UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN


# "SISTEMA DE MONITORIZACIÓN DE SERVIDOR EN TIEMPO REAL CON PROTOCOLO SNMPv3"

TRABAJO FIN DE GRADO

Febrero -2025

AUTOR: Mohamed Adam Toumi

DIRECTOR/ES: Oscar Martínez Bonastre

# RESUMEN

El presente proyecto final de grado aborda el diseño e implementación de un sistema de monitorización en tiempo real de servidores mediante el protocolo SNMPv3. El objetivo principal es garantizar una monitorización segura, eficiente y automatizada de recursos críticos como CPU, memoria y espacio en disco, aportando herramientas para la gestión de incidencias y la toma de decisiones. El sistema diseñado es una arquitectura cliente-servidor que combina un agente de monitorización sobre un servidor Ubuntu y un panel de control web accesible desde cualquier navegador. Este panel permite la visualización en tiempo real de los datos mediante gráficos dinámicos, la exportación de informes en formato PDF y la gestión de alarmas en función de umbrales predefinidos. El sistema incorpora además funciones de autenticación y autorización para distinguir entre usuarios estándar y administradores. La metodología indicada ha sido la configuración de SNMPv3, un backend desarrollado en Python y PHP, así como una base de datos MySQL para guardar los registros históricos de la base de datos. Los resultados obtenidos demuestran la capacidad del sistema para identificar y registrar patrones de comportamiento anómalos en el servidor, mejorando la prevención y gestión de fallos. Este proyecto final no sólo funciona, sino que también sirve como una herramienta eficaz para gestionar sistemas críticos, haciendo hincapié en la seguridad y la facilidad de uso.

**Palabras clave**: monitoreo en tiempo real, SNMPv3, sistema de servidores, seguridad, gestión de recursos, Python, PHP, MySQL.

# ABSTRACT

This final degree project focuses on the design and implementation of a real-time monitoring system for servers using the SNMPv3 protocol. The primary objective is to ensure a safe, efficient, and automated monitoring of critical resources such as CPU, memory, and disk space, providing tools for incident management and decision making. The system designed is a client-server architecture that combines a monitoring agent on an Ubuntu server and a web control panel that can be accessed from any browser. This panel enables real-time visualization of data via dynamic graphs, exporting reports as PDF, and managing alarms based on pre-defined thresholds. The system also incorporates authentication and authorization functions to distinguish between regular users and administrators. The methodology indicated SNMPv3 configuration, a backend developed in Python and PHP, as well as a MySQL database to save historical database records. The results obtained demonstrate the ability of the system to identify and record anomalous behavior patterns in the server, enhancing the prevention and management of failures. This final project not only works but also serves as an effective tool for managing critical systems, emphasizing security and ease of use.



**Keywords**: real-time monitoring, SNMPv3, server systems, security, resource management, Python, PHP, MySQL.

# INDICE:

# ÍNDICE DE ILUSTRACIONES

# 1. INTRODUCCIÓN

## 1.1 CONTEXTO Y MOTIVACIÓN

En un entorno en el que el avance tecnológico es continuo y la dependencia de sistemas informáticos aumenta cada día más, se hace imprescindible la tarea de garantizar que los servidores funcionen de manera adecuada. Empresas y organizaciones confían ciegamente en los servidores donde corren sus servicios en tiempo real, gestionando miles de millones de datos a la vez, asegurando así su operatividad. Por consiguiente, un sistema de monitorización de servidores es vital para detectar y prevenir posibles fallos que puedan afectar el rendimiento o seguridad del sistema. No obstante, la mayoría de los sistemas de monitorización actuales presentan limitaciones, por ejemplo, algunos de ellos son poco administrables en entornos dinámicos o en tiempo real, entre otros problemas. Incluso, herramientas como Nagios, Zabbix o Icinga, que son las más populares, suelen ser demasiado complejas para implementaciones de menor envergadura, tener un elevado costo de adquisición o simplemente no ser adaptables fácilmente.

Por consiguiente, se explora un sistema de monitorización basado en la versión SNMPv3 motivante en ese sentido con un enfoque en la automatización del proceso de consultas y generación de gráficos y reportes de los parámetros clave de un servidor. El motivo fundamental de este proyecto no habla solo de una necesidad o motivación técnica, además, se pretende que sea también una herramienta pedagógica permitiendo que un profesional o estudiante en la industria pueda adaptar este código a su infraestructura o que simplemente entienda cómo funciona un sistema basado en un protocolo.

## 1.2 ESTADO DEL ARTE

En la administración de infraestructuras informáticas, algo que resulta ser una necesidad esencial es el monitoreo de servidores y sistemas. Con el tiempo, se han desarrollado, implementado y perfeccionado diversas herramientas con el objetivo de abordar este desafío, con diferencias y similitudes significativas entre ellas. En este sentido, existen limitaciones inherentes a todas ellas que este proyecto pretende evitar. A continuación, se muestra información sobre algunas de las herramientas más conocidas, junto con sus ventajas y desventajas.

### 1.2.1 Zabbix

Zabbix [11]| es un software de código abierto con una amplia presencia en el monitoreo de grandes infraestructuras. Su enfoque radica en el monitoreo de datos de cualquier dispositivo, junto con los servicios relacionados, para proporcionar gráficos y alertas totalmente configurados.


*Ilustración 1. Logo Zabbix*

Esta herramienta es altamente escalable y es una elección sólida para el monitoreo empresarial. El inconveniente es que es difícil de instalar y mantener, y consume muchos recursos para los usuarios y empresas que no pueden implementar un equipo dedicado.


*Ilustración 2. Interfaz Zabbix*

### 1.2.2 Observium

Observium [12] es conocido por su orientación al monitoreo basado en SNMP. Ofrece una interfaz gráfica agradable y se puede desplegar en un corto período. Esta opción es adecuada para monitorear el flujo en tiempo real de una red y de los dispositivos en ella.


*Ilustración 3. Logo de Observium.*

Presenta una configuración inicial sencilla y rápida junto con una interfaz intuitiva lo que la hace una gran opción para empresas pequeñas y medianas. Sin embargo, presenta una

dependencia exclusiva de SNMP lo que la limita para otros entornos y las funcionalidades avanzadas solo están presentes en su versión de pago.

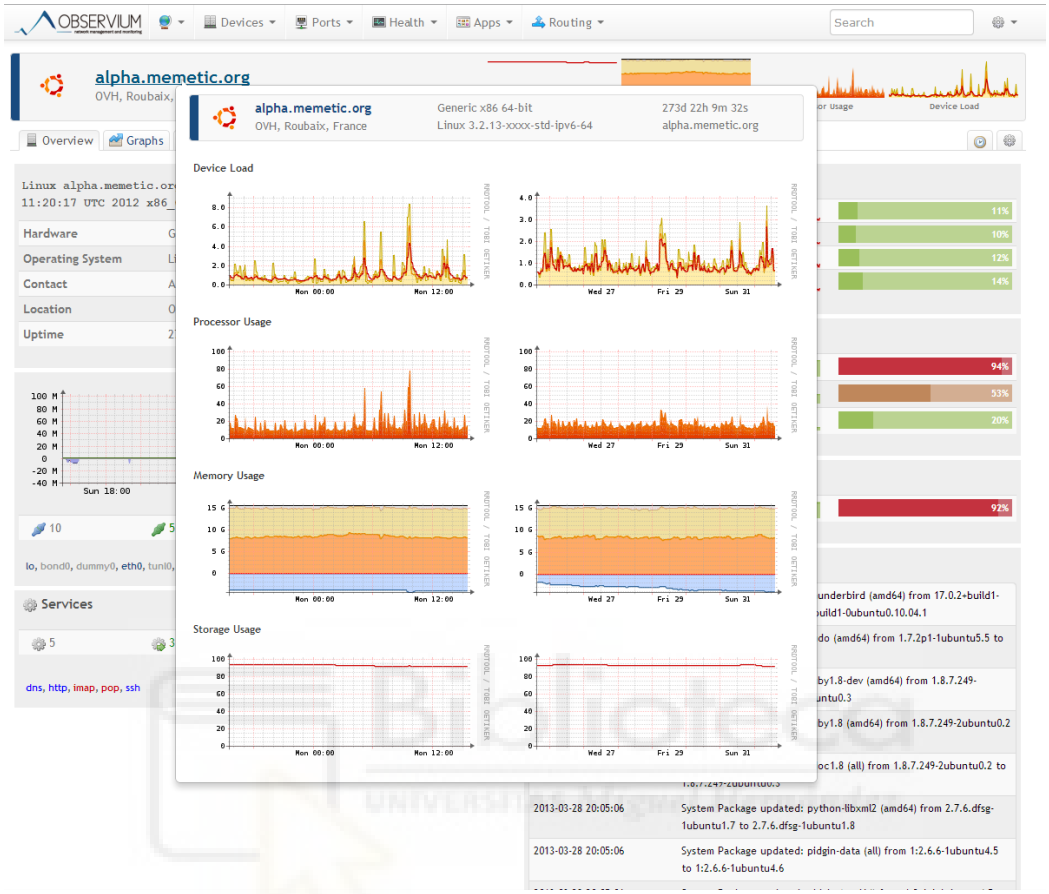



*Ilustración 4. Interfaz Observium.*

### 1.2.3 Nagios

Nagios [13] es una solución pionera en monitoreo de sistemas. Presenta un diseño que permite personalizar y extender sus capacidades mediante el uso de complementos y scripts, lo que le permite adaptarse a una amplia gama de entornos.

*Ilustración 5. Logo Nagios.*

Permite monitorear dispositivos de red, servidores y servicios utilizando SNMP y otras tecnologías de monitoreo. Presenta una amplia compatibilidad con distintos sistemas operativos y una gran variedad de plataformas.

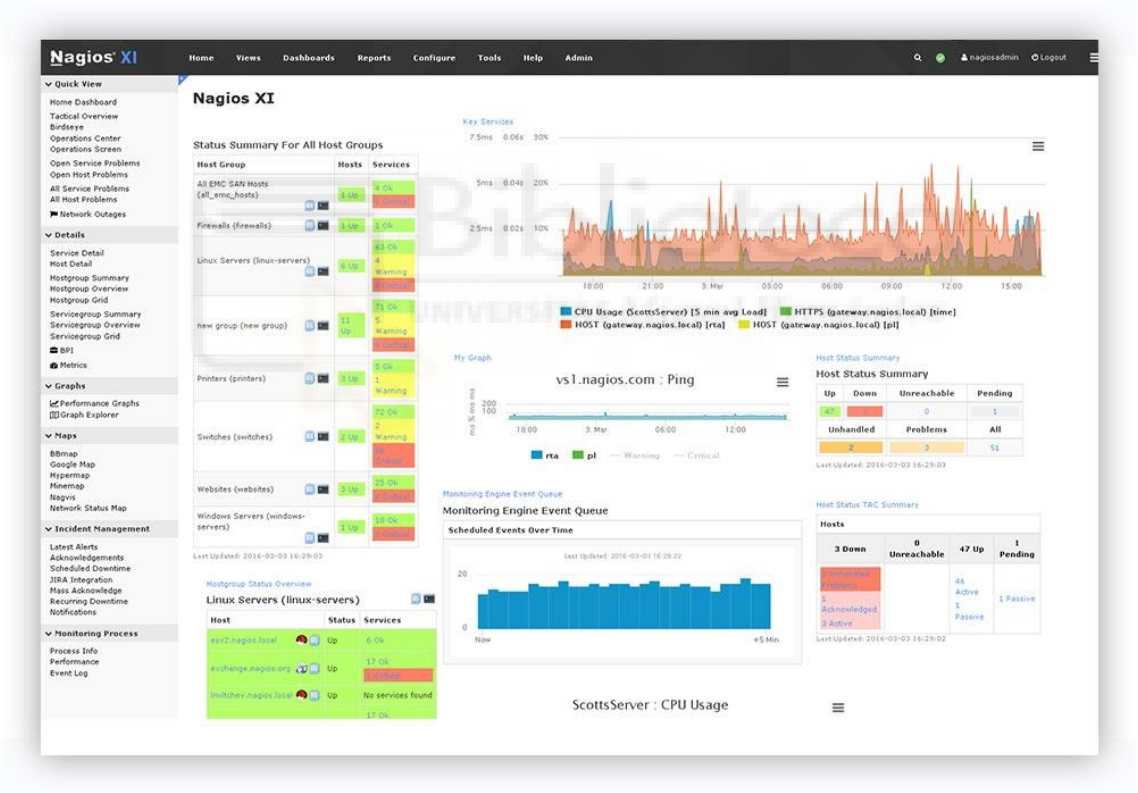



*Ilustración 6. Interfaz Nagios*

Sin embargo, presenta una interfaz gráfica limitada y compleja en comparación con las soluciones más modernas presentes hoy en día, esto implica que se requiera unos conocimientos avanzados para la configuración y el mantenimiento.

### 1.2.4 Icinga

Icinga [14] surge como una mejora de Nagios, es de código abierto y puede monitorear SNMPv3, SMTP, entre otros. Incorpora una interfaz más asequible para los usuarios y presenta características mejoradas para entornos distribuidos.


*Ilustración 7. Logo Icinga.*

Permite una excelente monitorización de infraestructuras complejas, presenta soporte para entornos distribuidos y compatibilidad con los plugins de Nagios. Las limitaciones que se podrían destacar serian que presenta una dependencia de módulos adicionales para funciones avanzadas y tiene una curva de aprendizaje algo compleja para usuarios sin experiencia previa.

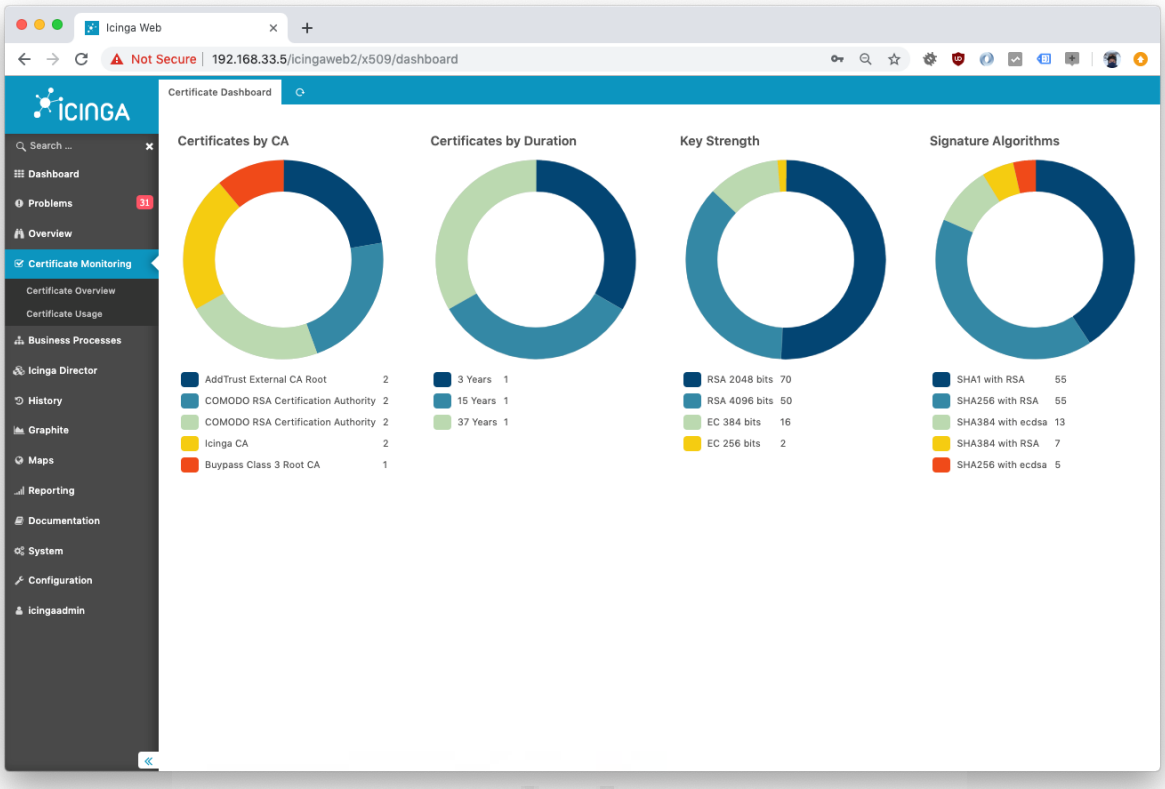



*Ilustración 8. Dashboard Icinga.*

## 1.3 PROBLEMA PLANTEADO

El monitoreo de servidores es vital para cualquier grupo de infraestructura tecnológica. No solo garantiza la continuidad operativa, sino también la optimización de los recursos en todo momento. Lamentablemente, las soluciones disponibles en el mercado tienen uno o más defectos. Por ejemplo, aplicaciones como Nagios o Zabbix son robustas, pero difíciles de configurar para los administradores que se inician. Al mismo tiempo, las opciones gratuitas a menudo son limitadas por la incompatibilidad con tecnologías modernas, como SNMPv3, que garantiza la autenticidad y la confidencialidad de las consultas. Por lo tanto, se necesita una solución con varios requisitos: seguridad, simplicidad, automatización, visualización efectiva y escalabilidad. Este proyecto proporciona una respuesta a estas necesidades al desarrollar un sistema de monitoreo en tiempo real basado en SNMPv3.

## 1.4 OBJETIVOS

El desarrollo de este trabajo de fin de grado persigue principalmente lograr diseñar e implementar un sistema de monitoreo de servidores en tiempo real a través del protocolo SNMPv3. Dicho sistema se enfocará en asegurar la seguridad, simplicidad y efectividad de la recolección y análisis de datos, así como rentabilizar la presentación de información relevante para la administración de infraestructuras tecnológicas.

**Objetivo General**

Desarrollar un sistema de monitoreo en tiempo real basado en SNMPv3, que permita a los administradores observar el estado y rendimiento de servidores con una interfaz gráfica amigable y funcional.

**Objetivos Específicos**

- Implementar la comunicación SNMPv3 segura: Configurar y asegurar la autenticación e integridad de consultas entre los sistemas a monitorear y el sistema de monitoreo.

- Diseñar una interfaz gráfica intuitiva: Crear dashboards diferenciados entre administradores y estándares con información clave mediante gráficas en tiempo real y reportes históricos.

- Automatizar procesos de monitoreo: Integrar funcionalidades para la generación diaria de reportes en formato PDF, gráficos de rendimiento, y eliminación automática de datos y reportes de más de tres días.

- Validar el monitoreo en tiempo real: Trabajar la recolección de datos cada minuto, guardar en base de datos relacional y enviar alarmas cada vez que los umbrales sean sobrepasados según su configuración.

## 1.5 ALCANCE DEL PROYECTO

El proyecto se centró en la implementación de un sistema de monitoreo en tiempo real de servidores basado en SNMPv3. Provee la posibilidad de controlar el uso de memoria, el espacio del disco en el servidor y el estado general del servidor. Incluye las siguientes funciones:

- **Monitoreo en tiempo real**: Se creó una aplicación que automatiza la recolección, el almacenamiento y el procesamiento de datos. El sistema realiza consultas periódicas al servidor sobre sus parámetros y detecta ciertos estados de este.

- **Visualización de gráficas**: Las consultas de SNMPv3 generan gráficas de cómo evolucionan los parámetros a lo largo de un minuto. Estas gráficas están disponibles en tiempo real y en reportes generados automáticamente cada día para el análisis posterior.

- **Reportes**: Los reportes diarios se generan en PDF e incluyen gráficas claras en función de los datos consultados con un análisis detallado de los mismos. Están disponibles por un máximo de tres días.

- **Automatización de purga de datos**: Para que el sistema sea efectivo y funcione eficientemente, se eliminan automáticamente los datos y reportes que excedan 3 días de antigüedad.

- **Control de acceso**: El sistema incluye roles de usuario, administrador y estándar. Además, la autenticación garantiza que un usuario en particular solo pueda acceder a la funcionalidad que le corresponde.

- **Escalamiento futuro**: Si bien hoy en día el sistema puede monitorear un solo servidor, está diseñado de tal manera que se pueda incrementar la complejidad y expandirlo para monitorear más servidores en un futuro.

Fuera del alcance:
- Integración con otros sistemas de transporte.
- Automatización de acciones correctivas basadas en la información del monitoreo, por ejemplo, reinicio de servicios.
- Soporte multilenguaje del web GUI.



## 1.6 PROTOCOLO SNMP

El Protocolo Simple de Gestión de Red (SNMP) es un protocolo estándar de red que se utiliza para monitorizar y gestionar dispositivos en una red IP. Es un protocolo ampliamente utilizado en la administración de sistemas y redes que, dada su simplicidad, efectividad y facilidad de implementación, permite recopilar información de dispositivos como routers, switches, servidores, estaciones de trabajo, entre otros.

El protocolo opera en una arquitectura cliente-servidor:

**El gestor (cliente):** es el sistema que centraliza las consultas y gestiona la red. En este proyecto, la máquina host Windows tiene el papel de un gestor.
**El agente (servidor):** es el software o dispositivo que proporciona datos al gestor. En este caso, la máquina virtual Ubuntu está configurada como agente SNMP.

**OID (Object Identifier):** Los **OID** son identificadores únicos que se utilizan para acceder a objetos específicos dentro de un dispositivo. Siguen una jerarquía estructurada en formato numérico (por ejemplo, .1.3.6.1.4.1.2021.4.6.0), que define su ubicación en la **MIB**.

**MIB (Management Information Base):** Una **MIB** es un archivo estructurado que define y organiza los OIDs accesibles en un dispositivo. Por ejemplo, para consultar la memoria utilizada en un servidor, se accede al OID correspondiente definido en la MIB

### 1.6.1 Versiones de SNMP

A lo largo del tiempo, el protocolo SNMP ha evolucionado con el objetivo de mejorar su capacidad y seguridad. Haciéndonos pasar a tener tres versiones principales:

- **SNMP v1**:
    - Introducida en 1988, es la versión original de SNMP.
    - **Características**: Simplicidad en la administración, pero con limitaciones de seguridad. Los datos se transmiten en texto plano, lo cual es una vulnerabilidad importante.
    - **Limitaciones**: Falta de autenticación y cifrado.

- **SNMP v2c**:
    - Introducida en 1993, con mejoras en el rendimiento y nuevas funcionalidades.
    - **Características**: Incorpora el comando *GetBulkRequest* para optimizar la transferencia de datos.
    - **Seguridad**: Implementa autenticación basada en “communities” (comunidades), aunque sin cifrado, lo que limita su seguridad. Estas comunidades controlan los permisos de acceso, permitiendo o denegando las solicitudes según el grupo al que pertenezca.

- **SNMP v3:**

  - Llegó en 2002, SNMP v3 mejoró significativamente en seguridad.
  - **Características:** Incluye autenticación de usuarios y cifrado de datos, permitiendo proteger la transmisión de la información.
  - **Seguridad:** Ahora presenta control de acceso, autenticación y privacidad. Permite la autenticación basada en MD5 o SHA y el cifrado con DES, haciendo que SNMP v3 se convierta en la versión más segura y recomendada para entornos críticos.

### 1.6.2 Funcionamiento

SNMP realiza tres funciones principales que le permiten administrar dispositivos de red:

- **Lectura de Información**:

  - **GetRequest**: El gestor envía esta solicitud para leer una variable específica del dispositivo.
  - **GetNextRequest**: Se utiliza para acceder al siguiente valor en la MIB, permitiendo una consulta secuencial de los datos.
  - **GetBulkRequest**: Introducido en SNMP v2c, esta operación permite solicitar grandes cantidades de datos en una única consulta, optimizando la comunicación en redes grandes.

- **Escritura de Información**:
  - **SetRequest**: Permite modificar una variable en el dispositivo, cambiando su configuración o activando una función. Esto es útil para aplicar configuraciones remotas.

- **Notificaciones de Eventos**:

  - **Traps**: Los traps son mensajes de alerta enviados por el agente al gestor cuando se produce un evento, como el fallo de un dispositivo o la caída de un enlace.

  - **InformRequest:** Similar a los traps, pero con confirmación de recepción, lo que asegura que el gestor ha recibido la alerta (disponible en SNMP v2c y v3).

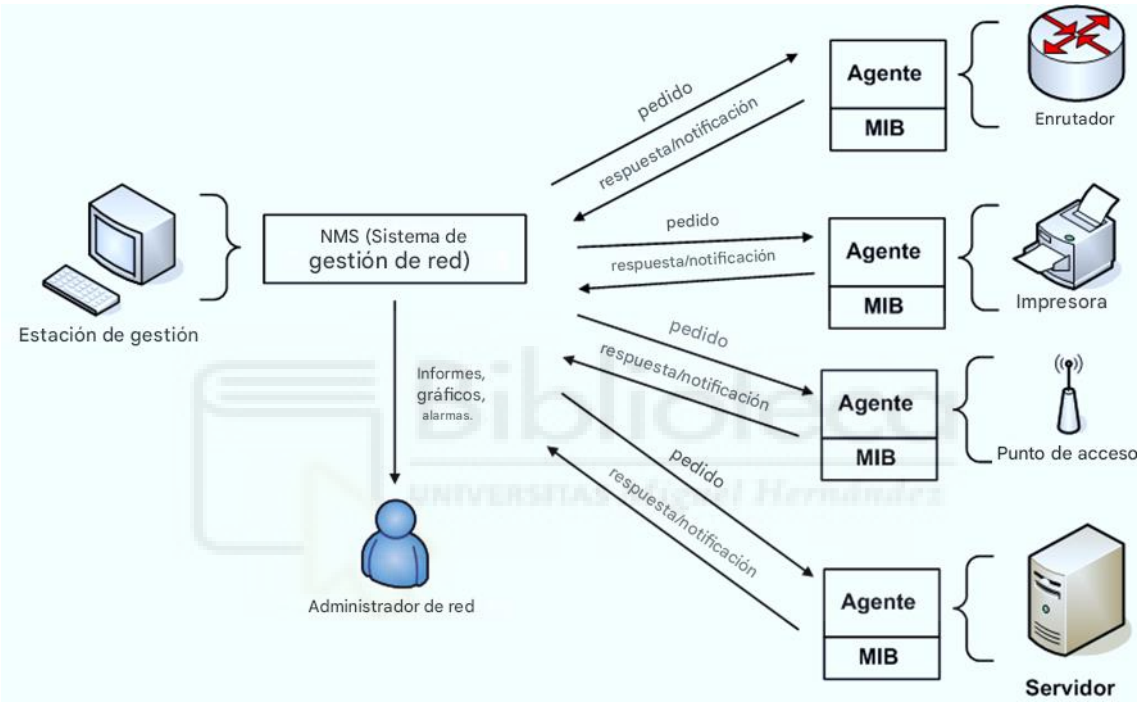



*Ilustración 9. Arquitectura SNMP.*

# 2. MATERIAL Y MÉTODOS

## 2.1 METODOLOGÍA APLICADA

Para el desarrollo del sistema de monitorización en tiempo real con el protocolo SNMPv3 se ha optado por un enfoque iterativo e incremental estructurado en fases. Esto permite el desarrollo del sistema por etapas, añadiendo las funcionalidades requeridas de manera progresiva y validando de manera continua los avances. Una de las razones por la que se implementa esta metodología es evitar los errores acumulativos en etapas avanzadas del proyecto que requieran cambios profundos en el sistema.

### 2.1.1 Análisis de Requerimientos

En esta primera etapa, se realizó un análisis detallado de los objetivos que buscamos cumplir con nuestro sistema de monitoreo. Estableciendo como principales:

- **Seguridad**: En este caso, se utiliza la implementación del protocolo SNMPv3 en modo 'authNoPriv' para poder garantizar una autenticación segura.
- **Accesibilidad**: En este apartado desarrollaremos un sistema accesible mediante navegador web, con interfaces diferenciadas para administradores y usuarios estándar.
- **Automatización**: Necesidad de un sistema que recolecte periódicamente datos mediante consultas SNMPv3 sin necesidad de intervención manual.
- **Optimización de recursos**: configuración de purgas automáticas periódicas para eliminar datos obsoletos y gráficos antiguos.
- **Escalabilidad**: Diseñar un sistema que permita futuras mejoras y optimizaciones sin modificaciones extremas.

### 2.1.2 Diseño de la arquitectura del Sistema

En esta fase se definió la estructura principal del sistema con sus componentes troncales y sus funciones:

- **Sistema de Monitoreo:** Es el encargado de realizar las consultas SNMP pertinentes y registrar los resultados en la base de datos SQL.
- **Frontend web:** Interfaz gráfica que da acceso a los distintos usuarios a través de navegador web.
- **Backend:** Se compone de los scripts de Python y las configuraciones PHP que gestionan la recolección, visualización, generación de informes y purga de datos.

### 2.1.3 Implementación y desarrollo

La implementación se dividió en cuatro etapas:

1- **Instalación y configuración de una máquina virtual**: Se descargó el programa de Oracle VirtualBox y se instaló y configuró un sistema operativo Ubuntu para que actúe como agente.
2- **Configuración del protocolo SNMPv3**: Se realizó la descarga, instalación y configuración de SNMPv3 en ambos sistemas (Gestor y agente).
3- **Desarrollo de funcionalidades clave**:
    a. Scripts de recolección y almacenamiento automático de datos.
    b. Generación de gráficos y reportes en formato pdf.
    c. Sistema de autenticación y control de acceso.
4- **Construcción de interfaz web:** Se desarrollo la interfaz web con las especificaciones y funcionalidades necesarias.



### 2.1.4 Validación y Pruebas

En cada etapa se hicieron las pruebas y comprobaciones correspondientes para verificar el correcto funcionamiento del sistema. Se comprobó las distintas partes clave:

- Consultas SNMP: Se comprobó mediante uso de Wireshark que se desarrollaba la comunicación de manera correcta entre gestor y agente. Se analizó los distintos paquetes SNMP enviados y recibidos, validando de manera satisfactoria los resultados.
- Gráficos y reportes: Se analizó y verificó los valores representados en la gráfica y se validó la correcta generación de los reportes en formato pdf.
- Interfaz Gráfica: Se comprobó la correcta accesibilidad y usabilidad de la página web por los usuarios con distintos roles.

### 2.1.5 Documentación y entrega

Finalmente, se documentó todos los aspectos técnicos del proyecto, incluyendo los scripts, configuraciones y esquemas con el objetivo de hacer que el sistema fuera reproducible y escalable.

Mediante esta metodología se logró desarrollar paso a paso un sistema robusto, seguro y escalable, manteniéndonos fieles a los objetivos iniciales que definimos al principio. Otro punto a destacar es que la metodología empleada nos ha ayudado a identificar problemas de manera más rápida y eficaz y poder así solucionarlos a tiempo y de manera óptima.

## 2.2 EQUIPOS Y HERRAMIENTAS

Para este sistema de monitoreo en tiempo real con SNMPv3, se ha implementado una infraestructura de dos terminales principales, y se ha hecho uso de varias herramientas de software para la captura, almacenamiento y visualización de los datos que hemos recopilado. En este apartado definiremos y detallaremos tanto los equipos físicos como las herramientas utilizadas, la configuración correspondiente a cada uno de ellos y la justificación de su uso.

### 2.2.1 Configuración de infraestructura

**EQUIPO 1: Servidor de Aplicaciones (Ubuntu)**

Para el servidor de aplicaciones se optó por un sistema virtualizado. Para ello, se utilizó el programa Oracle VM VirtualBox y en el cual se instaló el sistema operativo Ubuntu 24.04.1 LTS. Este sistema actuará como agente SNMP y será el responsable de proporcionar los datos de monitoreo como puede ser uso de CPU, memoria RAM disponible, entre otras. Responderá a las consultas pertinentes efectuadas desde el cliente. Se optó por una máquina virtual puesto que ofrecía mayor simpleza a la hora de operar, nos permitía tener todos los sistemas lógicos centralizados en un único sistema físico, lo que ayudaba en el manejo y el desarrollo del sistema y evitaba la necesidad de infraestructuras externas.

*Ilustración 10. Logo S.O Ubuntu.*

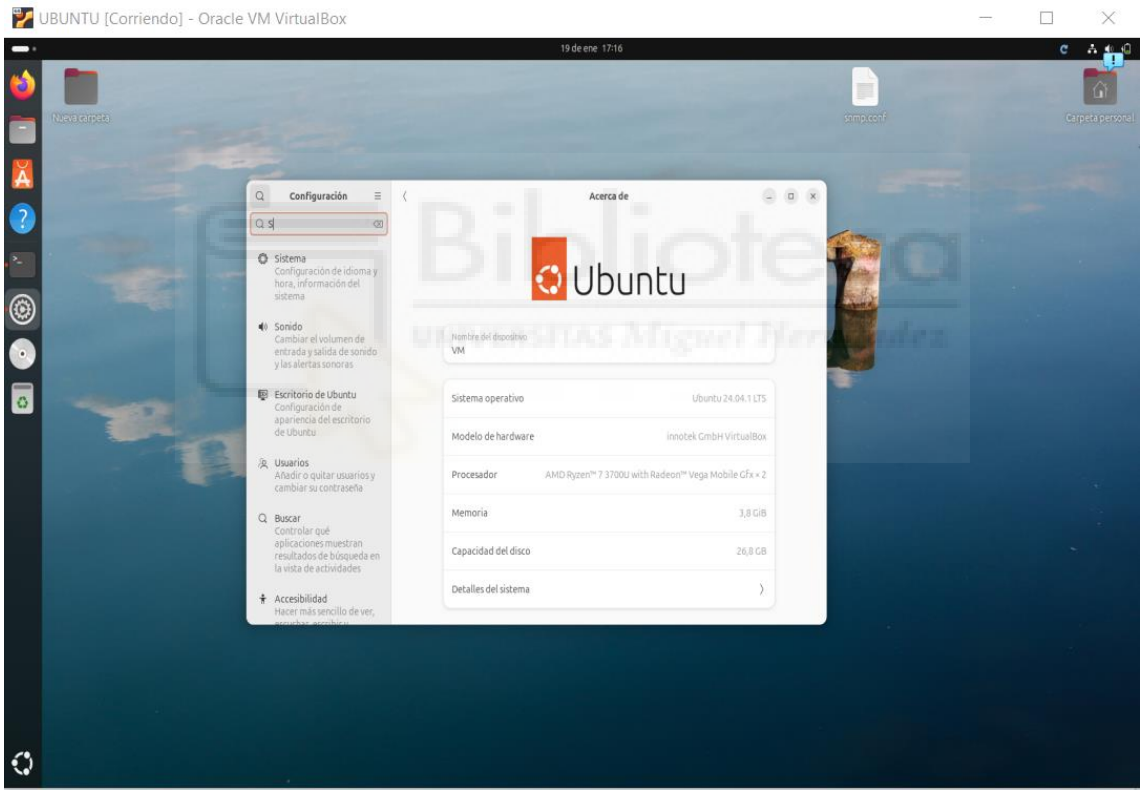



*Ilustración 11.Máquina Virtual Ubuntu.*

**Especificaciones:**

- **Sistema Operativo:** Ubuntu 24.04.1 LTS
- **IP asignada**: 192.168.18.168 (modo adaptador puente)
- **Software instalado:**
    - Agente SNMP (Net-SNMP)

- WireShark
- Herramientas de monitoreo adicionales como ‘htop’ y ‘psutil’


*Ilustración 12. Información del sistema de Ubuntu.*

Se instaló Net-SNMP en el sistema y se configuró como agente para responder a las consultas SNMPv3. Se creó un usuario con las siguientes características:

- usuario: ‘keldor’
- modo: ‘AuthNoPriv’, para ofrecer una capa de autenticación,
- Encriptación: ‘MD5’
- Clave: ‘12345678’

La configuración de red que se implementó en el sistema Ubuntu fue el de **adaptador puente** directamente desde la configuración del programa de Oracle VirtualBox. Esto permitió que la maquina estuviese dentro de la red local facilitando así la comunicación con el cliente.

**EQUIPO 2: Sistema de Monitoreo y Gestor (Windows)**

El equipo físico central y que actúa como cliente de monitoreo es un portátil Huawei D15 con un sistema operativo Windows 10. Este se encarga de realizar las consultas SNMPv3 al servidor de aplicaciones (agente), almacenar la información en la base de datos y proporcionar una interfaz web para la visualización de los datos y todas las operaciones intermedias.

**Especificaciones:**

- **Sistema operativo:** Windows 10
- **IP asignada:** 192.168.18.128
- **Software instalado:**
    - Wireshark (para captura de tráfico SNMP)
    - Apache (para servir la interfaz web)
    - MySQL (base de datos)
    - Python (para la ejecución de los scripts de automatización)
    - Cliente SNMP para consultas al servidor

Para la comprobación de la correcta comunicación y validación de datos entre los sistemas se ha utilizado Wireshark como programa para capturar los paquetes enviados y recibidos entre el gestor y el agente.

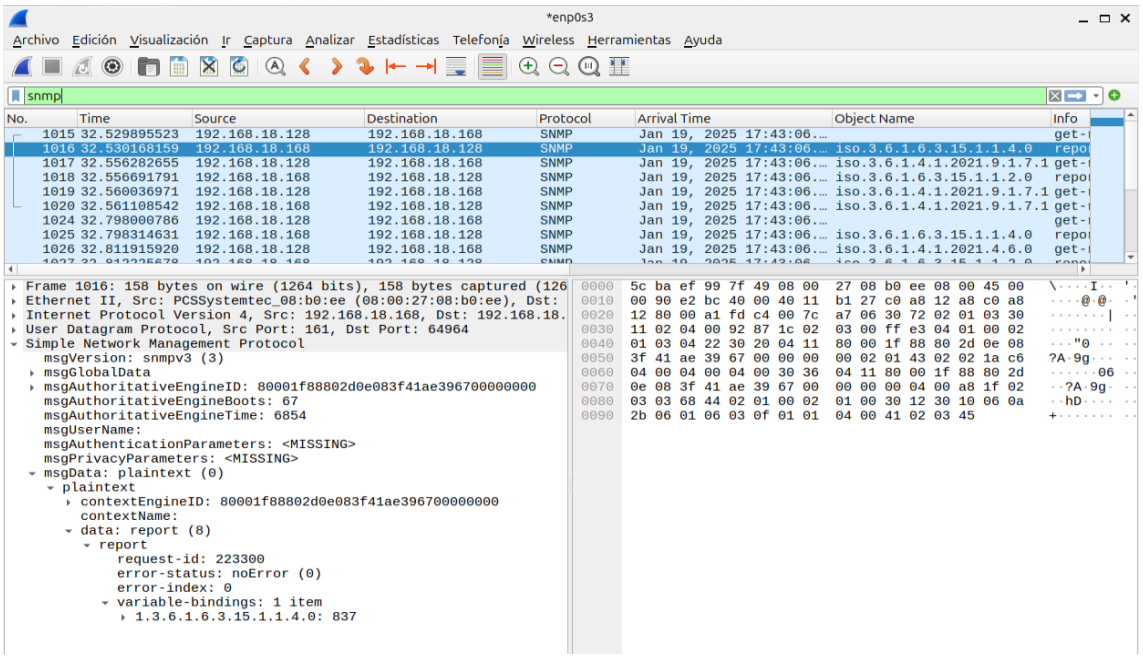



*Ilustración 13. Captura de tráfico SNMP con WireShark.*

### 2.2.2 Herramientas de Software Utilizadas

A la hora de implementar el sistema de monitoreo se ha hecho uso de múltiples herramientas de software, cada una tenía una función específica dentro del sistema. A continuación, se definen todas:

- **Oracle VM VirtualBox:** Permite la virtualización del sistema Ubuntu que actúa como agente SNMP. Su elección se debe a su facilidad de uso y compatibilidad con diferentes sistemas operativos. 【1】
- **Wireshark:** Herramienta de captura de paquetes de red utilizada para analizar la comunicación SNMP entre el cliente y el servidor, verificando la correcta transmisión de los datos.【2】
- **Apache:** Servidor web utilizado para alojar la interfaz web de monitoreo, permitiendo la consulta de los datos a través de una plataforma accesible.【4】
- **MySQL:** Sistema de gestión de bases de datos en el que se almacenan los valores obtenidos mediante SNMP, organizados en las tablas correspondientes para su posterior análisis.【5】
- **Python:** Lenguaje de programación utilizado para la implementación de los scripts monitor_snmp.py y generar_grafica.py, encargados de realizar las consultas SNMP y generar los reportes correspondientes.【6】.

- **PHP:** Lenguaje de programación utilizado en la implementación del sistema web para la gestión y visualización de los datos almacenados en la base de datos.【6】

```php
<?php
session_start();
include '../includes/db.php';

if (!isset($_SESSION['usuario']) || $_SESSION['rol'] !== 'Administrador') {
    header("Location: ../login.php");
    exit;
}

// Manejo de actualización de OID
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['editar_oid'])) {

    $oid = $_POST['oid'];
    $nombre = $_POST['nombre'];
    $threshold = $_POST['threshold'];
    $intervalo = $_POST['intervalo'];

    $stmt = $conn->prepare("UPDATE oids_config SET  nombre = ?, threshold = ?, intervalo = ? WHERE oid = ?");
    $stmt->bind_param("sdis",$nombre, $threshold, $intervalo, $oid);
    $stmt->execute();
    exit;
}
```

*Ilustración 14. Código parcial PHP de gestion_oids.*

## 2.3 DESCRIPCIÓN DEL SISTEMA IMPLEMENTADO

La solución de monitoreo desarrollada consta de varios módulos interconectados que permiten recopilar, almacenar y visualizar en tiempo real las métricas de rendimiento de un servidor. La solución creada está destinada a proporcionar un control centralizado de los datos recuperados y una visualización intuitiva de los mismos. Los aspectos más destacados, así como el desarrollo y los diagramas técnicos que ponen de manifiesto la implementación de la solución, se elaboran más abajo.

### 2.3.1 Funcionalidades Principales

El sistema está compuesto por diversas funcionalidades que permiten una gestión eficiente del monitoreo. Estas funcionalidades están diseñadas para abordar los requerimientos planteados inicialmente y asegurar una operación fluida y efectiva.

#### Control de acceso

El sistema cuenta con un mecanismo de control de acceso basado en roles, existiendo dos tipos de usuarios categorizados:

- **Administradores:** quienes cuentan con pleno acceso a todas las funciones del sistema, también con capacidad de modificar configuraciones de ciertos campos y funcionalidades, como gestión de usuario, OIDs y reporte.

- **Usuarios Estándar:** con acceso exclusivamente a las consultas de datos monitoreados, es decir, consultas en tiempo real de las métricas disponibles.

Además, el sistema utiliza un hash seguro en el almacenamiento de contraseñas en la base de datos para las credenciales de autenticación de los usuarios, los cuales aseguran la contraseña del administrador. Adicionalmente, cuenta con sesiones para seguir la autenticidad del usuario y la persistencia de este, evitando acceso no autorizado.

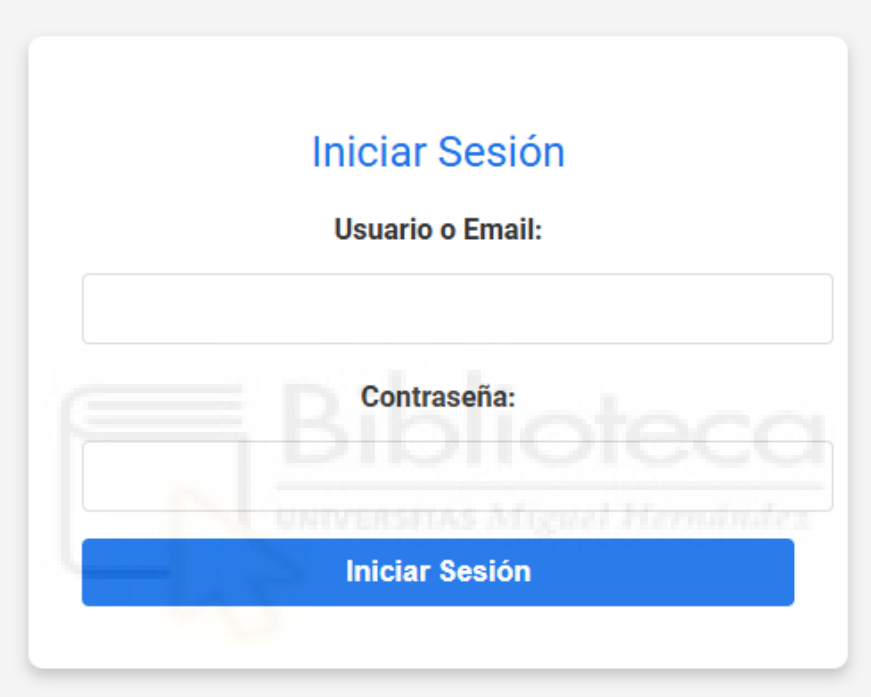



*Ilustración 15. Control de acceso al sistema.*

## Dashboard de Administración

El panel de administración, diseñado para los usuarios con rol de administrador, se ha desarrollado con el objetivo de ofrecer una interfaz todas las herramientas necesarias para un correcta gestión y monitorización de nuestro sistema. El diseño se ha desarrollado utilizando tecnologías web modernas, asegurando una experiencia de usuario optimizada y responsiva.

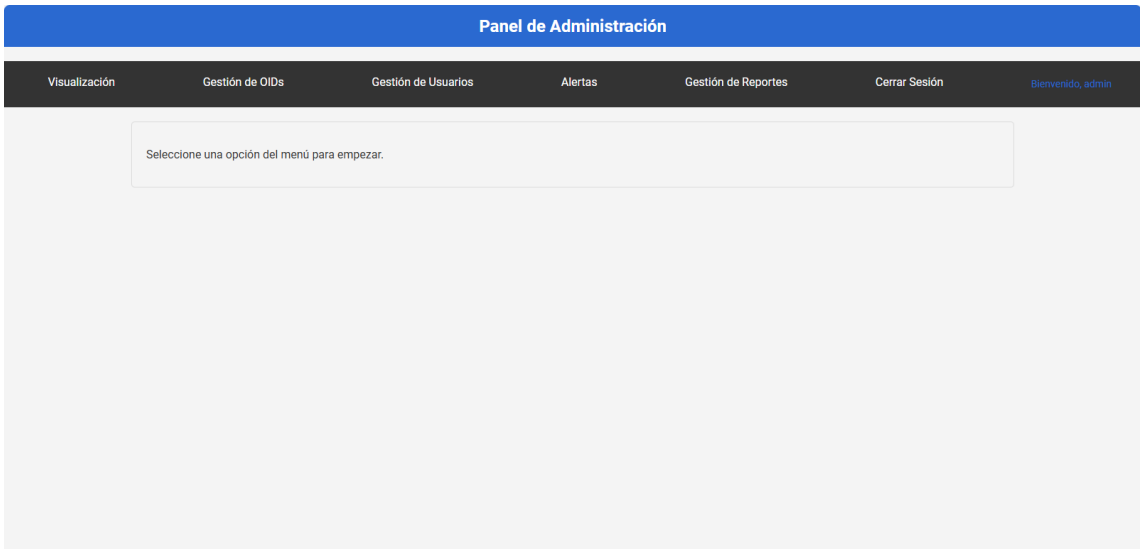



*Ilustración 16. Dashboard de Administrador.*

El dashboard de administrador presenta un menú con todas las funcionalidades requeridas para nuestro sistema a nivel de gestión:

**Visualización**

Para la visualización de las gráficas en tiempo real. Ofrece la opción de escoger la OID almacenadas en la tabla oid_config desde un desplegable y cargar la gráfica en tiempo real.

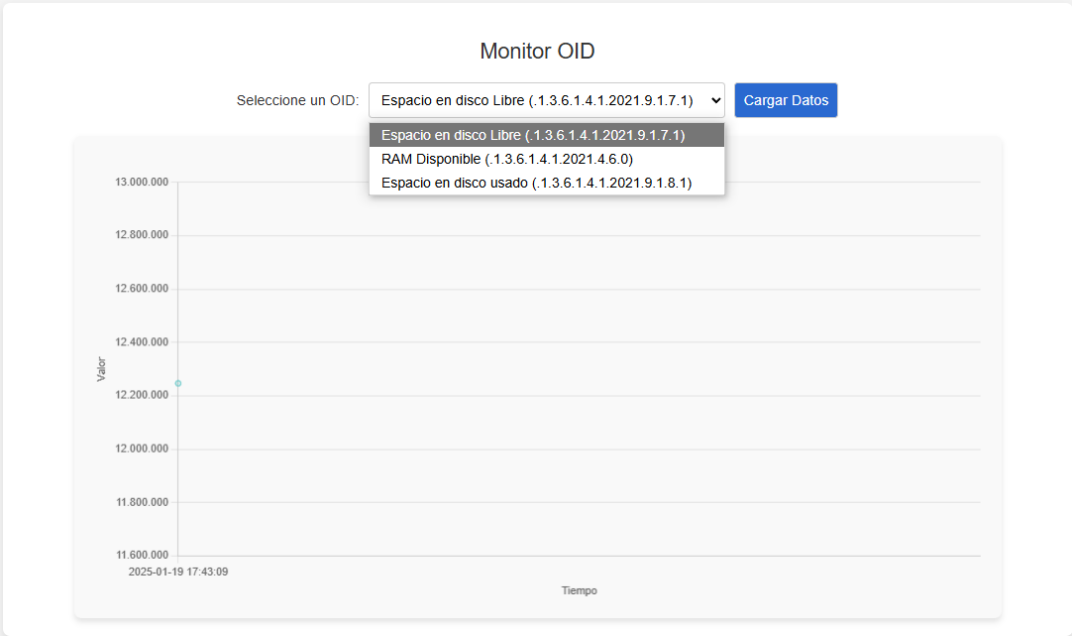



*Ilustración 17. Menú Visualización.*

### Gestión de OIDs

Para registrar, editar y eliminar OIDs de monitoreo. Los administradores pueden editar el nombre, el threshold y el intervalo de cada OID ya existente o añadir un OID nuevo.


*Ilustración 18. Menú Gestión de OIDs.*

- **Añadir nuevo OID:** Al seleccionar esta opción se abre una ventana para poder añadir los datos referentes a la nueva OID que queremos añadir. En este caso si se introduce una OID que ya está registrada nos avisará con una notificación de 'OID ya existente'

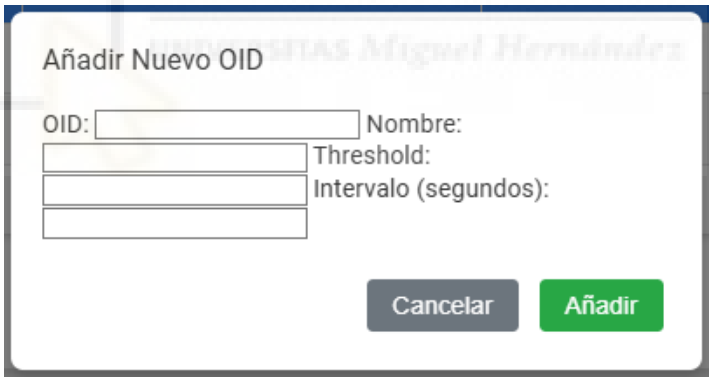



*Ilustración 19. Registro de nueva OID.*

- **Editar OID:** Al seleccionar la opción de editar, nos abrirá un desplegable igual que el anterior, pero con los datos de la OID que hemos seleccionado. Los podemos modificar y guardar a excepción del propio OID.

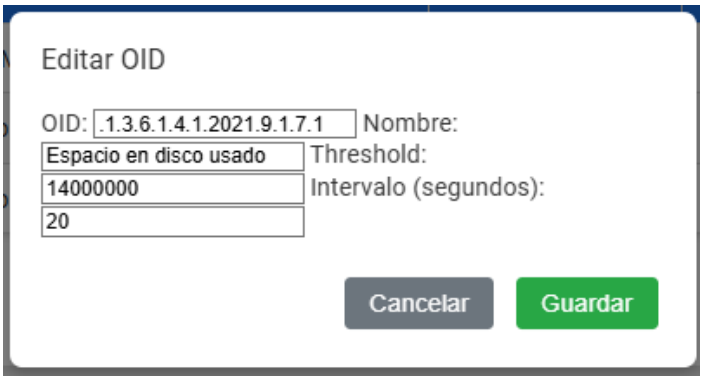



*Ilustración 20. Formulario para editar OID.*

- **Eliminar OID:** Y finalmente, si seleccionamos la opción de eliminar una OID nos aparecerá primero un desplegable de confirmación para evitar posibles eliminaciones accidentales.

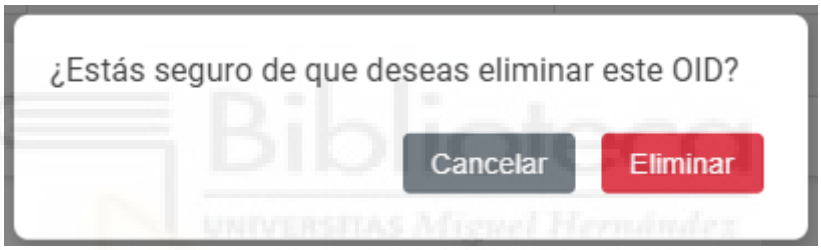



*Ilustración 21. Desplegable para confirmar eliminación de OID.*

**Gestión de Usuarios**

Esta sección será la encargada de crear, modificar y eliminar cuentas de usuario con roles especiales.

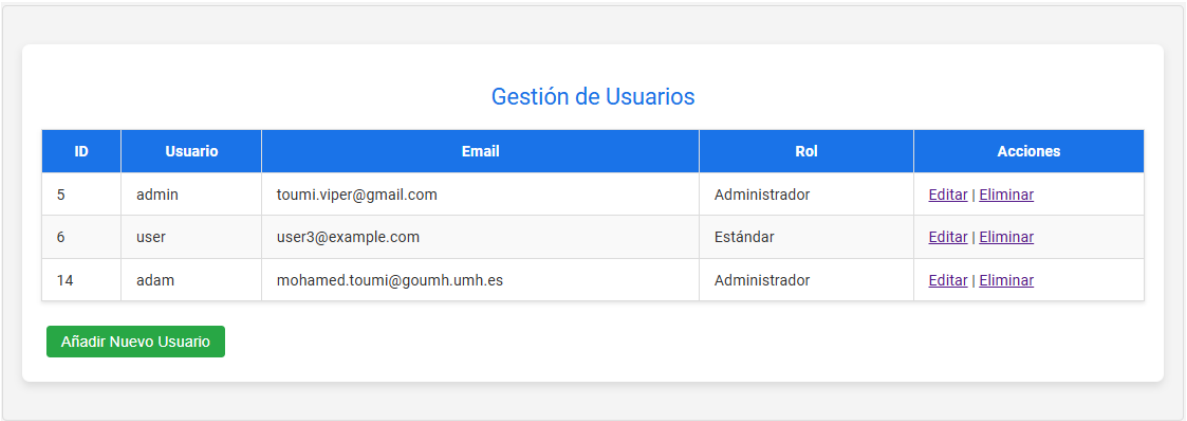



*Ilustración 22. menú Gestión de Usuarios.*

- **Añadir Nuevo Usuario:** Al seleccionar esta opción se abre una ventana con un formulario para introducir los datos del nuevo usuario junto a un desplegable para elegir el rol de este (Estándar o administrador).

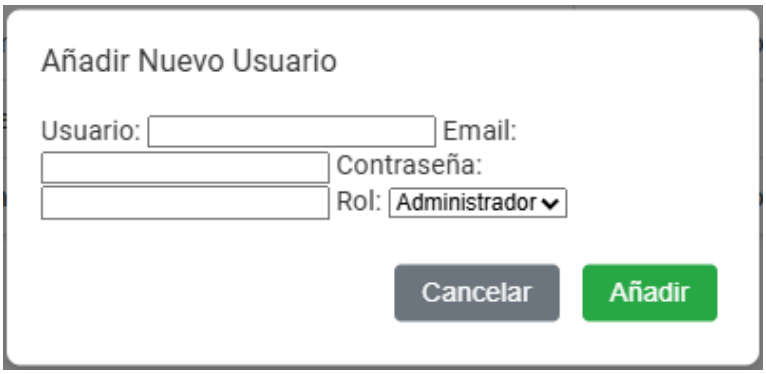



*Ilustración 23. Formulario para añadir nuevo usuario.*

- **Editar Usuario:** Al seleccionar la opción de editar, nos abrirá un desplegable igual que el anterior, pero con los datos del usuario que hemos seleccionado. Los podemos cambiar y guardar.

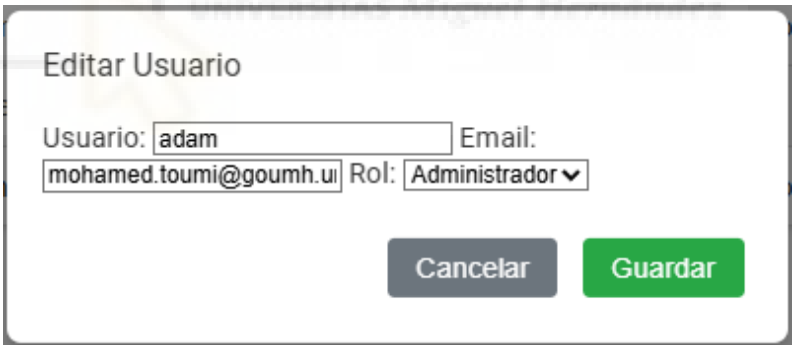



*Ilustración 24. Formulario para editar Usuario.*

- **Eliminar usuario:** Y finalmente, si seleccionamos la opción de eliminar usuario nos aparecerá primero un desplegable de confirmación para evitar posibles eliminaciones accidentales.

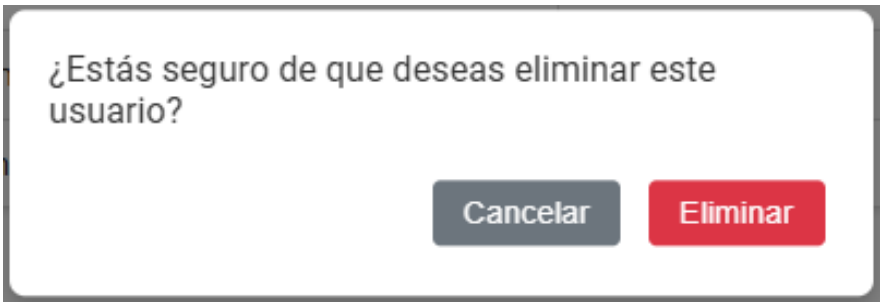



*Ilustración 25. Desplegable para confirmar eliminación de usuario.*

### Visualización de Alertas

Alertas generadas cuando un valor monitorizado supera el nivel de threshold asociado. Tenemos un desplegable que nos permite filtrar por OID y también otro desplegable que nos permite filtrar por fecha.


*Ilustración 26. Menú Alertas.*

### Gestión de Reportes

Esta sección nos ofrece una visualización junto con la descarga de los reportes generados en formato PDF. Nos permite filtrar por OID a través de un desplegable y también nos ofrece la posibilidad de una descarga masiva en la que se descargan todos los reportes comprimidos en un archivo ZIP.


*Ilustración 27. Menú Gestión de Reportes.*

### Cerrar Sesión

Y no menos importante, cabe destacar la opción en el menú de cerrar sesión que nos permite salir cuando hayamos finalizado la gestión.

## Dashboard de Usuario Estándar

El panel de usuario estándar está diseñado para brindar una experiencia simplificada y limitada de visualización de datos, permitiendo únicamente la selección y visualización de las gráficas en tiempo real de las distintas OIDs que hay configuradas en la tabla *oid_config* de la base de datos.


*Ilustración 28. Dashboard de Usuario Estándar.*

## Sistema de recolección de datos

La recolección de datos se lleva a cabo mediante **SNMPv3**, asegurando un alto nivel de seguridad en las consultas. Se han implementado un script en **Python**, que realiza consultas periódicas a los OIDs configurados en la base de datos. Los datos obtenidos se almacenan con marca de tiempo y se categorizan según el estado del sistema:

- **Activo:** Cuando el valor del OID se encuentra dentro de los límites establecidos.
- **Threshold Superado:** Si el valor excede el umbral definido por el administrador.
- **Inactivo:** Cuando el sistema monitoreado no responde.

| id | oid | valor | timestamp | estado |
|---|---|---|---|---|
| 1843 | .1.3.6.1.4.1.2021.9.1.7.1 | 12246700 | 2025-01-19 17:43:09 | Activo |
| 1844 | .1.3.6.1.4.1.2021.4.6.0 | 542308 | 2025-01-19 17:43:09 | Activo |
| 1845 | .1.3.6.1.4.1.2021.9.1.8.1 | 12050200 | 2025-01-19 17:43:10 | Activo |

*Ilustración 29. Ejemplo de datos almacenados en snmp_data.*

## Purga de Datos

El sistema implementa una rutina automatizada de purga de datos, asegurando la eficiencia en el almacenamiento. La purga se realiza diariamente mediante un script que elimina:

- Registros más antiguos a 3 días de la base de datos snmp_data.
- Reportes (archivos PDF y gráficas PNG) generados fuera del periodo de retención.

Este proceso garantiza que el sistema mantenga su rendimiento óptimo.

## Emails de Alerta

Como ya hemos mencionado anteriormente, se ha implementado un sistema de alertas que se activa cuando una OID supera el threshold que tiene asociado. El sistema evalúa esto cuando la OID pasa de estado ‘activo’ a ‘threshold superado’ y entonces notifica por email a todos los administradores registrados de esta incidencia.

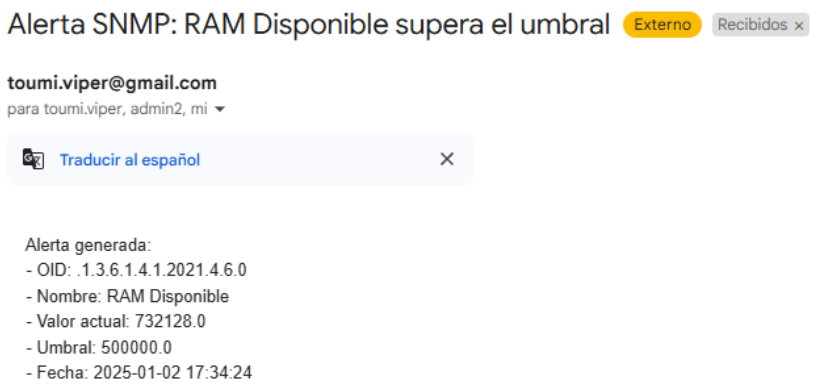



*Ilustración 30. Email de alerta de 'threshold superado'.*

### 2.3.2 Desarrollo del Sistema

Para el desarrollo correcto del sistema se ha seguido las siguientes etapas en orden que se detallan en profundidad a continuación:

1- **Instalación y configuración de Net-SNMP en Ubuntu.**

Una vez instalada la máquina virtual con el sistema Ubuntu, procedemos a instalar el programa Net-SNMP para el uso del protocolo. En este caso hemos procedido desde el terminal del sistema de manera secuencial para evitar errores y optimizar el proceso.

Empezamos actualizando la lista de paquetes del sistema:

*sudo apt-get update*

Instalamos Net-SNMP y las herramientas necesarias:

*sudo apt-get install snmp snmpd snmp-mibs-downloader -y*

Y descargamos las MIBs oficiales:

*sudo download-mibs*

Una vez descargado todos los componentes procedemos a la parte de la configuración. Y para ello lo primero que debemos hacer es detener el servicio snmpd para poder configurarlo:

*sudo systemctl stop snmpd*

Ahora procedemos a la creación de nuestro usuario con los datos que mencionamos en apartados anteriores:

*sudo net-snmp-create-v3-user -rw – a MD5 -A "12345678" “keldor” -l AuthNoPriv*

Una vez creado el usuario, procedemos a editar el archivo de configuración /etc/snmp/snmpd.conf:

*sudo nano /etc/snmp/snmpd.conf*

Configuramos el puerto 160 de UDP para que acepte la comunicación del protocolo SNMP:

*agentAddress udp:160*

Y añadimos el usuario creado con los privilegios:

*rwuser keldor*

Y finalmente, reiniciamos y permitimos el servicio snmpd:

*sudo systemctl restart snmpd*

*sudo systemctl enable snmpd*

Y con esto ya tendríamos SNMP configurado, activo y listo en el servidor. Para comprobación de un correcto funcionamiento procedemos a hacer una consulta local para comprobar que funciona correctamente y que devuelve el valor correspondiente antes de proceder con las consultas externas desde Windows.

Para ello utilizaremos el OID del espacio total en disco (.1.3.6.1.4.1.2021.9.1.6.1):

*snmpget -v3 -u keldor -l authNoPriv -a MD5 -A "12345678" localhost .1.3.6.1.4.1.2021.9.1.6.1*

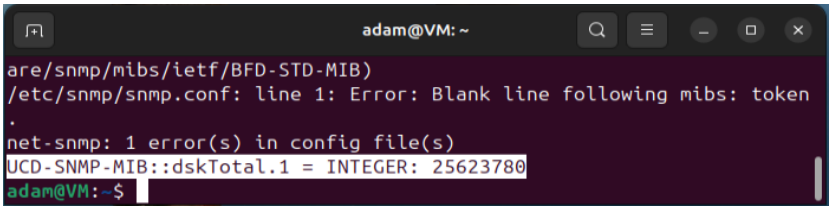



*Ilustración 31. Resultado de consulta SNMP.*

Con esto queda comprobado que funcionan las consultas en el agente a nivel local, lo que nos permite proseguir con la siguiente fase.

## 2- Instalación y configuración de Net-SNMP en Windows.

Una vez configurado el servicio SNMPv3 en el sistema Ubuntu, procedemos con la instalación de Net-SNMP en nuestro cliente (Windows). La primera etapa consiste en la descarga y la instalación desde la página web. Este proceso es más sencillo porque hay un archivo .exe para la instalación.

Procedemos con la instalación paso a paso y cuando ya lo tengamos instalado es importante añadir la ubicación del directorio bin del programa al PATH de las variables de entorno de Windows para poder ejecutar las consultas desde el CMD de Windows.

Y una vez que lo tenemos procedemos a comprobar el correcto funcionamiento mediante una consulta SNMPv3 dirigida al servidor Ubuntu. Utilizaremos la misma OID (Memoria en disco Total) que en la prueba anterior a fin de comprobar que los valores corresponden en ambos sistemas.

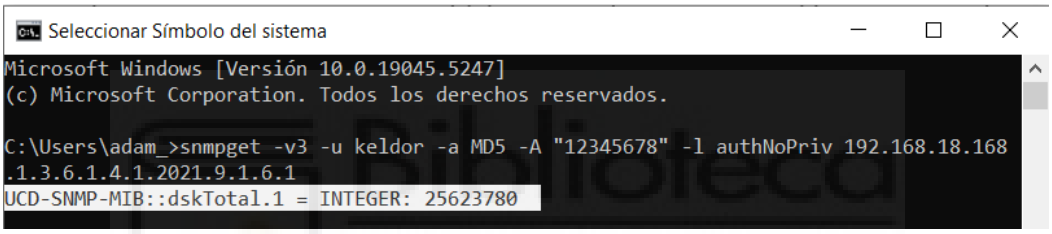



*Ilustración 32. Consulta SNMPv3 desde cliente (Windows).*

Y con esto queda comprobado que la comunicación y las consultas SNMP entre cliente (Windows) y el servidor (Ubuntu) funcionan de manera correcta.

## 3- Configuración de la base de datos en MySQL

Una vez tenemos comprobado satisfactoriamente la comunicación entre cliente y servidor procedemos al desarrollo de la base de datos. Para ello, como ya mencionamos anteriormente, utilizaremos el software de MySQL.

Creamos primero una base de datos que nombramos **sistema_snmp**. Esta estará compuesta por tres tablas principales que permiten almacenar información sobre los usuarios, la configuración de las OIDs a monitorear y los datos capturados en tiempo real.

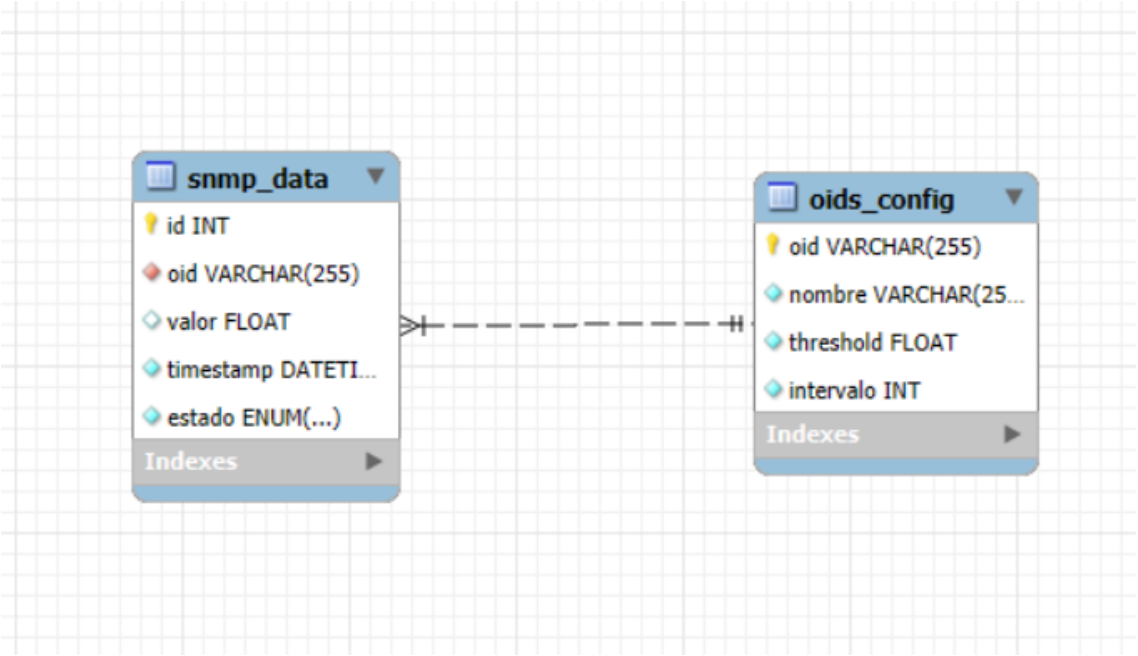



*Ilustración 33. Vista parcial de la base de datos MySQL. Relación 1:n.*

**Usuarios:**

Esta tabla almacena la información que tienen acceso al sistema diferenciando entre administradores y usuarios estándar y permitiendo así aplicar controles de acceso según el rol. Presenta la siguiente estructura:

| Campo | Tipo | Restricciones | Descripción |
|---|---|---|---|
| id | INT (AUTO_INCREMENT) | PRIMARY KEY | Identificador único del usuario. |
| usuario | VARCHAR(50) | NOT NULL, UNIQUE | Nombre de usuario único. |
| email | VARCHAR(100) | NOT NULL, UNIQUE | Correo electrónico del usuario. |
| password_hash | VARCHAR(255) | NOT NULL | Contraseña almacenada de manera segura (hash). |
| rol | ENUM('Administrador', 'Estándar') | NOT NULL | Determina los privilegios del usuario. |

*Tabla 1. Usuarios.*

Podemos apreciar que todas las contraseñas de los usuarios se almacenan usando algoritmos de hash seguros, como Bcrypt, para garantizar la seguridad de los datos de acceso.

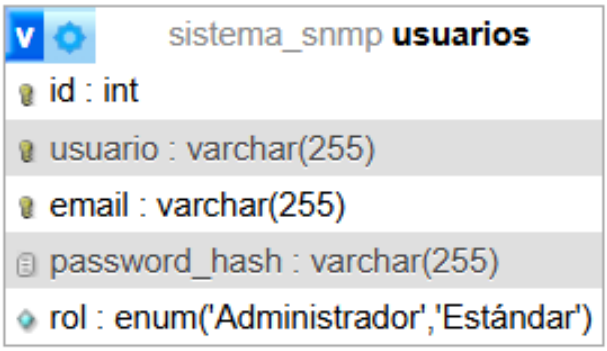



*Ilustración 34. Tabla usuarios en MySQL.*

**Oids_config:**

Esta tabla contiene la configuración de los OIDs (Object Identifiers) que serán monitoreados en el sistema. Esta tabla es fundamental, ya que define qué métricas del sistema se están observando, estableciendo umbrales para la generación de alertas. Presenta la siguiente estructura:

| Campo | Tipo | Restricciones | Descripción |
|---|---|---|---|
| oid | VARCHAR(255) | PRIMARY KEY, NOT NULL, | Identificador único SNMP del recurso a monitorear. |
| nombre | VARCHAR(255) | NOT NULL | Nombre descriptivo del OID. |
| threshold | INT | NOT NULL | Valor umbral que, si se supera, genera una alerta. |
| intervalo | INT | NOT NULL | Frecuencia en segundos de las consultas SNMP. |

*Tabla 2. oids_config.*

Esta tabla es la que utiliza el sistema de monitoreo para obtener las oids y sus respectivas configuraciones para realizar las consultas y almacenar los datos en la tabla snmp_data con sus respectivos valores.

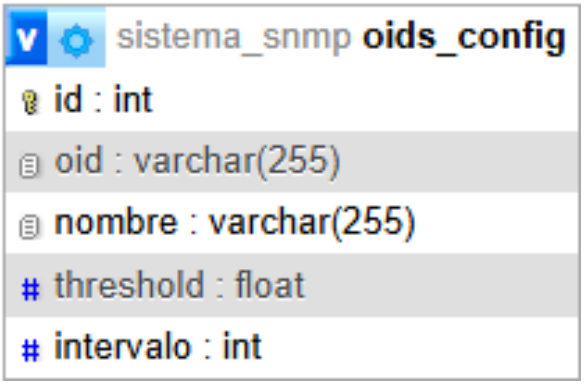



*Ilustración 35. Tabla oids_config en MySQL.*

**Snmp_data:**

Esta tabla es la encargada de almacenar los valores recolectados de los OIDs configurados en la tabla anterior. Cada registro incluye el valor obtenido, el estado del sistema en el momento de la consulta y la fecha/hora de captura y presenta la siguiente estructura:

| Campo | Tipo | Restricciones | Descripción |
|---|---|---|---|
| id | INT (AUTO_INCREMENT) | PRIMARY KEY | Identificador único del registro. |
| oid | VARCHAR(255) | FOREIGN KEY (oids_config.oid) | OID consultado en el monitoreo. |
| valor | BIGINT | NOT NULL | Valor capturado para el OID monitoreado. |
| timestamp | DATETIME | NOT NULL | Fecha y hora de la consulta SNMP. |
| estado | ENUM('Activo', 'Inactivo', 'Threshold Superado') | NOT NULL | Estado del recurso monitoreado. |

*Tabla 3. snmp_data.*

Esta será la tabla de datos que utilizaremos para la generación de gráficas y reportes. Como los registros son muchos y los datos almacenados puede ascender de manera muy elevada se ha decidido implementar un mecanismo automatizado de purgado de datos con más de tres días de antigüedad para optimizar el espacio de almacenamiento.

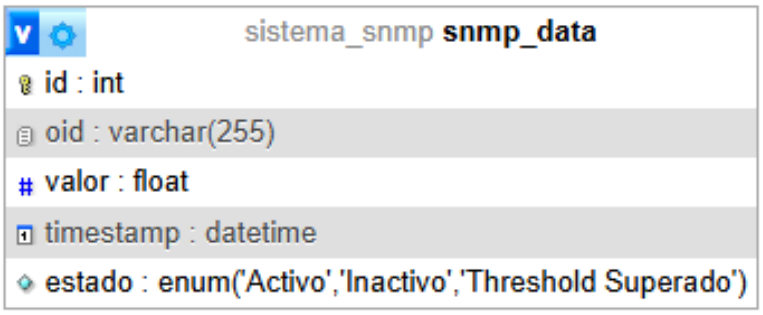



*Ilustración 36. Tabla snmp_data en MySQL.*

**4- Desarrollo del script 'monitor_snmp.py'.**

Este script tiene la función principal de la monitorización en tiempo real de un servidor utilizando el protocolo SNMPv3. Está diseñado para realizar consultas periódicas a las OIDs configuradas en la tabla 'oids_config', registrar los valores obtenidos en la tabla 'snmp_data' de la base de datos MySQL y generar alertas automáticas cuando los valores superen los umbrales establecidos. Se ha desarrollado para una ejecución continúa programada en el sistema.

El código se estructura en varias secciones clave, cada una de las cuales cumple un propósito especifico dentro del flujo de monitoreo. A continuación, detallaremos la funcionalidad de cada parte:

**importación de librerías y configuración del entorno:**

El script comienza importando las librerías necesarias, incluyendo:

- **sys**: Permite modificar rutas de importación de módulos.
- **datetime**: Manejo de fechas y horas.
- **mysql.connector**: Conexión con la base de datos MySQL.
- **pysnmp.hlapi**: Realización de consultas SNMPv3.
- **schedule**: Automatización de tareas periódicas.
- **smtplib y email**: Envío de alertas por correo electrónico.

**Conexión a la base de datos:**

Implementamos una función específica '*conectar_mysql()*' para establecer la conexión con la base de datos MySQL. Hemos configurado la conexión con nuestras credenciales de acceso y hemos ajustado el script para que en caso de que fallase la conexión a la base de datos se notifique y el programa finalice su ejecución.

```
# Conexión a la base de datos
def conectar_mysql():
    try:
        conexion = mysql.connector.connect(
            host="localhost",
            user="root",
            password="Cocktail-900",
            database="sistema_snmp"
        )
        print("Conexión exitosa a MySQL")
        return conexion
    except mysql.connector.Error as e:
        print(f"Error de conexión a MySQL: {e}")
        sys.exit(1)
```

*Ilustración 37. Función 'conectar_mysql()'.*



**Obtención de correos electrónicos de los administradores:**

Se ha definido una función específica para la obtención de los correos electrónicos de los administradores automáticamente desde la tabla usuarios de la base de datos para la posible notificación posterior de alertas.

```
# Función para obtener los correos electrónicos de los administradores
def obtener_administradores():
    try:
        conexion = conectar_mysql()
        cursor = conexion.cursor(dictionary=True)
        cursor.execute("SELECT email FROM usuarios WHERE rol = 'Administrador'")
        administradores = [row['email'] for row in cursor.fetchall()]
        cursor.close()
        conexion.close()
        return administradores
    except mysql.connector.Error as e:
        print(f"Error al obtener administradores: {e}")
        return []
```

*Ilustración 38. Función 'obtener_administradores()'.*

**Envío de alertas por correo electrónico:**

Esta función es la encargada de enviar un correo electrónico con la alerta correspondiente cuando una OID supera el umbral por primera vez. El remitente utilizado actualmente es un correo personal y el mensaje contiene todos los datos necesarios de la OID correspondiente tal y como se puede ver en la *ilustración 22*.

```python
# Función para enviar alertas por correo electrónico
def enviar_alerta(oid, nombre, valor, threshold):
    administradores = obtener_administradores()
    if not administradores:
        print("No hay administradores registrados para enviar la alerta.")
        return

    remitente = "toumi.viper@gmail.com"
    contraseña = "rpcg oazb fwdu veto"  # Contraseña de aplicación de Gmail
    destinatarios = ", ".join(administradores)
    asunto = f"Alerta SNMP: {nombre} supera el umbral"

    cuerpo = f"""
    Alerta generada:
    - OID: {oid}
    - Nombre: {nombre}
    - Valor actual: {valor}
    - Umbral: {threshold}
    - Fecha: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}
    """

    mensaje = MIMEMultipart()
    mensaje['From'] = remitente
    mensaje['To'] = destinatarios
    mensaje['Subject'] = asunto
    mensaje.attach(MIMEText(cuerpo, 'plain'))

    try:
        servidor = smtplib.SMTP('smtp.gmail.com', 587)
        servidor.starttls()  # Inicia el cifrado TLS
        servidor.login(remitente, contraseña)  # Autenticación
        servidor.send_message(mensaje)
        servidor.quit()
        print(f"Alerta enviada a: {destinatarios}")
    except Exception as e:
        print(f"Error al enviar alerta: {e}")
```

*Ilustración 39. Función 'enviar_alerta()'*

Esta función contiene los ajustes necesarios para notificar al remitente de si se ha enviado la alerta correctamente al destinatario o ha habido un error, como por ejemplo, cuando el correo de destinatario no existe.

**Consulta SNMPv3 a los dispositivos:**

Para obtener los valores de los OIDs, el script utiliza la función *snmp_get()* que recibe como argumento la OID correspondiente y realiza una consulta SNMPv3 con las credenciales prestablecidas y la IP correspondiente del agente Ubuntu.

```
# Obtener valor SNMP
def snmp_get(oid):
    try:
        iterator = getCmd(
            SnmpEngine(),
            UsmUserData('keldor', '12345678', authProtocol=usmHMACMD5AuthProtocol),
            UdpTransportTarget(('192.168.18.168', 161)),
            ContextData(),
            ObjectType(ObjectIdentity(oid))
        )
        errorIndication, errorStatus, errorIndex, varBinds = next(iterator)
        if errorIndication:
            print(f"Error SNMP: {errorIndication}")
            return None
        elif errorStatus:
            print(f"Error Status: {errorStatus.prettyPrint()}")
            return None
        else:
            for varBind in varBinds:
                return float(varBind[1])  # Devuelve el valor obtenido
    except Exception as e:
        print(f"Excepción SNMP: {e}")
        return None
```

*Ilustración 40. Función 'snmp_get(oid)'.*

**Determinación del estado de valores:**

La función *determinar_estado()* compara los valores obtenidos con los umbrales establecidos y define el estado.

```
# Determinar estado del OID
def determinar_estado(valor, threshold):
    if valor > threshold:
        return 'Threshold Superado'
    return 'Activo'
```

*Ilustración 41. Función 'determinar_estado()'.*

**Monitoreo de OIDs:**

La función principal *monitor_snmp()* obtiene las OIDs desde la base de datos, consulta sus valores mediante SNMP y almacena los resultados.

```
# Monitorear OIDs
def monitor_snmp():
    print("Iniciando monitorización de OIDs desde la base de datos...")
    conexion = conectar_mysql()
    cursor = conexion.cursor(dictionary=True)
    cursor.execute("SELECT oid, nombre, threshold, intervalo FROM oids_config")
    oids = cursor.fetchall()

    for oid_entry in oids:
        oid = oid_entry['oid']
        nombre = oid_entry['nombre']
        threshold = oid_entry['threshold']
        valor = snmp_get(oid)
        timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

        if valor is not None:
            estado = determinar_estado(valor, threshold)
            # Obtener último estado
            cursor.execute("SELECT estado FROM snmp_data WHERE oid = %s ORDER BY timestamp DESC LIMIT 1", (oid,))
            resultado = cursor.fetchone()
            ultimo_estado = resultado['estado'] if resultado else None

            # Enviar alerta solo si cambia a "Threshold Superado"
            if estado == "Threshold Superado" and ultimo_estado != "Threshold Superado":
                enviar_alerta(oid, nombre, valor, threshold)

            cursor.execute(
                "INSERT INTO snmp_data (oid, valor, timestamp, estado) VALUES (%s, %s, %s, %s)",
                (oid, valor, timestamp, estado)
            )
            print(f"[{timestamp}] {nombre} ({oid}): {valor} - Estado: {estado}")
        else:
            # No hay respuesta SNMP, registrar como Inactivo
            cursor.execute(
                "INSERT INTO snmp_data (oid, valor, timestamp, estado) VALUES (%s, NULL, %s, %s)",
                (oid, timestamp, 'Inactivo')
            )
            print(f"[{timestamp}] {nombre} ({oid}): No responde - Estado: Inactivo")
        conexion.commit()
    cursor.close()
    conexion.close()
```

*Ilustración 42. Función 'monitor_snmp()'.*

5- **Desarrollo script generar_grafica().**

El script *generar_grafica.py* tiene como propósito la generación automatizada de reportes visuales en formato PDF basados en los datos obtenidos a través del monitoreo SNMP. Este proceso permite una fácil visualización de la información, facilitando la interpretación del estado de los sistemas monitoreados.

Las funcionalidades principales del script incluyen:

- Conexión a la base de datos para la obtención de datos históricos de SNMP.
- Generación de gráficos de barras representando los valores de las OIDs monitoreadas.
- Creación de reportes en formato PDF con las gráficas generadas.
- Purga automática de datos antiguos para optimizar el almacenamiento.
- Registro detallado de eventos en un archivo de log.

El script se divide en varias secciones que abarcan diferentes funcionalidades, la cuales, describiremos a continuación:

**Importación de librerías necesarias y configuración inicial**

Empezamos importando las librerías que necesitamos para el resto del script, las cuales definimos a continuación:

- **os**: Para la manipulación de archivos y directorios.
- **mysql.connector**: Para la conexión con la base de datos MySQL.
- **matplotlib.pyplot:** Para la generación de gráficos.
- **datetime y timedelta:** Para el manejo de fechas y tiempos.
- **fpdf:** Para la generación de archivos PDF.

Y como complemento adicional definimos la configuración de la base de datos

```
# Configuración de conexión a la base de datos
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'Cocktail-900',
    'database': 'sistema_snmp'
}
```

*Ilustración 43. Configuración 'db_config'.*

**Parámetros de configuración**

Se establecen las rutas de almacenamiento de los reportes y los archivos log y se crean en caso de que no existan:

```
# Directorios
reports_dir = 'C:/xampp/htdocs/snmp_web/reports'
logs_dir = 'C:/xampp/htdocs/snmp_web/logs'
log_file = os.path.join(logs_dir, 'reporte_automatizado.log')

# Crear directorios si no existen
os.makedirs(reports_dir, exist_ok=True)
os.makedirs(logs_dir, exist_ok=True)
```

*Ilustración 44. Directorios de almacenamiento.*

### Registro de eventos

La función *escribir_log()* registra los eventos en un archivo de log para llevar acabo un seguimiento de las operaciones realizadas:

```
# Función para escribir en el log
def escribir_log(mensaje):
    with open(log_file, 'a') as log:
        log.write(f"[{datetime.now()}] {mensaje}\n")
```

*Ilustración 45. función 'escribir_log()'.*

### Conexión a la base de datos

El script establece una conexión con la base de datos para obtener datos relevantes para la generación de reportes:

```
# Conectar a la base de datos
try:
    conexion = mysql.connector.connect(**db_config)
    cursor = conexion.cursor(dictionary=True)
    escribir_log("Conexión a la base de datos exitosa.")
except Exception as e:
    escribir_log(f"Error conectando a la base de datos: {e}")
    exit()
```

*Ilustración 46. Conexión a la base de datos.*

### Purga de datos antiguos de la base de datos

En este mismo script se implementan dos funciones para evitar la acumulación innecesaria de información mediante la eliminación de registros antiguos de la base de datos y los archivos generados hace más de tres días.

```
# Purga de datos antiguos en la tabla snmp_data
def purgar_datos_antiguos():
    try:
        fecha_limite = (datetime.now() - timedelta(days=4)).strftime("%Y-%m-%d")
        cursor.execute("DELETE FROM snmp_data WHERE DATE(timestamp) < %s", (fecha_limite,))
        conexion.commit()
        escribir_log(f"Datos anteriores a {fecha_limite} eliminados de la tabla snmp_data.")
    except Exception as e:
        escribir_log(f"Error eliminando datos antiguos: {e}")
```

*Ilustración 47. Función 'purgar_datos_antiguos()'.*

```python
# Purga de archivos antiguos en el directorio reports
def purgar_archivos_antiguos():
    try:
        fecha_limite = datetime.now() - timedelta(days=4)
        for archivo in os.listdir(reports_dir):
            if archivo.endswith(".pdf") or archivo.endswith(".png"):
                partes = archivo.split('_')
                fecha_archivo = partes[-1].replace('.pdf', '').replace('.png', '')
                try:
                    fecha_archivo_dt = datetime.strptime(fecha_archivo, "%Y-%m-%d")
                    if fecha_archivo_dt < fecha_limite:
                        os.remove(os.path.join(reports_dir, archivo))
                        escribir_log(f"Archivo eliminado: {archivo}")
                except ValueError:
                    escribir_log(f"Formato de fecha no válido en el archivo: {archivo}")
    except Exception as e:
        escribir_log(f"Error eliminando archivos antiguos: {e}")
```

*Ilustración 48. Función 'purgar_archivos_antiguos()'.*

### Obtención de OIDs

Esta función se encarga de obtener todos los OIDs configurados en la base de datos para generar reportes de este e informa si hubiese un problema en el proceso.

```python
# Obtener todas las OIDs configuradas
def obtener_oids():
    try:
        cursor.execute("SELECT oid, nombre FROM oids_config")
        return cursor.fetchall()
    except Exception as e:
        escribir_log(f"Error obteniendo las OIDs: {e}")
        return []
```

*Ilustración 49. Función 'obtener:oids()'.*

### Generación de gráficos

La función generar_grafico() crea gráficos de barras con los datos obtenidos, diferenciado los estados mediante colores específicos.

```python
# Generar gráfico de barras para una OID específica
def generar_grafico(oid, nombre_oid, fecha, datos):
    try:
        if not datos:
            escribir_log(f"No hay datos para la OID {oid} en el día {fecha}.")
            return

        # Extraer datos relevantes
        tiempos = [dato['timestamp'].strftime('%H:%M') for dato in datos]
        valores = [dato['valor'] for dato in datos]
        estados = [dato['estado'] for dato in datos]

        # Colores por estado
        colores = {
            'Activo': 'blue',
            'Inactivo': 'black',
            'Threshold Superado': 'red'
        }

        # Generar gráfico
        plt.figure(figsize=(12, 6))
        plt.bar(
            tiempos,
            valores,
            color=[colores[estado] for estado in estados],
            width=0.6
        )
        plt.title(f"Reporte de OID {nombre_oid} - {fecha}", fontsize=16)
        plt.xlabel("Hora del día", fontsize=12)
        plt.ylabel("Valor", fontsize=12)
        plt.xticks(rotation=45, fontsize=8)

        # Establecer el límite del eje Y según la tabla LIMITES_OID
        limite_y = LIMITES_OID.get(oid, max(valores) * 1.1 if valores else 1)
        plt.ylim(0, limite_y)
        plt.grid(axis='y', linestyle='--', linewidth=0.7, alpha=0.7)

        # Leyenda estática
        plt.legend(
            [
                plt.Line2D([0], [0], color=colores['Activo'], lw=4),
                plt.Line2D([0], [0], color=colores['Inactivo'], lw=4),
                plt.Line2D([0], [0], color=colores['Threshold Superado'], lw=4)
            ],
            ['Activo', 'Inactivo', 'Threshold Superado'],
            title="Estado",
            loc='upper right'
        )

        # Guardar gráfica
        grafica_path = os.path.join(reports_dir, f"grafica_{oid}_{fecha}.png")
        pdf_path = os.path.join(reports_dir, f"reporte_{oid}_{fecha}.pdf")
        plt.savefig(grafica_path)
        escribir_log(f"Gráfica guardada en {grafica_path}")

        # Generar PDF
        pdf = FPDF()
        pdf.add_page()
        pdf.set_font("Arial", size=12)
        pdf.cell(200, 10, txt=f"Reporte de OID {nombre_oid} - {fecha}", ln=True, align='C')
        pdf.ln(10)
        pdf.image(grafica_path, x=10, y=30, w=190)
        pdf.output(pdf_path)
        escribir_log(f"PDF generado en {pdf_path}")

        plt.close()

    except Exception as e:
        escribir_log(f"Error generando gráfico para OID {oid}: {e}")
```

*Ilustración 50. Función 'generar_grafico()'.*

**Ejecución principal del script**

El script se ha programado con el programador de tareas de Windows para que se ejecute diariamente. Y a continuación se presenta el proceso principal del script:

```
# Proceso principal
def main():
    # Purga de datos y archivos antiguos
    purgar_datos_antiguos()
    purgar_archivos_antiguos()

    fecha_actual = (datetime.now() - timedelta(days=1)).strftime("%Y-%m-%d")
    oids = obtener_oids()

    if not oids:
        escribir_log("No se encontraron OIDs configuradas.")
        return

    for oid_data in oids:
        cursor.execute("""
            SELECT valor, timestamp, estado
            FROM snmp_data
            WHERE oid = %s AND DATE(timestamp) = %s
            ORDER BY timestamp
        """, (oid_data['oid'], fecha_actual))
        resultados = cursor.fetchall()
        generar_grafico(oid_data['oid'], oid_data['nombre'], fecha_actual, resultados)

    escribir_log("Proceso de generación de reportes completado.")

if __name__ == "__main__":
    main()
```

*Ilustración 51. Función main de 'generar_grafica'.*

### 2.3.3 Diagramas y Esquemas Técnicos

El desarrollo del sistema de monitoreo ha requerido la creación de diagramas técnicos que faciliten la comprensión de la estructura del sistema, sus interacciones y el flujo de información entre los diferentes componentes. A continuación, se presentan los principales diagramas utilizados en el diseño del sistema.

**Diagrama de Arquitectura del sistema**

Este sistema se basa en un enfoque cliente-servidor, que se estructura para permitir la captura, el almacenamiento y la visualización continua y en tiempo real de la información bajo monitoreo, y está compuesto por los siguientes elementos clave:

- **Cliente SNMP**: este es un cliente descargado en un sistema Windows que envía mensajes SNMP a la MV de Ubuntu para obtener las métricas actuales de esa máquina.
- **Servidor SNMP**: un agente en una instancia de Ubuntu responde a las peticiones del cliente con los datos medidos.
- **Base de datos MySQL**: contiene los datos medidos, el umbral preestablecido y un registro del historial.
- **Servidor web Apache**: un navegador ya instalado en Ubuntu que aloja una UI para la gestión de servicios y la visualización de la información.
- **Scripts Python**: realiza scripts que automatizan las capacidades de recolección y búsquedas de datos, higiene de datos y generación de reportes.

**Esquema de Flujo de Monitoreo SNMP**

El flujo del proceso de monitoreo SNMP es coherente y sigue el diseño lógico para asegurarse de que la captura y el almacenamiento de datos sean precisos. A continuación, se presenta el flujo de trabajo del script monitor_snmp.py:

1. Consulta periódica de OIDs configuradas en la base de datos.
2. Envío de solicitudes SNMP al servidor de monitoreo.
3. Recepción de respuestas con valores de métricas del sistema.
4. Evaluación del valor recibido y comparación con los umbrales definidos.
5. Registro del resultado en la base de datos, marcando el estado correspondiente.
6. Generación de alertas por correo electrónico en caso de superar los umbrales.

**Esquema de Flujo de Generación de Reportes**

El script *generar_grafica.py* se encarga de generar reportes gráficos en formato PDF basándose en los datos almacenados en la base de datos. El proceso sigue los siguientes pasos:

1. Consulta de registros almacenados del día anterior.
2. Procesamiento de los datos para organizar valores y estados.
3. Creación de gráficos de barras diferenciando estados del sistema.
4. Generación de reportes en formato PDF y almacenamiento en el servidor.
5. Eliminación de reportes y registros más antiguos a tres días para optimizar el almacenamiento.

**Relaciones entre Componentes**

El sistema está diseñado para asegurar una comunicación eficiente entre sus diferentes componentes. A continuación, se describen las interacciones clave:

- El servidor SNMP envía respuestas al cliente en función de las solicitudes recibidas.
- Los scripts en Python se encargan de la recolección de datos y de la generación de reportes automatizados.
- La base de datos permite la consulta y almacenamiento de los valores recolectados, asegurando su disponibilidad para su visualización en la web.
- El servidor web Apache expone una interfaz accesible para los administradores y usuarios finales.

# 3. RESULTADOS Y DISCUSIÓN

## 3.1 RESULTADOS OBTENIDOS

### 3.1.1 Ejecución del sistema en tiempo real

El sistema de monitoreo implementado se basa en consultas periódicas a dispositivos de red mediante el protocolo SNMPv3. El script monitor_snmp.py se encarga de ejecutar estas consultas a intervalos definidos por la configuración del sistema, almacenando los resultados en la base de datos MySQL para su posterior análisis y visualización.
El proceso de ejecución se inicia automáticamente, ejecutándose de manera continua y programada mediante la herramienta Schedule de Python. El script obtiene información de los OIDs configurados, procesa los valores recibidos y almacena los datos junto con sus estados en la base de datos.

**Fases de ejecución**

1. **Inicio del script:**
    - El script monitor_snmp.py establece la conexión con la base de datos MySQL y recupera la lista de OIDs a monitorear.
    - Se consulta la tabla oids_config para obtener los umbrales y los intervalos de consulta definidos.
2. **Consulta SNMP:**
    - Se envía una solicitud SNMPv3 a la dirección IP del agente configurado.
    - Se recibe y procesa la respuesta, registrando los valores en la base de datos.
3. **Evaluación del estado:**
    - Se compara el valor recibido con el umbral configurado.
    - Si el valor supera el umbral, se envía una alerta por correo electrónico a los administradores registrados.
4. **Registro de actividad:**
    - Cada consulta exitosa o fallida se registra en la base de datos con su correspondiente estado (Activo, Threshold Superado o Inactivo).

A continuación, podemos ver en la *ilustración 51* la ejecución del script monitor_snmp.py en consola mostrando consultas periódicas a los OIDs configurados. Podemos apreciar la

secuencia que sigue el programa mediante la notificación en la ventana de comandos, así como las distintas OIDs consultadas, su valor y su respectivo estado.

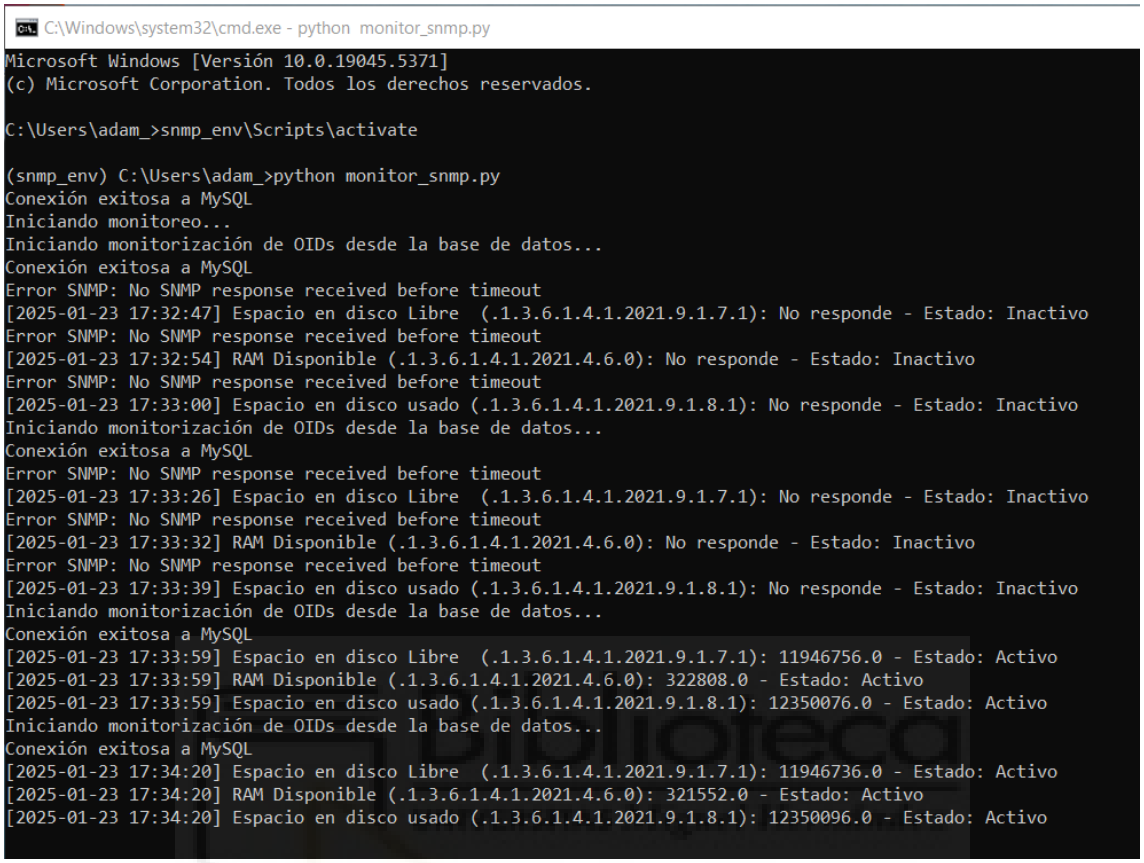
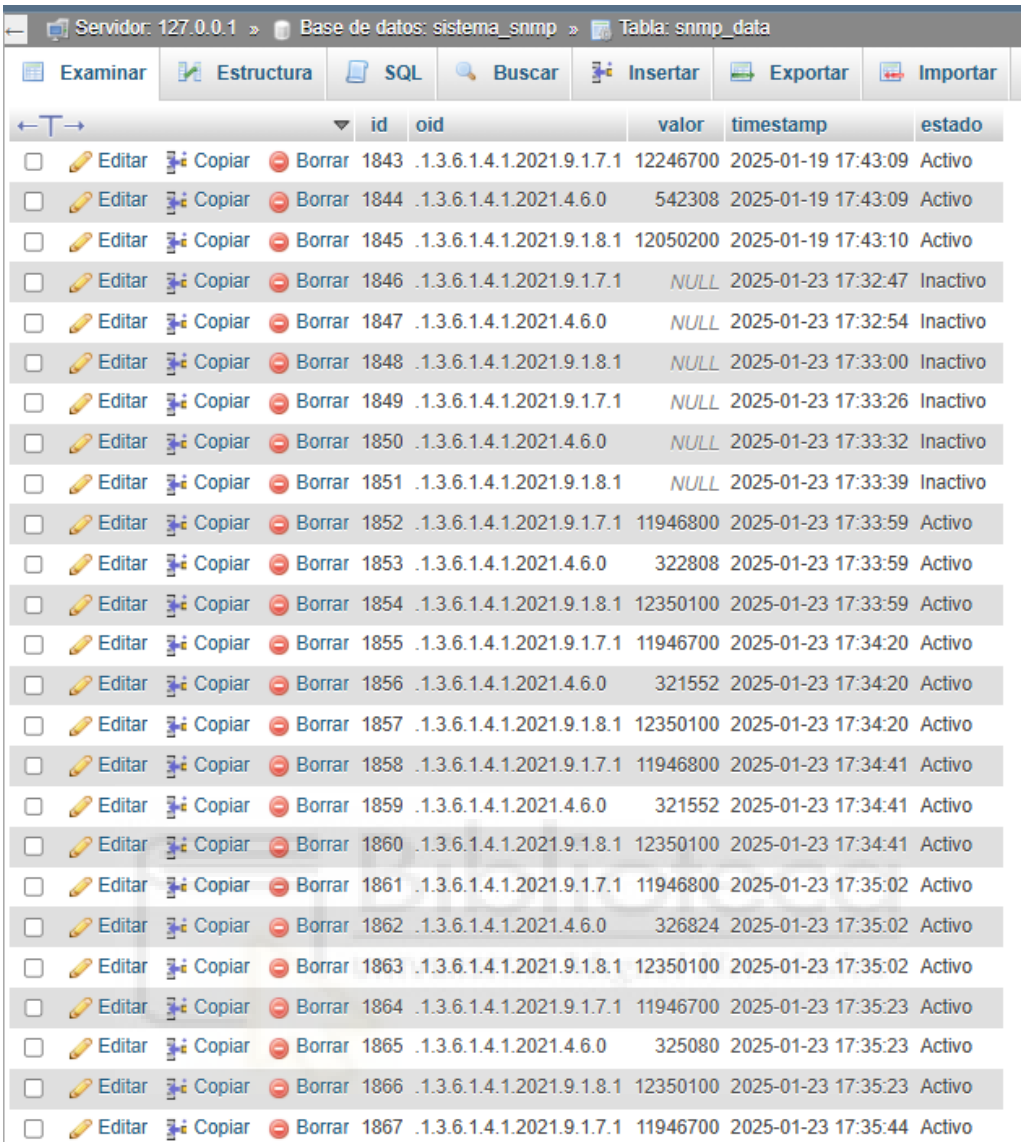



*Ilustración 52. Ejecución monitor_snmp.py.*

Y en la *ilustración 52* podemos ver el registro de datos en la base de datos después de la ejecución del script.




| | id | oid | valor | timestamp | estado |
|---|---|---|---|---|---|
| Editar Copiar Borrar | 1843 | .1.3.6.1.4.1.2021.9.1.7.1 | 12246700 | 2025-01-19 17:43:09 | Activo |
| Editar Copiar Borrar | 1844 | .1.3.6.1.4.1.2021.4.6.0 | 542308 | 2025-01-19 17:43:09 | Activo |
| Editar Copiar Borrar | 1845 | .1.3.6.1.4.1.2021.9.1.8.1 | 12050200 | 2025-01-19 17:43:10 | Activo |
| Editar Copiar Borrar | 1846 | .1.3.6.1.4.1.2021.9.1.7.1 | NULL | 2025-01-23 17:32:47 | Inactivo |
| Editar Copiar Borrar | 1847 | .1.3.6.1.4.1.2021.4.6.0 | NULL | 2025-01-23 17:32:54 | Inactivo |
| Editar Copiar Borrar | 1848 | .1.3.6.1.4.1.2021.9.1.8.1 | NULL | 2025-01-23 17:33:00 | Inactivo |
| Editar Copiar Borrar | 1849 | .1.3.6.1.4.1.2021.9.1.7.1 | NULL | 2025-01-23 17:33:26 | Inactivo |
| Editar Copiar Borrar | 1850 | .1.3.6.1.4.1.2021.4.6.0 | NULL | 2025-01-23 17:33:32 | Inactivo |
| Editar Copiar Borrar | 1851 | .1.3.6.1.4.1.2021.9.1.8.1 | NULL | 2025-01-23 17:33:39 | Inactivo |
| Editar Copiar Borrar | 1852 | .1.3.6.1.4.1.2021.9.1.7.1 | 11946800 | 2025-01-23 17:33:59 | Activo |
| Editar Copiar Borrar | 1853 | .1.3.6.1.4.1.2021.4.6.0 | 322808 | 2025-01-23 17:33:59 | Activo |
| Editar Copiar Borrar | 1854 | .1.3.6.1.4.1.2021.9.1.8.1 | 12350100 | 2025-01-23 17:33:59 | Activo |
| Editar Copiar Borrar | 1855 | .1.3.6.1.4.1.2021.9.1.7.1 | 11946700 | 2025-01-23 17:34:20 | Activo |
| Editar Copiar Borrar | 1856 | .1.3.6.1.4.1.2021.4.6.0 | 321552 | 2025-01-23 17:34:20 | Activo |
| Editar Copiar Borrar | 1857 | .1.3.6.1.4.1.2021.9.1.8.1 | 12350100 | 2025-01-23 17:34:20 | Activo |
| Editar Copiar Borrar | 1858 | .1.3.6.1.4.1.2021.9.1.7.1 | 11946800 | 2025-01-23 17:34:41 | Activo |
| Editar Copiar Borrar | 1859 | .1.3.6.1.4.1.2021.4.6.0 | 321552 | 2025-01-23 17:34:41 | Activo |
| Editar Copiar Borrar | 1860 | .1.3.6.1.4.1.2021.9.1.8.1 | 12350100 | 2025-01-23 17:34:41 | Activo |
| Editar Copiar Borrar | 1861 | .1.3.6.1.4.1.2021.9.1.7.1 | 11946800 | 2025-01-23 17:35:02 | Activo |
| Editar Copiar Borrar | 1862 | .1.3.6.1.4.1.2021.4.6.0 | 326824 | 2025-01-23 17:35:02 | Activo |
| Editar Copiar Borrar | 1863 | .1.3.6.1.4.1.2021.9.1.8.1 | 12350100 | 2025-01-23 17:35:02 | Activo |
| Editar Copiar Borrar | 1864 | .1.3.6.1.4.1.2021.9.1.7.1 | 11946700 | 2025-01-23 17:35:23 | Activo |
| Editar Copiar Borrar | 1865 | .1.3.6.1.4.1.2021.4.6.0 | 325080 | 2025-01-23 17:35:23 | Activo |
| Editar Copiar Borrar | 1866 | .1.3.6.1.4.1.2021.9.1.8.1 | 12350100 | 2025-01-23 17:35:23 | Activo |
| Editar Copiar Borrar | 1867 | .1.3.6.1.4.1.2021.9.1.7.1 | 11946700 | 2025-01-23 17:35:44 | Activo |

*Ilustración 53. Tabla snmp_data con valores almacenados.*

**Discusión de los resultados**

El sistema ha demostrado ser capaz de realizar consultas SNMP de manera eficiente, registrando valores de los OIDs definidos y detectando situaciones en las que se superan los umbrales establecidos. Se ha observado que:

- La consulta SNMP se realiza con éxito en la mayoría de los casos, reflejando valores precisos del estado del sistema monitoreado.
- Cuando el dispositivo no está disponible, el sistema registra correctamente el estado como Inactivo.
- Las alertas por correo electrónico se generan solo cuando es necesario, evitando notificaciones redundantes.

### 3.1.2 Validación del monitoreo SNMPv3

Para garantizar la correcta operación del sistema de monitoreo SNMPv3, se llevaron a cabo una serie de pruebas para validar tanto la precisión de las consultas como la fiabilidad del almacenamiento de datos y la detección de eventos críticos.

La validación del monitoreo se basó en la ejecución del script *monitor_snmp.py*, su integración con la base de datos MySQL y la correcta generación de alertas ante valores que superan los umbrales establecidos.

## Pruebas Realizadas

**Consulta de OIDs configurados:**

- Se verificó que las OIDs almacenadas en la tabla oids_config fueran consultadas correctamente por el script.
- Se realizó la captura del tráfico SNMP con Wireshark para comprobar la autenticidad y seguridad de las consultas enviadas bajo el protocolo SNMPv3.

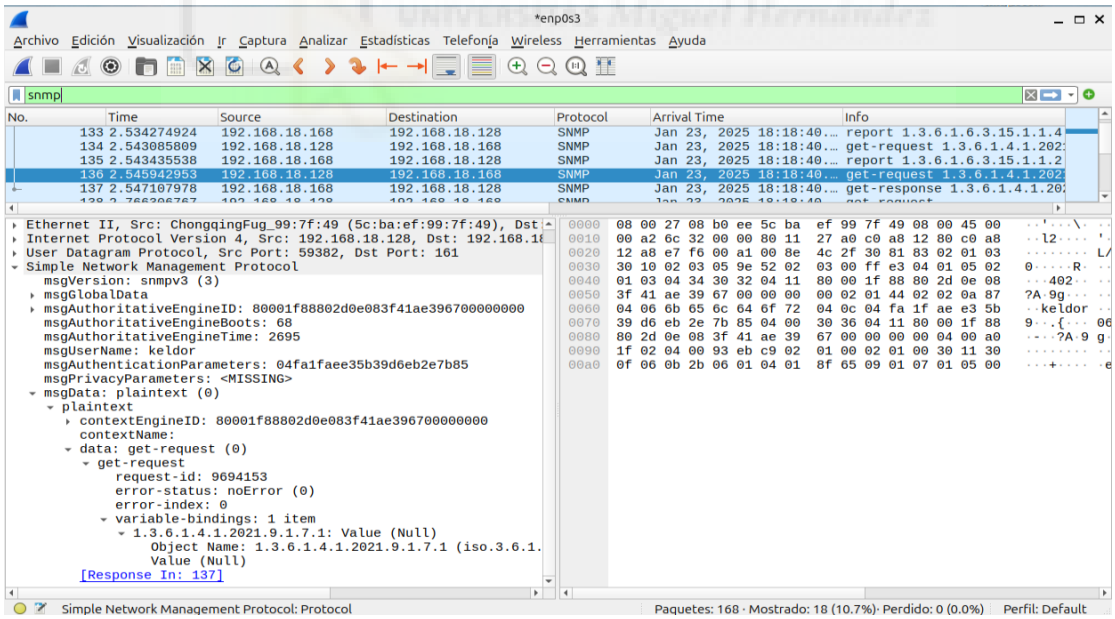



*Ilustración 54. Consulta SNMPv3 (Get-Request).*

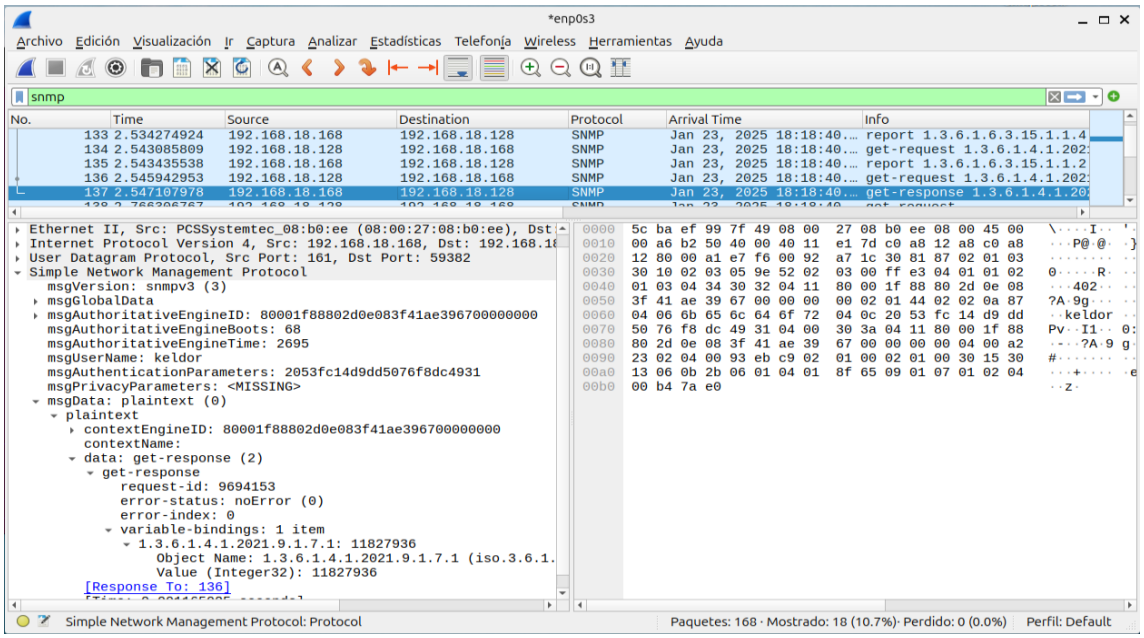



*Ilustración 55. Respuesta SNMPv3 (Get-Response).*

## Registro en base de datos:

- Se comprobó que los valores obtenidos de los OIDs se almacenan correctamente en la tabla snmp_data junto con la marca temporal y el estado correspondiente.

| id | oid | valor | timestamp | estado |
|---|---|---|---|---|
| 1871 | .1.3.6.1.4.1.2021.4.6.0 | 322060 | 2025-01-23 17:36:05 | Activo |
| 1872 | .1.3.6.1.4.1.2021.9.1.8.1 | 12350100 | 2025-01-23 17:36:05 | Activo |
| 1873 | .1.3.6.1.4.1.2021.9.1.7.1 | 11946700 | 2025-01-23 17:36:26 | Activo |
| 1874 | .1.3.6.1.4.1.2021.4.6.0 | 322060 | 2025-01-23 17:36:26 | Activo |
| 1875 | .1.3.6.1.4.1.2021.9.1.8.1 | 12350100 | 2025-01-23 17:36:26 | Activo |
| 1876 | .1.3.6.1.4.1.2021.9.1.7.1 | 11946700 | 2025-01-23 17:36:47 | Activo |
| 1877 | .1.3.6.1.4.1.2021.4.6.0 | 322060 | 2025-01-23 17:36:47 | Activo |
| 1878 | .1.3.6.1.4.1.2021.9.1.8.1 | 12350100 | 2025-01-23 17:36:47 | Activo |
| 1879 | .1.3.6.1.4.1.2021.9.1.7.1 | 11946700 | 2025-01-23 17:37:08 | Activo |

*Ilustración 56. Registros almacenados en la tabla snmp_data*

**Simulación de umbral superado:**

- Se modificó el valor del threshold en la OID de ‘Espacio en disco usado’ para forzar el cambio de ‘activo’ a ‘threshold superado’ y poder así monitorizarlo .
- Se verificó que el sistema cambie correctamente el estado a "Threshold Superado" y que se envíe un correo de alerta a los administradores.

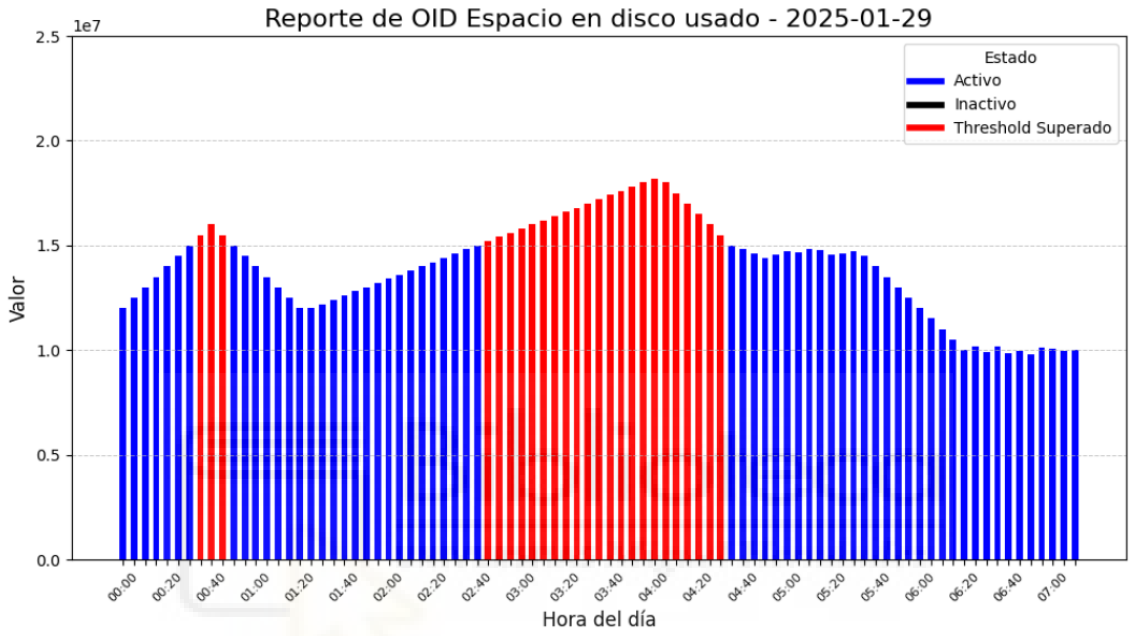



*Ilustración 57. Reporte Generado por el sistema.*

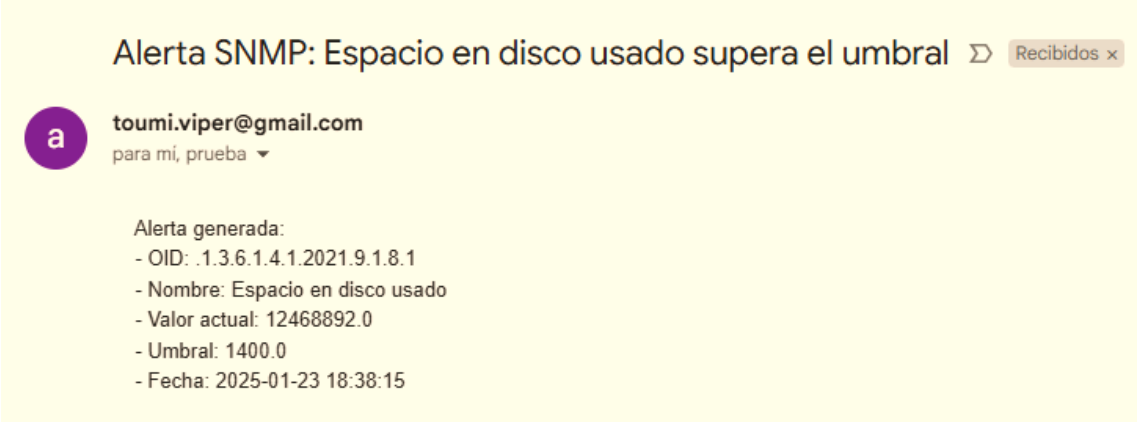



*Ilustración 58. Email de alerta de umbral superado.*

**Prueba de inactividad del agente:**

- Se desconectó el sistema Ubuntu para comprobar si el sistema detecta correctamente la falta de respuesta, registrando el estado como "Inactivo" en la base de datos.

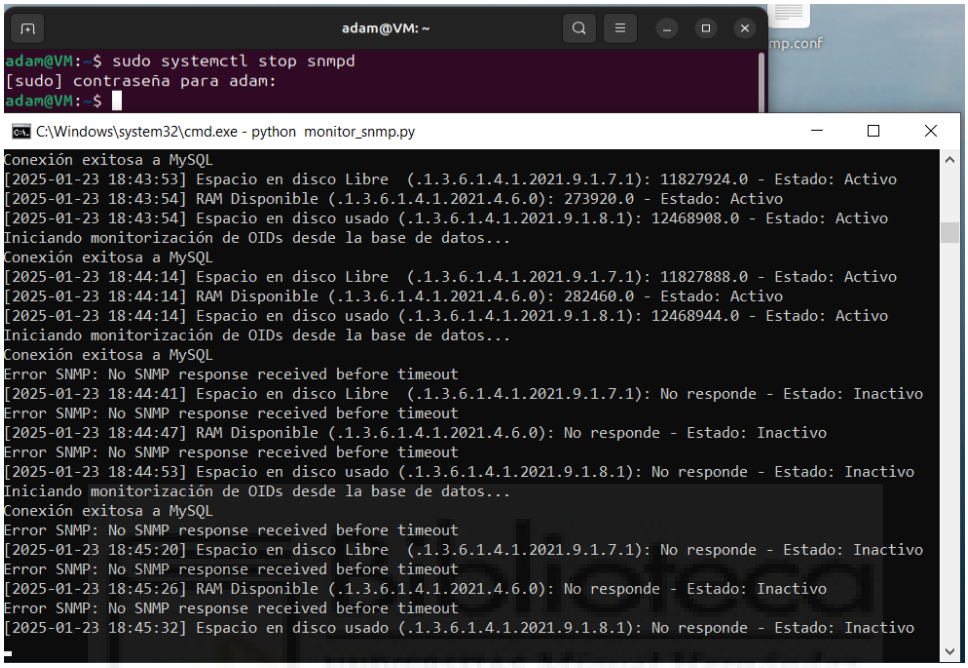



*Ilustración 59. Registro de estado Inactivo.*

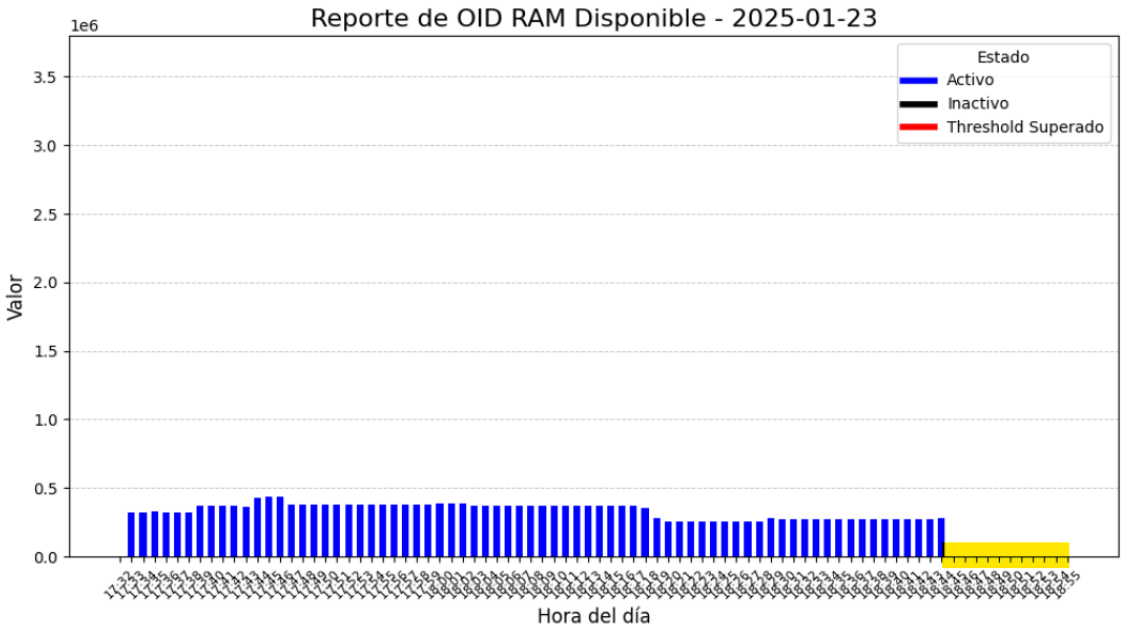



*Ilustración 60. Gráfica generada con valores nulos.*

**Resultados obtenidos**

El proceso de validación permitió comprobar que:

- Las consultas SNMPv3 se realizan correctamente y los valores obtenidos son fieles a los datos del sistema monitoreado.
- La detección de umbrales y el envío de alertas funcionan adecuadamente, notificando a los administradores solo cuando es necesario.
- La gestión de estados (Activo, Threshold Superado, Inactivo) se realiza de manera precisa, reflejando el estado real del sistema monitoreado.
- El tráfico SNMP está correctamente cifrado bajo el protocolo SNMPv3, garantizando la seguridad de las consultas.

La conclusión de este apartado es que el sistema de monitoreo implementado cumple con los requisitos de funcionamiento esperados. Ha demostrado ser capaz de realizar consultas de manera eficiente, detectar anomalías y generar alertas cuando es necesario.

### 3.1.3 Visualización y análisis de gráficos

La información almacenada se visualiza en tiempo real a través de gráficas gracias a las consultas SNMPv3. Estos gráficos se usan para determinar el comportamiento del uso de la memoria RAM, espacio en disco y demás parámetros monitoreados. La interfaz de administración dispone de un dashboard web con unos gráficos en los cuales cada uno muestra un histórico con los valores recolectados para cada OID.

**Proceso de generación de gráficos**

1. **Ejecución del script generar_grafica.py:**
    - Este script se ejecuta diariamente de forma automatizada.
    - Genera gráficos basados en los datos almacenados en la base de datos MySQL.
    - Guarda las gráficas en formato PNG y genera reportes en formato PDF.
2. **Consulta de datos almacenados:**
    - Se realiza una consulta SQL a la tabla snmp_data para obtener los valores de los OIDs monitoreados.
    - Se aplican filtros de fecha y estado para generar reportes diarios.

3. **Generación de gráficos:**
    - Los datos se procesan utilizando la librería matplotlib para trazar gráficos de barras.
    - Se establecen límites en el eje Y para una representación proporcional de los valores obtenidos.
    - Los diferentes estados (Activo, Inactivo, Threshold Superado) se representan con colores diferenciados

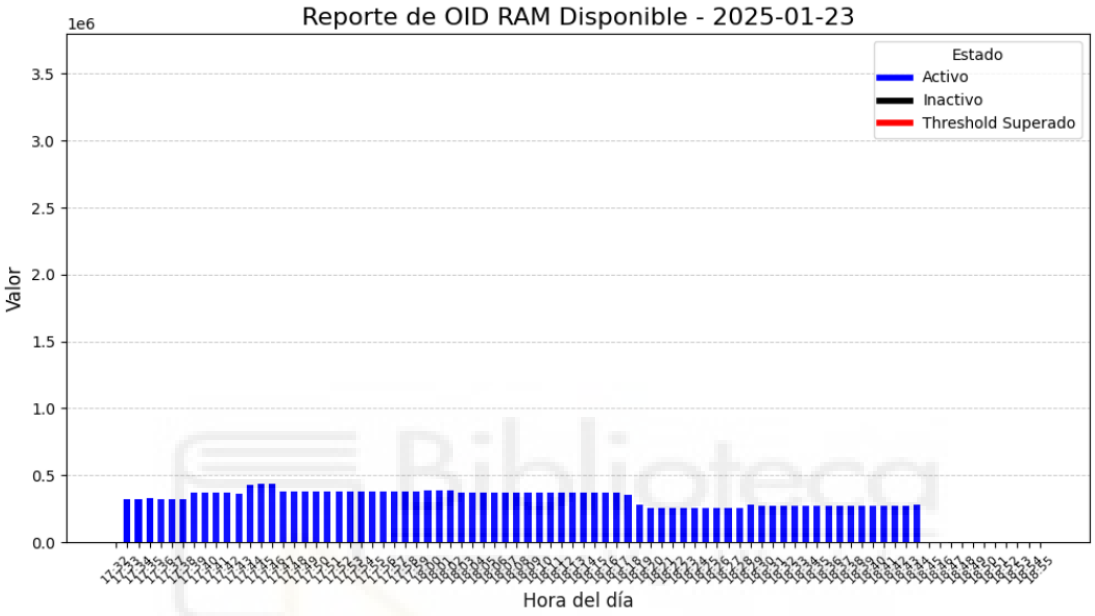



*Ilustración 61. Reporte generado automáticamente.*

**Análisis de gráficos obtenidos**

El análisis de los gráficos permite identificar patrones y anomalías en el comportamiento del sistema monitoreado. Entre los puntos clave observados se encuentran:

1. **Identificación de tendencias:**
    - Aumento progresivo en el uso de memoria a lo largo del tiempo.
    - Oscilaciones en el espacio en disco debido a procesos internos del sistema.
2. **Detección de eventos críticos:**
    - Se observan picos significativos cuando el umbral configurado es superado.
    - Las áreas en color rojo en el gráfico indican momentos en los que el sistema estuvo en un estado crítico.

3. **Evaluación de inactividad:**
    - Se visualizan períodos de inactividad reflejados en el gráfico mediante barras de color negro, indicando que el sistema monitoreado estuvo fuera de línea.

**Visualización en el dashboard web**

Los gráficos generados se integran en el dashboard web del sistema, permitiendo a los administradores:

- Seleccionar OIDs específicos para su visualización.
- Comparar métricas de diferentes días.
- Descargar reportes en formato PDF para análisis posterior.

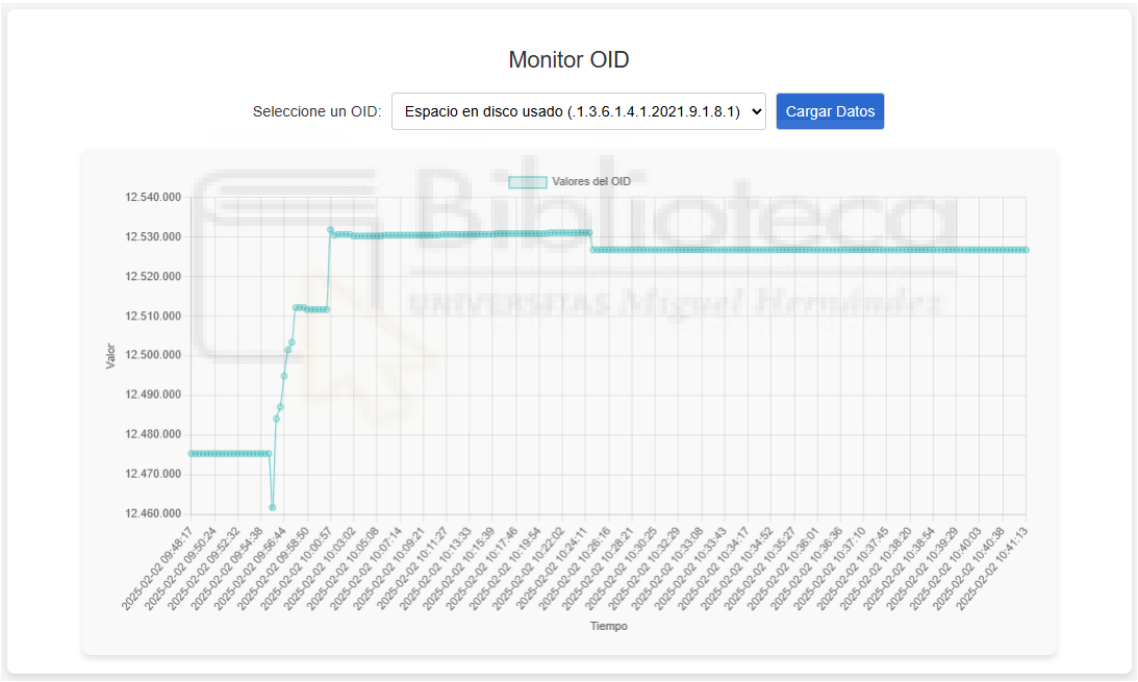



*Ilustración 62. Monitor web 'Espacio en disco Usado'.*

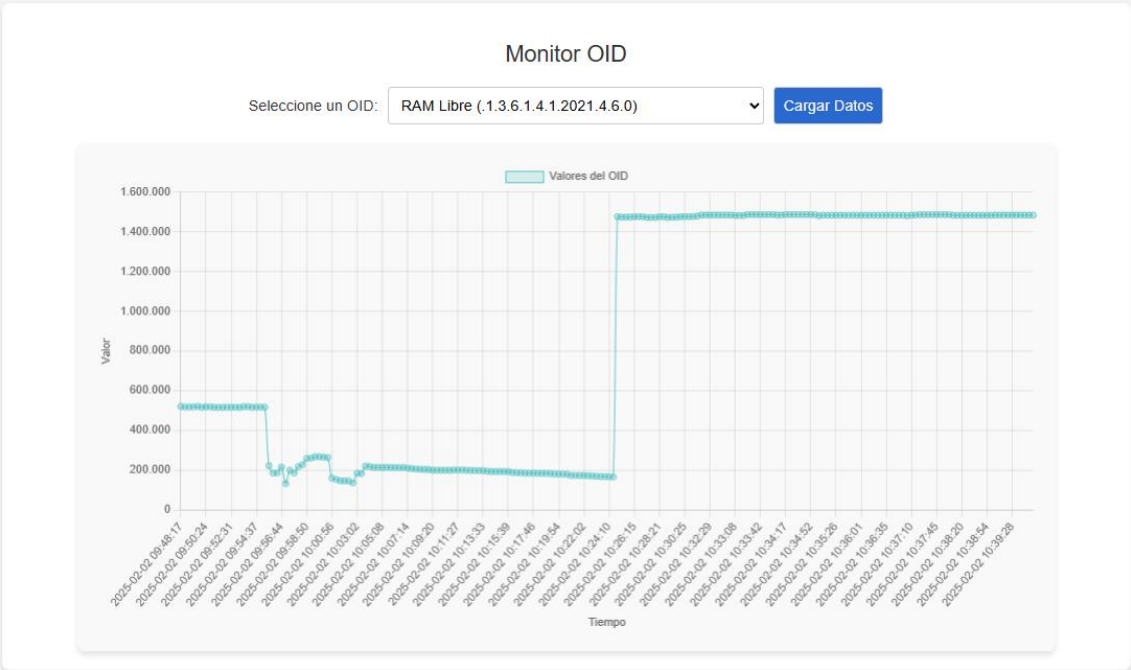



*Ilustración 63. monitor web 'RAM Libre'.*

**Conclusiones de la visualización de datos**

El sistema de visualización permite un análisis eficiente de los parámetros monitoreados, proporcionando información clave para la toma de decisiones sobre el rendimiento del sistema monitoreado. La representación gráfica facilita la detección de patrones y la respuesta ante situaciones críticas.

## 3.2 LIMITACIONES IDENTIFICADAS

Durante el proceso de desarrollo e implementación del sistema de monitoreo en tiempo real se ha podido identificar una serie de limitaciones que afectan al rendimiento, la escalabilidad y la usabilidad de este sistema. Se han agrupado estas limitaciones en diferentes categorías, abordando aspectos técnicos, operativos y de implementación.

### 3.2.1 Limitaciones Técnicas

**Latencia en las consultas SNMPv3**

Al introducir la autenticación en las consultas SNMPv3 se genera una leve latencia que puede causar retrasos al aumentar el volumen de las consultas en el sistema. Junto con eso, la configuración de intervalos más cortos aumenta la carga en la red y la base de datos.

**Capacidad de almacenamiento en la base de datos**

Aunque hemos implementado un mecanismo de purga de datos, el almacenamiento prolongado de registros de monitoreo puede resultar en un consumo significativo de espacio en disco en entornos con grandes volúmenes de OID a monitorizar resultando así en una disminución del rendimiento del sistema.

También podríamos tener en cuenta que la eliminación de datos antiguos podría afectar a futuras auditorías o análisis históricos a largo plazo.

**Gestión de múltiples OIDs simultáneamente**

El sistema actual está optimizado para un número limitado de OIDs en paralelo. Un crecimiento exponencial en los dispositivos monitoreados podría requerir una reestructuración del esquema de la base de datos y una mejora en los algoritmos de consulta.

### 3.2.2 Limitaciones Operativas

**Configuración inicial compleja**

La configuración de SNMPv3 requiere un conocimiento técnico previo en seguridad de redes, debido a la necesidad de establecer usuarios, claves de autenticación y configuraciones específicas de cada dispositivo monitoreado. Y la integración con diferentes sistemas operativos podría requerir configuraciones adicionales específicas para cada entorno.

**Carga manual de OIDs predefinidos**

Actualmente, la incorporación de nuevos OIDs debe hacerse manualmente a través de la interfaz de administración. La automatización de este proceso mediante precarga de MIB (Management Information Base) podría mejorar la escalabilidad del sistema.

### 3.2.3 Limitaciones de seguridad

**Acceso a la interfaz web**

Aunque se implementó el control de acceso con autenticación de usuarios y roles diferenciados, un acceso no autorizado podría comprometer la integridad de la información monitoreada. Y la falta de un registro detallado de accesos y modificaciones en el sistema representa una futura mejora en el sistema.

### 3.2.4 Limitaciones de Usabilidad

**Curva de aprendizaje para nuevos usuarios**

La interfaz de usuario, aunque intuitiva, podría beneficiarse de documentación más detallada para nuevos usuarios o administradores que no estén familiarizados con SNMP.

**Visualización de datos en tiempo real**

La actualización de gráficos en la interfaz web se realiza en intervalos fijos de 30 segundos. Para sistemas que requieren información en tiempo real con latencias más bajas, se necesitaría una optimización del sistema de consultas.

En base a las limitaciones identificadas, se proponen varias estrategias de mejora y optimización, las cuales, trataremos en el siguiente apartado. Estas mejoras se centrarán específicamente en la escalabilidad, la automatización de procesos y la seguridad del sistema.

# 4. CONCLUSIONES Y TRABAJO FUTURO

## 4.1 CONCLUSIONES GENERALES

El desarrollo de este proyecto ha permitido la implementación de un sistema funcional capaz de monitorizar en tiempo real los recursos críticos de un servidor mediante el protocolo **SNMPv3**, garantizando la seguridad y autenticación de las consultas.
A lo largo del proceso, se han alcanzado los siguientes objetivos principales:

- **Automatización de la monitorización:**
  Se ha desarrollado un sistema que realiza consultas periódicas, almacenando los datos en una base de datos MySQL y presentándolos de manera visual en un dashboard accesible vía web.
- **Generación de reportes y análisis histórico:**
  Se ha implementado una funcionalidad para la exportación de reportes diarios en formato PDF, permitiendo la consulta de datos históricos y la detección de patrones de comportamiento.
- **Gestión de alertas y thresholds:**
  Se han configurado umbrales para la detección proactiva de problemas, enviando alertas por correo electrónico a los administradores cuando se superan ciertos valores críticos.
- **Eficiencia en la purga de datos:**
  Se ha diseñado un mecanismo automatizado que mantiene la base de datos optimizada, eliminando registros antiguos y garantizando un rendimiento estable a lo largo del tiempo.

## 4.2 PROPUESTAS DE MEJORA

Para una futura mejora y optimización de este sistema, se proponen las siguientes mejoras:

**Integración con múltiples dispositivos**

Actualmente, el sistema está diseñado para monitorear un único servidor. Se podría ampliar la funcionalidad para gestionar múltiples servidores simultáneamente, permitiendo así, un monitoreo distribuido.

**Mejora de Interfaz de Usuario**

Se podría mejorar la experiencia del usuario mediante gráficos interactivos y un diseño más dinámico incorporando frameworks, como **ReactJS** o **Vue.js**, e incluyendo interfaces móviles.

**Escalabilidad del almacenamiento de datos**

Se podría implementar soluciones de almacenamiento en la nube para manejar grandes volúmenes de información a largo plazo.

**Sistema de notificaciones mejorado**

Se podría ampliar el sistema de alertas mediante notificación a través de WhatsApp, Telegram o similar ofreciendo así más opciones a los administradores.

## 4.3 POSIBLES LINEAS DE DESARROLLO FUTURO

El sistema que hemos desarrollado ofrece una base sólida para futuras ampliaciones. Algunas de las cuales pueden ser:

**Implementación de Machine Learning:**
Aplicar técnicas de aprendizaje automático para detectar anomalías en el comportamiento del sistema, prediciendo posibles fallos antes de que ocurran.

**Visualización avanzada de datos:**
Integración con herramientas de análisis como Grafana, permitiendo crear dashboards más avanzados con múltiples vistas y análisis de tendencias.

**Automatización de respuestas a eventos críticos:**
Incorporar un sistema de reacción automática ante eventos críticos, como el reinicio de servicios o el envío de comandos correctivos al servidor monitorizado.

**Despliegue en entornos híbridos:**
Adaptación del sistema para monitorear entornos en la nube como AWS, Azure o Google Cloud, extendiendo la capacidad de monitorización a infraestructuras híbridas.

# 5. BIBLIOGRAFÍA

**1. Apache Software Foundation, "Apache HTTP Server Project."**
[En línea]. Disponible en: https://httpd.apache.org/.
[Último acceso: 15-02-2025].

**2. MySQL Documentation, "MySQL: The world's most popular open-source database."**
[En línea]. Disponible en: https://dev.mysql.com/doc/.
[Último acceso: 15-02-2025].

**3. PHP Documentation, "PHP: Hypertext Preprocessor - Official Website."**
[En línea]. Disponible en: https://www.php.net/docs.php.
[Último acceso: 15-02-2025].

**4. Apache HTTP Server, "Official Documentation."**
[En línea]. Disponible en: https://httpd.apache.org/docs/.
[Último acceso: 15-02-2025].

**5. MySQL, "MySQL 8.0 Reference Manual."**
[En línea]. Disponible en: https://dev.mysql.com/doc/refman/8.0/en/.
[Último acceso: 15-02-2025].

**6. Python, "Python 3.11 Documentation."**
[En línea]. Disponible en: https://docs.python.org/3/.
[Último acceso: 15-02-2025].

**7. Oracle VM VirtualBox, "User Manual."**
[En línea]. Disponible en: https://www.virtualbox.org/manual/UserManual.html.
[Último acceso: 15-02-2025].

**8. RFC 3411 - An Architecture for Describing SNMP Management Frameworks**

[En línea]. Disponible en: RFC 3411 - An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks.

[Último acceso: 15-02-2025].

**10. Net-SNMP Official Documentation**

[En línea]. Disponible en: http://www.net-snmp.org/docs/.

[Último acceso: 15-02-2025].

**11. Zabbix Official Documentation**

[En línea]. Disponible en: https://www.zabbix.com/documentation/current/manual.

[Último acceso: 15-02-2025].

**12. Observium User Guide**

[En línea]. Disponible en: https://docs.observium.org/.

[Último acceso: 15-02-2025].

**13. Nagios Core Documentation**

[En línea]. Disponible en: https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/.

[Último acceso: 15-02-2025].

**14. Icinga Documentation**

[En línea]. Disponible en: https://icinga.com/docs/icinga-2/latest/.

[Último acceso: 15-02-2025].