Proyecto Final

Monitoreo de Recursos

James León 00213782

Pamela Pupiales 00213871

Se detalla el proceso de construcción de una aplicación gráfica de monitoreo de sistema que cumple con: utilización del CPU, memoria RAM, almacenamiento, listado de Procesos en Ejecución y sus características, utilización del ancho de banda de la red.

Visualización en el Tiempo: en estado actual, recopilación de información temporal de la última hora de ejecución.

La aplicación se ha desarrollado utilizando el lenguaje de programación Python y varias bibliotecas que se explicaran a continuación.

Dependencias

psutil: Una biblioteca multiplataforma para obtener información sobre la utilización del sistema, incluido el uso de CPU, memoria, discos, redes, etc.

tkinter: Una biblioteca estándar de Python para crear interfaces gráficas.

matplotlib: Una biblioteca para crear visualizaciones en 2D en Python.

ttkbootstrap: Una biblioteca que proporciona temas y estilos adicionales para la interfaz gráfica de Tkinter.

```python
import psutil
import tkinter as tk
from tkinter import scrolledtext, ttk
from threading import Thread
from time import sleep
from collections import deque
from ttkbootstrap import Style
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import queue
```

Resumen de Funcionalidad

- Utilización del CPU

psutil.cpu_percent() tiene la funcionalidad de obtener el porcentaje de uso del CPU en tiempo real.

- Utilización de Memoria RAM

psutil.virtual_memory().percent tiene la funcionalidad de obtener el porcentaje de uso de la memoria RAM en tiempo real.

- Utilización del Almacenamiento

psutil.disk_usage('/') tiene la funcionalidad de obtener el porcentaje de uso del almacenamiento en tiempo real.

Para las 3 funciones se implementa una gráfica que muestra el historial de uso en la última hora.

- Listado de Procesos en Ejecución

psutil.process_iter() tiene la funcionalidad de obtener información sobre los procesos en ejecución. Se muestra una lista de los procesos que incluye el nombre, identificador del proceso, estado, cantidad de memoria RAM y cantidad de disco utilizado.

- Utilización del Ancho de Banda de la Red

psutil.net_io_counters() tiene la funcionalidad de obtener información sobre la utilización del ancho de banda de la red. Se muestra el número de bytes enviados y recibidos en la red en tiempo real.

- Visualización en el Tiempo

Se implementa la actualización periódica de las diferentes métricas y gráficas para reflejar la información en tiempo real. Se recopila la información de la última hora y se muestra en las gráficas.

- Hilos

Se utilizan hilos para actualizar los componentes visuales de forma independiente y permitir la interacción con el usuario. Se utiliza la biblioteca threading para crear y controlar los hilos de actualización.

Clase y métodos

- Clase SystemMonitor

SystemMonitor representa la ventana principal de la aplicación del Monitor de Sistema. Hereda de la clase tk.Tk de la biblioteca Tkinter y contiene varios métodos para inicializar y actualizar la interfaz gráfica, así como para manejar las actualizaciones de recursos.

__init__() se encarga de inicializar la ventana y establecer el diseño de la interfaz de usuario. Aquí se configuran las etiquetas de recursos, la información del sistema, el área de texto y las figuras para mostrar los gráficos.

setup_figure() se utiliza para crear y configurar una figura de Matplotlib que muestra el historial de uso de recursos. Recibe parámetros title, xlabel y ylabel para personalizar los títulos y etiquetas del gráfico.

check_queue() verifica periódicamente si hay tareas en la cola de actualización de la interfaz de usuario. Si se encuentran tareas en la cola, se ejecutan para actualizar la

interfaz de usuario, este método es importante porque permite que las actualizaciones de la interfaz de usuario se realicen desde otros hilos.

- Métodos de actualización de recursos

update_cpu(): Actualiza periódicamente el uso de CPU obteniendo la información mediante psutil.cpu_percent(). Actualiza la etiqueta correspondiente y agrega el valor al historial de uso de CPU.

update_ram(): Actualiza periódicamente el uso de RAM obteniendo la información mediante psutil.virtual_memory().percent. Actualiza la etiqueta correspondiente y agrega el valor al historial de uso de RAM.

update_disk(): Actualiza periódicamente el uso del disco obteniendo la información mediante psutil.disk_usage('/').percent. Actualiza la etiqueta correspondiente y agrega el valor al historial de uso del disco.

update_network(): Actualiza periódicamente las estadísticas de red, incluyendo la cantidad de datos enviados y recibidos. Utiliza psutil.net_io_counters() para obtener las estadísticas y actualiza la etiqueta.

- Métodos de actualización de información del sistema

update_processes(): Actualiza periódicamente la lista de procesos en ejecución obteniendo la información mediante psutil.process_iter(['pid', 'name', 'memory_info']). Formatea los datos y los muestra en el área de texto correspondiente.

update_threads(): Actualiza periódicamente el recuento de hilos obteniendo la información mediante psutil.Process().num_threads()

update_processes_count(): Actualiza periódicamente el recuento de procesos obteniendo la información mediante len(psutil.pids())

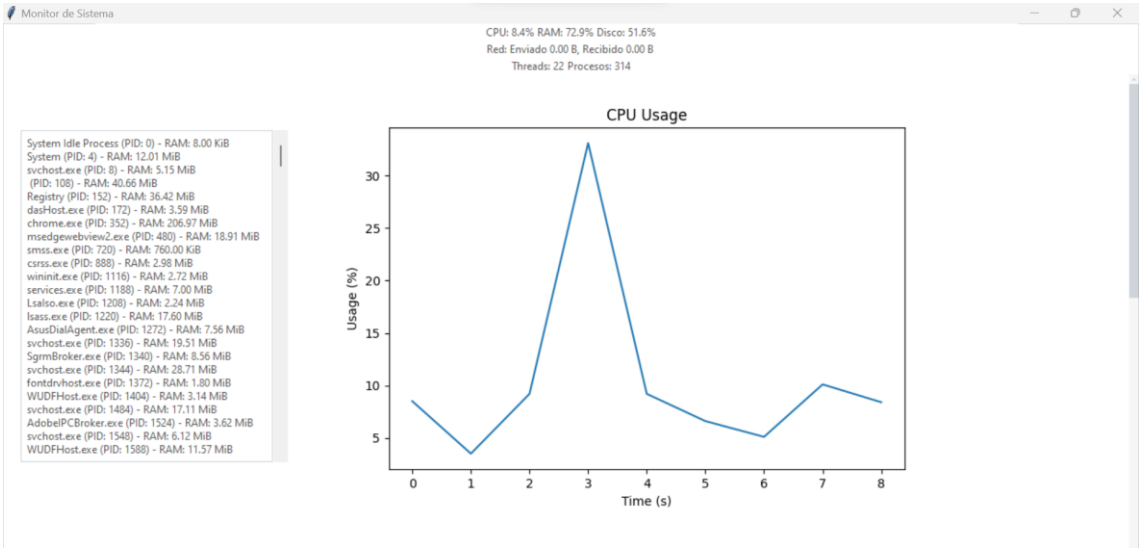
update_graph() es para actualizar los gráficos de historial de uso de recursos. Recibe el historial de uso, el objeto de trazado (plot), el lienzo (canvas), así como los títulos y etiquetas correspondientes. Este método borra el gráfico actual, establece los títulos y etiquetas adecuados, traza el nuevo historial y actualiza el lienzo.

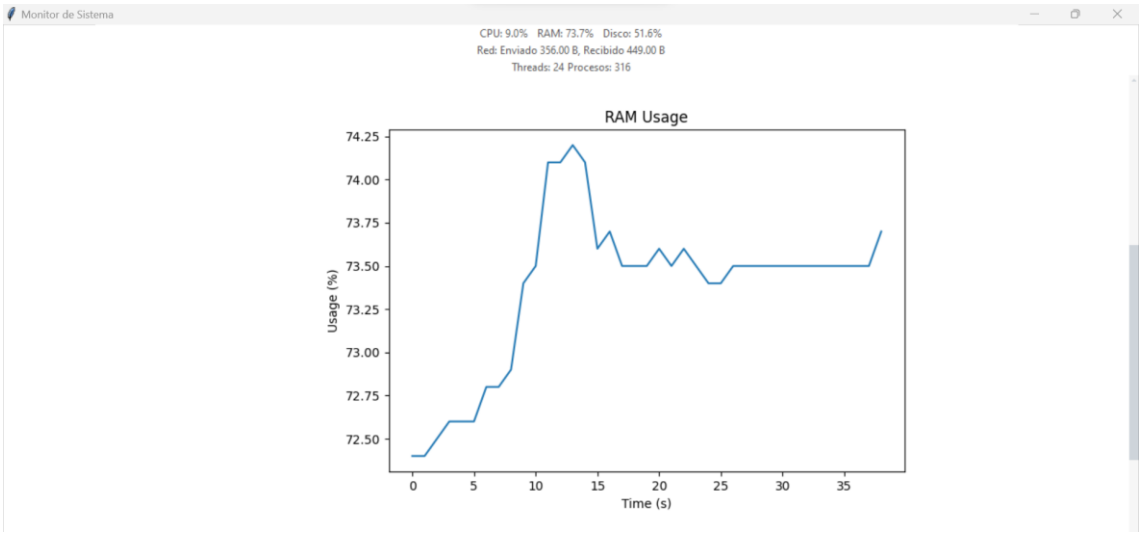
format_bytes() es para formatear una cantidad de bytes en unidades de tamaño (KB, MB, GB). Recibe el tamaño en bytes y realiza la conversión y el formateo correspondiente antes de devolver la cadena formateada.

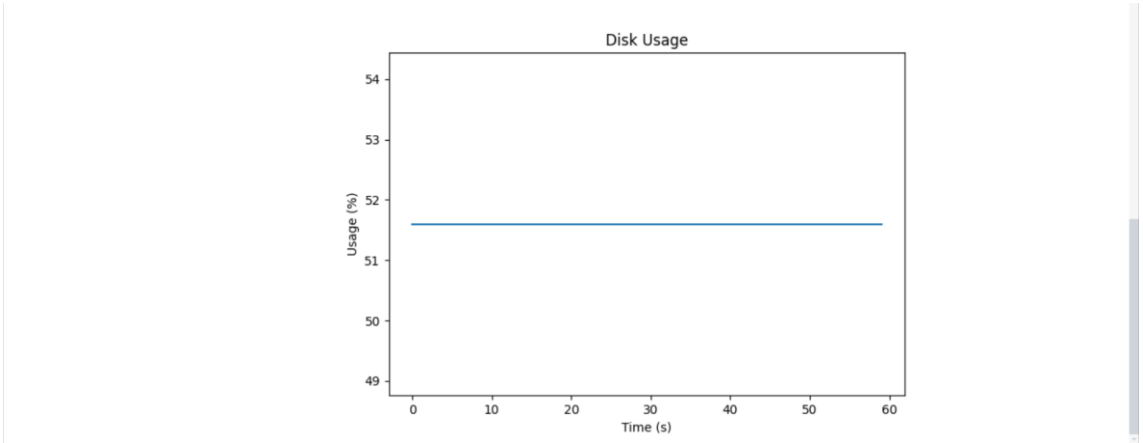
stop_threads() se llama cuando se cierra la ventana de la aplicación. Establece la variable running en False, lo que detiene la ejecución de los hilos y cierra la ventana.

main crea una instancia de la clase SystemMonitor y comienza el bucle principal de la aplicación llamando al método mainloop().

Funcionamiento







Aportes de los Integrantes

James: Implementación de la visualización del uso del CPU y la memoria RAM, actualización de las métricas y gráficas correspondientes, diseño de la interfaz de usuario y elaboración del informe.

Pamela: Implementación de la visualización del almacenamiento y del ancho de banda de la red, también recopilación de información histórica, representación en las gráficas correspondientes, correcciones del diseño de la interfaz y elaboración del informe

Pruebas Realizadas

Durante el proceso de desarrollo de la herramienta, se realizaron pruebas exhaustivas para garantizar su correcto funcionamiento.

- Precisión de las métricas de uso del CPU, memoria RAM, almacenamiento y ancho de banda de la red.
- Comprobación de la actualización adecuada de las métricas y gráficas en tiempo real.
- Estabilidad y rendimiento de la aplicación al ejecutarla durante un período prolongado.
- Pruebas de interacción con la interfaz de usuario para garantizar una experiencia fluida y sin errores.

Conclusiones

En conclusión, el proceso de construcción de la herramienta de monitoreo de recursos ha sido exitoso. Se lograron implementar todas las funcionalidades requeridas, permitiendo capturar y visualizar información relevante sobre la utilización del CPU, memoria RAM, almacenamiento, procesos en ejecución y ancho de banda de la red. Además, se implementó la capacidad de visualizar la información en el tiempo, incluyendo el historial de uso en la última hora. La aplicación se desarrolló utilizando hilos para actualizar los componentes visuales de forma independiente y permitir la interacción con el usuario. Asimismo, se realizaron pruebas exhaustivas para garantizar su correcto funcionamiento.

En resumen, la herramienta de monitoreo de sistema construida cumple con los requisitos establecidos y proporciona una interfaz intuitiva y eficiente para obtener información detallada sobre el rendimiento del sistema en tiempo real.