UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS Y SISTEMAS

SISTEMAS OPERATIVOS 1 SECCIÓN A

ING. EDGAR RENE ORNELIS HOIL

AUX.DIEGO ALEJANDRO JUAREZ BRAN

ESCUELA DE VACACIONES JUNIO 2025

Proyecto Fase 1

Monitor de servicios Linux

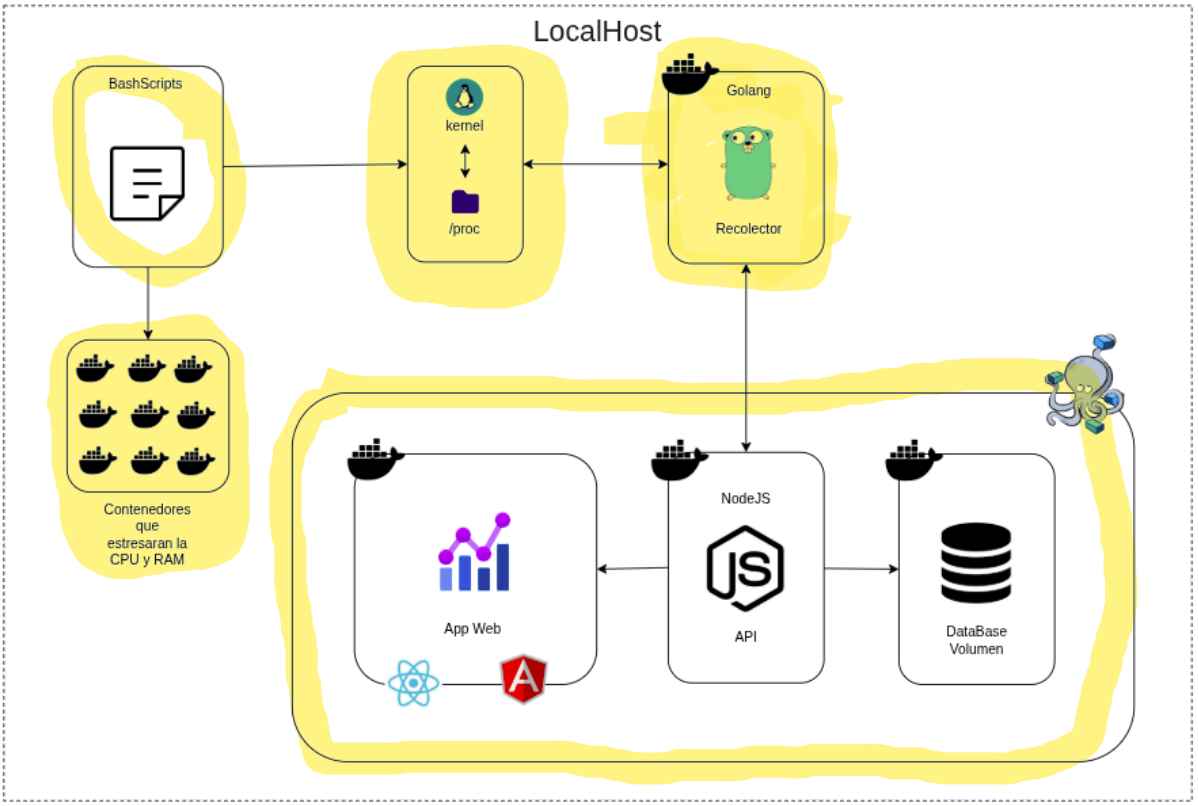
Objetivos

- Conocer el Kernel de Linux mediante módulos de C.
- Hacer uso del lenguaje de programación Golang para la gestión del sistema.
- Comprender el funcionamiento de los contenedores usando Docker.
- Comprender el funcionamiento de los scripts de bash para la automatización de procesos.

Introducción

El objetivo de este proyecto es aplicar todos los conocimientos adquiridos en la unidad 1, con la implementación de un gestor de contenedores mediante el uso de scripts, módulos de kernel, lenguajes de programación y la herramienta para la creación y manejo de contenedores más popular, Docker. Con la ayuda de este gestor de contenedores se podrá observar de manera más detallada los recursos y la representación de los contenedores a nivel de procesos de Linux y como de manera flexible pueden ser creados, destruidos y conectados por otros servicios.

Arquitectura



Con Bash se automatizan tareas, estas tareas se detallan más adelante, una vez cargados los módulos del kernel el agente de monitoreo (Golang) empezará a recopilar información sobre esos módulos y los parseara a formato JSON, la API en NodeJS realizará peticiones periódicamente hacia el agente de monitoreo para poder obtener métricas actualizas de los distintos módulos, estas métricas se enviarán al dashboard para la visualización de la data y a la base de datos para mantener un registro de todas las métricas.

BashScripting

Por medio de archivos bash se podrán realizar varias automatizaciones de tareas, entre estas tareas se encuentran:

- Creación de script para desplegar 10 contenedores a modo de estresar la CPU y memoria RAM.
- Creación de script para instalar y configurar los módulos del Kernel.
- Creación de script para eliminar todo los servicios utilizados.
- Creación de script para desplegar los contenedores de la app y docker-compose.

Monitorear

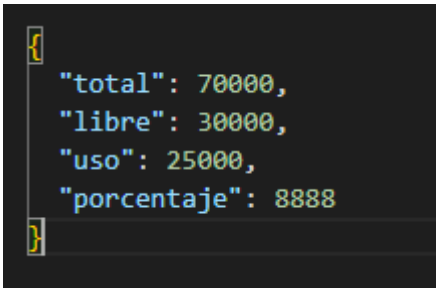
Kernel Modules

Módulo de RAM

Este módulo deberá de estar alojado en un archivo ubicado en el directorio /proc.

Características:
• Importar la librería < sys/sysinfo.h>, <linux/mm.h>
• La información que se mostrará en el módulo debe ser obtenida por medio de los struct de información del sistema operativo y no por otro medio.
• El nombre del módulo será: ram_<carnet>

Ejemplo de Json:



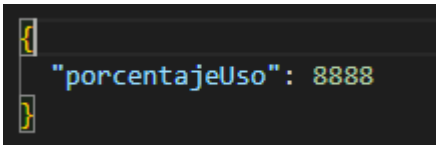
Módulo de CPU

Este módulo deberá de estar alojado en un archivo ubicado en el directorio /proc.

Características:

• Importar la librería <linux/sched.h>, <linux/sched/signal.h>
• La información que se mostrará en el módulo debe ser obtenida por medio de los struct de información del sistema operativo y no por otro medio.
• El nombre del módulo será: cpu_<carnet>

Ejemplo de Json:



Nota: Los módulos del kernel son la parte principal del proyecto por lo que es de suma importancia que se implemente esta parte.

Agente de Monitoreo

Este es un programa escrito en Golang y contenerizado.

Se encarga de realizar llamadas a los módulos de Kernel por medio de rutinas para obtener la información del estatus de CPU, RAM. Así mismo este enviará la información al API de NodeJS para que sean almacenadas en la base de datos.

El objetivo del agente de monitoreo es que comprender el uso concurrencia empleando rutinas y canales en el lenguaje de Golang

Plataforma de Monitoreo

Frontend

Se debe de crear una página Web que muestre:

Monitoreo en Tiempo Real

Se debe mostrar:

• Gráfica en Tiempo Real del porcentaje de utilización de la memoria RAM.
• Gráfica en Tiempo Real del porcentaje de utilización del CPU.

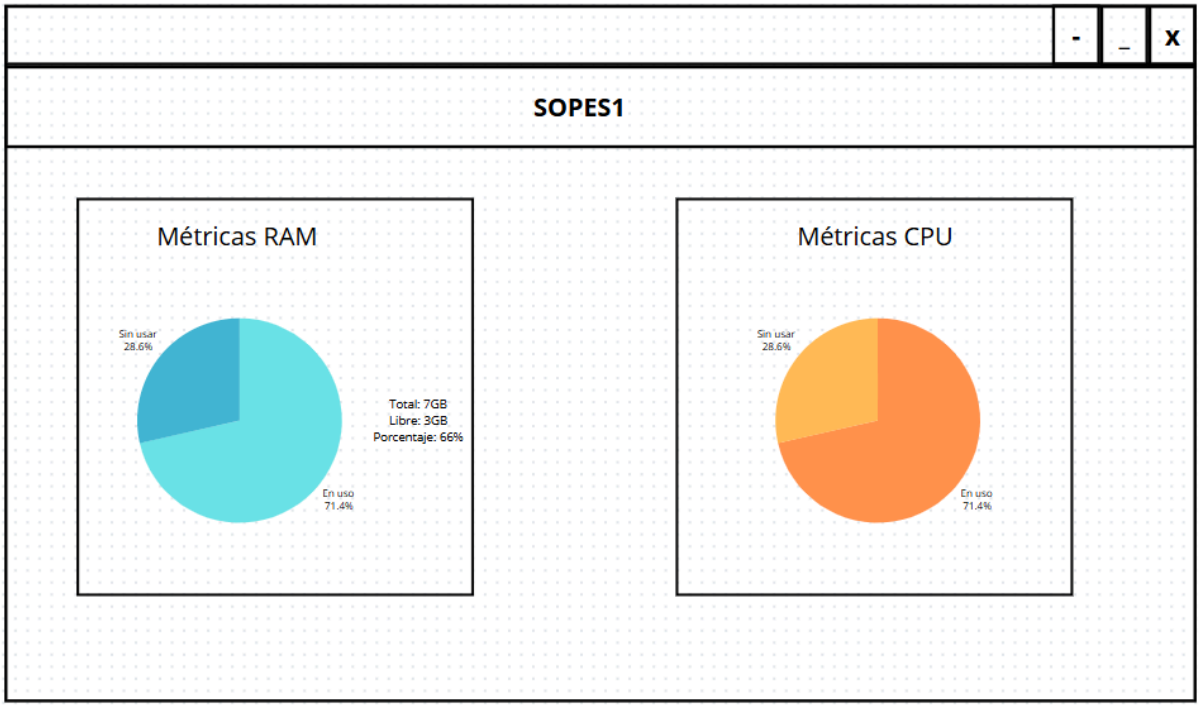


Imagen con fines ilustrativos

API NODEJS

Este servicio expone una API que posibilita la comunicación con la base de datos. Dicha API será empleada tanto para la lectura de información desde el Frontend como para la escritura de datos desde el monitoreo.

Base de datos

Se debe implementar una base de datos por medio de un contenedor de Docker, esto para guardar información de memoria ram y cpu. La base de datos debe de tener persistencia por lo que se requiere un Volumen de Docker para evitar que los datos se pierdan cada vez que el contenedor se reinicia.

Docker Compose

Para facilitar el despliegue de los contenedores de la Plataforma de Monitoreo se utilizará Docker Compose el cual permite gestionar múltiples contenedores en un solo bloque lo cual facilita la administración.

DockerHub

Se debe utilizar DockerHub para alojar las imágenes de los contenedores utilizados.

Restricciones

• El proyecto se realizará de forma individual.
• La obtención de la información se hará a través de los módulos de Kernel en C.
• El Agente de Monitoreo debe ser realizado con Golang.
• La API debe ser realizada con NodeJS.
• El Frontend debe ser realizado con cualquier framework como React, Angular, VueJS o cualquier otro.
• Las máquina local debe tener instalada una distribución de Ubuntu en su versión 20.04 o 22.04.
• Las imágenes de los contenedores deben de ser publicadas de DockerHub.

Github

- El código fuente debe de ser gestionado por un repositorio privado de Github
- Crea una Carpeta en el repositorio llamado: Proyecto1_Fase1
- Agregar al auxiliar al repositorio de GitHub: **dialjub19**

Calificación

- Al momento de la calificación se verificará la última versión publicada en el repositorio de GitHub
- El estudiante debe tener todo listo al momento de la calificación en caso contrario obtendrá una nota 0.
- No se permite la modificación o alteración de archivos o código de la aplicación a menos que el auxiliar lo permita o solicite.
- La calificación tendrá una duración de 15min.
- Se podrá solicitar explicaciones o alteraciones de código por medio del auxiliar para validar la autenticidad del proyecto presentado.
- Cualquier copia parcial o total tendrán nota de 0 puntos y serán reportadas al catedrático y a la Escuela de Ciencias y Sistemas.
- Si el proyecto se envía después de la fecha límite, se aplicará una penalización del 25% en la puntuación asignada cada 12 horas después de la fecha límite. Esto significa que, por cada período de 12 horas de retraso, se reducirá un 25% de los puntos totales posibles. La penalización continuará acumulándose hasta que el trabajo alcance un retraso de 48 horas (2 días). Después de este punto, si el trabajo se envía, la puntuación asignada será de 0 puntos.

Entregables

- La forma de entrega es mediante UEDI subiendo el enlace del repositorio.
- Manual técnico en formato Mark Down o PDF.

Fecha de Entrega

- Domingo 15 de junio de dos mil veinticinco (15/06/2025) antes de las 23:59 hrs.
- **La Calificación es en modalidad virtual.**