UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS Y SISTEMAS

SISTEMAS OPERATIVOS 1 SECCIÓN N

ING. JESUS GUZMAN POLANCO

AUX. JOSÉ DANIEL VELÁSQUEZ OROZCO

AUX. JHONATHAN DANIEL TOCAY

SEGUNDO SEMESTRE 2023

PROYECTO 1

Plataforma de Monitoreo y Señales a Procesos

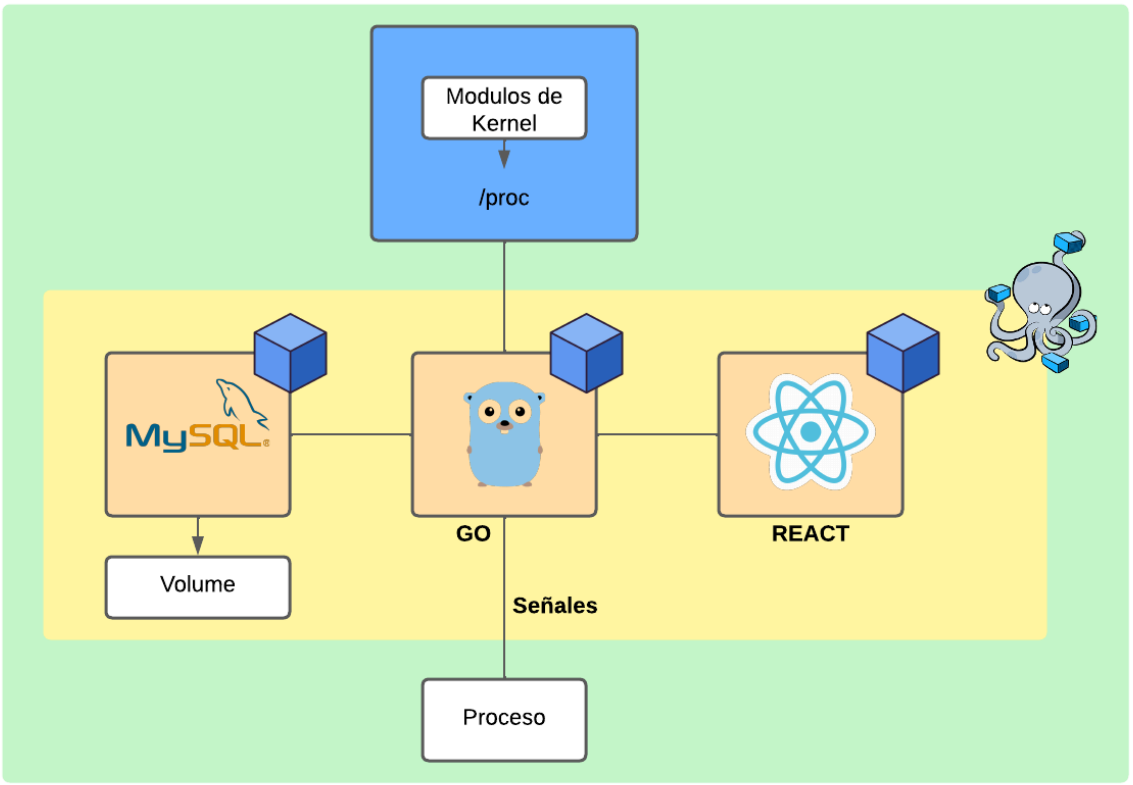
Objetivos

- Conocer el Kernel de Linux mediante módulos de C.
- Hacer uso de programación asíncrona con rutinas de Golang.
- Comprender el funcionamiento de los contenedores usando Docker.

Introducción

En este proyecto, se tiene como objetivo principal implementar un sistema de monitoreo de recursos del sistema y gestión de procesos, empleando varias tecnologías y lenguajes de programación. El sistema resultante permitirá obtener información clave sobre el rendimiento del computador, procesos en ejecución y su administración a través de una interfaz amigable.

Arquitectura



VM a Monitorear

El proyecto se deberá implementar en una máquina virtual montada en vmware o virtual box, la distribución que se utilizara será Ubuntu Server 22.04.

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_so1_1s2024**

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_so1_1s2024**

Plataforma de Monitoreo

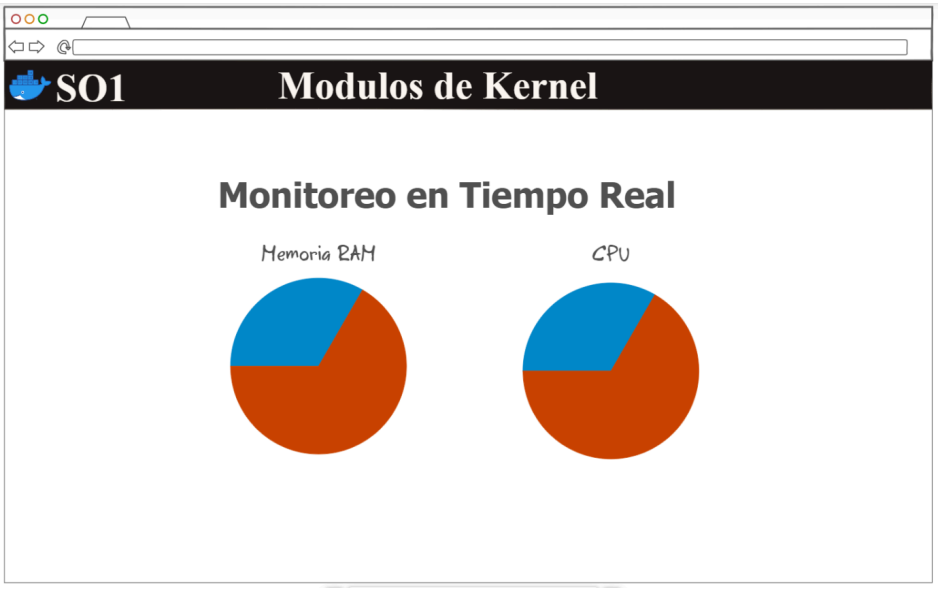
Frontend

Se debe de crear una Aplicación Web desarrollada en React que se divide en:

Monitoreo en Tiempo Real

Se debe mostrar:

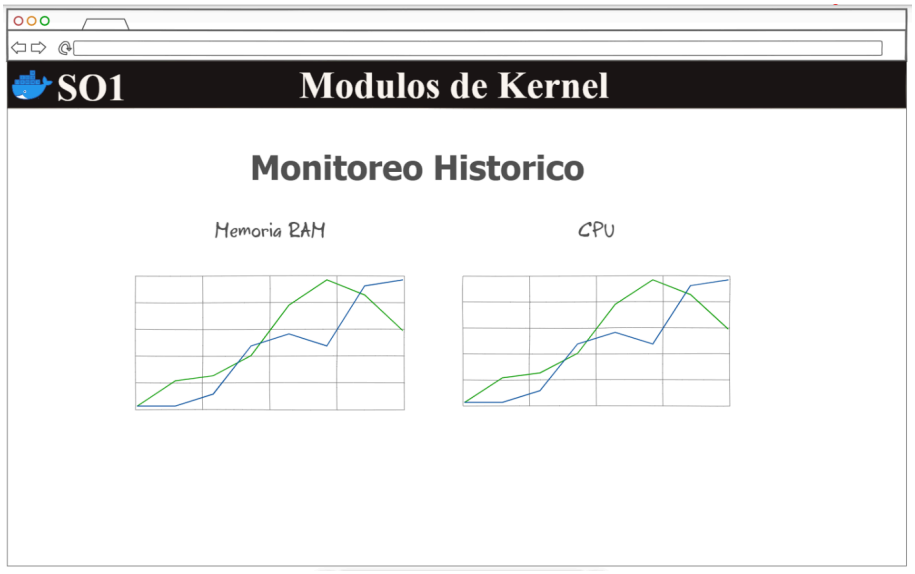
- Gráfica en Tiempo Real del porcentaje de utilización de la memoria RAM.
- Gráfica en Tiempo Real del porcentaje de utilización del CPU.



Monitoreo a lo largo del Tiempo

En esta página de debe de Agregar:

- Una Gráfica del Rendimiento a lo largo del tiempo de la RAM.
- Una Grafica de Rendimiento a lo largo del tiempo del CPU.



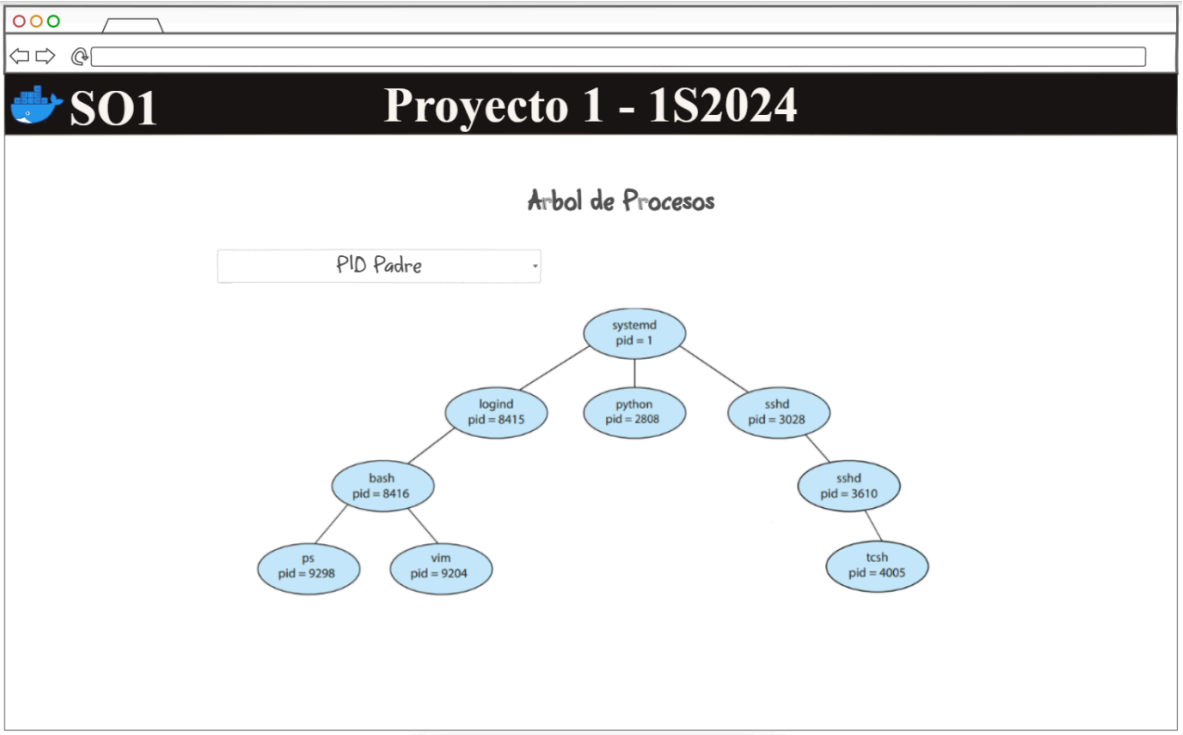
Árbol de Procesos

En esta página de debe de Agregar:

- Un Select para elegir el PID de los procesos Padres.
- Una Gráfica que desglosa los procesos hijos (si existen).

Nota:

- Los procesos deben de ser obtenidos de la información que brinda el módulo de CPU.
- Se recomienda utilizar la librería visjs para el árbol de Procesos.



Referencia: **Pág. 116 del libro Operating System Concepts, tenth edition.**

Simulación de Cambio de Estados en los Procesos

Este apartado simulara los estados de los procesos a través de botones, que enviaran las señales KILL a los procesos.

Estados:

- **New:** El proceso se está creando
- **Running:** Se están ejecutando instrucciones.
- **Waiting:** El proceso está esperando que ocurra algún evento.
- **Ready:** El proceso está esperando ser asignado a un procesador.
- **Terminated:** El proceso ha finalizado su ejecución.

Botones

- **New:** Este botón creará un nuevo proceso, debe de retornar el PID de dicho proceso y generará los siguientes estados: **New, Ready y Running.**
- **Kill:** Debe de Terminar el Proceso definitivamente. **Estado Terminated.**
- **Stop:** este botón debe de cambiar el estado de **Running a Ready.**
- **Resumen:** Este botón debe de cambiar el estado de **Ready a Running.**

Diagrama de Estado de Procesos

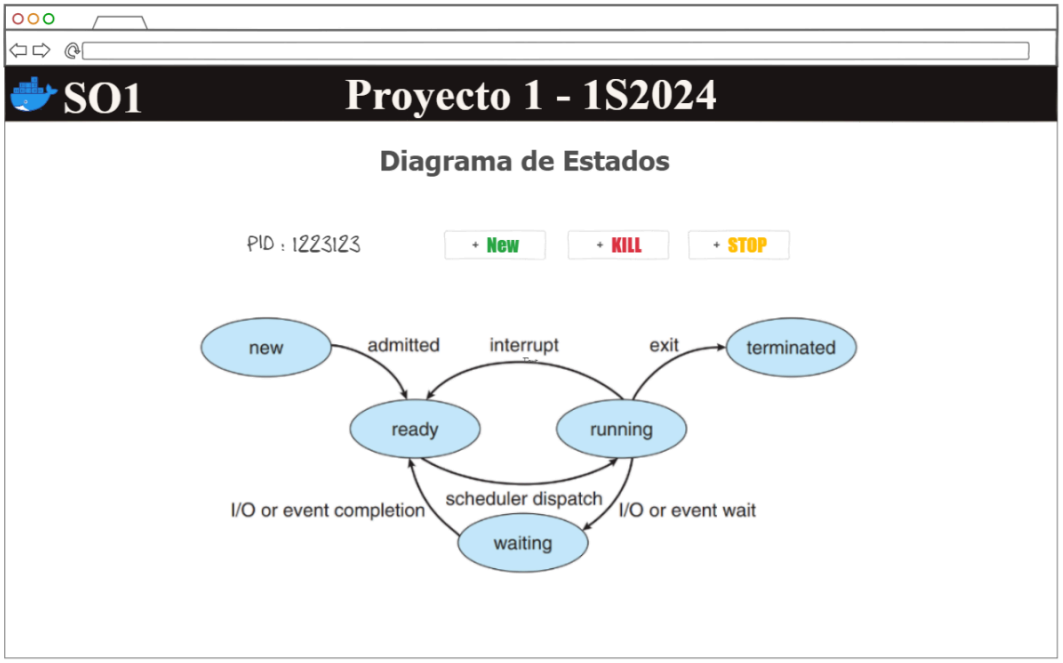
En esta página de debe de Agregar:

- Un Diagrama que muestre los Estados que han tenido un proceso.
- Cuando ocurre un cambio de estado debe de mostrarse el estado actual en el que se encuentra el proceso, es decir el nodo y arista de la transición debe de tener un color diferente.

Nota:

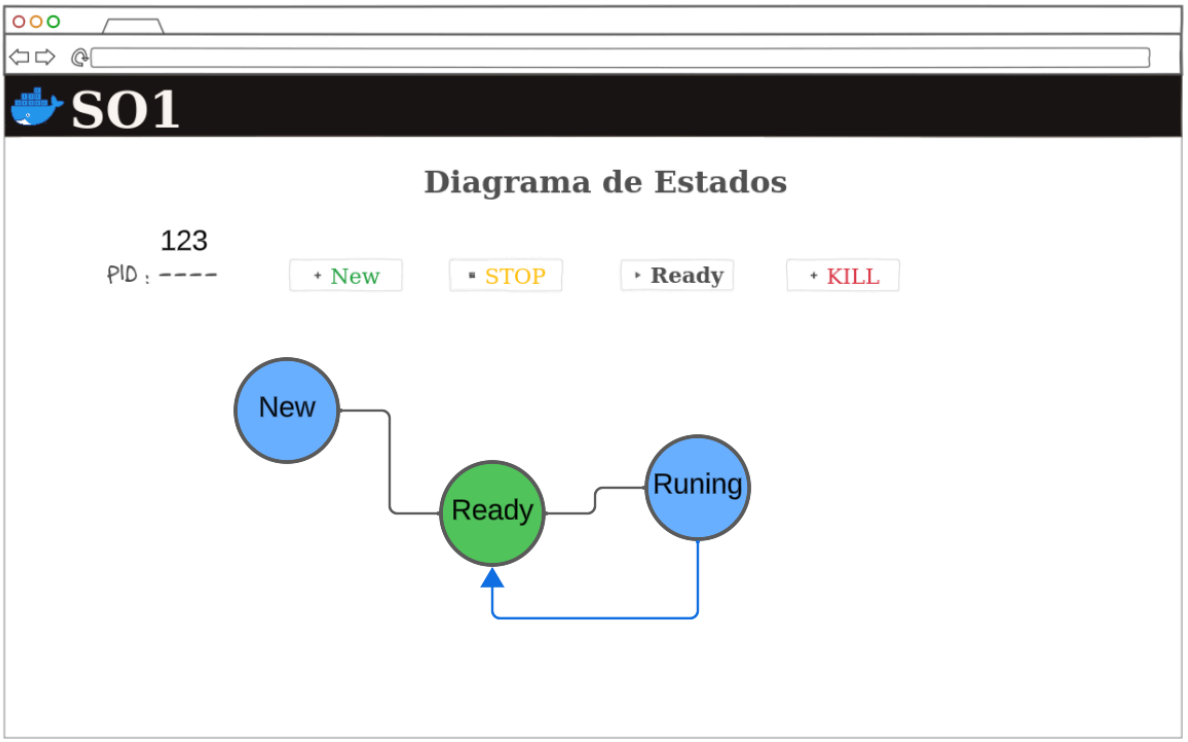
- Se recomienda utilizar la librería visjs para graficar.

Referencia: **Pág. 109 del libro Operating System Concepts, tenth edition.**

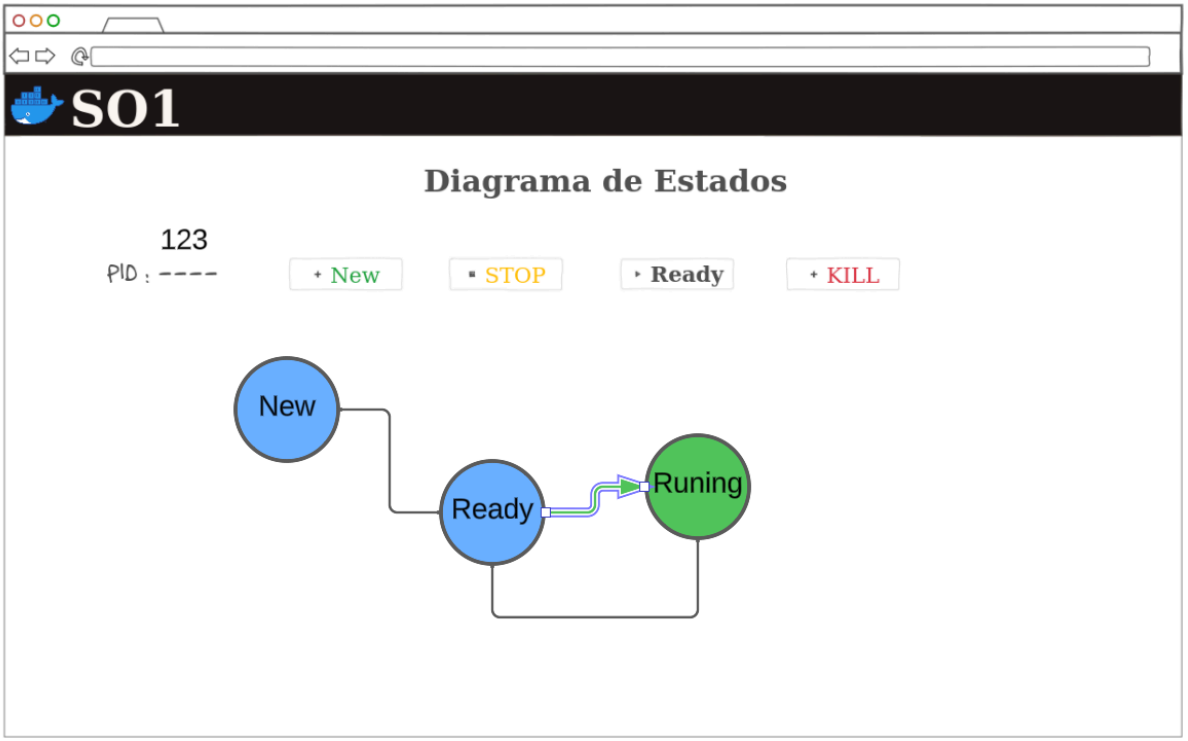


Ejemplo: 1. New

Stop


SO1
Diagrama de Estados
123
PID : ----
New
STOP
Ready
KILL
New
Ready
Runing

Resume


SO1
Diagrama de Estados
123
PID : ----
New
STOP
Ready
KILL
New
Ready
Runing

API GOLANG

- Se encargará de realizar los llamados a los módulos ubicados en la carpeta proc y almacenar los datos en la BD de MYSQL.
- Realizará el envío de datos hacia el monitoreo histórico y monitoreo a tiempo real.
- Realizará el envío de datos necesarios para el árbol de procesos en tiempo real.
- Tiene como tarea realizar las operaciones del Simulador de Procesos y Graficarlo de manera inmediata, las operaciones de señales a procesos también deben de ser almacenados en la BD.

Base de datos

Se debe implementar una base de datos MySQL por medio de un contenedor de Docker, esto para guardar los datos de los procesos padre, procesos hijos, información de memoria ram y cpu, diagrama de estados. 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.

Requisitos Mínimos

- Documentación.
- No debe existir ninguna dirección IP quemada directamente en el código.
- Desplegar frontend en el puerto 80.

Restricciones

- El proyecto se realizará de forma individual.
- La obtención de la información se hará a través de los módulos de Kernel en C.
- La API se realizará con Golang.
- El Frontend debe ser realizado con React.
- La máquina virtual deberá ser de Ubuntu Server 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
- Agregar a los auxiliares al repositorio de GitHub: **DannyLaine158** y **JhonathanTocay2020**

Calificación

- Al momento de la calificación se verificará la última versión publicada en el repositorio de GitHub
- 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 entrega se debe realizar antes de las 23:59 del 15 de marzo de 2024.
- La forma de entrega es mediante UEDI subiendo el enlace del repositorio.
- Manual técnico con explicación de todos los componentes utilizados en el proyecto por ejemplo Web UI, Database, Simulador de Estados, Módulos, etc. Este manual debe ser un archivo .md (Markdown), por lo que deberá estar en el README del repositorio.

Referencias

- Docker
- Docker Compose
- Libro, Operating System Concepts, tenth edition