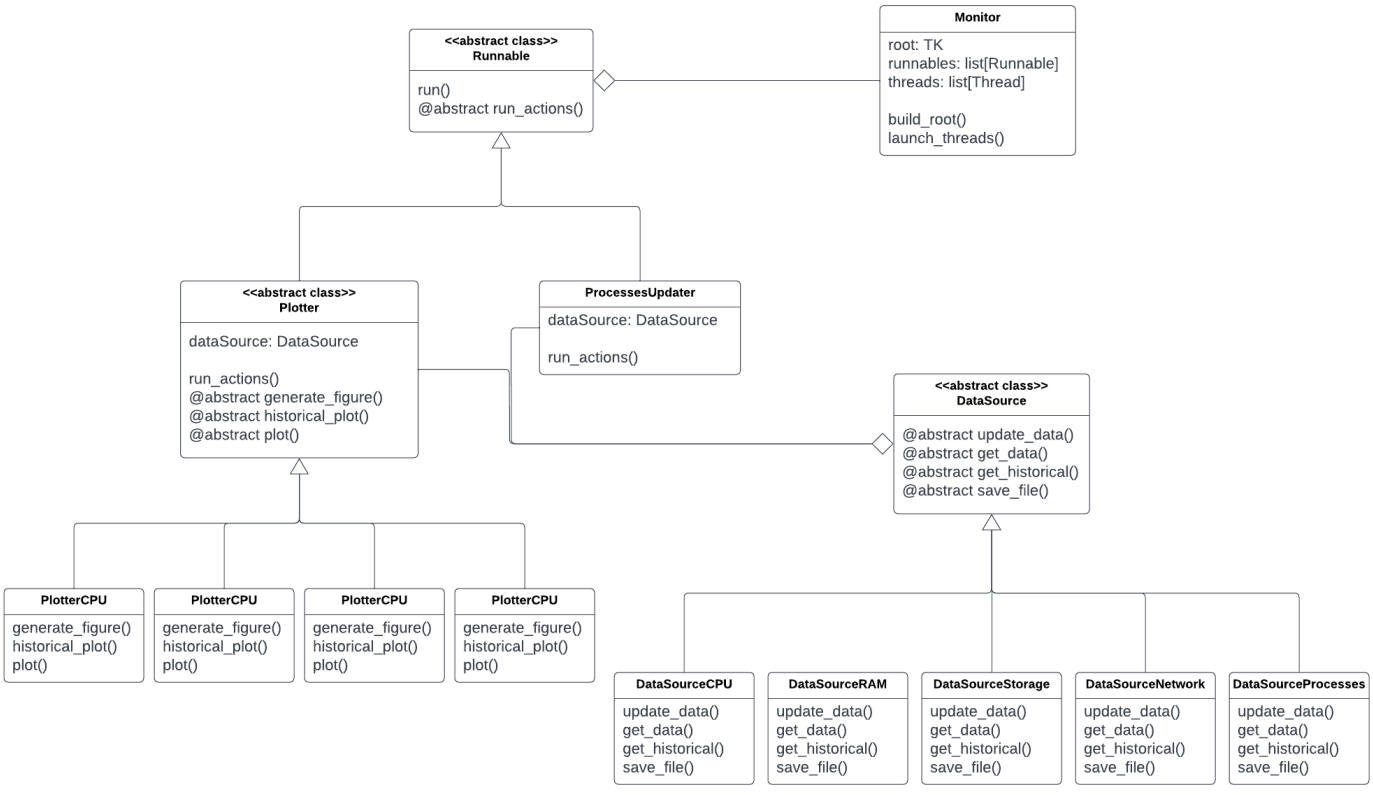
Universidad San Francisco de Quito
Sistemas Operativos
Carlos Montiel y Alejandro Duque

Proyecto Final - Monitoreo de Recursos

El objetivo del monitor de recursos es representar de forma gráfica y textual los diferentes recursos de la computadora: CPU, Memoria Principal, Memoria Secundaria, Swap, y Procesos. Se segmenta gráficamente mostrando la lista de procesos, curva de uso de CPUs, porcentaje de CPU y memoria usados, así como porcentaje de memoria swap usado y finalmente curva de tiempo y velocidad en uso de red. Se optó por realizar un diseño orientado a objetos para modularizar la implementación.

Arquitectura de la solución



Para desarrollar el monitor de recursos se utilizó Python, juntos con las siguientes bibliotecas: TKinter para la interfaz gráfica, threading para la gestión de hilos, matplotlib para la generación de gráficas, pandas para gestionar el modelo de datos y psutil para recuperar la información del sistema operativo.

La solución implementada es la de un esquema MVC. La clase Monitor corresponde a la vista, los DataSource son los modelos de datos y los Runnable son los controladores. La jerarquía de clases sigue un diseño modular basado en clases abstractas que describen el comportamiento general, y clases concretas que implementan la funcionalidad específica para cada tipo de recurso. Las clases abstractas son DataSource, Runnable y Plotter.

La clase DataSource funciona como una interfaz que define las operaciones de actualización, recuperación y guardado de datos. Cada DataSource concreto tiene métodos privados que encapsulan las funciones de psutil y devuelven valores adecuados según la estadística del recurso que se requiera. La información de cada DataSource se guarda dentro de DataFrames de pandas. Los DataFrames se actualizan cada vez que se llama a la función updata_data. Al usar DataFrames se facilita la comunicación entre el modelo de datos y el controlador, ya que éstos son compatibles con matplotlib.

La clase Runnable implementa el patrón Template Method en el método run, que se encarga de llamar a run_actions en un bucle cada cierto tiempo. En run_actions se debe actualizar el modelo de datos y reflejar el cambio en la interfaz. La clase Plotter también implementa el patrón Template Method en el método run_actions, que plotea el estado actual o el histórico del recurso monitoreado según corresponda. En cada Plotter se realiza la implementación concreta de la generación de la figura para plotear, así como del mismo ploteo.

Flujo del programa

El programa inicia con la construcción de la interfaz gráfica por parte del monitor. Una vez creada la interfaz se lanzan 5 threads que gestionan de forma independiente cada frame de la pantalla. Los threads son controlados por su respectivo runnable, y su control consiste en que cada dos segundos recuperan datos del datasource y actualizan un frame de la interfaz con esta nueva información.

Contribuciones

Carlos Montiel: Implementación de las 4 clases que encapsulan las funciones que obtienen las estadísticas de psutil, es decir de los DataSource.

Alejandro Duque: Diseño de la arquitectura de la aplicación. Implementación de la interfaz gráfica, implementación de los Runnables y Plotters. Control de threads.

Bibliografía

Rodola, G. (2023). psutil documentation. Recuperado de: https://psutil.readthedocs.io/en/latest/

Matplotlib development team. (2019). Embedding in Tk. Recuperado de: https://matplotlib.org/3.1.0/gallery/user_interfaces/embedding_in_tk_sgskip.html