Monitor de Recursos del Sistema

Proyecto Final de Curso - Sistemas Operativos

Jiménez Sanchez Emma Alicia

Salazar Gonzalez Pedro Yamil

Sanchez Cruz Norma Selene

Suárez Ortiz Joshua Daniel

Objetivo del Proyecto

Herramienta de Diagnóstico

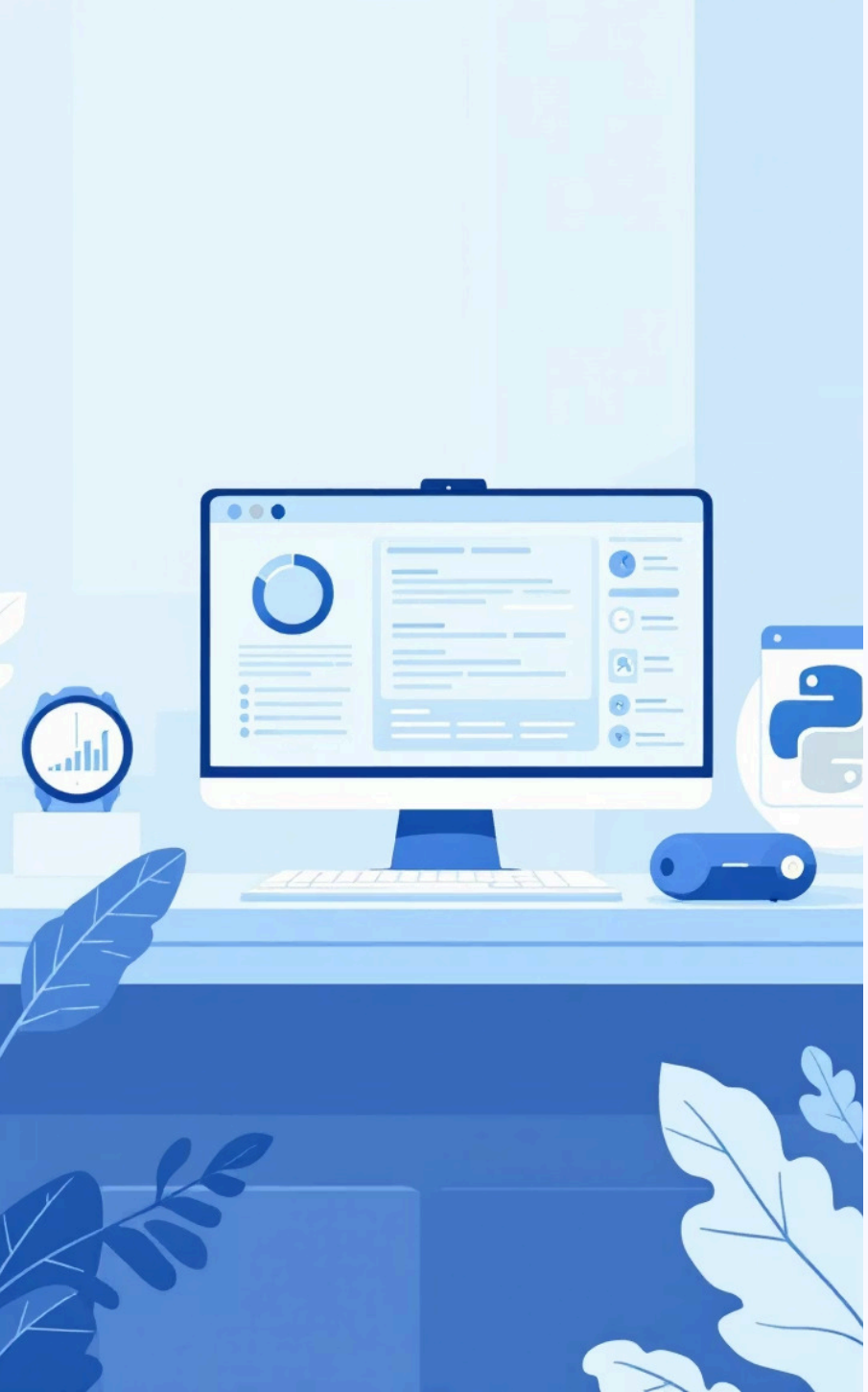
Desarrollar una solución ligera y portable para monitoreo de recursos del sistema, eliminando dependencias pesadas y facilitando el despliegue en múltiples entornos.

Visualización en Tiempo Real

Desafío técnico principal: implementar actualización dinámica de métricas en la terminal sin comprometer la experiencia del usuario ni la fluidez de la interfaz.

Eficiencia de Recursos

Diferenciador clave: consumo de memoria y CPU significativamente menor comparado con soluciones GUI tradicionales, ideal para sistemas embebidos o servidores.



Stack Tecnológico

Backend: Recolección de Datos

Python 3.x

Lenguaje principal seleccionado por su ecosistema maduro para scripting de sistemas y manipulación de datos en tiempo real.

Psutil

Librería multiplataforma que proporciona acceso directo a syscalls del sistema operativo. Permite lectura eficiente de CPU, memoria, disco, red y sensores térmicos.

Frontend: Interfaz TUI

Textual Framework

Framework moderno para construcción de TUIs reactivas con soporte para CSS, widgets componibles y manejo de eventos.

Características clave: arquitectura asíncrona, renderizado optimizado y sistema de layout responsive similar a tecnologías web.

Funcionalidades

- **CPU:** Desglose por núcleos (Físicos/Lógicos) y Frecuencia en MHz.
- **Memoria:** Uso de RAM y estado de SWAP (Paginación).
- **Disco:** Detección de particiones físicas y virtuales (Snap/Loop).
- **Red:** Velocidad de subida/bajada en tiempo real (Cálculo por Deltas).
- **Temperaturas:** Sensores de hardware (NVMe, GPU, Placa Base).
- **Procesos:** Top consumidores de CPU e identificación de aplicaciones.

Monitor.py - Métodos de Recopilación de Datos

1

clear()

Limpia la consola usando cls (Windows) o clear (UNIX/Linux/MacOS).

2

bytes_to_gb(b)

Función auxiliar que convierte bytes a gigabytes para mejor visualización del consumo de recursos.

3

show_system_info()

Muestra información general del sistema:

- Nombre del sistema operativo
- Versión del sistema operativo
- Arquitectura
- Procesador del equipo
- Tiempo del sistema

4

show_cpu()

Información detallada del CPU:

- Uso total del CPU
- Número de núcleos físicos y lógicos
- Frecuencia actual y máxima
- Uso por núcleo

Permite identificar problemas de rendimiento y saturación del procesador.

5

show_memory()

Estado en tiempo real de RAM y SWAP:

- Total de RAM
- Disponibilidad de RAM
- Porcentaje usado
- Total y porcentaje de SWAP

Previene fallos, ralentizaciones y asegura estabilidad del sistema.

6

show_disk()

Monitoreo de particiones montadas:

- Nombre de la partición
- Espacio total
- Porcentaje usado y libre

Previene fallos críticos, corrupción de archivos e interrupciones de procesos.

7

show_network()

Tráfico de red y velocidades:

- Subidas y bajadas totales
- Velocidades de subida y bajada

Garantiza estabilidad, detecta comportamientos sospechosos y mejora la seguridad.

8

show_temperatures()

Temperatura de sensores del sistema (si disponible):

- Sensor y temperatura en °C

Mantiene estabilidad, previene daños y facilita diagnóstico de problemas térmicos.

9

show_processes()

Top 10 procesos ordenados por uso de CPU:

- PID
- Nombre
- Porcentaje de CPU
- Porcentaje de RAM

Identifica procesos que consumen recursos en exceso y optimiza el rendimiento.

Aplicación Principal: `main()`

La función `main()` actúa como el orquestador central, unificando todos los métodos de recopilación de datos previamente descritos para ofrecer un monitoreo integral del sistema operativo en tiempo real. Este código transforma un monitor de sistema basado en consola en una aplicación interactiva con interfaz de usuario en terminal (TUI) utilizando el framework Textual.

Funcionalidades Clave

- **Interfaz con Pestañas:** Muestra diferentes métricas (Sistema, CPU, Memoria, Disco, Red, Temperaturas y Procesos) organizadas en pestañas claras.
- **Actualización Automática:** Refresca la información del sistema cada segundo para una visión en tiempo real.
- **Reutilización Eficiente:** Integra y adapta las funciones existentes del módulo `Monitor` optimizadas para la terminal.

Permite visualizar la última hora de actualización de los datos, asegurando que la información sea siempre reciente.

Resultados y Validación

El proyecto cumple exitosamente con los objetivos planteados: proporciona una herramienta de diagnóstico eficiente, portable y con excelente experiencia de usuario, demostrando la viabilidad de Python para aplicaciones de sistemas en tiempo real.

Demostración

Conclusiones y Trabajo Futuro

01

Viabilidad de Python para Sistemas en Tiempo Real

El proyecto valida que Python, correctamente optimizado, es una opción viable para herramientas de sistema con requisitos de baja latencia y alto rendimiento.

02

Mejoras Propuestas: Persistencia

Implementar logging estructurado a archivo (JSON/CSV) para análisis histórico de métricas y detección de patrones de uso a largo plazo.

03

Sistema de Alertas

Módulo de notificaciones configurables: envío de alertas por correo o webhooks cuando se superen umbrales críticos (temperatura, uso de disco, etc.).

04

Arquitectura Cliente-Servidor

Extensión del monitor para soporte remoto mediante WebSockets, permitiendo supervisión de múltiples hosts desde una consola centralizada.