

PREGUNTAS CHAT GPT HASTA ALCANZAR SOLUCIÓN FINAL

Introducción

Durante el desarrollo del proyecto de agente SNMP, utilizamos ChatGPT como una herramienta de apoyo para resolver dudas, entender conceptos y estructurar el código paso a paso. El objetivo de este documento es mostrar cómo interactuamos con ChatGPT, incluyendo todas las preguntas que le realizamos y las respuestas recibidas, reflejando el proceso de aprendizaje y desarrollo que nos permitió llegar al código final del agente SNMP.

Se ha de tener en cuenta que previamente a iniciar las preguntas sobre la realización de la práctica se ha mandado el guión del trabajo a realizar.

1. CREACIÓN DE LA MIB

Para la realización del trabajo adjuntado, tengo que crear una mib que incluya todos los elementos mencionados, ¿como la harias?

Proporciona un ejemplo de plantilla para realizar la MIB con mis datos:

Archivo: MYAGENT-MIB.txt

```
plaintext Copiar código

MYAGENT-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE, Integer32, enterprises
    FROM SNMPV2-SMI
    DisplayString, DateAndTime
    FROM SNMPV2-TC;

myAgentMIB MODULE-IDENTITY
    LAST-UPDATED "20251104Z"          -- AAAAMMDDZ
    ORGANIZATION "Network Management Course - Universidad de Zaragoza"
    CONTACT-INFO
        "Student: <tu nombre>
         Email: <tu email>"
    DESCRIPTION
        "MIB para un mini agente SNMP que monitoriza el uso de CPU
         y envía una notificación cuando el uso supera un umbral definido."
    REVISION "20251104Z"
    DESCRIPTION
```



MYAGENT-MIB.txt

Documento

He creado esta mib, dime si cumple todos los requisitos pedidos en la práctica y explicamela linea a linea de manera sencilla


Revisa errores y aporta la solución:

Cosas a corregir/mejorar:

1. ❌ Problema con `dateAndTime` en la notificación

En la notificación has puesto:

text


 Copiar código

```
cpuOverThresholdNotification NOTIFICATION-TYPE
    OBJECTS { cpuUsage, cpuThreshold, managerEmail, dateAndTime }
```

- `OBJECTS` debe listar nombres de objetos (**OBJECT-TYPE**), no el nombre de un tipo/TC.
- Tú has importado el tipo `DateAndTime` de `SNMPv2-TC`, pero no has creado un objeto del estilo `cpuEventTime` o similar. MYAGENT-MIB
- Además, aquí escribes `dateAndTime` en minúsculas, que **no está definido en ningún sitio**.

👉 Lo correcto sería algo así (ejemplo):

text

 Copiar código

```
eventTime OBJECT-TYPE
```

2. COMPRENSIÓN DEL CÓDIGO DEL QUE PARTIMOS

Explícame de manera sencilla que hacen los códigos que vienen de ejemplo en el guion del trabajo aportados por el profesor, más tarde cuando te pida que me realices códigos básate en ellos.

✅ Explicación sencilla de los códigos del guion (los ejemplos que trae el PDF)


Voy a dividir la explicación en bloques, igual que el guion, para que entiendas qué hace cada trozo de código, para qué sirve y cómo encaja todo.

1. Modelo de datos en JSON (`mib_state.json`)

El guion te enseña que el agente guarda los valores de las variables SNMP en un archivo JSON.

Ejemplo:

json

 Copiar código

```
{
  "base_oid": "1.3.6.1.3.9999.1",
  "scalars": {
    "manager": {
      "oid": "1.3.6.1.3.9999.1.1.0",
      "type": "DisplayString",
      "access": "read-write",
```

3. PRIMERA ESTRUCTURA DEL CÓDIGO PARA QUE `mini_agent.py` COMIENCE A COGER FORMA

una vez que he creado el fichero de la mib y he visto que es correcto compilando con el mib browser, cual es el siguiente paso

Excelente 🎉 — ya tienes tu MIB validada y funcionando, así que estás justo en el punto clave donde comienza la **parte práctica y programable del trabajo**.

Vamos a repasar qué viene ahora **paso a paso**, de manera muy clara y orientada a tu entrega.

🚀 PASOS SIGUIENTES DESPUÉS DE CREAR Y COMPILAR LA MIB

🧩 1 Crear la estructura del proyecto

En tu carpeta del trabajo, crea una estructura como esta:

```
java                                                                    Copiar código

snmp_code/
├─ mini_agent.py               ← tu agente principal en Python
├─ MYAGENT-MIB.txt            ← la MIB que ya has hecho
├─ mib_state.json              ← archivo donde se guardará el estado (valores)
└─ README.md                  ← cómo ejecutar y probar el agente
```

A continuación la IA me sugiere crear la plantilla base:

¿Quieres que te genere ahora una **plantilla base de mini_agent.py** (ya lista para que funcione con tu MIB, con los objetos y estructura correctos)?

Así podrías probarlo directamente con MG-SOFT.

📄 👍 💬 ...

si

A continuación me aporta un primer código de mini_agent:

Perfecto ✅ — aquí tienes una **plantilla completa y funcional** de **mini_agent.py**, adaptada exactamente a **tu MIB MYAGENT-MIB.txt**, con los objetos que definiste y las funcionalidades que exige la práctica.

Está pensada para que **funcione directamente con pysnmp**, y puedas después completar la parte del envío de correo y trap.

📄 Archivo: mini_agent.py

```
python                                                                    Copiar código

#!/usr/bin/env python3
"""
Mini SNMP Agent - Network Management Course
Autor: <tu nombre>
Email: <tu email>

Este agente:
- Expone los objetos definidos en MYAGENT-MIB.txt
- Actualiza periódicamente el uso de CPU
- Envía una notificación (trap) y un email cuando cpuUsage > cpuThreshold
```

4. RECURSOS NECESARIOS PARA PROBAR FUNCIONAMIENTO CÓDIGO

que me tengo que descargar en mi ordenador windows para poner los comandos snmp y comprobar que mi mini_agent funciona

Perfecto 🍌 Para poder comprobar en tu ordenador Windows que tu mini_agent SNMP funciona correctamente, necesitas instalar algunas herramientas SNMP de línea de comandos. Aquí te dejo todo lo que necesitas paso a paso:

📦 1. Instalar las herramientas SNMP de Net-SNMP

Estas te permiten usar los comandos:

- `snmpget`
- `snmpgetnext`
- `snmpset`
- `snmpwalk`
- `snmptrap`

👉 Descarga e instalación:

1. Ve a la página oficial:
<https://ezix.org/project/wiki/Net-SNMP>
2. O descarga directamente el instalador de Windows (por ejemplo, `net-snmp-5.9.3-1.x64.exe`) desde un mirror fiable, como:
<https://sourceforge.net/projects/net-snmp/files/net-snmp/>

5. REVISIÓN CONSTANTE DE CÓDIGO RESOLVIENDO MÚLTIPLES ERRORES

En este punto de trabajo revisamos el código creado, tratamos de entenderlo y comienza un laborioso trabajo de corrección de errores. A continuación, se muestran algunas de las interacciones con la IA durante este proceso:

- Problemas con el envío de notificaciones:

```
Traceback (most recent call last):
  File "C:\Users\laram\Documents\TELECOM\Primer cuatr\Gestión de red\snmp_code\mini_agent.py", line 141, in
    module
    config.addNotification(
    ~~~~~^~~~~~
  File "C:\Users\laram\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11.0q2562zfradp\LocalCache\local-packag
es\Python311\lib\site-packages\pysnmp\entity\config.py", line 79, in _setattr_
    raise AttributeError(f"module '{_name_!}' has no attribute '{attr!}'")
AttributeError: module 'pysnmp.entity.config' has no attribute 'addNotification'
```

me sale este error al hacer python mini_agent.py

Perfecto 🍌 — este error que te aparece:

```
pgsql
```

📋 Copiar código

```
AttributeError: module 'pysnmp.entity.config' has no attribute 'addNotification'
```

es **muy común** cuando se está configurando el envío de traps o notificaciones en un agente SNMP con `pysnmp`.

Te explico qué significa y cómo solucionarlo fácilmente 📌

- Problemas con el puerto elegido:

esta ejecutandose en el 161 y no funciona

Perfecto, gracias — eso me da la pista clave 🔍

Si tu agente se está ejecutando en el **puerto 161** y el comando `snmpget` devuelve

`No Such Object available on this agent at this OID`, el problema más común en Windows es que **otro servicio** (por ejemplo, el **SNMP Service** de Windows o el **firewall**) está ocupando o bloqueando el puerto **161**.

Vamos paso a paso para confirmar y resolverlo 🙋

🧩 1 Comprueba si otro proceso está usando el puerto 161

Abre PowerShell como **Administrador** y ejecuta:

powershell

📋 Copiar código

`netstat -ano | findstr ":161"`



- Falta de especificación de oids en el código:

la mib la tenemos guardada en el mismo directorio que el programa mini agent, pero como sabe que tiene que coger esos oids? puede venir el programa de ahí?

Excelente pregunta 🙋 — y muy importante para entender cómo funciona tu mini-agente SNMP.

La respuesta corta es: **no**, tu programa *no "lee" automáticamente* la MIB que tienes en el mismo directorio.

El `mini_agent.py` **no usa la MIB** para saber los OIDs: esos OIDs se definen *manualmente dentro del código* (en tu variable `STORE` dentro de `load_state()`).

🧩 Te explico con detalle:

1 Qué hace la MIB (`MYAGENT-MIB.txt`)

La MIB es simplemente una **descripción textual y formal** (SMIv2) de tus objetos gestionados:

asn1

📋 Copiar código

```
manager OBJECT-TYPE
    SYNTAX      DisplayString (SIZE(1..64))
    MAX-ACCESS  read-write
    STATUS      current
```

- Errores con diferentes versiones de pysnmp

adaptame ese codigo para pysnmp 7.1.4

Excelente — esto es justo lo que toca hacer ahora 🍌

Tu código original fue escrito para **pysnmp 4.x**, y ya lo habías ido adaptando parcialmente a 5.x (cambiando cosas como `addV1System` → `add_v1_system` y añadiendo `snmpContext`).

👉 Pero **pysnmp 7.1.4** cambia todavía más:

la API se moderniza completamente — ya no usa `config`, `cmdrsp`, ni `NotificationOriginator` como antes. Ahora todo se basa en clases de alto nivel como `SnmpEngine`, `SnmpDispatcher`, `SnmpContext`, y decoradores `@command_responder`.

⚙️ Te explico brevemente qué cambia en pysnmp ≥7

pysnmp 4.x / 5.x

```
from pysnmp.entity import engine, config
```

```
cmdrsp.GetCommandResponder
```

pysnmp 7.x

```
from pysnmp.hlapi.v3arch.asyncio import SnmpEngine
```

```
@command_responder('get') decorator
```



- Falta de código para la persistencia de la MIB

el fichero `mib_state.json` que se debería ir actualizando no se actualiza

Perfecto 🍌 — ahora que ya tienes el agente funcionando y midiendo valores reales,

te falta el último detalle: que el fichero `mib_state.json` se actualice con los cambios del agente.

Tu código actual (`mini_agent_7.py`) no tiene ningún mecanismo para guardar en disco los cambios del diccionario `STORE`, así que aunque los valores cambien en memoria, no se escriben en el archivo.

Vamos a corregir eso.

🔧 1 Qué pasa ahora

Tu agente mantiene el estado en memoria con el diccionario:

python

Copiar código

```
STORE = {  
    "1.3.6.1.4.1.28308.1.1.0": ("DisplayString", "Admin"),  
    ...  
}
```



- Problemas con el guardado de valores en las direcciones de las oids



MYAGENT-MIB.txt
Documento

lo que hay en el oid 127.0.0.1:1161 1.3.6.1.4.1.28308.1.4.0 no se le asigna bien el valor, sacando todo el rato "", como lo arreglo sabiendo que la mib es esta

1.3.6.1.4.1.28308.1.4.0 (→ `cpuThreshold`) no está recibiendo ni guardando bien el valor y sale como "" (cadena vacía).

Sin embargo, en tu archivo `mini_agent_7.py`:

- Entonces, cuando lees el JSON, `cpuThreshold` puede quedar así:

```
"1.3.6.1.4.1.28308.1.4.0": ["Integer32", ""]
```

[illegible]

El error que te sale ahora:

```
NoSuchInstanceError: (name=(1, 3, 6, 1, 6, 3, 13, 1, 1, 2, 112, 117, 98, 108, 105, 99, 97, 114, 10),)
Target param not configured at LCD
```

Tú estás llamando:

```
ntfOrg.sendVarBinds(
```



mini_agent_7 (2).py
Python

modificame este codio para que el envio del correo cuando se supera el umbral de cpu se haga a traves de servidores de google

Perfecto 🙌 Veo que tu código `mini_agent_7 (2).py` ya implementa el envío de trap cuando el uso de CPU supera el umbral, pero **no envía todavía el correo electrónico**.

Vamos a modificarlo para que **envíe el correo mediante los servidores SMTP de Google (Gmail)** cuando se detecte ese cruce de umbral.

Aquí tienes la versión adaptada —solo necesitas rellenar tu cuenta y contraseña de aplicación de Gmail (no la normal, sino una [App Password](#) 🡕):

```
python
import asyncio
import psutil
import time
import json
import os
import smtplib, ssl
from email.message import EmailMessage
```

6. CONFIGURACIÓN CORRECTA DE PUBLIC Y PRIVATE

revisame que se cumpla esto

GET/GETNEXT with -c public can read but not write.

- SET with -c private can write (subject to your own per-object RO/RW checks).

Perfecto ✅ — vamos a verificar paso a paso que tu **mini agente SNMP** cumple exactamente con lo que pide la práctica en ese punto:

“GET/GETNEXT with -c public can read but not write.

SET with -c private can write (subject to your own per-object RO/RW checks).”

1 Configuración de comunidades y seguridad (VACM)

En tu código tienes:

```
python
config.addV1System(snmp_engine, 'public-area', 'public')
config.addV1System(snmp_engine, 'private-area', 'private')

for secModel in (1, 2):
    config.addVacmUser(
        snmp_engine, secModel, 'public-area', 'noAuthNoPriv',
        readSubTree=(1, 3, 6, 1)
```

7. PRUEBAS DE MANEJO CORRECTO DE EXCEPCIONES

Tras probar comandos como `snmpget -v2c -c public 127.0.0.1:1161 1.3.6.1.4.1.28308.1.1.0` para comprobar el buen funcionamiento (de igual manera con `getnext` y `set`), se fuerzan

errores para comprobar que se identifican los errores correctos. Surge alguna duda y error que se pregunta a la IA:



mini_agent (4).py
Python



Assignment 1 NM course.pdf
PDF

revisame las excepciones que saltan en caso de error para que salte lo que me dice en la practica en la seccion final de test de errores

Perfecto He revisado tu código `mini_agent (4).py` y lo he comparado con los requisitos de tu práctica en el PDF *"Assignment 1 NM course"*.
Tu implementación de los manejadores SNMP (`MiniGet`, `MiniGetNext`, `MiniSet`) ya cumple casi todo, pero hay que ajustar las excepciones y códigos de error para que los tests de error de la práctica (sección 2.7.4 del PDF `Assignment 1 NM course`) pasen correctamente.

En la práctica se piden exactamente estos errores:

Tipo de error	Situación	Código SNMPv2	Respuesta esperada
noAccess (6)	OID fuera de tu árbol manejado	6	OID no gestionable
wrongType (7)	Tipo incorrecto (ej. string en vez de entero)	7	"Wrong Type"
wrongValue (10)	Valor fuera de rango, cadena demasiado larga, etc.	10	"Wrong Value"
notWritable (17)	Intentar escribir un objeto de solo lectura	17	"Not Writable"

8. REALIZACIÓN TEST DE PRUEBA

Realízame un test de pruebas como me piden en la práctica, debes probar a ejecutar todo tipo de comandos posibles para que se prueben todas las funciones. Además debes escribir comandos que fuercen errores para capturar el error y ver si es el error deseado.
A continuación te indico el estilo de comandos que se deben ejecutar al generar el test:

```
snmpget -v2c -c public 127.0.0.1:1161 1.3.6.1.4.1.28308.1.3.0
snmpset -v2c -c public 127.0.0.1:1161 1.3.6.1.4.1.28308.1.1.0 s
"NoDeberia"
snmpset -v2c -c private 127.0.0.1:1161 1.3.6.1.4.1.28308.1.1.0 s
"CarlayArancha"
snmpset -v2c -c private 127.0.0.1:1161 1.3.6.1.4.1.28308.99.0 i 10
```

Aquí tienes un **script de pruebas completo**, totalmente acorde a lo que pide tu práctica, que ejecuta **todo tipo de comandos**, incluyendo **errores forzados**, **GET**, **GETNEXT**, **SET correctos**, **SET incorrectos**, **SNMPWALK**, y valida que las respuestas son correctas.

El script:

- Ejecuta los comandos usando `subprocess`.
- Captura salida y errores.
- Imprime cada prueba con un marco elegante y ordenado.
- Comprueba si el error devuelto es el esperado.
- Permite que añadas o quites pruebas fácilmente.

✓ SCRIPT DE PRUEBAS COMPLETO (`test_agent.py`)

Copia y pega este archivo directamente:

```
python
import subprocess
import sys
...
```

Copiar código

Conclusión

La asistencia de ChatGPT fue fundamental para estructurar y depurar el proyecto. Gracias a sus explicaciones y ejemplos, hemos podido entender conceptos complejos de SNMP, verificar correctamente los valores de los objetos de la MIB, implementar la lógica de los handlers SET y GETNEXT, y asegurar la persistencia del estado del agente.

Este proceso no solo nos ayudó a generar el código, sino que también nos permitió aprender de manera guiada, clarificando dudas y reforzando nuestra comprensión sobre la programación de agentes SNMP en Python.