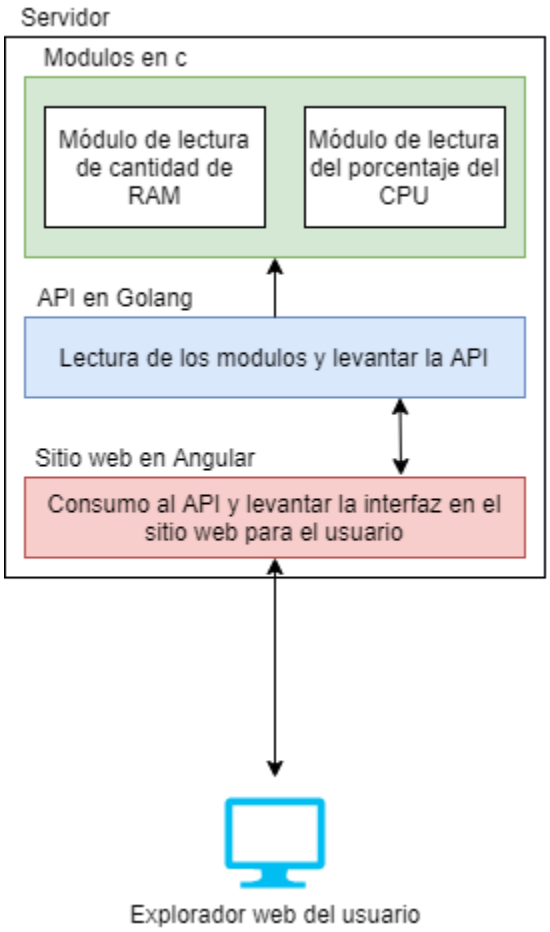
# Manual técnico

## Control, Creación y Monitorización de Procesos

*El programa fue realizado en Angular, obtiene información de los recursos del ordenador utilizando módulos en programados y cargados en C, la información obtenida por los modulos en C es presentada a Angular por medio de una API realizada en Golang.*

# Infrastructura

Se cuenta con un servidor para atender las solicitudes de los clientes, los recursos y procesos presentados al usuario serán los del servidor.



## Características del servidor de desarrollo

La aplicación fue testeada en una máquina virtual con las siguientes características:

- 5 GB de RAM DDR4
- 1 núcleo del procesador i7-10750H (2.60GHz) al 46%
- 50 GB de disco duro.

## Características del cliente

Dado que se para la aplicación se utilizo Angular 11.Información obtenida del sitio oficial de angular en la fecha 22 de febrero del 2021 (Angular - Browser support).

| | **Browser** | **versiones soportadas** |
|---|---|---|
| ○ | Chrome | latest |
| ○ | Firefox | latest and extended support release (ESR) |
| ○ | Edge | 2 most recent major versions |
| ○ | IE | 11 |
| ○ | Safari | 2 most recent major versions |

# Software instalado en el servidor

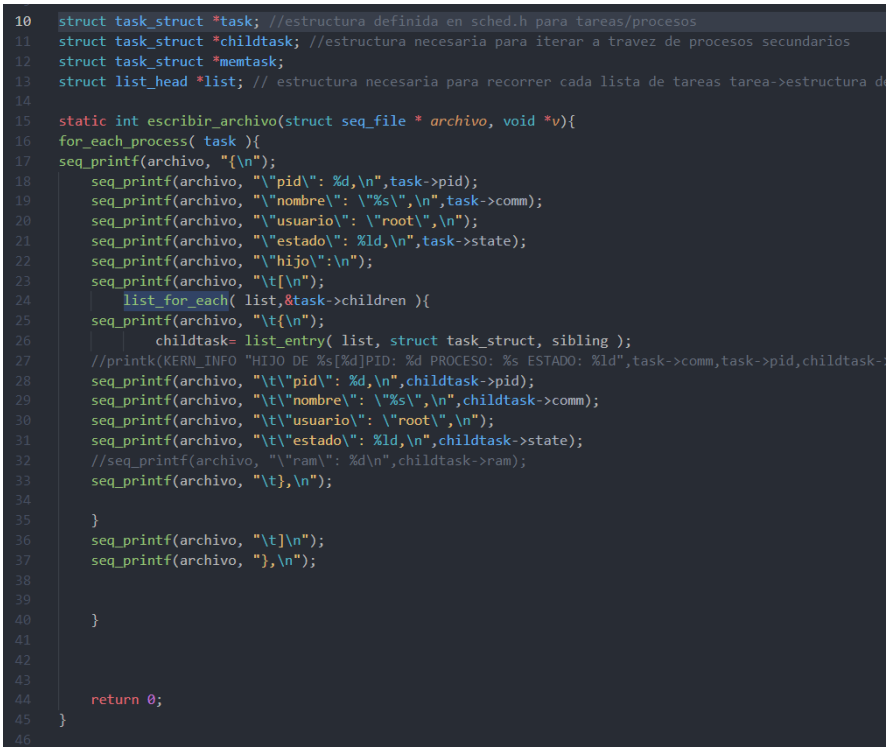
En el servidor se instaló:

- GNU Make 4.2.1
- GCC 9.3.0
- Java (OpenJDK) versión 11.0.10
- Git versión 2.25.1
- Golang versión 1.13.8
- Sistema Operativo: Ubuntu 16.04 en adelante.
- Espacio en disco ocupado por el programa: 700KB.
- Angular 11.0.4 en adelante.
- GNU make 4.2.1 en adelante.
- Golang 1.13 en adelante.

# Descripción del funcionamiento de los módulos en C.

## Módulo de lectura los procesos

El código anterior realiza genera un archivo que contiene datos de todos los procesos actualmente ejecutados en el sistema operativo, así como de los procesos hijos, todos los procesos encontrados son agregados al archivo que estamos formando, se termina formando un árbol similar al JSON.



```
struct task_struct *task; //estructura definida en sched.h para tareas/procesos
struct task_struct *childtask; //estructura necesaria para iterar a travez de procesos secundarios
struct task_struct *memtask;
struct list_head *list; // estructura necesaria para recorrer cada lista de tareas tarea->estructura d

static int escribir_archivo(struct seq_file * archivo, void *v){
for_each_process( task ){
seq_printf(archivo, "{\n");
    seq_printf(archivo, "\"pid\": %d,\n",task->pid);
    seq_printf(archivo, "\"nombre\": \"%s\",\n",task->comm);
    seq_printf(archivo, "\"usuario\": \"root\",\n");
    seq_printf(archivo, "\"estado\": %ld,\n",task->state);
    seq_printf(archivo, "\"hijo\":\n");
    seq_printf(archivo, "\t[\n");
        list_for_each( list,&task->children ){
    seq_printf(archivo, "\t{\n");
            childtask= list_entry( list, struct task_struct, sibling );
    //printk(KERN_INFO "HIJO DE %s[%d]PID: %d PROCESO: %s ESTADO: %ld",task->comm,task->pid,childtask-
    seq_printf(archivo, "\t\"pid\": %d,\n",childtask->pid);
    seq_printf(archivo, "\t\"nombre\": \"%s\",\n",childtask->comm);
    seq_printf(archivo, "\t\"usuario\": \"root\",\n");
    seq_printf(archivo, "\t\"estado\": %ld,\n",childtask->state);
    //seq_printf(archivo, "\"ram\": %d\n",childtask->ram);
    seq_printf(archivo, "\t},\n");

    }
    seq_printf(archivo, "\t]\n");
    seq_printf(archivo, "},\n");


    }



    return 0;
}

```

# Módulo de lectura de RAM

El código obtiene la información del sistema relacionada con el uso del a RAM, primero obtenemos la memoria utilizada para el cache, revisamos si hay si quiera, en caso de que no haya comprendemos que es cero, luego tomamos el total de RAM física con la que cuenta el sistema, a esta memoria le restaremos la que sabemos que está libre y la memoria utilizada por el buffer y por el cache. Finalmente tomamos la RAM total física disponible y le restamos la RAM libre para saber cuánta RAM se está utilizando.

```c
static int my_proc_show(struct seq_file *m, void *v)
{
    struct sysinfo i;
    long cached,usada,total,libre;

    si_meminfo(&i);

    cached = global_node_page_state(NR_FILE_PAGES) - total_swapcache_pages() - i.bufferram;
    if (cached < 0)
        cached = 0;
    total = i.totalram;
    libre = (i.freeram + i.bufferram + cached);
    usada = total - libre;

    seq_printf(m, "%ld\n", i.totalram);

    seq_printf(m, "%ld\n", usada);

    return 0;
}
```

# Contacto con el equipo de desarrollo

| Nombre | Teléfono |
|---|---|
| Juan Carlos Juarez Barneond | 5213 9098 |
| Juan Luis Robles Molina | 4249 5762 |
| David Omar Enriquez Reyes | 5574 0905 |

**Gracias por contar con nuestros servicios.**