UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE SISTEMAS

PROYECTO 1

200915347 - 200915466

ERICK FERNANDO REYES MANCILLA
OLIVER ALEXANDER RODAS MENDOZA
SISTEMAS OPERATIVOS 1

INDICE

MÓDULOS

Pagina Principal

Se debe de entender que para la página principal se tiene que tener un backend en el cual recibimos la información de la carpeta /proc la cual es donde están todos los procesos que se van generando, de la siguiente forma se accede a la carpeta /proc desde Go:

```
b, err := ioutil.ReadFile("/proc/" + b1[i].Name() + "/status")
if err != nil {
  return
}

bU, errU := ioutil.ReadFile("/proc/" + b1[i].Name() + "/loginuid")
if errU != nil {
  return
}
```

Esta información debe de ser mostrada en una vista web de la siguiente forma se hace la comunicación entre backend y frontend, se coloca el nombre de la ruta, seguido del nombre de la función que se va a utilizar de igual forma se tiene que colocar el puerto:

```
func main () {

  //http.HandleFunc("/memoria", ramInfo)
  router := mux.NewRouter().StrictSlash(true)
  mux.CORSMethodMiddleware(router)

  router.HandleFunc("/", Index)
  router.HandleFunc("/todos", TodoIndex)
  router.HandleFunc("/todos/{todoId}", TodoShow)
  router.HandleFunc("/proceso", procesosInfo)



  log.Fatal(http.ListenAndServe(":4000", router))
```

Del lado de la vista (web) se obtiene la información a con Ajax request:

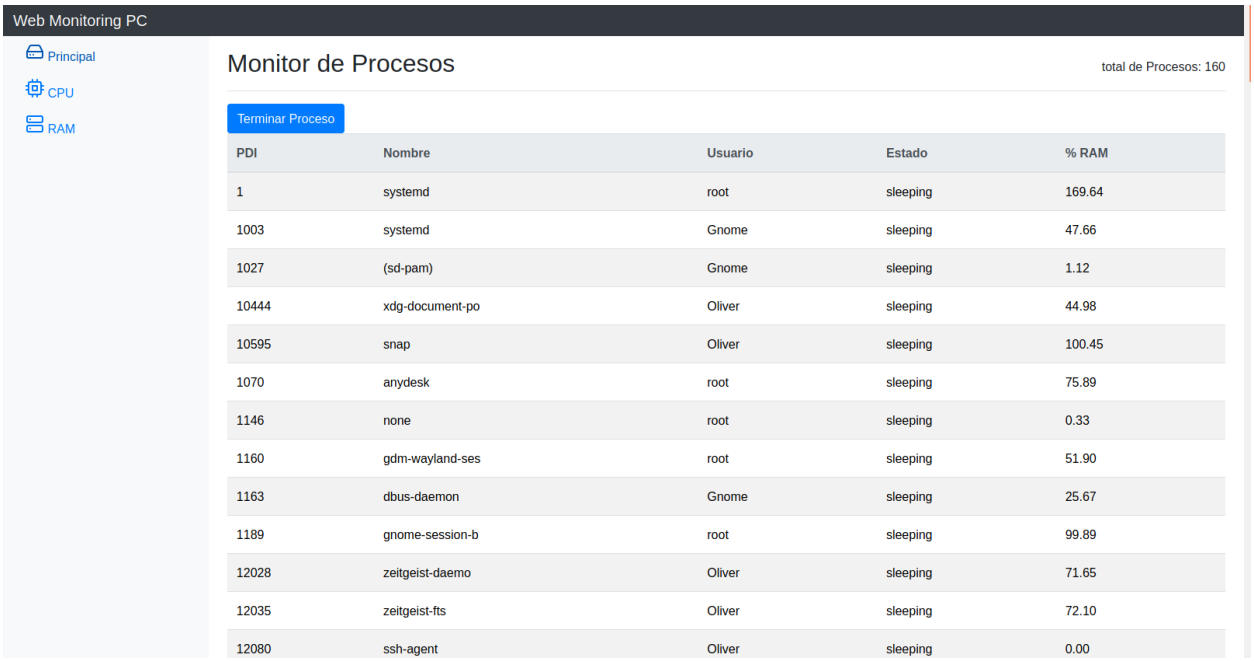
```
function obtenerProceso(){
  $.ajax({
      type: 'GET',
      url: 'http://localhost:4000/proceso',
      contentType: "application/json",
      dataType: 'json',
      crossDomain: true,
      async: false,
      success: function (response) {

          JSON.stringify(response);
        console.log(response);
        html ='';
         porcentaje = 0;
```

Este es del lado de los script o archivos Js que se comunican con el html para mostrar la información al usuario final:

```
    $("#paginas").html(html);
    document.getElementById('totalP').innerHTML = "total de Procesos: " + response.length;
```

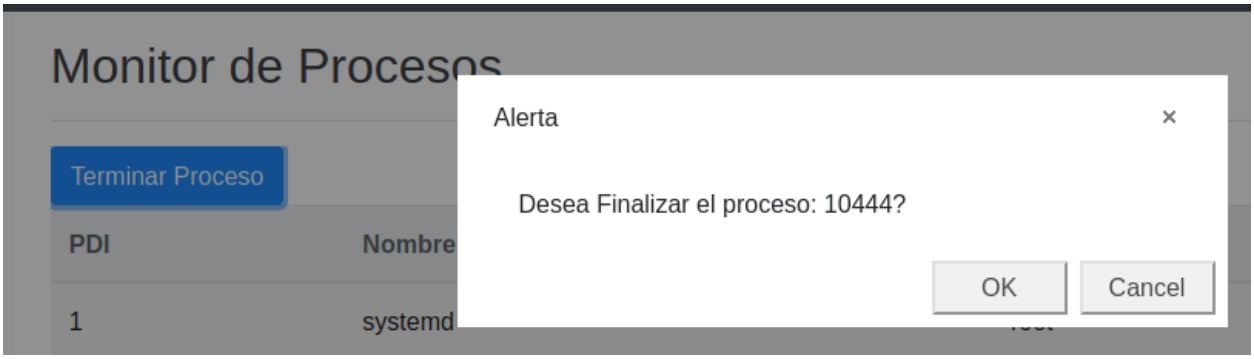
```
<table class="table table-striped no-margin" id="viaticoPendiente">
<thead class="thead-light">
<tr>
<th scope="col">PDI</th>
<th scope="col">Nombre</th>
<th scope="col">Usuario</th>
<th scope="col">Estado</th>
<th scope="col">% RAM</th>

</tr>
</thead>
<tbody id="paginas" style="color: black">
</tbody>
```

De esta forma se muestra la información visual al usuario:

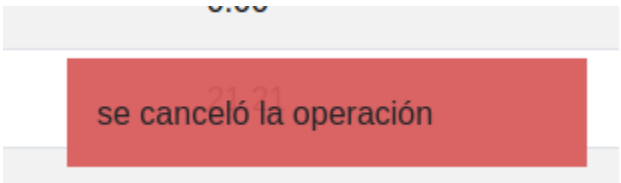


PDI	Nombre	Usuario	Estado	% RAM
1	systemd	root	sleeping	169.64
1003	systemd	Gnome	sleeping	47.66
1027	(sd-pam)	Gnome	sleeping	1.12
10444	xdg-document-po	Oliver	sleeping	44.98
10595	snap	Oliver	sleeping	100.45
1070	anydesk	root	sleeping	75.89
1146	none	root	sleeping	0.33
1160	gdm-wayland-ses	root	sleeping	51.90
1163	dbus-daemon	Gnome	sleeping	25.67
1189	gnome-session-b	root	sleeping	99.89
12028	zeitgeist-daemo	Oliver	sleeping	71.65
12035	zeitgeist-fts	Oliver	sleeping	72.10
12080	ssh-agent	Oliver	sleeping	0.00

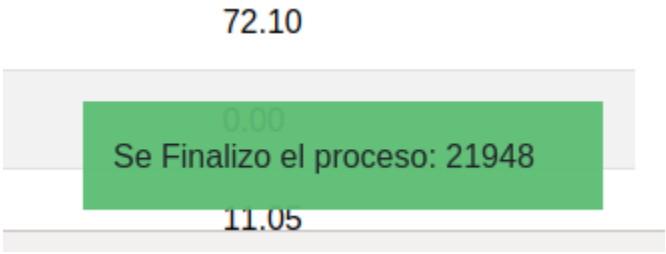
Para terminar el proceso se selecciona el proceso que deseamos finalizar y se le da clic en el boton terminar proceso:



Nos da la opción de finalizar el proceso o cancelar, para canelar nos muestra el mensaje que se cancelo la operación:

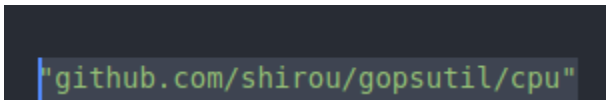


Al momento de dar clic en Ok este finalizará el proceso:



Monitor de CPU

Se tiene la parte del monitor de cpu la cual funciona obteniendo la información mediante la consulta de la información a nivel de la librería gopsutil/cpu



Se obtiene la información del CPU de la siguiente forma:

```
}
var cpu = CPU{
  cpuStat[0].Cores,
  cpuStat[0].VendorID,
  cpuStat[0].Family,
  cpuStat[0].ModelName,
  strconv.FormatFloat(cpuStat[0].Mhz, 'f', 2, 64),
  read}
```

De la siguiente forma se envia la información hacia la vista:

```
85
86  func main() {
87
88      mux := http.NewServeMux()
89      mux.HandleFunc("/", GetData)
90      http.ListenAndServe(":8080", mux)
91  }
92
```

Del lado de la vista se necesita obtener esa informacion que se esta obteniendo desde el servidor, y se obtiene de la siguiente manera:

```
function obtenerData(){
  $.ajax({
      type: 'GET',
      url: 'http://localhost:8080/GetData',
      contentType: "application/json",
      dataType: 'json',
      crossDomain: true,
      async: false,
      success: function (response) {

          JSON.stringify(response);
        console.log(response);
```

Este es del lado de los script o archivos Js que se comunican con el html para mostrar la información al usuario final:

```
//alert(response.os);
document.getElementById('infoCpu').innerHTML = response.cpu.model;
document.getElementById('VelocidadB').innerHTML = response.cpu.speed
document.getElementById('nucleoCpu').innerHTML = response.cpu.cores;
document.getElementById('proL').innerHTML = response.cpu.family;
```

Para que se este haciendo la petición de la información se debe de hacer un timer o hilo; este se puede hacer de varias formas, en esta ocasión se realizó así:

```
setTimeout(obtenerDataDinamica, 1000);
//setTimeout(graficar, 1000);
actualizarG();
```

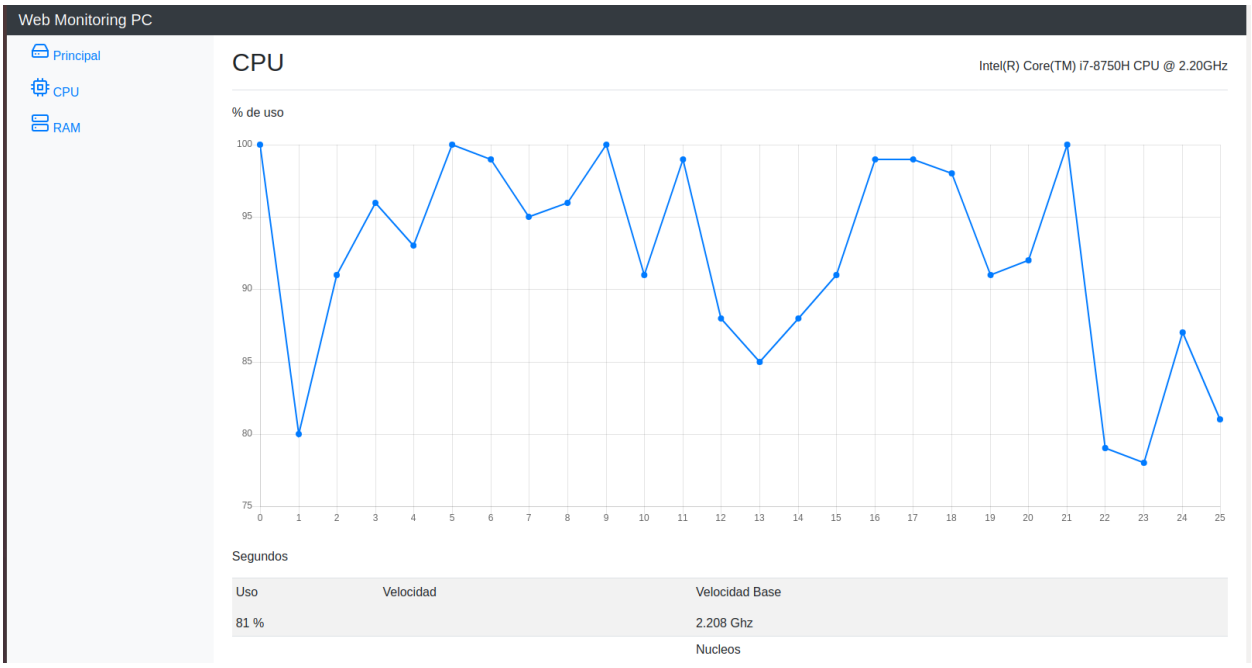
Se debe de realizar una grafica cada proceso que va realizando del lado del CPU:

```
var canvasV = document.getElementById("myChart").getContext("2d");
// var ctx = canvasV.;
 myChart = new Chart(canvasV, {
  type: 'line',
  data: {
    labels: [0],
    datasets: [{
      data: [dataG],
      lineTension: 0,
      backgroundColor: 'transparent',
      borderColor: '#007bff',
      borderWidth: 2,
      pointBackgroundColor: '#007bff'
    }]
  },
  options: {
    scales: {
      yAxes: [{
        ticks: {
          beginAtZero: false
        }
      }]
    },
    legend: {
      display: false,
    }
  }
});
```

Del lado del html se llama el canvas de la siguiente forma:

```
<canvas class="my-4" id="myChart" width="900" height="350"></canvas>
<p>Segundos</p>
```

De esta forma se muestra la vista de los procesos que esta ejecutando el CPU:



Monitor de RAM

Para el monitor de RAM se tiene que la información de este se lee desde la carpeta /proc como el monitor de procesos:

```
//////////////////////MEMORIA RAM
func ramInfo(w http.ResponseWriter, r *http.Request) {

  b, err := ioutil.ReadFile("/proc/meminfo")
  if err != nil {
    return
  }

  str := string(b)
  listadoInfo := strings.Split(string(str), "\n")

  memoriaTotal := strings.Replace((listadoInfo[0])[10:24], " ", "", -1 )
  memoriaLibre := strings.Replace((listadoInfo[1])[10:24], " ", "", -1 )
  ramTotalKB, err1 := strconv.Atoi(memoriaTotal)
  ramLibreKB, err2 := strconv.Atoi(memoriaLibre)
  if err1 == nil && err2 == nil{
      ramTotalMB := ramTotalKB / 1024
      ramLibreMB := ramLibreKB / 1024
```

Esta información obtenida se envía por medio del metodo GET:

```
http.HandleFunc("/memoria", ramInfo)
```

Del lado de la vista esta información se realiza la petición por medio de Ajax:

```
function obtenerData(){

  $.ajax({
      type: 'GET',
      url: 'http://localhost:8080/GetData',
      contentType: "application/json",
      dataType: 'json',
      crossDomain: true,
      async: false,
      success: function (response) {

          JSON.stringify(response);
        console.log(response);
```

Este es del lado de los script o archivos Js que se comunican con el html para mostrar la información al usuario final:

```
totalRam = parseInt(response.ram.total / 1024000);
//alert(response.os);
document.getElementById('infoRam').innerHTML = totalRam +" MB";

dataG = parseInt(response.ram.usage);
```

Para que se este haciendo la petición de la información se debe de hacer un timer o hilo; este se puede hacer de varias formas, en esta ocasión se realizó así:

```
setTimeout(obtenerDataDinamica, 1000);
//setTimeout(graficar, 1000);
actualizarG();
```

Se debe de realizar una gráfica cada proceso que va realizando del lado de la RAM:

```
var canvasV = document.getElementById("myChart").getContext("2d");
// var ctx = canvasV.;
 myChart = new Chart(canvasV, {
  type: 'line',
  data: {
    labels: [0],
    datasets: [{
      data: [dataG],
      lineTension: 0,
      backgroundColor: 'transparent',
      borderColor: '#007bff',
      borderWidth: 2,
      pointBackgroundColor: '#007bff'
    }]
  },
  options: {
    scales: {
      yAxes: [{
        ticks: {
          beginAtZero: false
        }
      }]
    },
    legend: {
      display: false,
    }
  }
});
```

Del lado del html se llama el canvas de la siguiente forma:

```
<canvas class="my-4" id="myChart" width="900" height="350"></canvas>
<p>Segundos</p>
```

De esta forma se visualiza la grafica y el total de ram que se esta utilizando como también el total de ram de la maquina: