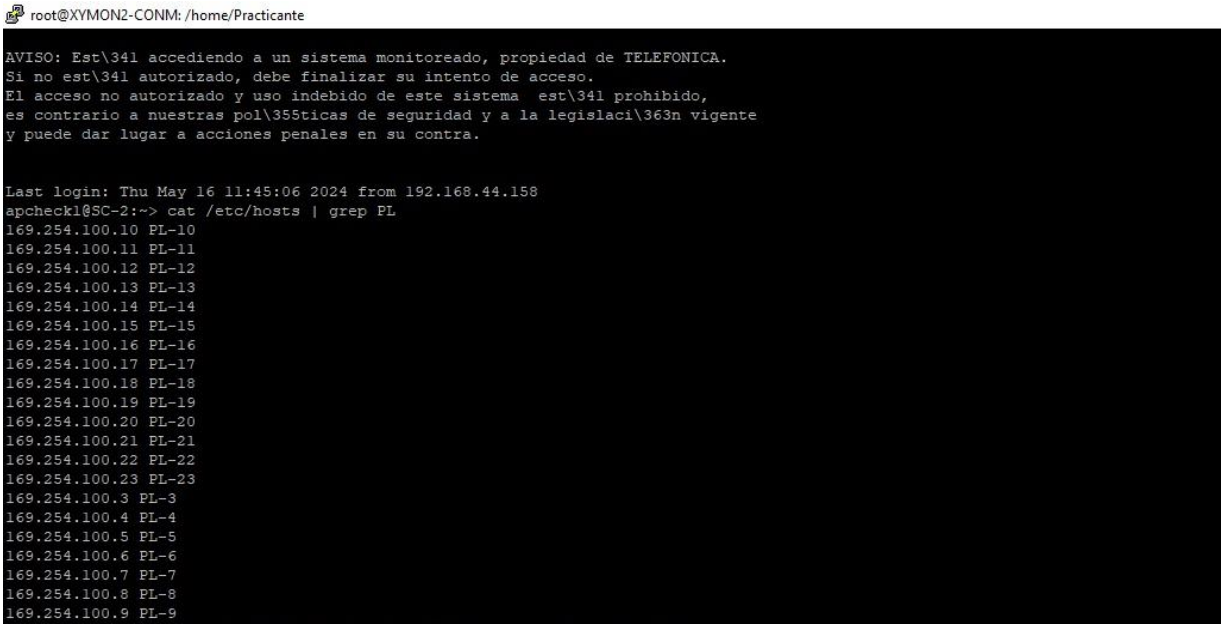


Figura 4. Hosts en el vMTAS

En la siguiente imagen se pueden observar los archivos presentes en el sistema de archivos de cada host, junto con el porcentaje de almacenamiento utilizado.

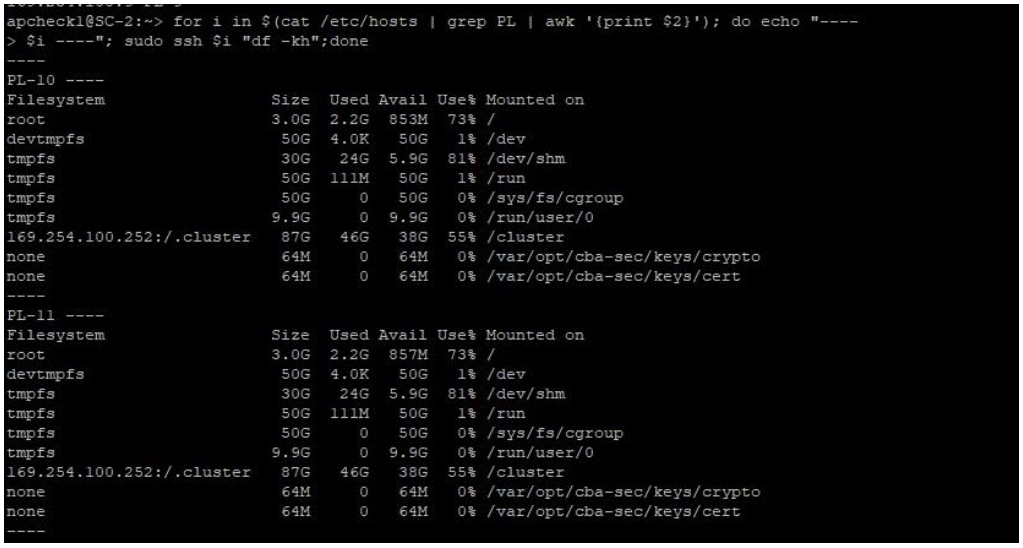


Figura 5. Filesystem de cada PL.

Como se observa en la figura 5, el porcentaje de uso de ciertos archivos es demasiado alto, lo cual podría afectar negativamente el funcionamiento de los sistemas. Se propone implementar un mecanismo automático mediante un 'trigger' que limpie automáticamente los archivos cuando su uso supere el 80%.

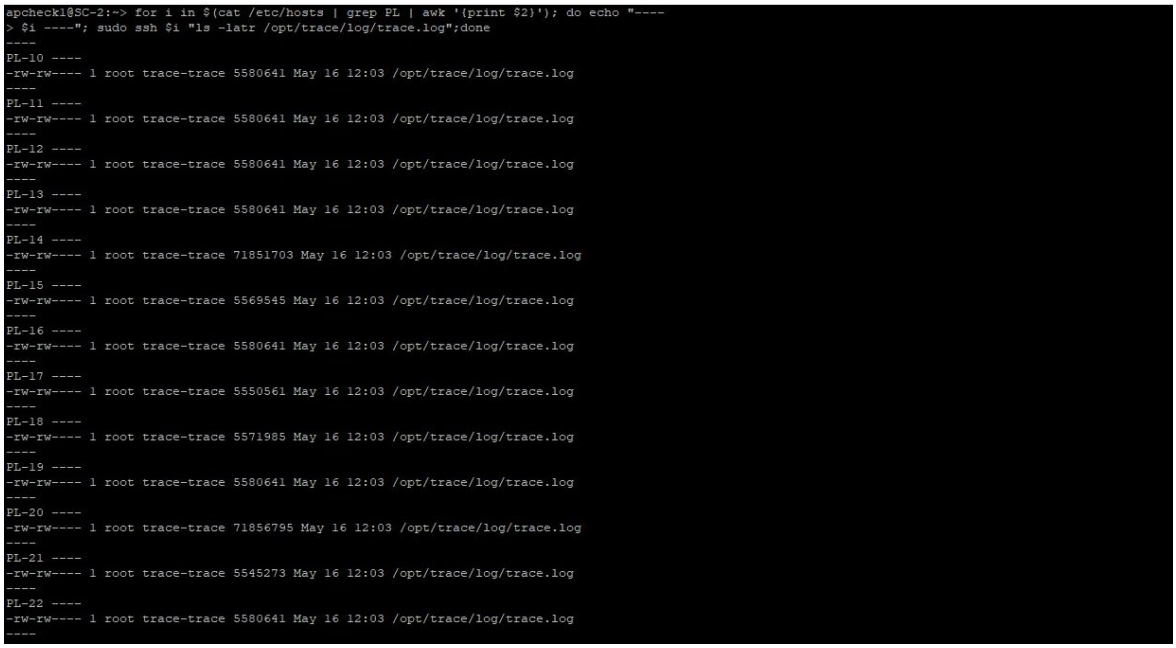


Figura 6. Almacenamiento del archivo trace.log.

Observando la Figura 6, se puede notar que cada host tiene un archivo trace.log con un tamaño de almacenamiento significativo que va aumentando con el tiempo. Este archivo de almacenamiento se encuentra ubicado en "/opt/trace/log/trace.log". El objetivo es reducir el tamaño de este archivo a 0 bytes de forma automática para evitar interrupciones en las operaciones.

El análisis de estos datos se realizó durante un período de 5 meses, involucrando a los miembros del equipo de Core Voz y Cico, quienes supervisaron el script y el desarrollo en Zabbix, así como a expertos externos de la empresa Imagunet, que brindaron supervisión y apoyo en la implementación en Zabbix. El rol del practicante consistió en desarrollar el script y participar en la creación de alarmas en Zabbix para automatizar la eliminación de archivos pesado.

3.3. Desarrollo del Script

Para llevar a cabo lo mencionado anteriormente, se ha creado un script que permite automatizar el proceso y vincularlo con Zabbix para configurar los triggers y el proceso de autoremediación

3.3.1. Librerías

La librería json facilita el manejo de datos en formato JSON. En este caso, será utilizada para almacenar la información del script y luego enviarla a Zabbix en formato JSON.

Paramiko posibilita la comunicación SSH con las Vmtas y la ejecución de comandos SSH.

La librería os se emplea para ejecutar comandos del sistema operativo a través de os.system, lo que permite interactuar con el sistema desde el script (Ver ANEXO A).

3.3.2. Función para establecer la conexión SSH

La función conexión permite establecer una conexión SSH. Comienza creando un cliente SSH (SSHClient) y configura la política para manejar automáticamente las claves de host desconocidas. Luego intenta establecer la conexión SSH utilizando los parámetros de hostname, puerto y usuario especificados. Si la conexión tiene éxito, muestra un mensaje indicando que se ha conectado al servidor correctamente, junto con el nombre del servidor y su dirección IP. La función devuelve el objeto de conexión SSH (ssh) y el hostname. En caso de que ocurra un error durante la conexión, como una autenticación fallida (AuthenticationException) o problemas con la conexión SSH (SSHException), la función captura la excepción correspondiente, imprime un mensaje de error y devuelve None para indicar que la conexión no se pudo establecer correctamente. (Ver ANEXO B)

3.3.3. Función para ejecutar los comandos

La función **`ejecutar_comando'** se encarga de ejecutar los comandos de python. Primero, utiliza el método `exec_command()` del objeto `ssh` para ejecutar el comando `comando`. Este método devuelve tres canales de comunicación: `stdin`, `stdout` (salida estándar) y `stderr` (error estándar). Luego, la función lee la salida del comando desde `stdout` y los errores desde `stderr`, decodificándolos en cadenas de caracteres UTF-8. Estos resultados (salida y errores) son devueltos para que puedan ser utilizados por el programa (VER ANEXO C)

3.3.4. función para pasar de Gigas (G), Megas (M) y Kilos (K) a Bytes

La función toma un valor de tamaño de archivo en notación abreviada, como kilobytes (K), megabytes (M) o gigabytes (G), y lo convierte a su equivalente en bytes. Primero, define un diccionario que asigna a cada abreviatura de unidad su correspondiente multiplicador para convertir a bytes. Luego, verifica si la última letra del valor es una de las abreviaturas reconocidas. Si lo es, extrae el número y la unidad de medida, y multiplica el número por el multiplicador correspondiente. Finalmente, devuelve el tamaño convertido a bytes como un entero. Si la unidad de medida no está en la lista de abreviaturas reconocidas, simplemente devuelve el valor original, asumiendo que ya está en bytes, esto con el fin de facilitar la lectura de los datos en la autoremediacion de Zabbix. (VER ANEXO D)

3.3.5. Función para establecer la comunicación con Zabbix

La función está diseñada para enviar datos al servidor Zabbix utilizando el comando `**zabbix_sender**` desde Python. Primero, se definen algunas variables, como la dirección IP del servidor Zabbix y la dirección IP de origen. Luego, se asignan los valores de `host_name` y `key_name` a las variables `host` y `key`, respectivamente. A continuación, se construye un comando de línea de comandos utilizando estas variables junto con el argumento `-o` que representa el valor de la clave en formato JSON. Finalmente, se ejecuta el comando, lo que envía los datos al servidor Zabbix. (VER ANEXO E)

3.3.6. Funcion para sacar el tamaño del trace.log

Esta función se encarga de obtener el tamaño el archivo trace.log a través de una conexión ssh, utiliza el comanto stat el tamaño del archivo en bytes, luego se ejecuta el comando con la función ejecutar_comando que devuelve la salida del tamaño del archivo. (VER ANEXO F)

3.3.7. función para guardar los resultados en un json

Esta función toma los resultados proporcionados, los escribe en formato JSON en el archivo "resultados.json". Esto facilita el análisis de los resultados. (VER ANEXO G)

3.3.8. Main para establecer la conexión con el vmtas

La función **`main (hostname, host, key)` ** es la encargada de coordinar la ejecución de diferentes tareas para los servidores. En primer lugar, establece el puerto SSH y el nombre de usuario para la conexión. Luego, intenta establecer una conexión SSH con el servidor remoto utilizando la función **`conexion()`.** Si la conexión es exitosa, continúa ejecutando una serie de acciones a través de un if.

Una vez conectado al servidor, el script ejecuta un comando para obtener una lista de servidores que contienen "PL" en el archivo **`/etc/hosts`**. Luego, para cada servidor en la lista obtenida, ejecuta un comando remoto para obtener información sobre los filesystems del servidor, procesa esta información y la almacena en una estructura de datos.

Después de recopilar y procesar la información de los filesystems para todos los servidores, el script crea un objeto JSON que contiene toda esta información y lo envía a un servidor Zabbix utilizando la función `zabbix_sender()`. Además, guarda los resultados en un archivo JSON localmente llamado ‘resultados.json’. Finalmente, cierra la conexión SSH.

Si no se puede establecer la conexión SSH, imprime un mensaje indicando que no se pudo establecer la conexión. (VER ANEXO H)

3.3.9. Script vmtas completo

En el Anexo I se puede vizualzar todo el script propuesto para el desarrollo de automatización.

3.3.10. Archivo json

Los datos del sistema de archivos y el registro de seguimiento se almacenan en un archivo JSON llamado 'resultados.json'. (VER ANEXO J)

Debido a que son bastantes datos solo se adjuntaron algunos en el documento. En el anexo J se puede ver en una lista las PL y el tamaño que tiene esta en el archivo trace.log, también se ve el filesystem con todos sus datos y los valores numéricos en bytes.

4. INTEGRACIÓN DEL SCRIPT CON ZABBIX.

4.1. Integración del vMTAS en Zabbix

Primero, es fundamental vincular el script a Zabbix. Para lograrlo, se debe crear un template y asignarlo a un grupo de hosts.

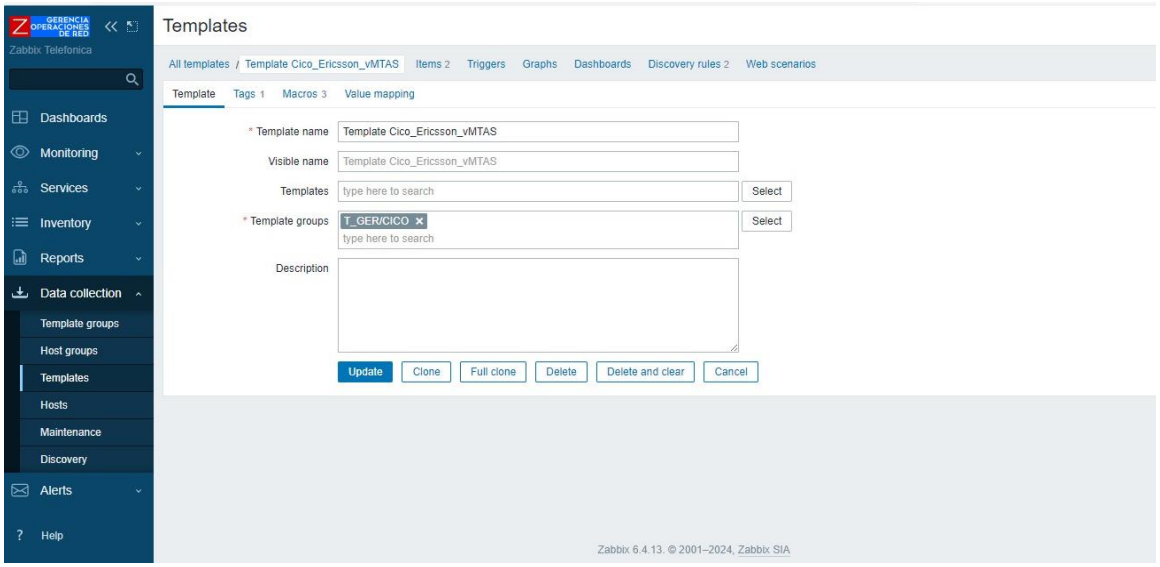


Figura 7. Creación del template del Vmtas

En la figura 7 se puede ver el template denominado "Template Cico_Ericcsson_vMtas", asignado al grupo de templates de la jefatura de voz, "T_GERCICO". Posteriormente, los datos se envían mediante el script mencionado anteriormente.

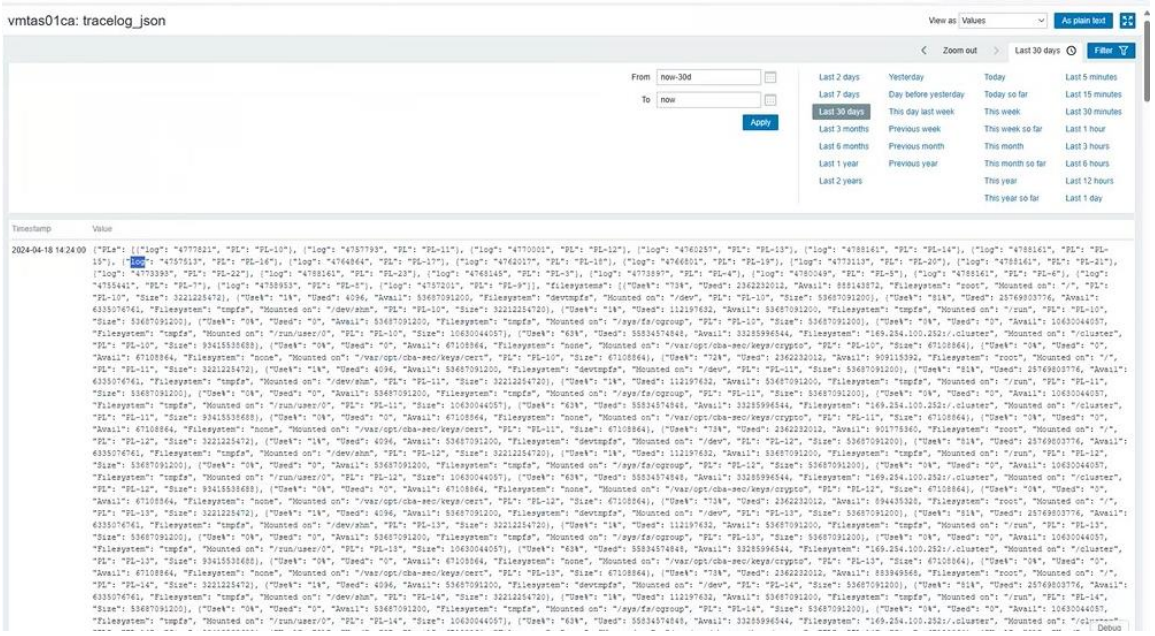


Figura 8. Datos enviados a zabbix del script.

Los datos presentados en la figura 8 son extraídos del archivo JSON. Luego, se establece un prototipo de elemento (item prototype) para filtrar dichos datos.

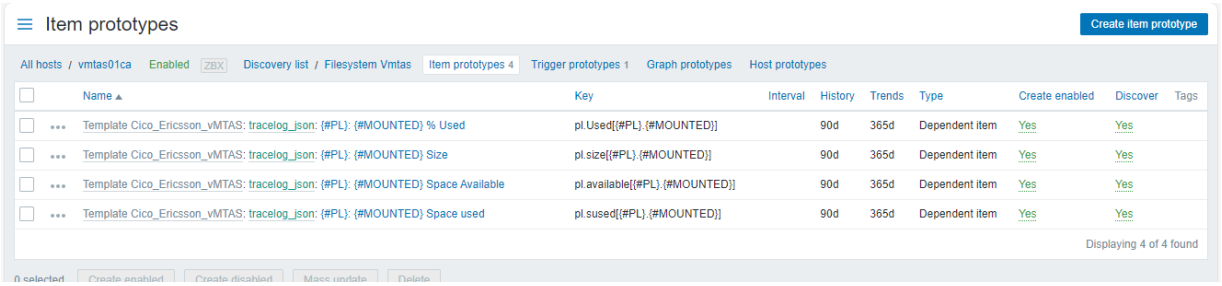


Figura 9. Item para el fileystem

En la figura 9, se observa que cada dato del sistema de archivos está etiquetado con un identificador denominado "MOUNTED", el cual se utiliza para filtrar los datos según su ubicación, como se muestra a continuación:

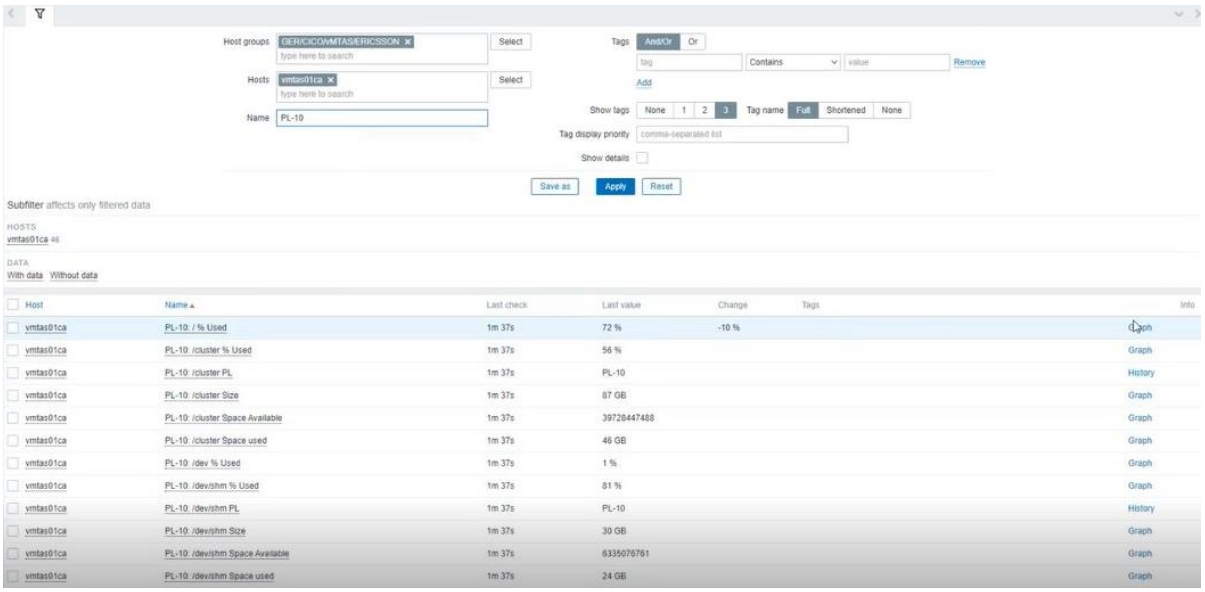


Figura 10. Datos filtrados del filesystem.

Los datos se utilizan para filtrar lo que se desea corregir automáticamente, en este caso, el porcentaje de ocupación del sistema de archivos como se ve en la figura 10. Para el almacenamiento del archivo trace.log, se crea otro item diferente.

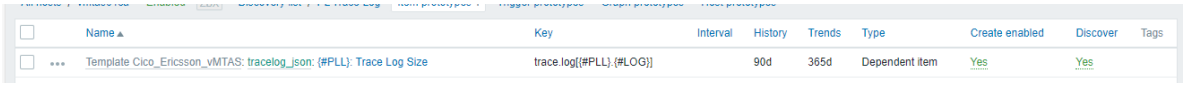


Figura 11. Ítem para el tracelog

En el elemento que se muestra en la figura 11, se lleva a cabo la comparación entre cada host (PL) y el tamaño del archivo (LOG).

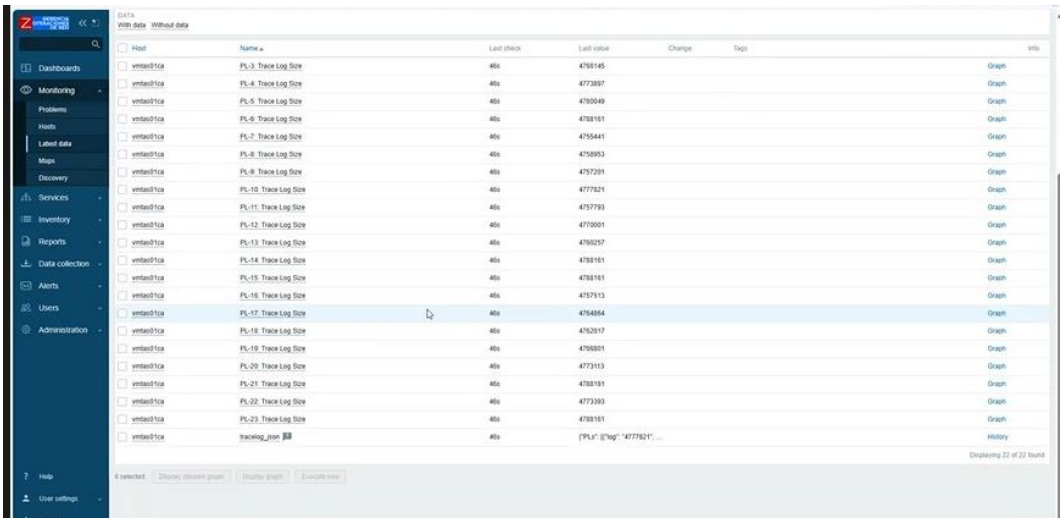


Figura 12. Datos del trace.log

Se procede a crear las macros para que genere las alarmas y el proceso de auto-remediación.

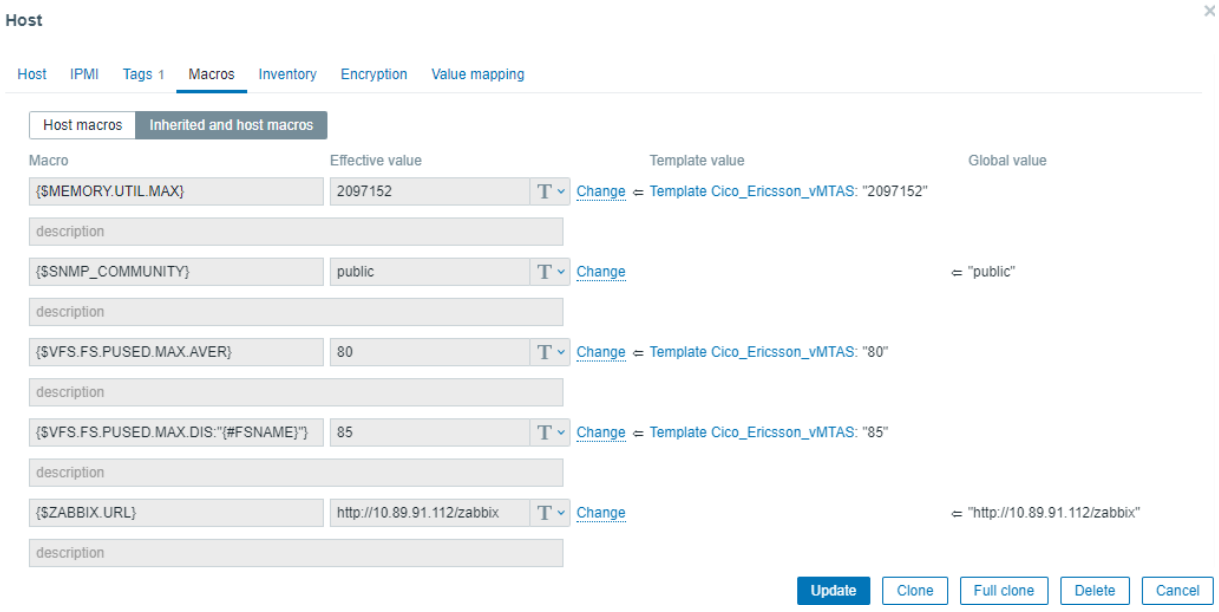


Figura 13. Macros establecidos

Se definen las macros del programa de la siguiente manera: cuando el sistema de archivos alcance el 80% o más de ocupación, se considerará en estado "alto" (high); y cuando alcance el 85%, se considerará en estado "desastre" (disaster) y procederá a limpiar el filesystem además, si el archivo trace.log supera los 200 megabytes, también limpiara el archivo.

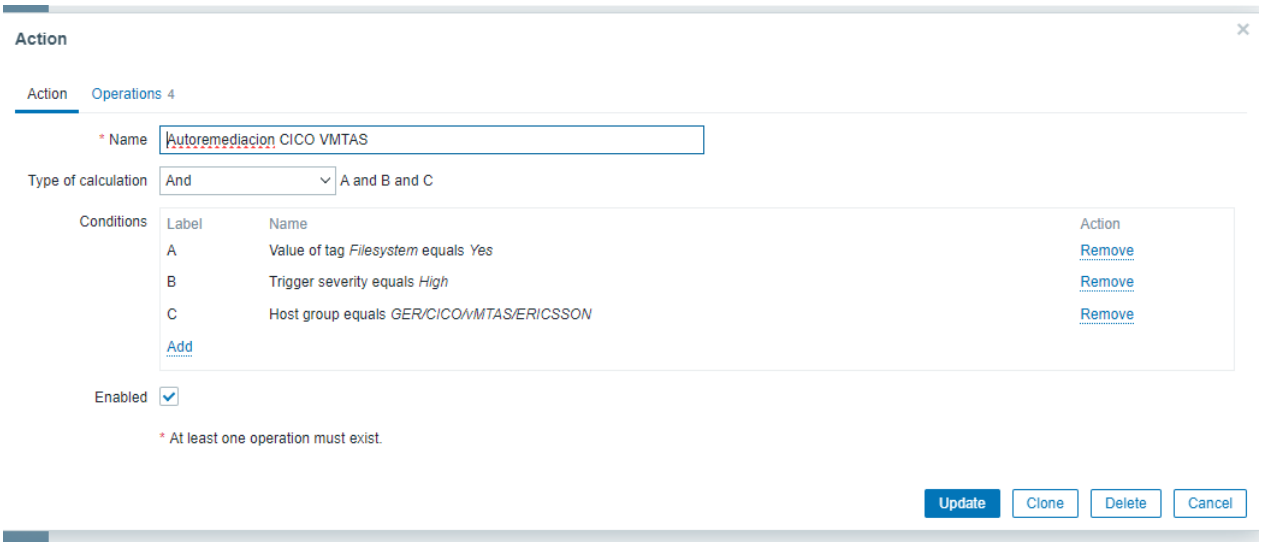


Figura 14. Creación del Trigger

Al crear el disparador (trigger), se establece el nombre comenzando con "Autoremediación", seguido por la jefatura (CICO) y la plataforma (VMTAS). Se elige el tipo de cálculo "AND" para que todas las condiciones se cumplan.

Después, en la sección de operaciones, se añade una duración de 1 minuto para ejecutar la operación de enviar el mensaje a través de Telegram. Además, se incluye el script de autoremedicación para que Zabbix lo ejecute de manera automática, tal como se muestra en la figura 14.

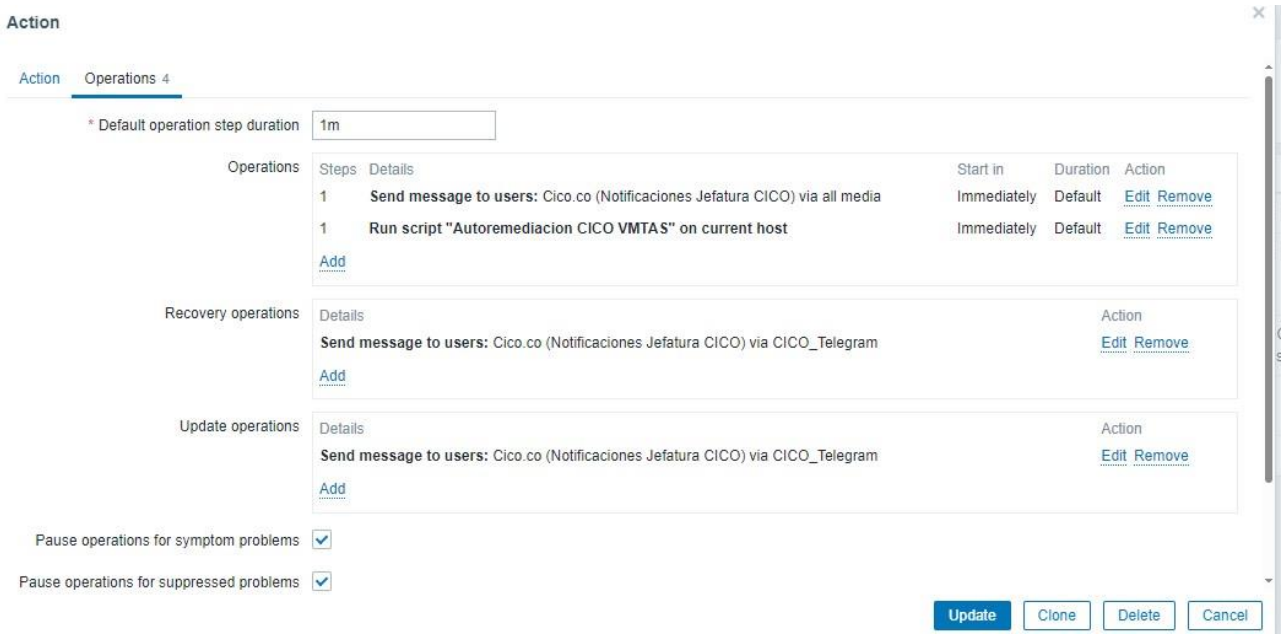


Figura 15. Operaciones del trigger

Después, en la sección de operations se añade una duración de 1 minuto para ejecutar la operación de enviar el mensaje a través de Telegram. Además, se incluye el script de autoremedicación para que Zabbix lo ejecute de manera automática, tal como se muestra en la figura 15.

Figura 16. Alarma en high del filesystem

Como se mencionó anteriormente, cuando un host alcance o supere el 80% de ocupación, se encontrará en estado "alto" (high), tal como se muestra en la figura 16.

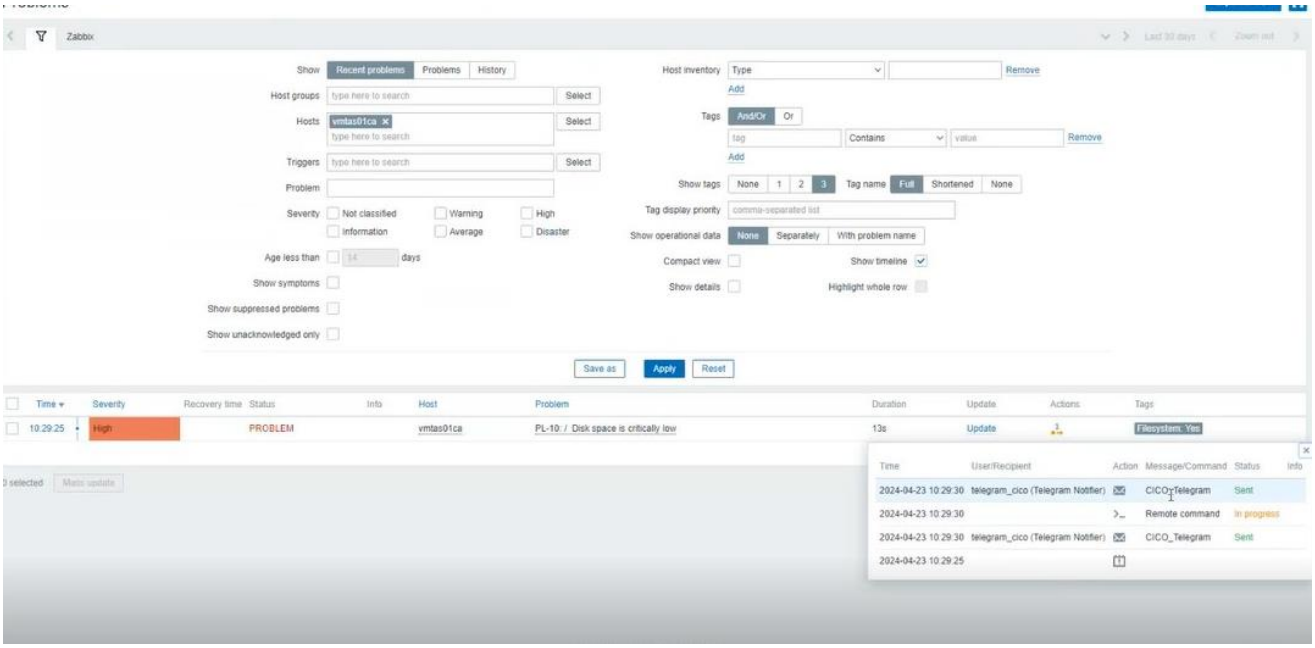


Figura 17. Ejecución de la auto-remediación

Luego se envía una notificación al grupo de Telegram de CICO informando sobre el estado del host. Después, se ejecuta el script de auto-remediación y, posteriormente, se envía otra notificación a Telegram confirmando que la auto-remediación se ha realizado.

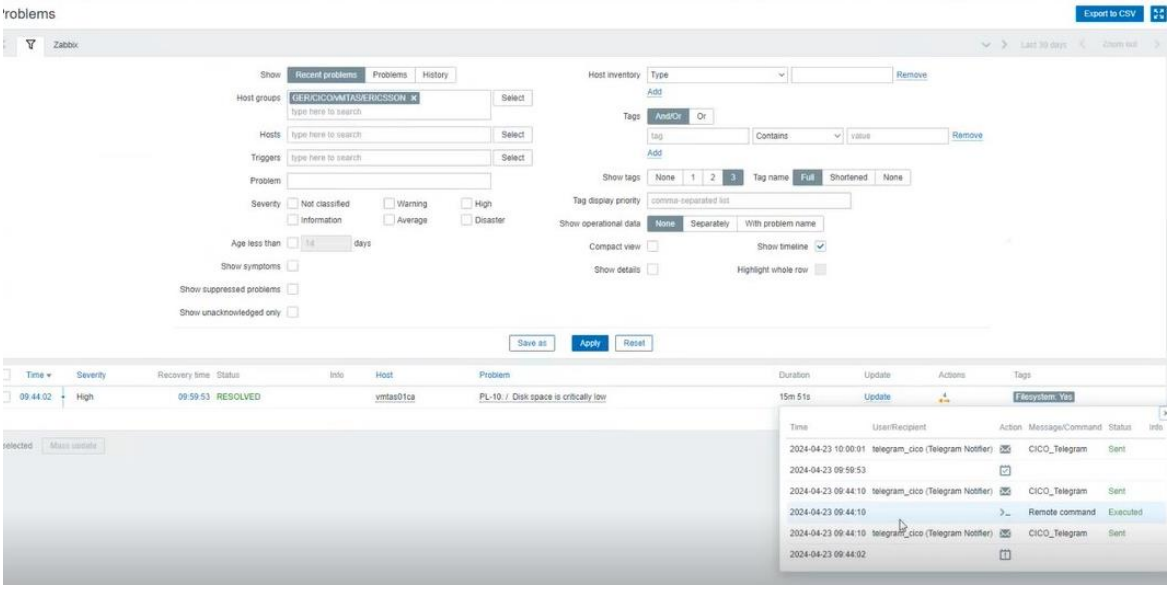
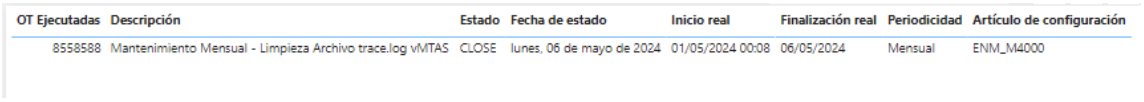


Figura 18. Ejecución de la auto-remediación resuelta

Como se muestra en la figura 18, después de ejecutar el script, el problema se resuelve y la alarma se elimina. Además, se envía un mensaje a Telegram indicando que la auto-remediación se ha realizado.

El proyecto para desarrollar el script de limpieza del archivo trace.log se completó en mayo. Hasta la fecha, se ha ejecutado una vez este mes, logrando satisfactoriamente el mantenimiento de la orden de trabajo. Actualmente, el proceso se está llevando a cabo de forma automática.



OT Ejecutadas	Descripción	Estado	Fecha de estado	Inicio real	Finalización real	Periodicidad	Artículo de configuración
8558588	Mantenimiento Mensual - Limpieza Archivo trace.log vMTAS	CLOSE	lunes, 06 de mayo de 2024	01/05/2024 00:08	06/05/2024	Mensual	ENM_M4000

Figura 19. Orden de trabajo del Vmtas.

En Zabbix, tras una serie de pruebas y verificaciones, se ha confirmado que el proceso de auto-remediación está activo. Esto permite su implementación en otros Vmtas que la empresa desee incorporar.

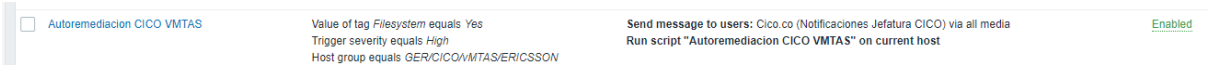


Figura 20. Auto-remediación del Vmtas activa en Zabbix

4.2. Mejoras Implementadas con el Script de Automatización y la Integración con Zabbix

La implementación del script de automatización junto con la integración con Zabbix ha permitido una serie de mejoras significativas en los procesos operativos de la jefatura Core Voz y Cico. A continuación, se detallan los aspectos más relevantes que se han mejorado:

Reducción del Trabajo Manual

- **Antes:** Las tareas de supervisión y gestión de archivos pesados se realizaban manualmente, lo que consumía tiempo y recursos significativos del equipo de Core Voz y Cico.
- **Después:** Con el desarrollo de la automatización para la identificación y eliminación de archivos pesados, se ha reducido considerablemente la necesidad de intervención manual. Esto ha liberado recursos del equipo, permitiéndoles centrarse en tareas que requieren más tiempo y aportan mayor valor agregado.

Optimización del Almacenamiento

- **Antes:** El archivo trace.log en los servidores vMTAS podía crecer hasta superar el umbral crítico del 80% de uso del disco, causando alarmas y problemas operativos.
- **Después:** El script automatiza la reducción del tamaño del archivo trace.log, manteniendo el uso del disco dentro de límites aceptables y evitando interrupciones en el servicio e incidentes.

Mejora en la Continuidad del Servicio

- **Antes:** Las constantes ventanas de mantenimiento para eliminar archivos pesados provocaban interrupciones en el servicio, afectando la continuidad operativa y los procesos de la jefatura, como los trazados de algunas llamadas.
- **Después:** La automatización de la limpieza de archivos ha disminuido la frecuencia y duración de las interrupciones operativas, mejorando la continuidad del servicio. Ahora, no es necesario crear una ventana de mantenimiento, ya que el proceso se realiza automáticamente y se notifica al equipo en el momento de la ejecución.

Monitoreo y Control en Tiempo Real

- **Antes:** La supervisión de los servidores y la detección de problemas dependían de revisiones manuales y no siempre se realizaban en tiempo real. Un miembro del equipo tenía que revisar si se había superado el umbral y programar la ventana de mantenimiento.
- **Después:** La integración con Zabbix permite un monitoreo continuo y en tiempo real de los servidores. Las alarmas configuradas garantizan una supervisión efectiva y una respuesta rápida ante cualquier anomalía detectada.

Estas mejoras no solo han optimizado los procesos operativos, sino que también han contribuido a incrementar la satisfacción del equipo de trabajo y la calidad del servicio ofrecido a los clientes. La automatización y el uso de herramientas avanzadas como Zabbix han demostrado ser estrategias clave para alcanzar una mayor eficiencia y resiliencia operativa en la jefatura Core Voz y Cico de Telefónica.

4.3. Implementación de las metodologías agiles

Según lo desarrollado anteriormente se plantea adicionarles metodologías agiles con el fin de que haya un desarrollo y mejoramiento continuo en los procesos.

El uso de metodologías agiles se centra en la mejora continua y en la flexibilidad para entregar valor mediante la implementación de fases e incorporación de prácticas de mejora continua. Esto permite al equipo de trabajo alcanzar un mejor rendimiento y satisfacer eficazmente las necesidades.

Mediante el uso de metodologías ágiles, el objetivo del grupo de Core Voz y CICO fue crear un entorno ágil para la gestión y planificación de proyectos. Esto se logró mediante la división de tareas a través de la plataforma Azure DevOps, donde las tareas se organizan en un tablero (board) y se asignan por prioridad dentro de un período determinado (Sprint).

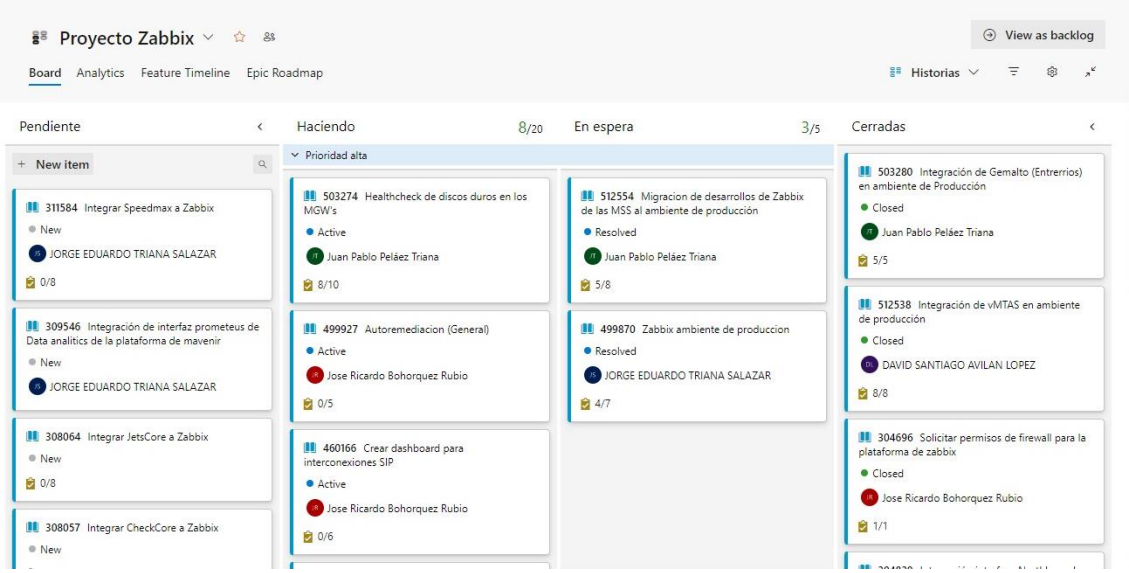


Figura 21. Board para la organización de proyectos y actividades.

Estos tableros están asignados por proyectos y, como se muestra en la figura 21, se utilizan para organizar las tareas en pendientes, en proceso, en espera (casi completas o en revisión para cerrarlas), y completadas. Cada tarea es asignada a un miembro del equipo. La revisión de estos tableros se realiza dos veces a la semana en reuniones de máximo media hora, con el fin de resolver inquietudes, agregar proyectos y tareas, y asegurar un progreso continuo.

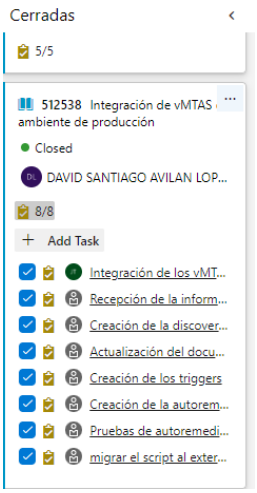


Figura 22. Tareas asignadas

Las tareas se cierran solo después de cumplir con los requisitos específicos; de lo contrario, no podrán cerrarse.

Además, cuatro días a la semana, antes de las actividades matutinas, se realiza una revisión diaria (Daily Review) de máximo media hora. El objetivo es exponer las tareas pendientes, fallos de hardware, procesos y documentación que surgen diariamente. Esta práctica fomenta la comunicación dentro del equipo, permite identificar problemas, actualizar el estado de los proyectos y, fundamentalmente, promueve la mejora continua.

Figura 23. Actividades del Daily Review

Por último, todos los lunes se lleva a cabo una reunión de planificación de la jefatura, en la que se presentan informes semanales a través de dashboards. Estos informes incluyen mantenimientos preventivos, alarmas, fallos de hardware, casos de abonados, entre otros. En la figura 24 se muestran algunos de estos dashboards.

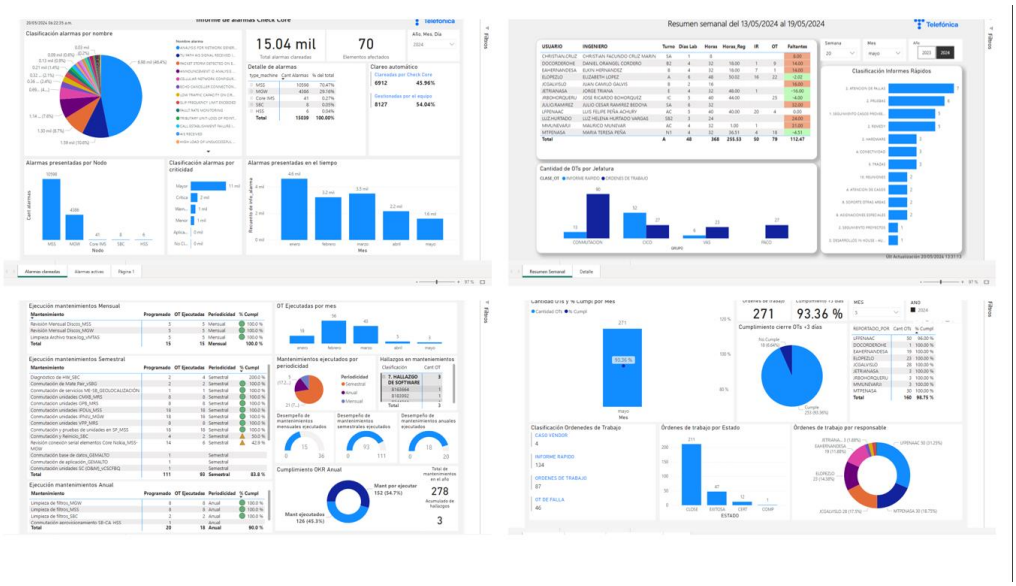


Figura 24. Dashboards de la reunión de planning

Dado que se manejan numerosos archivos, es fundamental mantener un orden eficiente como parte de las metodologías ágiles. Para facilitar el acceso a estos archivos, la jefatura maneja un micrositio utilizando estrategias tecnológicas de la empresa y diversas herramientas.

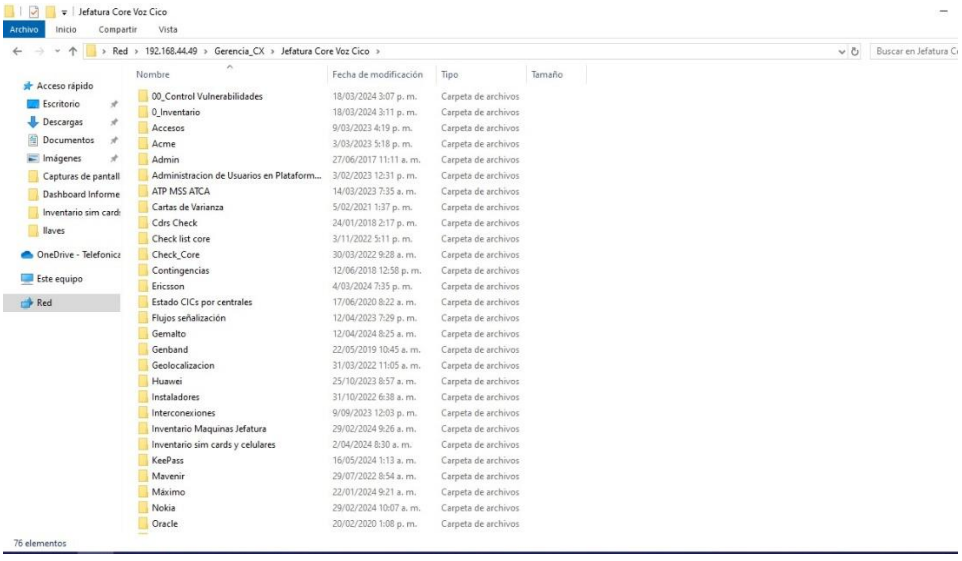


Figura 25. micrositio de la jefatura core voz y CICO.

En la plataforma de Teams, también se creó un micrositio de respaldo en caso de fallos.

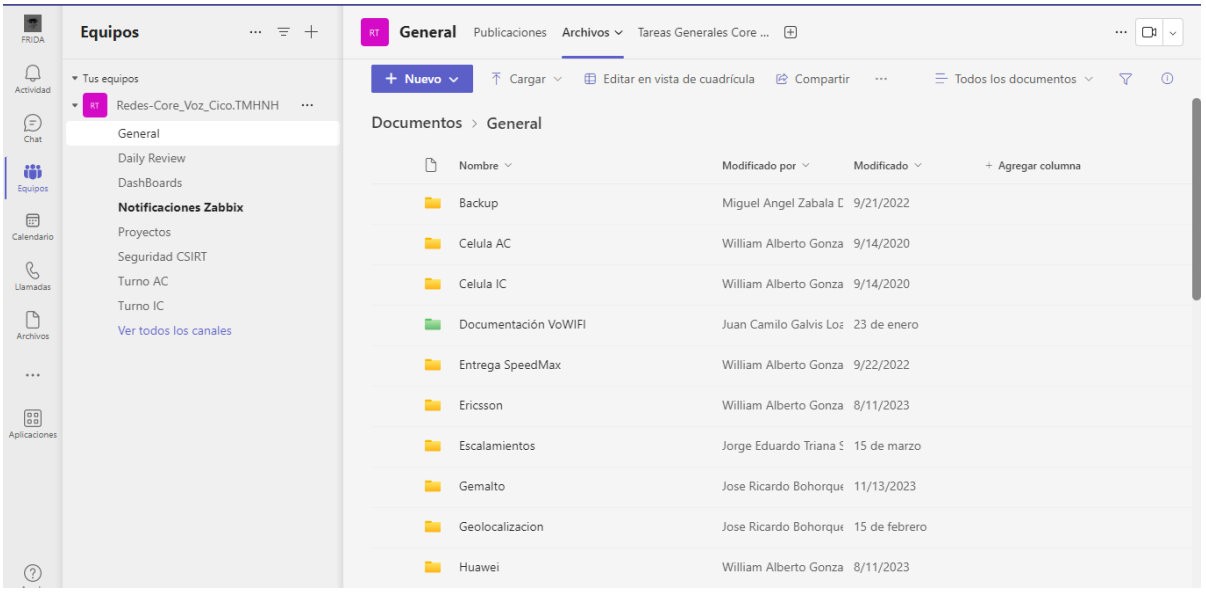


Figura 26. Micrositio de Teams

5. IMPLEMENTACIÓN DE LAS BUENAS PRÁCTICAS DE ITIL4

5.2. Análisis de Modelos de Gestión

A continuación, se presenta un análisis de algunos modelos de gestión relevantes en la Tabla 1, con el fin de justificar la razón por la cual se implementó ITIL4.

Esta comparación detallada permite evaluar las ventajas y desventajas de cada modelo, proporcionando una base sólida para entender por qué ITIL4 es la opción más adecuada para la gestión de servicios en la jefatura Core Voz y Cico de Telefónica.

Modelo/Framework	Descripción	Ventajas	Desventajas
ITIL v3 (Information Technology Infrastructure Library v3)	Conjunto de buenas prácticas para la gestión de servicios de TI, centrado en un ciclo de vida del servicio más estructurado.	• Estructura clara del ciclo de vida del servicio. • Amplia aceptación global. • Foco en la alineación de TI con el negocio	• Menos flexible y adaptable en comparación con ITIL4. • Transición a ITIL4 recomendada para mayor agilidad
ITIL 4 (Information Technology Infrastructure Library)	Conjunto de buenas prácticas para la gestión de servicios de TI que se centra en alinear los servicios de TI con las necesidades del negocio y generar valor.	• Enfoque en la mejora continua. • Flexibilidad y adaptabilidad. • Amplio reconocimiento. • Enfoque en la creación de valor. • Enfoque en la gestión de servicios. • Fomenta la innovación	• Complejidad en la implementación. • Puede ser percibido como demasiado centrado en procesos. • Tiempo de implementación
COBIT (Control Objectives for Information and Related Technologies)	Marco de gestión y gobierno de TI que se centra en la alineación de la TI con los objetivos del negocio.	• Gobierno de TI sólido. • Orientación integral. • Mejora del control y gestión	• Enfoque de alto nivel. • Curva de aprendizaje compleja. • Resistencia al cambio. • Enfoque en la gobernanza
ISO/IEC 20000	Estándar internacional para la gestión de servicios de TI basado en las mejores prácticas de ITIL.	• Certificación reconocida internacionalmente. • Estandarización. • Gestión de riesgos	• Rigidez. • Enfoque en cumplimiento. • Resistencia al cambio. • Mantenimiento continuo
DevOps	Conjunto de prácticas que combina el desarrollo de software (Dev) y las operaciones de TI (Ops) para mejorar la colaboración y acelerar la entrega de software.	• Velocidad y agilidad. • Cultura colaborativa. • Mejora de calidad del software	• Requiere cambio cultural significativo. • Enfoque técnico que puede no cubrir todas las áreas de la gestión de servicios de TI

Tabla 1. Análisis de modelos de gestión de servicios Fuente: Elaboración propia.

ITIL4 se presenta como la mejor opción para la gestión de servicios en la jefatura Core Voz y Cico de Telefónica debido a su enfoque en la mejora continua, flexibilidad y adaptabilidad. Este marco promueve un ciclo de mejora constante a través de su modelo de cadena de valor del servicio y principios guía, lo que es esencial para mantener la competitividad y eficiencia operativa en un entorno tan dinámico como el de las telecomunicaciones. Además, ITIL4 permite adaptar las

prácticas según las necesidades específicas de la organización, facilitando una respuesta a las demandas del mercado y asegurando que todos los aspectos de la gestión de servicios estén cubiertos, desde el diseño y la transición hasta la operación y la mejora continua.

Otra ventaja clave de ITIL4 es su enfoque en la creación de valor y su reconocimiento global. ITIL4 alinea los servicios de TI con los objetivos del negocio, asegurando que contribuyan al logro de los objetivos estratégicos de Telefónica y mejoren la satisfacción del cliente. Su adopción global facilita la implementación de mejores prácticas probadas y el alineamiento con estándares internacionales, crucial para una empresa que opera en múltiples regiones. Además, ITIL4 fomenta la innovación, permitiendo a la jefatura implementar nuevas tecnologías y soluciones de manera efectiva, mejorando la eficiencia operativa y promoviendo un entorno donde la innovación es valorada y fomentada.

Para la implementación de ITIL4, es fundamental definir las prácticas y componentes relevantes para los procesos revisados, de manera que contribuyan a las actividades de mejoramiento continuo.

5.1. Gestión de desarrollo y software.

Para iniciar el desarrollo de un software, es esencial realizar un análisis exhaustivo de los requisitos del proceso que se busca automatizar. Este análisis debe preceder al diseño de una solución que se ajuste al proceso, considerando cuidadosamente los datos relevantes y las posibles implicaciones del mal uso de cualquier comando, ya que este puede afectar el servicio de múltiples usuarios. Luego, es crucial someter esta solución a pruebas exhaustivas para garantizar que cumpla con los requerimientos.

Si la intención es implementar el desarrollo a nivel nacional, se recomienda ejecutarlo inicialmente en un entorno centralizado. Esta medida permite minimizar el impacto en caso de errores, evitando así perjudicar a un gran número de usuarios. Una vez verificada su eficacia y estabilidad en este entorno controlado, se puede proceder a su implementación a nivel nacional. Además, mediante el uso de herramientas como Zabbix, se puede garantizar un monitoreo constante del sistema, lo que facilita la identificación de áreas de mejora para futuros desarrollos.

A continuación, se muestra un gráfico, donde se explica un ciclo estructurado que garantiza la entrega de los productos e incluye las etapas clave:

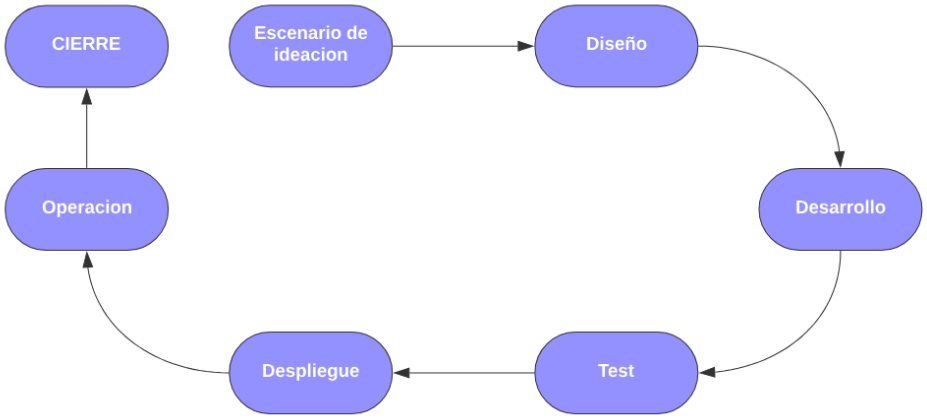


Gráfico 1. Diagrama de Flujo de Gestión de desarrollo y software. Fuente: elaboración propia.

- **Escenario de ideación:** Generación de ideas y definición de requisitos. En esta etapa se pueden realizar actividades como recopilación de requisitos, análisis de viabilidad y mapas mentales.
- **Diseño:** Creación del software y especificaciones técnicas. Las actividades incluyen diseño de diagramas de flujo, planificación de la arquitectura y elaboración de especificaciones técnicas.

- **Desarrollo:** Codificación e implementación de funcionalidades.
- **Pruebas**: Aseguramiento del correcto funcionamiento del software mediante pruebas unitarias, de integración y de rendimiento.
- **Despliegue**: Implementación del software en el entorno de producción, incluyendo actividades de instalación, configuración y migración de datos.
- **Operación:** Uso del software y mantenimiento, con actividades de monitoreo, soporte técnico y mantenimiento preventivo.
- **Cierre:** Evaluación final, documentación y cierre del proyecto.

5.2. Gestión de control de versiones

Es crucial contar con una documentación precisa de todos los productos, plataformas y servicios que se gestionan, dado que están sujetos a actualizaciones frecuentes. Cada versión debe estar debidamente registrada y documentada. El uso de herramientas como el

micrositio facilita a los miembros del equipo la tarea de documentar los cambios realizados y comunicarlos eficazmente al resto del equipo, reduciendo así la posibilidad de malos entendimientos

Gráfico 2. Diagrama de control de versiones. Fuente: elaboración propia

En el gráfico 2 se observa un diagrama para el control de versiones que asegura una adecuada organización y manejo:

1. **Creación de versiones:** Inicia con la creación de una nueva versión del software o servicio.
2. **Pruebas y validación:** Las versiones creadas se someten a pruebas exhaustivas para asegurar que cumplen con los requisitos establecidos.
3. **Documentación:** Una vez validadas, se documentan las versiones en las distintas plataformas.
4. **Comunicación de cambios:** Los cambios documentados se comunican a todos los miembros del equipo.
5. **Implementación**: La nueva versión se implementa en el entorno de producción.

5.3. Gestión de la implementación

Todas las configuraciones y cambios se ejecutan siguiendo un plan detallado por fases, que integra las actividades a desarrollar por el equipo de trabajo y considera la entrega continua de mejoras, asignando el tiempo necesario para cada actividad. Además, se realiza una evaluación exhaustiva una vez completada la implementación, con el fin de retroalimentar y optimizar las actividades futuras.

Por medio de ayudas visuales, se pueden establecer las fases de implementación y las actividades asociadas. Algunas de las fases implementadas incluyen:

- Fases del Proyecto: Planificación, Desarrollo, Pruebas, Implementación, Evaluación
- Tareas Asociadas: Definición de requisitos, diseño de arquitectura, codificación, pruebas unitarias, integración, despliegue en producción, monitoreo y mantenimiento.
- Duración de Cada Fase: Asignación de tiempo específico para cada fase según la complejidad y el alcance del proyecto.

5.4. Gestión de incidentes

La gestión de incidentes es crucial en la jefatura Core Voz y Cico, ya que algunos miembros del equipo no conocen los procedimientos y planes de acción adecuados para abordar un incidente, lo que puede derivar en problemas futuros. Es fundamental contar con una

documentación clara de los procedimientos, que incluya la prioridad, clasificación, antecedentes, el debido escalamiento y el cierre de cada incidente.

Asimismo, se debe implementar un proceso de validación de la información registrada. Si la interrupción persiste, es necesario realizar un seguimiento y control exhaustivo para facilitar un análisis concreto y una gestión más eficiente en futuras ocurrencias.

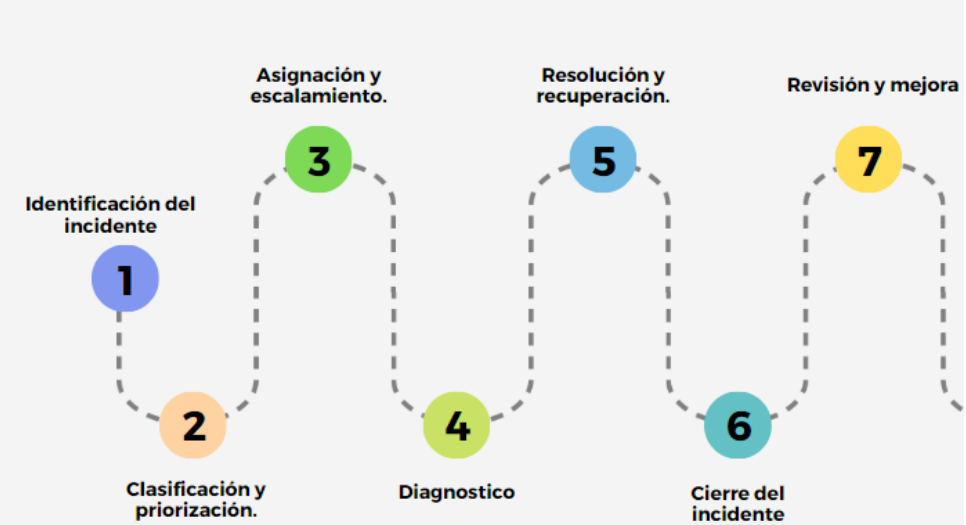


Gráfico 3. Diagrama de flujo de la Gestión de incidentes. Fuente: Elaboración propia.

En el gráfico 3 se presenta un diagrama de flujo que ilustra cómo gestionar un incidente. A continuación, se explica cada paso y las acciones correspondientes:

1. **Identificación del Incidente:**

   El proceso comienza cuando se detecta un incidente. Un incidente puede ser reportado por usuarios finales, detectado automáticamente por herramientas de monitoreo, o identificado por el personal de TI.

   Acciones: Registro del incidente con todos los detalles relevantes, como la fecha y hora, la persona que reportó, y una descripción del problema.

2. **Clasificación y Priorización:**

   Los incidentes se clasifican según su tipo (por ejemplo, problemas de hardware, software, red) y se priorizan en función de su impacto y urgencia.

Acciones: Asignación de un nivel de prioridad (por ejemplo, crítica, alta, media, baja) para determinar el orden en el que se atenderán los incidentes.

3. **Asignación y Escalamiento:**

   El incidente se asigna al equipo o persona adecuada para su resolución. Si el incidente no puede ser resuelto por el nivel de soporte inicial, se escala a un nivel superior.

   Acciones: Asignación del incidente a un técnico o equipo, y escalamiento a niveles superiores si es necesario.

4. **Investigación y Diagnóstico:**

   Se investiga el incidente para determinar su causa raíz y encontrar una solución. Esta etapa puede involucrar pruebas y análisis.

   Acciones: Realización de pruebas, análisis de logs y sistemas, y diagnóstico del problema.

5. **Resolución y Recuperación:**

   Una vez diagnosticado, se implementa una solución para resolver el incidente y restaurar el servicio.

   Acciones: Aplicación de parches, reinicio de sistemas, reconfiguración de hardware/software, etc.

6. **Cierre del Incidente:**

   Después de que el incidente ha sido resuelto, se verifica que el servicio ha sido restaurado a su estado normal y se cierra el incidente.

   Acciones: Confirmación con el usuario final de que el problema ha sido resuelto, documentación de la solución y cierre del incidente en el sistema de gestión de incidentes.

7. **Revisión y Mejora:**

   Se realiza una revisión post-incidente para identificar oportunidades de mejora y prevenir futuros incidentes similares.

   Acciones: Análisis de la gestión del incidente, documentación de lecciones aprendidas, y recomendaciones para mejorar procesos y evitar recurrencias.

Esta gestión requiere la colaboración de todos los miembros del equipo y es de vital importancia en empresas como Telefónica, debido a la gran cantidad de usuarios que se manejan diariamente.

5.5. Gestión de Problemas

La gestión de problemas se enfoca en identificar y abordar la causa potencial de uno o más incidentes, con el propósito de minimizar su probabilidad e impacto. Para ello, es fundamental revisar los incidentes registrados, identificar similitudes y determinar la causa raíz.

Para evitar confusiones con la gestión de incidencias, se debe crear un registro detallado de los problemas, donde se especifiquen estos y se elabore un plan de acción. Esto asegura la implementación de soluciones definitivas, evitando que los problemas escalen y afecten el servicio de manera más significativa.

Gráfico 4. Proceso de gestión de problemas. (InvGate, 2022)

La gestión de problemas se enfoca en identificar y abordar las causas subyacentes de los incidentes, con el propósito de minimizar su probabilidad e impacto. El proceso de gestión de problemas incluye las siguientes etapas:

Detección del Problema:
El proceso comienza cuando se detecta un problema. Esto puede surgir de la observación de patrones en incidentes repetitivos o mediante el monitoreo proactivo.

Clasificación del Problema:
Los problemas se clasifican según su tipo y naturaleza para facilitar su gestión y resolución.

Priorización del Problema:
Los problemas se priorizan en función de su impacto y urgencia, determinando el orden en el que deben ser abordados.

Análisis del Problema:
Se lleva a cabo una investigación detallada para identificar la causa raíz del problema.

Resolución del Problema:
Desarrollo e implementación de una solución para eliminar o mitigar el problema identificado.

Cierre del Problema:
Una vez que el problema ha sido resuelto, se verifica que la solución ha sido efectiva y se cierra el problema.

Este proceso asegura que los problemas se gestionen de manera eficaz, minimizando el impacto en las operaciones y mejorando continuamente el servicio.

5.6. Gestión de pruebas

Se debe validar que el servicio funcione correctamente mediante la ejecución de pruebas de funcionamiento, seguridad y rendimiento, con el fin de garantizar su calidad. Estas pruebas deben realizarse según lo planificado, registrando tanto los resultados positivos

como los negativos, así como cualquier observación pertinente, para que todo quede documentado y se proporcione claridad en los resultados.

Esta gestión se integra con la gestión de incidentes y la gestión de problemas, asegurando que forme parte de un enfoque holístico de la gestión de servicios de TI.

.

ID de la Prueba	Descripción de la Prueba	Tipo de Prueba	Criterios de Éxito	Resultado Esperado	Resultado Real	Estado	Observaciones

Tabla 2. Matriz de pruebas. Fuente: Elaboración propia.

En la tabla 2 se presenta una matriz de pruebas diseñada para documentar, planificar y organizar el seguimiento de todas las pruebas. Esta matriz ayuda a garantizar que se cubran todos los aspectos del sistema y que los resultados se documenten de manera clara y concisa.

5.7. Gestión de la estrategia

Se debe asegurar que los proyectos y soluciones propuestas estén alineados con los objetivos y la visión de la jefatura, para que tengan un impacto significativo en su cumplimiento. Es crucial analizar cómo se recopila la información para identificar más fácilmente indicadores de mejora. Además, se debe estar atento a los cambios inesperados en el mercado de las telecomunicaciones, ya que es altamente competitivo. Adaptarse a

estos cambios es esencial para garantizar que el proyecto genere valor. A continuación, en el grafico 5 se observan las etapas a seguir de una buena gestión estratégica.

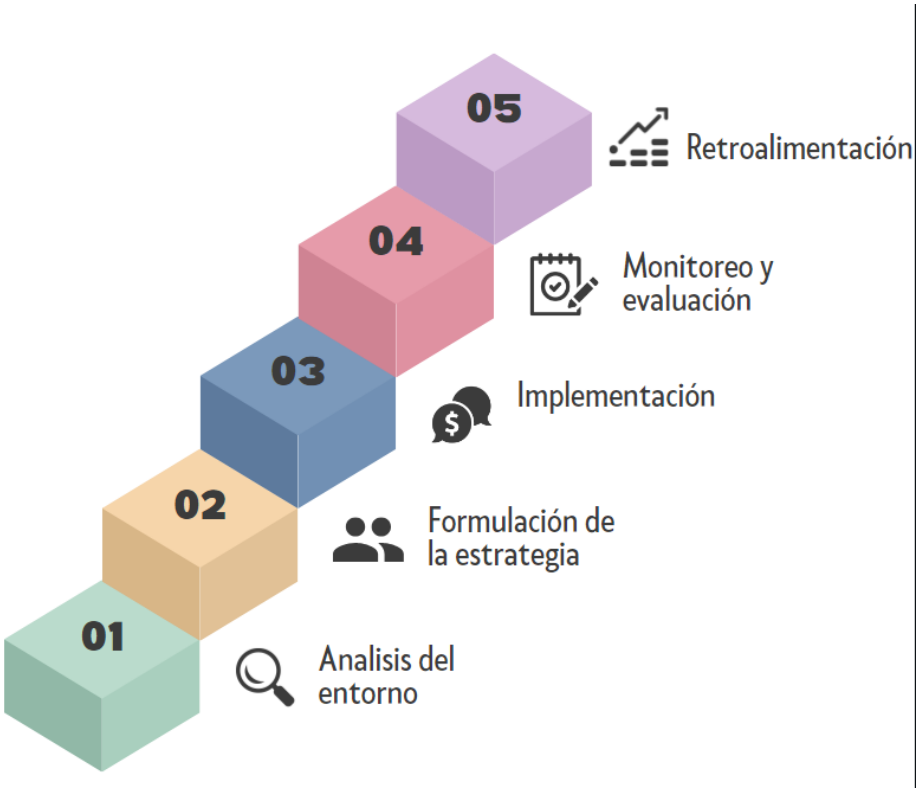


Gráfico 5. Etapas de la gestión estratégica. Fuente: elaboración propia.

1. **Análisis del Entorno:** Comprender el contexto en el que opera la organización lo cual permite identificar oportunidades y amenazas, así como fortalezas y debilidades internas.
2. **Formulación de la Estrategia:** Desarrollar estrategias específicas para alcanzar estos objetivos, asegurando que estén alineadas con las metas de la jefatura.
3. **Implementación de la Estrategia:** Poner en práctica las estrategias mediante planes de acción detallados y la asignación adecuada de recursos. Ejecutar proyectos y programas estratégicos con un enfoque en la entrega de valor.
4. **Monitoreo y Evaluación:** Supervisar el progreso y evaluar la efectividad de las estrategias implementadas. Utilizar para realizar revisiones periódicas y ajustar los planes según sea necesario.
5. **Retroalimentación:** Incorporar lo aprendido y las observaciones en el proceso estratégico para mejorar continuamente y asegurar que se cumplan los objetivos de la organización.

.

5.8. Gestión de Mejora continua

Para garantizar un mejoramiento continuo, se implementa el ciclo de Deming (PVHA) según lo especificado en la norma ISO 9001. Los pasos para su implementación en la empresa son: planificar, hacer, verificar y actuar, como se puede ver en el grafico 6.

Gráfico 6. Ciclo de Deming (PDCA: Significado, Etapas E Importancia | SafetyCulture, 2024)

- **Planeación:**

  La planeación se lleva a cabo considerando el contexto de la empresa, sus necesidades y expectativas. Se establecen los objetivos del proyecto y las políticas de la empresa, tomando en cuenta los recursos disponibles, los riesgos y las oportunidades. Cada parte del proceso se identifica a través de un cronograma con tiempos estipulados.

- **Hacer:**

  Se ejecuta la planeación utilizando los recursos internos y externos necesarios, manteniendo un enfoque en la mejora continua y el valor. Se identifican los cambios y las acciones necesarias para aprovechar las oportunidades de mejora.

- **Verificar:**

  Se establecen periodos de prueba para verificar la efectividad de los cambios realizados. Se hace un seguimiento de los procesos y requisitos mediante métodos de evaluación y análisis, determinando cuándo y cómo se realizará el seguimiento para una evaluación constante.

- **Actuar:**

  Se evalúa si los resultados se ajustan a las expectativas y si generan valor. Se realizan correcciones si es necesario, y se toman decisiones basadas en esta evaluación para mejorar el desempeño y la eficacia de los procesos.

Este ciclo permite a la jefatura de Core Voz y Cico lograr mejoras a corto plazo, evitar incidencias, incrementar la productividad y mejorar la calidad del servicio. Además, facilita la adaptación a cambios imprevistos y la eliminación de procesos repetitivos y manuales.

6. MEJORA CUANTIFICADA

Las métricas clave para la mejora incluyen la disminución del tiempo de procesamiento, la reducción del personal necesario y la reducción de fallas. Cuantificar estas mejoras es esencial para

evaluar el impacto de las soluciones implementadas y para justificar futuros esfuerzos de automatización.

6.1. Métricas iniciales

Antes de la implementación de la automatización, se recolectaron las siguientes métricas para establecer una línea base:

- **Tiempo de procesamiento manual de borrado trace.log**: Las tareas de supervisión y gestión de archivos pesados, como la eliminación del archivo trace.log, tomaban un promedio de 3 horas por tarea. Es importante destacar que primero se debía programar una ventana de mantenimiento y discutir la tarea en un comité de control de cambios, lo que a veces retrasaba su ejecución.

- **Número de personal involucrado**: Se requerían dos ingenieros trabajando en el turno nocturno para ejecutar el borrado del trace.log. Este horario se elegía para minimizar el impacto en los clientes, ya que durante la noche hay menos tráfico y las posibles fallas afectan a menos usuarios.

- **Número de fallas o incidentes**: Se reportaba aproximadamente 2 incidentes por semana relacionado con el uso excesivo del disco.

- **Tiempo de respuesta a incidentes**: El tiempo promedio de respuesta a estos incidentes era de 3 horas. Además, se realizaban supervisiones de mantenimiento dos veces al mes.

- **Capacidad de almacenamiento utilizada**: El uso del disco en los servidores vMTAS frecuentemente superaba el 85%.

6.2. Métricas posteriores a la implementación

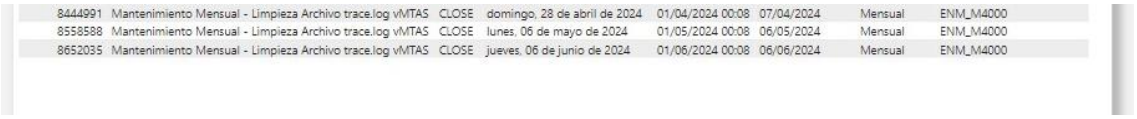
Después de la implementación de la automatización, se recolectaron las siguientes métricas:

- **Tiempo de procesamiento automatizado del borrado trace.log**: El tiempo requerido para completar esta tarea se redujo a 1 hora, gracias a la automatización. Ahora, cuando se supera el umbral de llenado del disco, el proceso se realiza automáticamente sin necesidad de convocar un comité ni de crear una ventana de mantenimiento.

- **Número de personal involucrado**: Ahora solo se necesita un ingeniero para supervisar las tareas automatizadas. Este ingeniero únicamente verifica que se haya activado la alarma y que la tarea se haya ejecutado exitosamente, eliminando la necesidad de realizar el proceso manualmente.

- **Número de fallas o incidentes**: La frecuencia de incidentes disminuyó a 2 incidentes por mes.

- **Tiempo de respuesta a incidentes**: El tiempo promedio de respuesta se redujo a 30 minutos. Además, se continúa realizando el borrado una vez al mes para asegurar un funcionamiento óptimo.



8444991	Mantenimiento Mensual - Limpieza Archivo trace.log vMTAS	CLOSE	domingo, 28 de abril de 2024	01/04/2024 00:08	07/04/2024	Mensual	ENM_M4000
8558588	Mantenimiento Mensual - Limpieza Archivo trace.log vMTAS	CLOSE	lunes, 06 de mayo de 2024	01/05/2024 00:08	06/05/2024	Mensual	ENM_M4000
8652035	Mantenimiento Mensual - Limpieza Archivo trace.log vMTAS	CLOSE	jueves, 06 de junio de 2024	01/06/2024 00:08	06/06/2024	Mensual	ENM_M4000

Figura 27. Mantenimiento mensual del borrado del vMTAS

En la figura 27 se puede visualizar el mantenimiento del mensual del borrado del vMTAS para asegurar el correcto funcionamiento.

Ejecución mantenimientos mensual

Mantenimiento	Programado	OT Ejecutadas	Periodicidad	% Cumpl
Revisión Mensual Discos_MSS	6	6	Mensual	100.0 %
Revisión Mensual Discos_MGW	6	6	Mensual	100.0 %
Limpieza Archivo trace.log_vMTAS	6	6	Mensual	100.0 %
Total	18	18	Mensual	100.0 %

Figura 28. Ejecución de mantenimientos mensuales del Vmtas.

En la figura 28 se observa la periodicidad del borrado en los servidores vMTAS, con una frecuencia mensual. Hasta el mes de junio, se han realizado 6 órdenes de trabajo, las cuales, a partir de abril, se ejecutaron automáticamente mediante el uso del script y Zabbix.

- **Capacidad de almacenamiento utilizada**: El uso del disco se mantuvo por debajo del 50%. Cuando el uso del disco alcanza el 80%, el script se ejecuta automáticamente, generando alarmas y permaneciendo muy poco tiempo en estado crítico.

6.3. Beneficios adicionales

Además de las mejoras cuantificadas, se observaron beneficios adicionales, tales como:

- **Mayor moral del equipo**: La reducción de tareas repetitivas y manuales permitió al personal enfocarse en actividades de mayor valor agregado, mejorando así la satisfacción y moral del equipo.

- **Mejora en la continuidad del servicio**: La automatización de la limpieza de archivos ha disminuido tanto la frecuencia como la duración de las interrupciones operativas, mejorando significativamente la continuidad del servicio.
- **Uso del script para futuros proyectos relacionados**: El script desarrollado puede ser reutilizado en futuros proyectos relacionados con la gestión del almacenamiento, sirviendo como una guía y herramienta valiosa.

7. CONCLUSIONES

La implementación del script para identificar y eliminar automáticamente archivos pesados ha optimizado significativamente el almacenamiento y la eficiencia operativa de los servidores

remotos. Esta automatización ha reducido el tiempo y esfuerzo manual necesarios para la gestión de archivos, mejorando así la disponibilidad y el rendimiento del sistema. Además, ha permitido una gestión más proactiva de los recursos, anticipándose a posibles problemas de capacidad y asegurando un uso más eficiente del espacio de almacenamiento disponible.

Por medio del proceso de automatización, se ha disminuido la frecuencia y duración de las interrupciones operativas causadas por el llenado del almacenamiento. Esto ha resultado en una mejora notable en la continuidad del servicio y en la satisfacción de los usuarios. La capacidad de anticiparse y resolver problemas de almacenamiento antes de que afecten el servicio ha sido crucial para mantener una operación fluida y sin interrupciones.

El monitoreo y control eficiente con Zabbix ha permitido una supervisión exhaustiva del proceso de eliminación de archivos. Por medio de la configuración de alarmas se ha garantizado una vigilancia efectiva de la ejecución y los resultados, facilitando una respuesta rápida ante cualquier anomalía detectada. Esta integración ha proporcionado una visibilidad continua y detallada del estado de los servidores, permitiendo intervenciones más rápidas y precisas.

La capacidad de Zabbix para proporcionar datos en tiempo real ha mejorado significativamente la toma de decisiones informadas, permitiendo seleccionar las mejores acciones correctivas y minimizando el impacto negativo en las operaciones. Los datos en tiempo real han permitido una mejor planificación y una respuesta más ágil a las necesidades operativas.

La propuesta de implementar las buenas prácticas de ITIL4 representa una mejora significativa en el orden y control de la documentación, impactando positivamente en la eficiencia y eficacia de la gestión. Esto ha permitido una supervisión más efectiva y una optimización constante de los recursos y procesos operativos. Estas prácticas ayudan a establecer un marco de trabajo sólido que garantiza la calidad y continuidad del servicio, alineándose con los objetivos estratégicos de la jefatura Core Voz y Cico.

La implementación de metodologías ágiles y las buenas prácticas de ITIL4 en la jefatura Core Voz y Cico ha sido un paso crucial hacia el establecimiento de un enfoque prioritario en la calidad del servicio. Al abordar los problemas de documentación y procesos, estas metodologías promueven la eficiencia operativa y la transparencia, fomentando la colaboración entre equipos. Además, al adoptar prácticas ágiles, se facilita la adaptación a cambios rápidos y se promueve una cultura de mejora continua, garantizando que la jefatura esté equipada para enfrentar desafíos futuros.

8. Referencias

1 Servidor. (n.d.). Retrieved May 20, 2024, from https://www.zabbix.com/documentation/current/es/manual/concepts/server

2 Agente. (n.d.). Retrieved May 20, 2024, from https://www.zabbix.com/documentation/current/es/manual/concepts/agent

Componentes de ITIL 4. (n.d.). Retrieved May 20, 2024, from https://www.itil.com.mx/componentes/

December 2020-https://prokanban.org/the-kanban-guide. (n.d.). Retrieved May 20, 2024, from https://prokanban.org/the-kanban-guide/

Misión - Telefónica. (n.d.). Retrieved May 20, 2024, from https://www.telefonica.com/es/nosotros/mision/

Monitoreo de TI empresarial con Zabbix. (n.d.). Retrieved May 20, 2024, from https://www.zabbix.com/la/enterprise_monitoring

Prieto Fernando. (n.d.). *ITIL 4*.

Telefónica Colombia | Innovación en Telecomunicaciones. (n.d.). Retrieved May 20, 2024, from https://www.telefonica.co/

Una guía completa para la entrega de proyectos utilizando Scrum Una guía para el CONOCIMIENTO DE SCRUM (GUÍA SBOK™) 2013 Edición A Guide to the SCRUM BODY OF KNOWLEDGE (SBOK™ Guide). (2013). www.scrumstudy.com

Zabbix features overview. (n.d.). Retrieved May 20, 2024, from https://www.zabbix.com/la/features

InvGate. (2022, 4 agosto). Gestión de problemas - Una guía definitiva - InvGate. https://invgate.com/es/guides/problem-management/

PDCA: significado, etapas e importancia | SafetyCulture. (2024, 28 marzo). SafetyCulture. https://safetyculture.com/es/temas/ciclo-pdca/

ANEXOS

ANEXO A. Librerías

```
import json
import paramiko
import subprocess
import os
```

ANEXO B. Función para establecer la conexión SSH.

```
def conexion(hostname, puerto, usuario):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    try:
        ssh.connect(hostname, port=puerto, username=usuario)
        print("Conectado al servidor " + hostname + " (" +
ssh.get_transport().sock.getpeername()[0] + ") exitosamente.\n")
        return ssh, hostname
    except paramiko.AuthenticationException:
        print("No se puede conectar al servidor. Error de
autenticacion.\n")
    except paramiko.SSHException as e:
        print("No se puede establecer una conexion SSH con el servidor.
Error: " + str(e) + "\n")
    except Exception as e:
        print("Ocurrio un error al conectar al servidor: " + str(e) +
"\n")

    return None, None
```

ANEXO C. Función para ejecutar los comandos

```
def ejecutar_comando(ssh, comando, hostname):
    stdin, stdout, stderr = ssh.exec_command(comando)
    salida = stdout.read().decode()
    errores = stderr.read().decode()
    return salida, errores
```

ANEXO D. Función para pasar de Gigas (G), Megas (M) y Kilos (K) a Bytes

```
def convertir_tamano_a_bytes(valor):
    multiplicadores = {'K': 1024, 'M': 1024*2, 'G': 1024*3}
    if valor[-1] in multiplicadores:
        numero, unidad = float(valor[:-1]), valor[-1]
```

```
        return int(numero * multiplicadores.get(unidad, 1))
    else:
        return valor
```

ANEXO E. Función para establecer la conexión con Zabbix

```
def zabbix_sender(json, host_name, key_name ):
    zabbix_server = '10.89.91.8'

    source_address = '192.168.44.158'
    host =host_name
    key = key_name
    key_value =json
    comando = "/usr/bin/zabbix_sender -z " + zabbix_server + " -I " +
source_address + " -s " + host + " -k " + key + " -o '" + key_value +
"'"
    os.system (comando)
```

ANEXO F. Función para sacar el tamaño trace.log

```
def obtener_tamano_trace_log(ssh, servidor):
    print('obtener trace.log')
    comando = "sudo ssh {} 'stat -c%s
/opt/trace/log/trace.log'".format(servidor)
    salida, errores = ejecutar_comando(ssh, comando, servidor)
    #print('se obtuvo el trace.log')
    return salida.strip()
```

ANEXO G. Función para guardar los resultados en un Json

```
def guardar_resultados(resultados):
    with open("resultados.json", "w") as f:
        f.write(resultados)
```

ANEXO H. Main para establecer la conexión con el Vmtas

```
def main(hostname , host, key):
    #hostname = '10.180.129.50'
    puerto = 22
    usuario = 'apcheck1'


    # Conectar al servidor
    ssh, hostname = conexion(hostname, puerto, usuario)
    print('conexion OK')
```

```
    if ssh:
        resultados = {}
        # 4.1) comando para chequear los filesystem del PL
        comando = "cat /etc/hosts | grep PL | awk '{print $2}'"
        salida, errores = ejecutar_comando(ssh, comando, hostname)
       # print('Se obtuvieron las PLs')
        servidores_PL = salida.splitlines()

        filesystems_pl = {}
        list_pls = []
        filesystem_info = []
        for servidor in servidores_PL:
            list_pls.append({'PL': servidor, 'log':''})
            comando = "echo ' {} :'; sudo ssh {} 'df -
kh'".format(servidor, servidor)
           # print('PL', servidor)
            salida, errores = ejecutar_comando(ssh, comando, hostname)
           # print('Se obtuvo los filesystems')



            # Dividir la salida por lineas y eliminar la primera linea
(encabezado)
            lines = salida.splitlines()[1:]

            for line in lines:
                # Dividir la linea en campos: filesystem, size, used,
avail, use%, mounted on
                fields = line.split()
                filesystem = fields[0]
                size = convertir_tamano_a_bytes(fields[1])
                used = convertir_tamano_a_bytes(fields[2])
                avail = convertir_tamano_a_bytes(fields[3])
                use_percent = fields[4]
                mounted_on = fields[5]
                if size != 'Size':

                    filesystem_info.append({
                        "Filesystem": filesystem,
                        "Avail": avail,
                        "Use": use_percent,
                        "Used": used,
                        "Mounted_on": mounted_on,
                        "Size": size,
                        "PL": servidor
                    })

            # Eliminar la primera parte de cada entrada de filesystem
            #filesystem_info = filesystem_info[1:]
```

```
        filesystems_pl['PLs'] = list_pls
        filesystems_pl['filesystems'] = filesystem_info

        for servidor in servidores_PL:
            # Obtener espacio de trace.log
            tamano_trace_log = obtener_tamano_trace_log(ssh, servidor)

            for item in enumerate(list_pls):
                if item[1]['PL'] == servidor:
                    list_pls[item[0]]['log']= tamano_trace_log
                    break


        resultado_json =json.dumps(filesystems_pl)
        resultado_json_indent = json.dumps(filesystems_pl, indent=4)
        zabbix_sender(resultado_json, host, key)
        guardar_resultados(resultado_json_indent)
        print("Resultados guardados en resultados.json")

        # Comando de zabbix_sender


        ssh.close()
    else:
        print("No se pudo establecer la conexion SSH.")



if _name_ == "_main_":



     main('10.180.129.50', 'vmtas01ca','tracelog.json') # main del vsbg
cali
     main('10.180.193.50', 'vsbg01bq','tracelog.json.bq') # main del vsg
barraquilla
```

ANEXO I. Script Vmtas completo

```
# Librerias a usar:

import json
import paramiko
import subprocess
import os

# funicion para establecer conexion a travves de SSH
def conexion(hostname, puerto, usuario):
    ssh = paramiko.SSHClient()
```

```
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    try:
        ssh.connect(hostname, port=puerto, username=usuario)
        print("Conectado al servidor " + hostname + " (" +
ssh.get_transport().sock.getpeername()[0] + ") exitosamente.\n")
        return ssh, hostname
    except paramiko.AuthenticationException:
        print("No se puede conectar al servidor. Error de
autenticacion.\n")
    except paramiko.SSHException as e:
        print("No se puede establecer una conexion SSH con el servidor.
Error: " + str(e) + "\n")
    except Exception as e:
        print("Ocurrio un error al conectar al servidor: " + str(e) +
"\n")

    return None, None
# funcion para ejecutar los comandos linux
def ejecutar_comando(ssh, comando, hostname):
    stdin, stdout, stderr = ssh.exec_command(comando)
    salida = stdout.read().decode()
    errores = stderr.read().decode()
    return salida, errores
# funcion para pasa de Gigas (G), Megas(M) y Kilos (K) a Bytes
def convertir_tamano_a_bytes(valor):
    multiplicadores = {'K': 1024, 'M': 1024*2, 'G': 1024*3}
    if valor[-1] in multiplicadores:
        numero, unidad = float(valor[:-1]), valor[-1]
        return int(numero * multiplicadores.get(unidad, 1))
    else:
        return valor  # Devuelve el valor original si no tiene una
unidad de medida

#funcion para establecer comunicacion con el zabbix
def zabbix_sender(json, host_name, key_name ):
    zabbix_server = '10.89.91.8'
    source_address = '192.168.44.158'
    host =host_name
    key = key_name
    key_value =json
    comando = "/usr/bin/zabbix_sender -z " + zabbix_server + " -I " +
source_address + " -s " + host + " -k " + key + " -o '" + key_value +
"'"
    os.system (comando)

#funcion para sacar el tamano del archvio trace.log
def obtener_tamano_trace_log(ssh, servidor):
    print('obtener trace.log')
```

```
    comando = "sudo ssh {} 'stat -c%s
/opt/trace/log/trace.log'".format(servidor)
    salida, errores = ejecutar_comando(ssh, comando, servidor)
    #print('se obtuvo el trace.log')
    return salida.strip()

# funcion para guardar los resultados en el json
def guardar_resultados(resultados):
    with open("resultados.json", "w") as f:
        f.write(resultados)

# main para establecer la conexion con el vmtas
def main(hostname , host, key):
    puerto = 22
    usuario = 'apcheck1'


    # Conectar al servidor
    ssh, hostname = conexion(hostname, puerto, usuario)
    print('conexion OK')

    if ssh:
        resultados = {}
        # 4.1) comando para chequear los filesystem del PL
        comando = "cat /etc/hosts | grep PL | awk '{print $2}'"
        salida, errores = ejecutar_comando(ssh, comando, hostname)
       # print('Se obtuvieron las PLs')
        servidores_PL = salida.splitlines()

        filesystems_pl = {}
        list_pls = []
        filesystem_info = []
        for servidor in servidores_PL:
            list_pls.append({'PL': servidor, 'log':''})
            comando = "echo ' {} :'; sudo ssh {} 'df -
kh'".format(servidor, servidor)
           # print('PL', servidor)
            salida, errores = ejecutar_comando(ssh, comando, hostname)
           # print('Se obtuvo los filesystems')



            # Dividir la salida por lineas y eliminar la primera linea
(encabezado)
            lines = salida.splitlines()[1:]

            for line in lines:
                # Dividir la linea en campos: filesystem, size, used,
avail, use%, mounted on
                fields = line.split()
```

```
                filesystem = fields[0]
                size = convertir_tamano_a_bytes(fields[1])
                used = convertir_tamano_a_bytes(fields[2])
                avail = convertir_tamano_a_bytes(fields[3])
                use_percent = fields[4]
                mounted_on = fields[5]
                if size != 'Size':

                    filesystem_info.append({
                        "Filesystem": filesystem,
                        "Avail": avail,
                        "Use": use_percent,
                        "Used": used,
                        "Mounted_on": mounted_on,
                        "Size": size,
                        "PL": servidor
                    })

            # Eliminar la primera parte de cada entrada de filesystem
            #filesystem_info = filesystem_info[1:]

        filesystems_pl['PLs'] = list_pls
        filesystems_pl['filesystems'] = filesystem_info

        for servidor in servidores_PL:
            # Obtener espacio de trace.log
            tamano_trace_log = obtener_tamano_trace_log(ssh, servidor)

            for item in enumerate(list_pls):
                if item[1]['PL'] == servidor:
                    list_pls[item[0]]['log']= tamano_trace_log
                    break


        resultado_json =json.dumps(filesystems_pl)
        resultado_json_indent = json.dumps(filesystems_pl, indent=4)
        zabbix_sender(resultado_json, host, key)
        guardar_resultados(resultado_json_indent)
        print("Resultados guardados en resultados.json")

        # Comando de zabbix_sender


        ssh.close()
    else:
        print("No se pudo establecer la conexion SSH.")



if _name_ == "_main_":
```

```
     main('10.180.129.50', 'vmtas01ca','tracelog.json') # main del vsbg
cali
     main('10.180.193.50', 'vsbg01bq','tracelog.json.bq') # main del vsg
barraquilla
```

ANEXO J. Archivo Json

```
{
    "PLs": [
        {
            "PL": "PL-10",
            "log": "8321"
        },
        {
            "PL": "PL-11",
            "log": "8321"
        },
        {
            "PL": "PL-12",
            "log": "8321"
...
  ]
    "filesystems": [
        {
            "Filesystem": "root",
            "Avail": 908066816,
            "Use": "72%",
            "Used": 2362232012,
            "Mounted_on": "/",
            "Size": 3221225472,
            "PL": "PL-10"
        },
        {
            "Filesystem": "devtmpfs",
            "Avail": 53687091200,
            "Use": "1%",
            "Used": 4096,
            "Mounted_on": "/dev",
            "Size": 53687091200,
            "PL": "PL-10"
        },
...
  ]
```