INSTITUTO TECNOLÓGICO DE LAS AMERICAS(ITLA)

Alumno:

Yaher Enrique Hichez García

Matrícula:

202010339

Carrera:

Desarrollo en software

Tema:

Tarea - Trabajo de Programación

Asignatura:

Programación Paralela

Luis Bessewell Feliz

a. Lista de los Procesos ejecutándose en el computador al momento de correr el programa. Valor 1 pto.

b. Lista de Servicios ejecutándose en el computador al momento de correr el programa. Valor 1 pto.

c. Al señalar cualquiera de los elementos de la lista A o B, mostrar el PID o Process ID Number de dicha ejecución en caso de aplicar. Valor 1 pto.

d. Al señalar cualquiera de los elementos de la lista A o B, mostrar el porcentaje de consumo de CPU o Procesamiento. Valor 1 pto.

e. Al señalar cualquiera de los elementos de la lista A o B, mostrar el porcentaje de consumo de Memoria RAM. Valor 1 pto.

```
using System;
using System.Collections.Generic;
using System.ComponentModel; using
System.Data; using System.Drawing;
using System.Linq; using
System.Text; using
System.Threading.Tasks; using
System.Windows.Forms; using
System.Runtime.InteropServices;
using System.Diagnostics;


namespace Tarea2
{
    public partial class Form1 : MetroFramework.Forms.MetroForm
    {
        //Declaracion de variable string para obtener el nombre del proceso en la tabla
para su eliminacion         string Str_Obt_Proc;         public Form1()
        {
            InitializeComponent();
            //Activacion del timer que actualizara la tabla
ActualizarTabla();             timer1.Enabled = true;

        }
        private void ActualizarTabla()
        {
            //limpieza del datagrid
dgv_Proceso.Rows.Clear();
            //creacion columnas con sus respectivos nombres
dgv_Proceso.Columns[0].Name = "Num. Procesos";
dgv_Proceso.Columns[1].Name = "Procesos";
dgv_Proceso.Columns[2].Name = "Prioridad Proceso";
dgv_Proceso.Columns[3].Name = "ID";             dgv_Proceso.Columns[4].Name
= "Memoria Fisica";             dgv_Proceso.Columns[5].Name = "Memoria
Virtual";


            //Propiedad para autoajustar el tamaño de las celdas segun su contenido
dgv_Proceso.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
//Propiedad para que el usuario seleccione solamente filas en la tabla y no celdas
sueltas
            dgv_Proceso.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
//Propiedad  para  que  el  usuario  no  pueda  seleccionar  mas  de  una  fila
dgv_Proceso.MultiSelect = false;
            //Declaracion de la variable que sera un contador para el total de procesos
int Int_Cant_Proc = 1;

            foreach (Process Proc_Proceso in Process.GetProcesses())
{
                //Ingreso de los datos en el datagrid
dgv_Proceso.Rows.Add(Int_Cant_Proc, Proc_Proceso.ProcessName,
```

```
Proc_Proceso.BasePriority, Proc_Proceso.Id, Proc_Proceso.WorkingSet64,
                    Proc_Proceso.VirtualMemorySize64);
                //aumento en 1 de la variable
                Int_Cant_Proc += 1;
            }
            //El label muestra la cantidad de procesos actuales
            lbl_Contador.Text = "Procesos Actuales: " + (Int_Cant_Proc - 1);    //  cant
de procesos


        } //fin metodo ActualizarTabla


        //Boton Actualizar
        private void BtnActualizar_Click(object sender, EventArgs e){
            //Ocultamos todos los objetos para los graficos y mostramos solo el Dgv de
Procesos
            LblNombreCPU.Visible = false;
            LblNombreRam.Visible = false;
            ProgressBarCPU.Visible = false;
            ProgressBarRAM.Visible = false;
            LblPorCPU.Visible = false;
            LblPorRAM.Visible = false;
Grafico.Visible = false;
dgv_Proceso.Visible = true;

            //Llamado al proceso para actualizar la tabla
            ActualizarTabla();
        }







        private void dgv_Proceso_MouseClick_1(object sender, MouseEventArgs e)
        {
            //La variable obtiene el Nombre del Proceso de la Tabla al hacerle clic
            Str_Obt_Proc = dgv_Proceso.SelectedRows[0].Cells[1].Value.ToString();
        }
        private void Form1_Load(object sender, EventArgs e){
            //Evento que maneja el contador de rendimiento de la RAM y del CPU
timer.Start();
        }

        private void Timer_Tick(object sender, EventArgs e){
            //Asignamos un float a un PerformanceCounter que ya tiene asignado el
contador de rendimiento de la RAM y del CPU             float fCPU =
pCPU.NextValue();             float fRAM = pRAM.NextValue();
            //Agregamos los Valores a la Barra de progeso respectivamente
            ProgressBarCPU.Value = (int)fCPU;
            ProgressBarRAM.Value = (int)fRAM;
            //Damos el formato de porcentaje correspondiente al label de porcentaje con
lo float
            LblPorCPU.Text = string.Format("{0:0.00}%", fCPU);
```

```
            LblPorRAM.Text = string.Format("{0:0.00}%", fRAM);
            //Agregamos los valores de Y que se usaran para mostrarlos en grafica
            Grafico.Series["CPU"].Points.AddY(fCPU);
            Grafico.Series["RAM"].Points.AddY(fRAM);
        }
        private void Button1_Click(object sender, EventArgs e){
            //Ocultamos el Dgv de Procesos y mostramos todos los objetos para los
gráficos
            LblNombreCPU.Visible = true;
            LblNombreRam.Visible = true;
            ProgressBarCPU.Visible = true;
            ProgressBarRAM.Visible = true;
            LblPorCPU.Visible = true;
            LblPorRAM.Visible = true;
Grafico.Visible = true;
dgv_Proceso.Visible = false;
        }
        private void Grafico_Click(object sender, EventArgs e)
        {
        }
    } }
```

```
  using System; using
System.Diagnostics; using
System.Collections.Generic;
using System.ComponentModel;
using System.Data; using
System.Drawing; using
System.Linq; using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting; using
System.ServiceProcess;

namespace Practica2
{
    public partial class Form1 : Form
{
    public Form2()
    {
        InitializeComponent();
UpdateProcessList();         timer1.Enabled
= true;
    }
//metodo de listar el servicio en listviews
private void UpdateProcessList() {
            ServiceController[] services = ServiceController.GetServices();
lstProcesses.Items.Clear();


            int id = 1;
            foreach (ServiceController service in services)
            {
```

```
                lstProcesses.Items.Add(id + ":" + service.ServiceName);
lst_id.Items.Add(service.Status);
                lst_memoriafisica.Items.Add(service.ServiceType);




                id = id + 1;
            }
            tslProcessCount.Text = "Servicios Actuales: " +
lstProcesses.Items.Count.ToString();
        }



        private void button1_Click(object sender, EventArgs e) {
            Close();
        }




        private void btnUpdateProcessList_Click_1(object sender, EventArgs e)
        {
         UpdateProcessList();
        }
        private void timer1_Tick(object sender, EventArgs e)
{
            UpdateProcessList();
        }          private void lstProcesses_SelectedIndexChanged(object sender,
EventArgs e)
        {

        }
        private void Form2_Load(object sender, EventArgs e)
{


        }          private void chart1_Click(object sender,
EventArgs e)         {



        }
    }
}
```