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


# MANUAL TECNICO

## 2012-12487

*Alan René Bautista Barahona*
*Sistemas operativos 1*

# INDICE

# Introducción

El siguiente manual contiene información técnica acerca del funcionamiento, así como las partes y lógica de programación que posee la aplicación. La aplicación básicamente consiste en sitio web que muestra los recursos del servidor en donde este está publicado.

Se trata de una aplicación que permita monitorear y gestionar los procesos de un servidor, por medio de una interfaz de fácil acceso desde el navegador de una computadora o de un dispositivo móvil como teléfono o Tablet.

Objetivos

- ✓ Desarrollar una aplicación con una interfaz gráfica agradable para el usuario y fácil de manipular.

- ✓ Investigar la función del directorio /proc para obtener la información del sistema.

- ✓ Generar gráficas dinámicas dentro de una solución web, para facilitar la lectura de información por parte del usuario final.

- ✓ Conocer los estados de los procesos en los sistemas basados en Unix.

# ANÁLISIS

## DESCRIPCIÓN DEL PROBLEMA

Grafica de recursos: permite visualizar graficas dinámicas que muestren el uso del CPU y de la memoria RAM del servidor.

Administrador de procesos: mostrará la información básica de los procesos que se ejecutan y permite matar los procesos(kill) que se encuentran en ejecución.

## ANÁLISIS DE REQUERIMIENTOS DEL USUARIO.

- ✓ El usuario necesita una interfaz agradable y fácil de entender.
- ✓ Generar gráficas dinámicas dentro de una solución web, para facilitar la lectura de información por parte del usuario final.

## PLANTEO INICIAL DE LA SOLUCIÓN.

Se desea desarrollar una aplicación que analice los recursos del servidor sobre el cual está corriendo, esto se podrá realizar por medio de la lectura he interpretación de archivos que hacen funcionar el sistema operativo que en este caso es una distribución de Linux. Por medio de la carpeta “Proc” se puede encontrar información como uso de memoria, cantidad de procesos ejecutados, el estado de cada proceso, que usuario los ejecuto y cuanta memoria del sistema ocupa cada proceso.

Además de solo leer los archivos se necesita interpretar la información de cada uno de estos y mostrarla de una manera elocuente y legible para el usuario, además de poder implementar herramientas de díselo web como lo es bootstrap, esto para poder obtener un diseño presentable en la aplicación además de lograr que el sitio web sea adaptable a la mayoría de dispositivos actuales y no solamente limitado a ordenadores.

## BUENAS PRÁCTICAS DE PROGRAMACIÓN A IMPLEMENTAR.

Realizar un análisis estructurado y lógico de lo que se de hacer siempre es una buena práctica para implementar de la forma más apropiada los distintos métodos para resolver un problema, además el hecho de planificar y revisar las acciones que realizara nuestro programa al momento de ejecutarse, siempre nos permite encontrar posibles erres de compilación o ejecución, errores que se pueden corregir con el simple hecho se ponerles más atención tratando de evitarlos.

Al momento de realizar cada uno de los algoritmos o implementar distintos métodos para resolver cada uno de los problemas, es bueno verificar que el software se desarrolle de la forma deseada o esperada, y que los métodos, clases, algoritmos, etc… que acabamos de implementar no interfieran con el buen funcionamiento de los que ya tenemos creados y funcionando.

Investigar siempre la mejor manera de resolver un problema, ya que existen muchas formas de resolverla misma problemática, pero unas se ajustan de mejor forma dependiendo las circunstancias en la que esta se esté requiriendo.

# DISEÑO

APP.JS

```
//importacion de librerias o módulos
//var dt = require('./myfirstmodule');
var http = require('http');
var url = require('url');
var sql = require("mssql");
var express = require('express');
var fs = require('fs');
var bodyParser = require('body-parser');
var request = require('request');
var ejs = require('ejs');
var proc = require('node-proc');
var child_process = require('child_process');
var ps = require('ps-node');
var os = require('os-utils');

var app = express();

//configuracion para sql

var config = {
        user: "aln",
        password: "e$0lutions",
        server: "localhost\\SQLEXPRESS",
        database: "COMPI2" ,
              port: 1433

};

var UsrActual;

app.use('/css', express.static('css')); //acceder al css
app.use('/js', express.static('js')); //acceder a los js
app.use(bodyParser.urlencoded({ extended: true })); //obtener los datos del html

//para renderizar y pasar datos hacia la vista
app.set('views', __dirname + '/');
app.engine('html', ejs.renderFile);
app.set('view engine', 'html');

// metodo para inicia el servidor
var server = app.listen(1346, function () {
```

```
    console.log('Servidor ON carnal!');
});

//---------------------------------------------------------------
//                                        LOGIN
//---------------------------------------------------------------

app.get('/', function (req, res) {
   res.render("login.html",{ mensaje: "" });
});

app.post('/', function (req, res) {
   var usuario = req.body.user.trim();
   var password = req.body.pass.trim();

   console.log(usuario);
   console.log(password);

            if (usuario == "admin" && password=="admin"){

                  res.redirect('/index');
            }else{

                  res.render('login.html',{   mensaje:   "Usuario   /   Clave
incorrecta" });
            }



});


//---------------------------------------------------------------
//                                        INDEX
//---------------------------------------------------------------

app.get('/index', function (req, res) {


      var data = fs.readFileSync('/proc/meminfo').toString();
      var lines = data.split(/\n/g).map(function(line){
            return line.split(':');
      });
      //console.log(lines);

      child_process.exec('ps  -A  -o  pid,user,state,%mem,command',  (err,  stdout,
stdin) => {
```

```
if (err) console.log(err);
var jsonArr = [];
var json1="";
var lines = stdout.split("\n");
var contador = 0;
var dormidos = 0;
var corriendo = 0;
var detenidos = 0;
var zombies = 0;
for (var i = 1; i < lines.length; i++) {
	var line = lines[i].trim();
	var pid = line.split(" ")[0];
	line = line.substring(pid.length, line.length).trim();
	var user = line.split(" ")[0];
	line = line.substring(user.length, line.length).trim();
	var state = line.split(" ")[0];
	line = line.substring(state.length, line.length).trim();
	var mem = line.split(" ")[0];
	line = line.substring(mem.length, line.length).trim();
	var command = line;

	//console.log("pid",pid);
	//console.log("user",user);
	//console.log("state",state);
	//console.log("mem",mem);
	//console.log("name",command);

	switch(state)
	{
		case "S":
			dormidos ++;
			break;
		case "D":
			dormidos ++;
			break;
		case "R":
			corriendo ++;
			break;
		case "T":
			detenidos ++;
			break;
		case "Z":
			zombies ++;
			break;
	}
	contador++;
```

```
                    jsonArr.push({"pid":pid,     "user":user,     "state":state,
"mem":mem, "name":command});
        }

                json1 = JSON.stringify({jsonArr:jsonArr});
                //console.log("JSON COMPLETO:"+json1);
                var jsonObj1 = JSON.parse(json1);
                //console.log("TAMAÑO ARREGLO:"+jsonObj1.jsonArr.length);
                //console.log(contador);
                contador = contador -1;
                res.render('index.html',{ procesos: jsonArr, NumProcesos: contador,
NumSleep: dormidos , NumRun: corriendo , NumStop:detenidos , NumZombies: zombies});
        });

});


app.get('/icon', function (req, res) {
    console.log("iconos");
       res.render('messages.html');
});

//------------------------------------------------------------------
//                                        ELIMINAR
//------------------------------------------------------------------

app.get('/eliminar', function (req, res) {

        var pid = req.query['id'];

        console.log("Muere :'v");

        //process.kill(pid, 'SIGKILL');
        ps.kill( pid, function( err ) {
           if (err) {
                  //throw new Error( err );
                      console.log( 'Proceso no eliminado: ', pid );
                      res.redirect('/index');
           }
           else {
                  console.log( 'Proceso eliminado: ', pid );
                  res.redirect('/index');
           }
        });

        console.log("Murio :'v");
        res.redirect('/index');
```

```
});


//------------------------------------------------------------
//                                        CPU
//------------------------------------------------------------

app.get('/cpu', function (req, res) {

	var uso = 0;
	var total = 0;
	var consumida = 0;

	os.cpuUsage(function(p){
		console.log( 'Uso de CPU  (%): ' + p );
		uso = p;
	});

	//console.log('Total de memoria: ' + os.totalmem() + " MB");
	total = os.totalmem();
	//console.log('Memoria consumida: ' + (os.totalmem() - os.freemem())+" MB");
	consumida = os.totalmem() - os.freemem();

	res.render('cpu.html',{ UsoCPU: uso, MemTotal: total, MemUso: consumida});

});

app.post('/vista_clase', function (req, res) {
	var clase = req.body.clase_nombre;
	var codigo = req.body.txtCodigo;

	fs.writeFile("./Descargas/"+clase, codigo, function (err) {
		if (err) {
			return console.log(err);
		}else{
			console.log("Archivo guardado");
			res.json({success : true})
			res.redirect('/repositorio_201212487');
		}

	});
});

//------------------------------------------------------------
//                                        RAM
//------------------------------------------------------------
```

```
app.get('/ram', function (req, res) {

	var uso = 0;
	var total = 0;
	var consumida = 0;

	//console.log('Total de memoria: ' + os.totalmem() + " MB");
	total = os.totalmem();
	//console.log('Memoria consumida: ' + (os.totalmem() - os.freemem())+" MB");
	consumida = os.totalmem() - os.freemem();

	res.render('ram.html',{ UsoCPU: uso, MemTotal: total, MemUso: consumida});

});

//-----------------------------------------------------------------
//                                        LOG OFF
//-----------------------------------------------------------------

app.get('/salir', function (req, res) {

	res.render("login.html",{ mensaje: "" });

});
```

## ESPECIFICACIONES

A continuación se especificaran ciertas restricciones o contenido relevante para el funcionamiento correcto de la aplicación entre las cuales están:

- El framework a utilizar es Node.js
- El servidor web que se debe usar, es el que node.js nos brinda al probar la aplicación.
- El servidor a utilizar debe ser un servicio EC2 de AWS con sistema operativo Ubuntu.
- La aplicación debe ser responsiva y debe tener aspecto profesional.
- La obtención de la información debe hacerse a través de directorio /proc, a excepción de las estadísticas de CPU, de lo contrario se tendrá una nota de cero puntos.

SOPORTE

| | |
|---|---|
| *Autor* | ALAN BAUTISTA |
| *ID* | 2012-12487 |
| *Tel* | 42523870 |
| | |
| | |

# CONCULCIONES

- Los métodos y atributos de cada clase dependen tanto de lo que se quiere lograr con cada una, como de las demás clases que tienen interacción con ella.

- El usuario se familiariza y es más fácil para este realizar una operación en un software nuevo si este está bien ambientado.

- Hay distintos archivos con estructuras sencillas y comprensibles tanto como para una persona así como para una aplicación