# MANUAL TECNICO

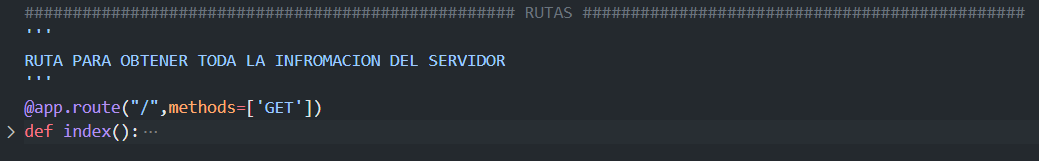
## API

La api fue desarrollada en Python, utilizando el framework Flask.

### Rutas

#### “/”

La ruta “/” es la encargada de obtener la información de los módulos y devolverlos en formato JSON



#### “/guardarOracion”

Esta ruta es la encargada de almacenar la oración y el usuario, en la base de datos respectiva a ese servidor



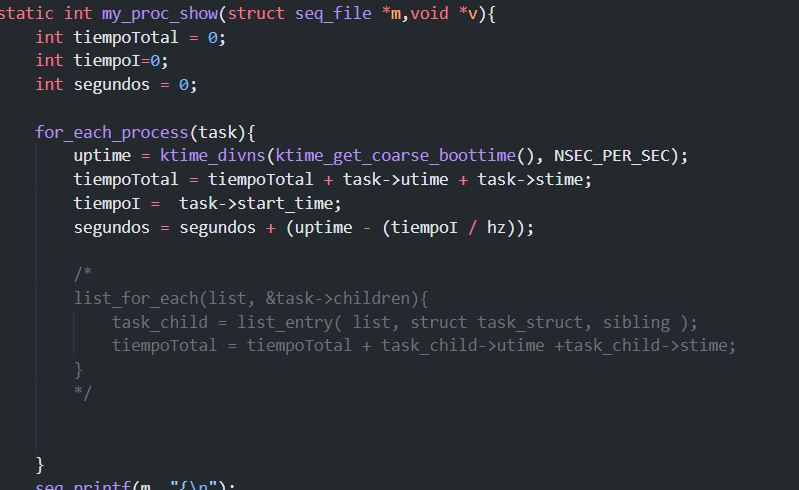
## MODULOS

Los módulos fueron desarrollados en el lenguaje de programación “C”, para almacenar al modulo dentro del kernel (temporalmente) se utilizo el comando “sudo insmod”, el comando “make” para compilarlo y generar los tipos de archivos necesarios

### CPU

Para el modulo del CPU se procedió a obtener el uptime de cada proceso y su tiempo final, así mismo, se fueron sumando en una variable de tiempo global, almacenando los segundos pasados durante toda la ejecución y los Hertz(100).

Aplicando la formula porcentaje= ((TiempoTotal/Hertz) / segundos) \* 100

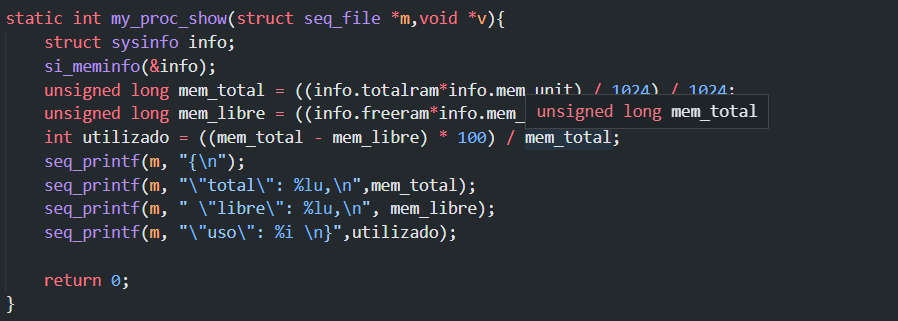


### MEMORIA RAM

Para el modulo de la memoria ram se hizo uso del struct sysinfo que obtiene la información total de la ram, como es el total de la ram, la memoria libre, así aplicando la formula de

Utilizado = ((memoriaTotal – memoriaLibre) \* 100) / memoriaTotal

Se puede obtener el porcentaje de utilización de la ram.

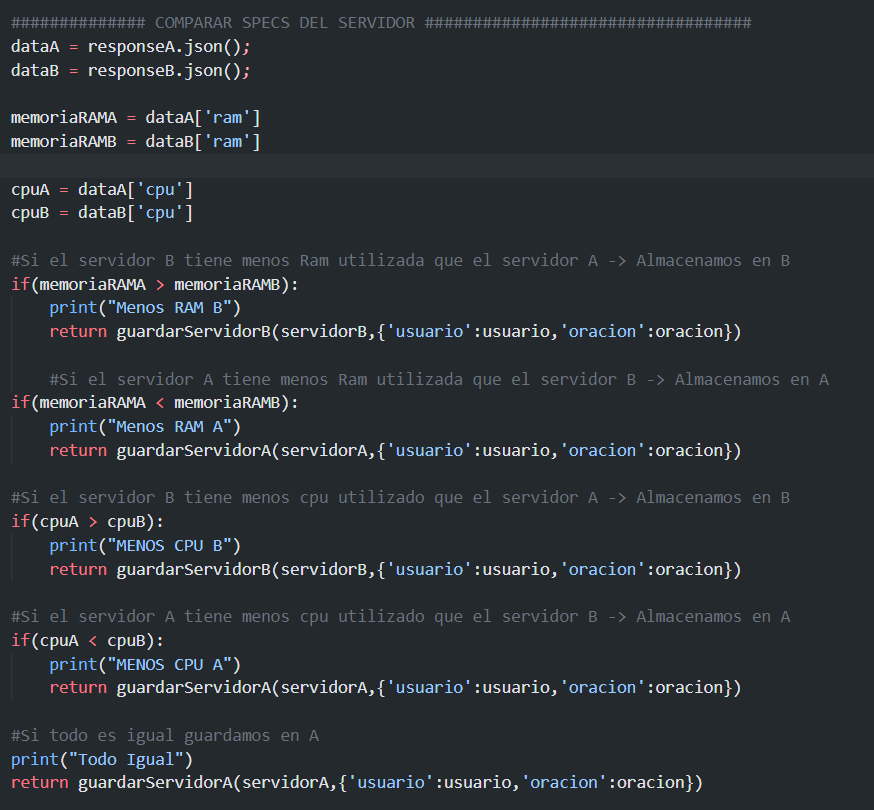


## BALANCEADOR

El balanceador fue desarrollado en el lenguaje Python, utilizando el framework Flask.

Para el balanceador se tomaron en cuenta los siguientes criterios en orden de prioridad:

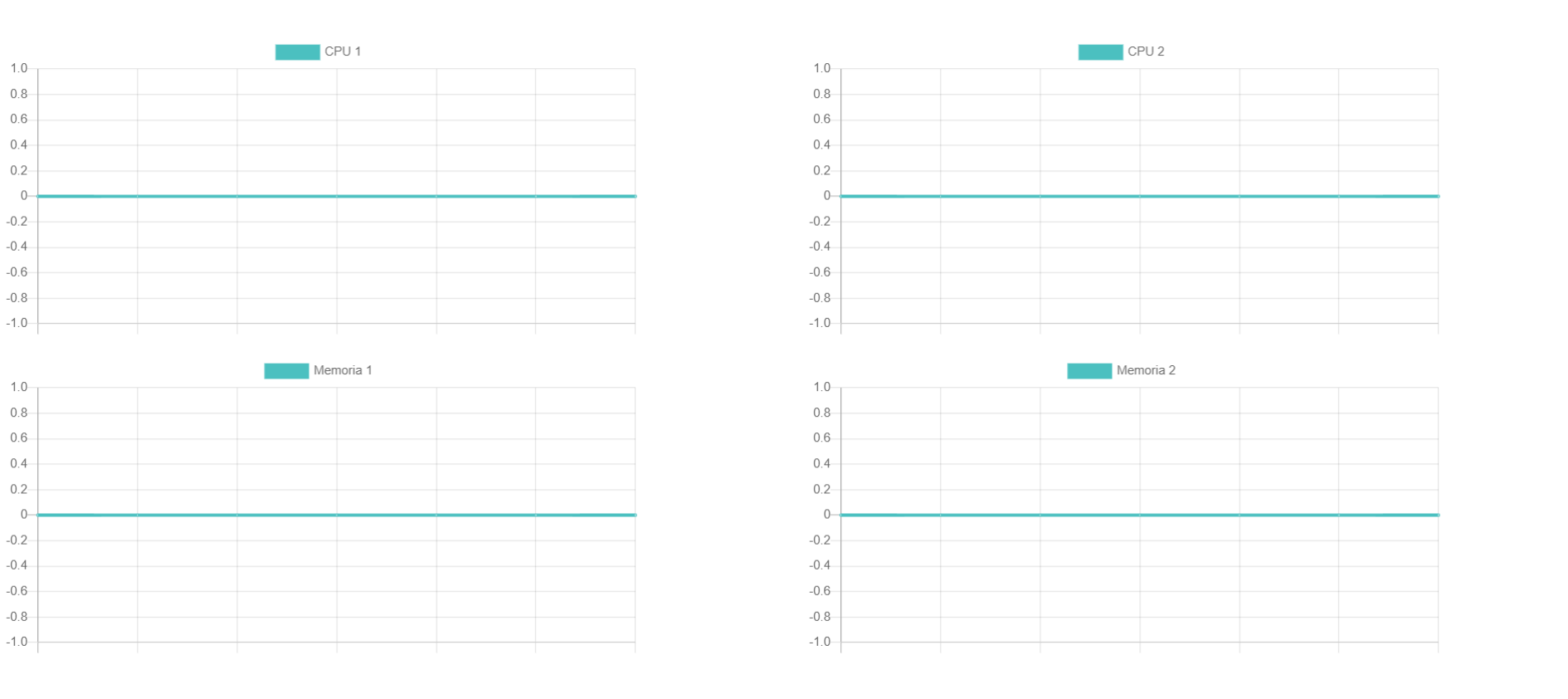
* Servidor A y B están caídos -> Error
* Servidor A esta caido -> Almacenar en B
* Servidor B esta caido -> Almacenar en A
* Servidor A tiene menos datos que Servidor B -> Almacenar en A
* Servidor B tiene menos datos que Servidor A -> Almacenar en B
* Servidor A tiene menos ram ocupada que Servidor B -> Almacenar en A
* Servidor B tiene menos ram ocupada que Servidor A -> Almacenamos en B
* Servidor A tiene menos cpu ocupado que Servidor B -> Almacenamos en A
* Servidor B tiene menos cpu ocupado que Servidor A -> Almacenamos en B
* En Defecto lo Almacenamos en A



## FRONTEND

Para el FrontEnd se utilizo el framework RectJS, para poder refrescar la pagina cada 5seg ya se en las oraciones o para las graficas, se utilizo el método setInterval, las graficas tenemos 7 puntos que se van llenando cada 5 seg. Aplicando el método de cola, sacando al inicio, ingresando al final.





## CLIENTE

Es una aplicación de consola desarrollada en Python, que nos permite ingresar la url del archivo a analizar que contiene todas las oraciones para hacer las peticiones, y la dirección URL donde realizaremos las peticiones. Enviando Oración por Oración devolviendo un mensaje de respuesta del servidor.

