UNIVERSIDAD CATÓLICA DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
INGENIERÍA EN SISTEMAS INFORMÁTICOS
CICLO: I - 2019
MATERIA: PROGRAMACIÓN II
DOCENTE: ING. GIOVANNI ACOSTA HENRIQUEZ (giovanni.acosta@catolica.edu.sv)

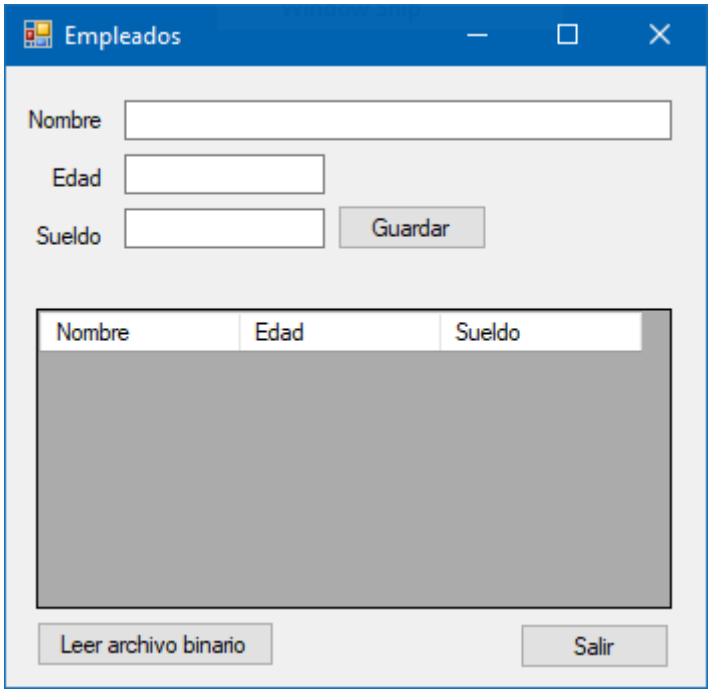
PRÁCTICA 7: MANEJO DE ARCHIVOS BINARIOS Y SERIALIZACIÓN DE OBJETOS

Objetivos:

- ✓ Elaborar programas para escribir y leer archivos binarios
- ✓ Crear programas para escribir y leer archivos binarios con objetos serializables
- ✓ Practicar el paso de datos entre formularios

1. Ejecutar Visual Studio .NET
2. Crear un nuevo proyecto de tipo Aplicación de Windows Forms (Windows Form App)
3. Crear un formulario para cada uno de los siguientes ejemplos.

Ejemplo 1: escritura y lectura de archivo binario



Control	Name	Text
TextBox1	txtNombre	
TextBox2	txtEdad	
TextBox3	txtSueldo	
Button1	btnGuardar	Guardar
Button2	btnLeer	Leer archivo binario
Button3	btnSalir	Salir
DataGridView1	dgvEmpleados	

Agregar el espacio de nombres System.IO

```
using System.IO;
```

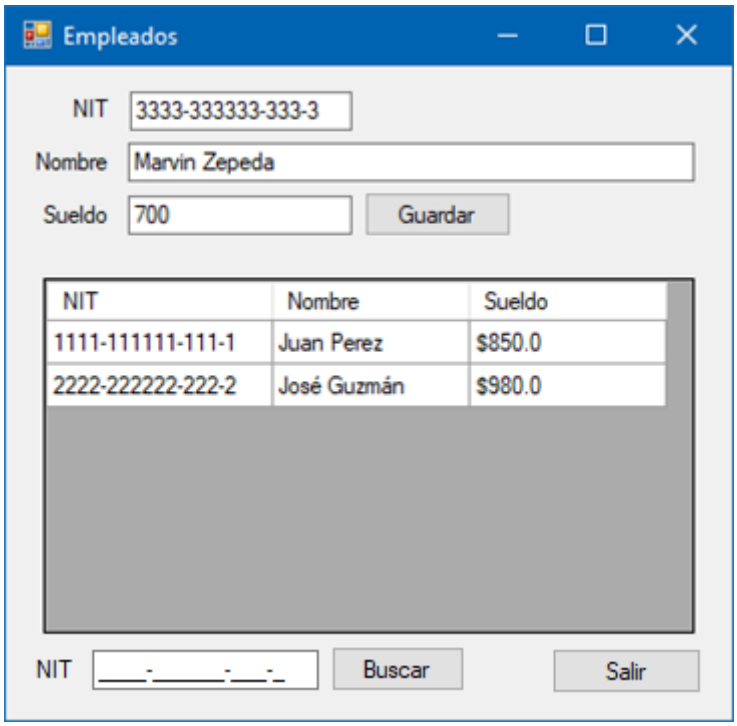
Crear las siguientes variables a utilizar en el programa, en las declaraciones generales, fuera de los métodos.

```
string nombre;
int edad;
decimal sueldo;
FileStream fs = null;
BinaryWriter bw = null;
BinaryReader br = null;
```

```
private void btnGuardar_Click(object sender, EventArgs e)
{
    try
    {
        fs = new FileStream("empleados.dat", FileMode.Append, FileAccess.Write);
        bw = new BinaryWriter(fs);
        nombre = txtNombre.Text;
        edad = Convert.ToInt32(txtEdad.Text);
        sueldo = Convert.ToDecimal(txtSueldo.Text);
        bw.Write(nombre);
        bw.Write(edad);
        bw.Write(sueldo);
        MessageBox.Show("Los datos fueron almacenados");
        txtNombre.Clear();
        txtEdad.Clear();
        txtSueldo.Clear();
        txtNombre.Focus();
    }
    catch (Exception)
    {
        MessageBox.Show("Ingrese los datos correctamente");
    }
    finally
    {
        if (bw != null) bw.Close();
    }
}
private void btnLeer_Click(object sender, EventArgs e)
{
    try
    {
        fs = new FileStream("empleados.dat", FileMode.Open, FileAccess.Read);
        br = new BinaryReader(fs);
        dgvEmpleados.Rows.Clear();
        while (true)
        {
            nombre = br.ReadString();
            edad = br.ReadInt32();
            sueldo = br.ReadDecimal();
            dgvEmpleados.Rows.Add(nombre, edad, sueldo);
        }
    }
    catch (Exception)
    {
    }
    finally
    {
        if (br != null) br.Close();
        dgvEmpleados.ClearSelection();
    }
}
```

```
private void btnSalir_Click(object sender, EventArgs e)
{
    this.Close();
}
```

Ejemplo 2: uso del diccionario (dictionary) y estructuras de datos serializadas en archivos binarios



Control	Name	Text	Mask
MaskedTextBox1	mktNIT		0000-000000-000-0
TextBox1	txtNombre		
TextBox2	txtSueldo		
Button1	btnGuardar	Guardar	
Button2	btnSalir	Salir	
DataGridView1	dgvEmpleados		
MaskedTextBox2	mktNitBuscar		0000-000000-000-0
Button3	btnBuscar	Buscar	

Agregar los siguientes espacios de nombre:

```
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
```

Crear las siguientes variables y estructura a utilizar en el programa, en las declaraciones generales, fuera de los métodos

```
[Serializable]
6 references
public struct Empleado
{
    public string nit;
    public string nombre;
    public decimal sueldo;
}

//El diccionario empleadosDictionary almacenará como clave el NIT de tipo string
//y como valor guardará una estructura de tipo Empleado (nit, nombre y sueldo)
private static Dictionary<string, Empleado> empleadosDictionary
    = new Dictionary<string, Empleado>();
private static BinaryFormatter formatter = new BinaryFormatter();
private const string NOMBRE_ARCHIVO = "datos.dat";
```

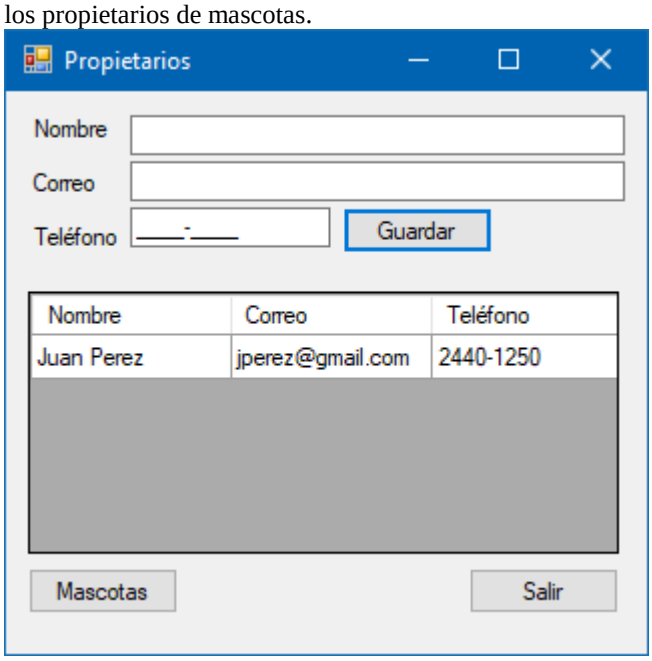
```
//Función personalizada para guardar el diccionario serializado en el archivo binario
1 reference
private static void guardarArchivo()
{
    try
    {
        FileStream writerFS = new FileStream(NOMBRE_ARCHIVO, FileMode.Create, FileAccess.Write);
        formatter.Serialize(writerFS, empleadosDictionary);
        writerFS.Close();
        MessageBox.Show("Los datos fueron guardados");
    }
    catch (Exception)
    {
        Console.WriteLine("No fue posible almacenar los datos de empleados");
    }
}

//Función personalizada para leer el archivo binario y desarializar el objeto
2 references
private static void leerArchivo()
{
    if (File.Exists(NOMBRE_ARCHIVO))
    {
        try
        {
            FileStream readerFS = new FileStream(NOMBRE_ARCHIVO,FileMode.Open, FileAccess.Read);
            empleadosDictionary = (Dictionary<String, Empleado>) formatter.Deserialize(readerFS);
            readerFS.Close();
        }
        catch (Exception)
        {
            Console.WriteLine("El archivo no esta disponible o no fue posible leerlo");
        }
    }
}

//Función personalizada para actualizar el contenido del dataGrid
2 references
public static void actualizarGrid(ref DataGridView grid)
{
    if (empleadosDictionary.Count > 0)
    {
        grid.Rows.Clear();
        foreach (Empleado empleado in empleadosDictionary.Values)
        {
            grid.Rows.Add(empleado.nit, empleado.nombre, empleado.sueldo);
        }
        grid.ClearSelection();
    }
    else
    {
        MessageBox.Show("No existe información almacenada sobre empleados");
    }
}
```

```
private void Form1_Load(object sender, EventArgs e)
{
    leerArchivo();
    actualizarGrid(ref dgvEmpleados);
}

private void btnGuardar_Click(object sender, EventArgs e)
{
    try
    {
        Empleado empleado = new Empleado();
        empleado.nit = mktNIT.Text;
        empleado.nombre = txtNombre.Text;
        empleado.sueldo = Convert.ToDecimal(txtSueldo.Text);
        if (empleadosDictionary.ContainsKey(empleado.nit))
        {
            MessageBox.Show("Ya fue agregado antes un empleado con NIT " + empleado.nit);
        }
        else
        {
            empleadosDictionary.Add(empleado.nit, empleado);
            guardarArchivo();
            leerArchivo();
            actualizarGrid(ref dgvEmpleados);
            mktNIT.Clear();
            txtNombre.Clear();
            txtSueldo.Clear();
            mktNIT.Focus();
        }
    }
    catch (Exception)
    {
        MessageBox.Show("Ingrese los datos corretamente");
    }
}

private void btnBuscar_Click(object sender, EventArgs e)
{
    //Buscar empleado por NIT
    if (dgvEmpleados.Rows.Count > 0 && mktNitBuscar.MaskFull)
    {
        if (empleadosDictionary.ContainsKey(mktNitBuscar.Text))
        {
            Empleado empleadoEncontrado = empleadosDictionary[mktNitBuscar.Text];
            MessageBox.Show("Datos del empleado:\n\n" + "NIT: " + empleadoEncontrado.nit + "\n" +
                            "Nombre: " + empleadoEncontrado.nombre + "\n" + "Sueldo: " +
                            empleadoEncontrado.sueldo.ToString("C1"));
        }
        else
        {
            MessageBox.Show("No existe un empleado con el NIT: "+ mktNitBuscar.Text);
        }
    }
}
```

	Agregar un nuevo proyecto de nombre **práctica7** a la solución del portafolio en Visual Studio .NET

Indicaciones: agregar a la solución un proyecto de C# de tipo Aplicación Windows Forms y programar los siguientes formularios.

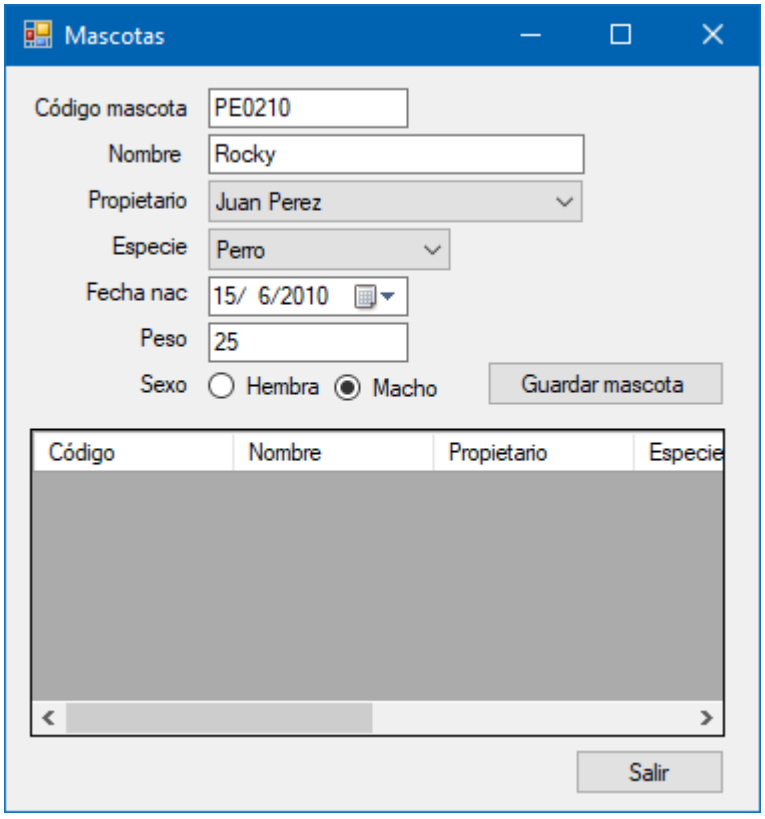
1. Desarrollar una aplicación que almacene en un **archivo binario** con nombre "propietarios.dat", el nombre, correo y teléfono de los propietarios de mascotas.



El botón Guardar: almacenará en el archivo binario, el nombre, correo y teléfono del propietario, mostrará los datos en el Grid, limpiará los controles y enviará el foco al TextBox Nombre.

El botón Mascotas: abrirá el formulario Mascotas, que será creado en el siguiente ejercicio.

2. Crear una aplicación para almacenar un **diccionario** (dictionary) con **estructuras de datos serializadas** en un **archivo binario** con el nombre "mascotas.dat"



Estructura Mascota: código, nombre, propietario, especie, fechaNacimiento, peso y sexo.

El código de la mascota, debe ser único y estará compuesto por 2 letras y 4 dígitos. (será ingresado manualmente por teclado)

El ComboBox Propietario: debe poblarse con los nombres almacenados en el archivo binario "propietarios.dat"

El ComboBox Especie: mostrará los valores fijos de: Perro, Gato, Perico, y otros que desee agregar.

El botón Guardar Mascota: almacenará los datos en el diccionario y luego de manera serializada en el archivo binario "mascotas.dat", mostrará los datos en el Grid, limpiará los controles y enviará el foco al TextBox Código mascota.