PREGRADO

UNIDAD 2 | DESARROLLO DE APLICACIONES BASADO EN COMPONENTES

COMPONENTES ARCHIVOS

SEMANA 6

S1AS10SI393 | Fundamentos de Sistemas de Información

Al finalizar la unidad, el estudiante implementa soluciones teniendo en cuenta la los componentes de una aplicación, así como diversos elementos en el entorno visual.

AGENDA

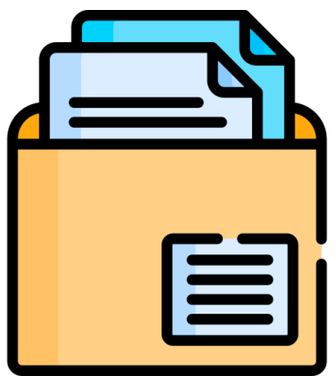
- Definición.
- Validaciones en Formularios (obligatorio, formato especifico, numeral, fechas).
- ErrorProvider.
- MessageBox.

Archivos

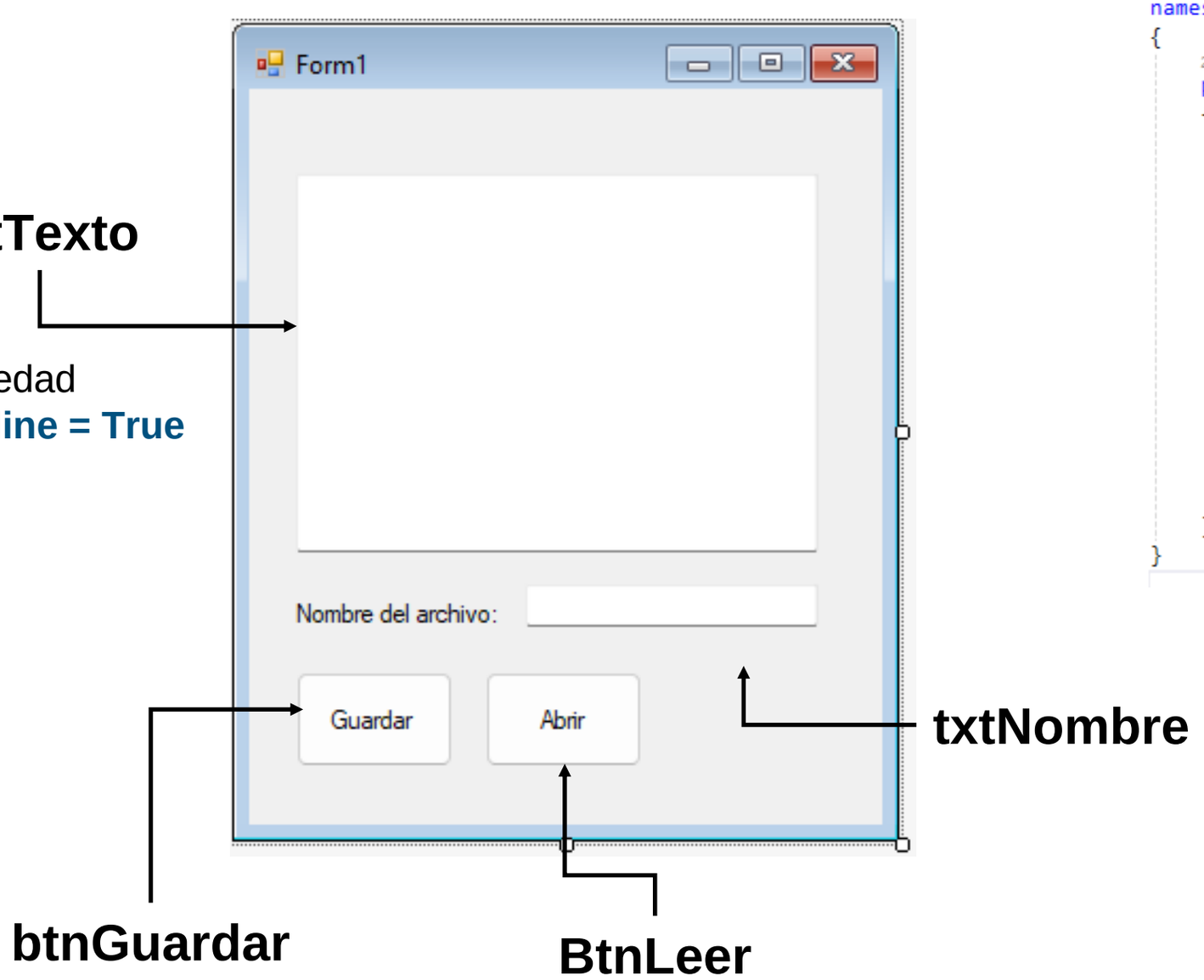
- un archivo se refiere generalmente a un objeto que representa un recurso de almacenamiento en el sistema de archivos.
- Los archivos son utilizados para almacenar datos de manera persistente.
- C# proporciona diversas clases para la manipulación de archivos en el espacio de nombres System.IO.
- Se pueden realizar operaciones de lectura, escritura, creación y eliminación de archivos.
- Existen dos tipos principales de archivos:
  - **Archivos de texto:** Contienen datos en formato de texto plano.
  - **Archivos binarios:** Contienen datos en formato binario, utilizados para almacenar imágenes, videos y otros formatos no textuales.

Clases Principales para Manejo de Archivos

1. **File y FileInfo**: Manipulación de archivos.
2. **Directory y DirectoryInfo**: Manipulación de directorios.
3. **StreamReader y StreamWriter**: Lectura y escritura de archivos de texto.
4. **BinaryReader y BinaryWriter**: Lectura y escritura de archivos binarios.



Creación y lectura de un archivo




```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace files_archivo
{
    2 referencias
    public class archivo
    {
        1 referencia
        public void guardaArchivo(string nombre, string texto)
        {
            File.WriteAllText(nombre,texto);
        }

        1 referencia
        public string leerArchivo(string nombre)
        {
            string contenido = "";
            if (File.Exists(nombre))
                contenido = File.ReadAllText(nombre);
            return contenido;
        }
    }
}
```

Agrega la clase **files_archivo**, esta clase tiene los métodos para guardar y leer un archivo de texto:

1. Crea un archivo de texto con **File.WriteAllText**
2. Hace una lectura completa con **File.ReadAllText**

En el Form

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

namespace files_archivo
{
    3 referencias
    public partial class Form1: Form
    {
        archivo nuevoArchivo = new archivo();
        1 referencia
        public Form1()
        {
            InitializeComponent();
        }
        1 referencia
        private void limpiar()
        {
            txtTexto.Clear();
            txtNombre.Clear();
        }

private void btnGuardar_Click(object sender, EventArgs e)
{
    string nombreArchivo = txtNombre.Text.Trim();
    string textoArchivo = txtTexto.Text.Trim();

    try
    {
        nuevoArchivo.guardaArchivo(nombreArchivo,textoArchivo);
        MessageBox.Show("ArcHivo Guardado", "Mensaje", MessageBoxButtons.OK,
                        MessageBoxIcon.Information);
    }
    catch (UnauthorizedAccessException)
    {
        MessageBox.Show("No tienes permisos para escribir en este archivo",
                        "Mensaje",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    catch (DirectoryNotFoundException)
    {
        MessageBox.Show("La ruta especificada no existe",
                        "Mensaje", MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
    }
    catch (IOException ex)
    {
        MessageBox.Show("Error de entrada/salida: "+ ex.Message.ToString(),
                        "Mensaje",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ocurrio un error inesperado: " + ex.Message.ToString(),
                        "Mensaje",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    limpiar();
}

    1 referencia
    private void BtnLeer_Click(object sender, EventArgs e)
    {
        string nombreArchivo = txtNombre.Text.Trim();
        string contenido = nuevoArchivo.leerArchivo(nombreArchivo);
        if(string.IsNullOrEmpty(contenido))
            MessageBox.Show("No existe el archivo o no se puede abrir",
                            "Mensaje",
                            MessageBoxButtons.OK,
                            MessageBoxIcon.Error);
        else
            txtTexto.Text = contenido;
    }
}
```

Prueba el programa

observaciones

Escritura de múltiples líneas en un archivo

```
public class archivo
{
    1 referencia
    public void guardaArchivo(string nombre, string texto)
    {
        File.WriteAllText(nombre, texto);
    }

    1 referencia
    public string leerArchivo(string nombre)
    {
        string contenido = "";
        if (File.Exists(nombre))
            contenido = File.ReadAllText(nombre);
        return contenido;
    }

    0 referencias
    public void guardaArchivoLineas(string nombre, string[] texto)
    {
        //string[] texto = { "Línea 1", "Línea 2", "Línea 3" };
        File.WriteAllLines(nombre, texto);
    }

    0 referencias
    public string[] leerArchivoLineas(string nombre)
    {
        string[] lineas = Array.Empty<string>();
        if (File.Exists(nombre))
            lineas = File.ReadAllLines(nombre);
        return lineas;
    }
}
```

Se ha añadido los métodos para crear y leer un archivo por líneas (vector de cadenas), donde cada línea es un elemento de un vector

Empleando StreamReader y StreamWriter

```
public class Archivos
{
    0 referencias
    public bool crearArchivoTexto(string nombreArchivo, string texto)
    {
        try
        {
            using (StreamWriter NuevoArchivo = new StreamWriter(nombreArchivo))
            {
                int bufferSize = 1024 * 10; // Tamaño del buffer de escritura
                for (int i = 0; i < texto.Length; i += bufferSize)
                {
                    string parte = texto.Substring(i, Math.Min(bufferSize, texto.Length - i));
                    NuevoArchivo.Write(parte);
                    // Asegura que los datos se escriban en el archivo periódicamente
                    // Flush libera el buffer
                    NuevoArchivo.Flush();
                }
            }
            return true;
        }
        catch (Exception)
        {
            return false;
        }
    }
    0 referencias
    public string leerArchivoTexto(string nombreArchivo)
    {
        try
        {
            using (StreamReader viejoArchivo = new StreamReader(nombreArchivo))
            {
                return viejoArchivo.ReadToEnd();
            }
        }
        catch (Exception)
        {
            return null;
        }
    }
}
```

- **StreamReader** y **StreamWriter** son útiles para archivos de texto.
- **using** asegura que los recursos sean liberados adecuadamente.
- Es importante manejar excepciones (**try-catch**) para evitar errores en la manipulación de archivos.

Practica lo aprendido: Gestión de Inventario con Archivos de Texto

Crea un programa en C# que gestione un inventario de productos utilizando archivos de texto. El programa debe permitir al usuario:

1. Agregar productos: El usuario introduce el nombre del producto, la cantidad y el precio. Estos datos deben guardarse en un archivo de texto (inventario.txt).
2. Leer el inventario: Mostrar en consola el contenido del archivo, mostrando cada producto en una línea con su información.
3. Buscar un producto: Permitir al usuario ingresar el nombre de un producto y mostrar su información si existe en el archivo.
4. Actualizar la cantidad de un producto: El usuario puede modificar la cantidad de un producto específico.
5. Eliminar un producto: Permitir al usuario eliminar un producto del archivo.

Cada línea representa un producto y sigue el formato:

nombreProducto, cantidadProducto, precioProducto

Ejemplo:

- Laptop,5,1200.50
- Mouse,20,15.99
- Teclado,10,25.00

Requisitos:

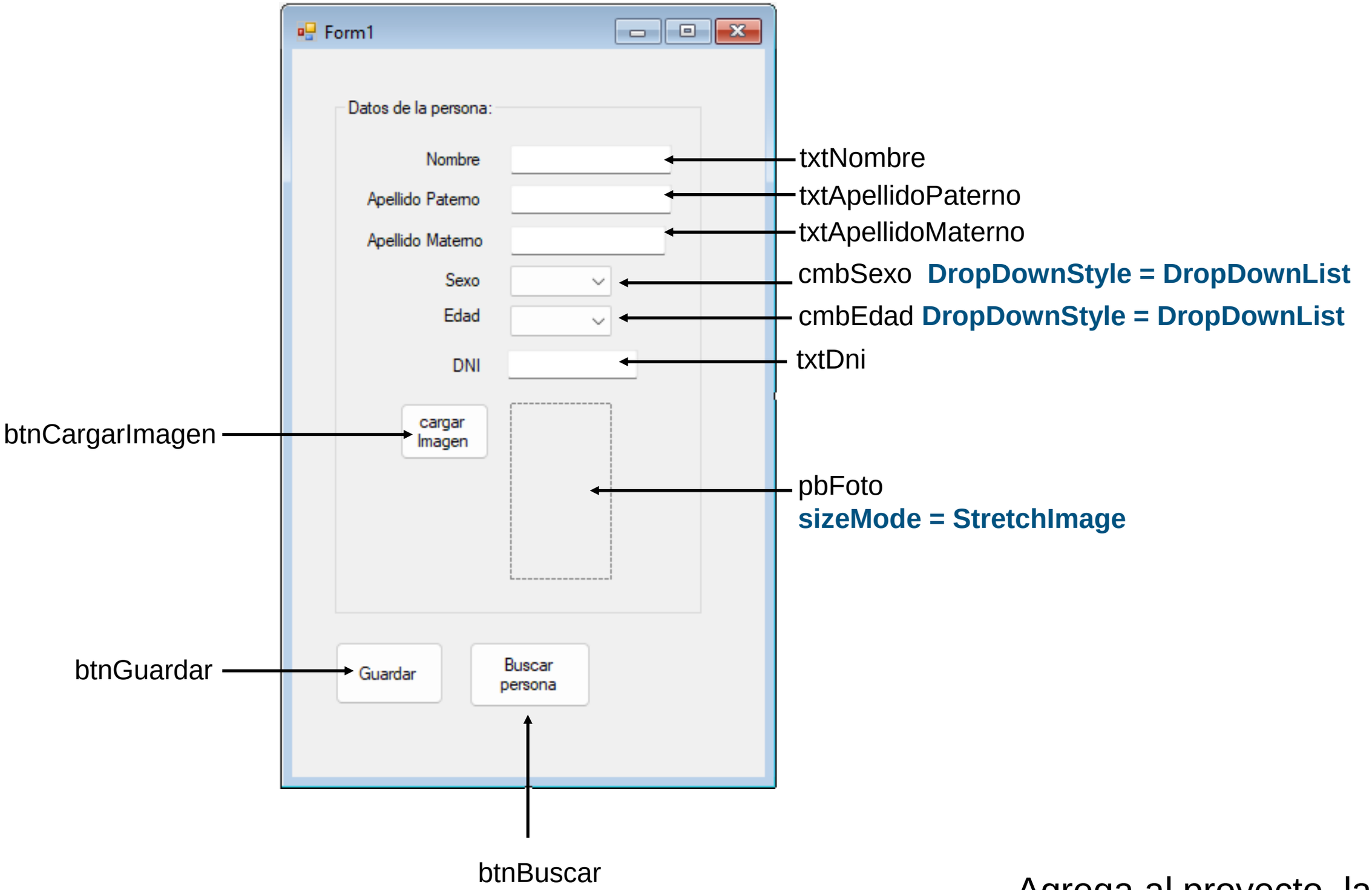
- Utilizar StreamWriter y StreamReader para leer y escribir en el archivo
- Manejar excepciones para evitar errores al acceder al archivo.

¿Qué son los archivos binarios?

- Son archivos que almacenan datos en formato binario (no texto).
- Se emplean para guardar imágenes, audios, videos, datos estructurados, etc.
- Sus principales clases son: **FileStream**, **BinaryWriter** y **BinaryReader**.

Clase	acción
FileStream	Abre y manipula archivos binarios.
BinaryWriter	Escribe datos binarios en un archivo.
BinaryReader	Lee datos binarios de un archivo.

Para explicar mejor desarrollaremos una aplicación trivila para registrar los datos de personas {nombre, apellidoPaterno, apellidoMaterno, sexo, edad, dni, y su foto}, se empleará la clase persona para gestionar los datos y los métodos que manipularán la información del archivo



Agrega al proyecto, la clase **persona**

```cs
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace binario
{
    7 referencias
    public class persona
    {
        4 referencias
        public string Nombre { get; set; }
        4 referencias
        public string ApellidoPaterno { get; set; }
        4 referencias
        public string ApellidoMaterno { get; set; }
        4 referencias
        public string Sexo { get; set; }
        4 referencias
        public int Edad { get; set; }
        4 referencias
        public string Dni { get; set; }
        5 referencias
        public byte[] Foto { get; set; }
        string nombreArchivo = "datosPersona.bin";

    public bool DniExiste(string dniBuscado)
    {
        if (!File.Exists(nombreArchivo))
            return false; // Si el archivo no existe, el DNI no está registrado.
        using (FileStream archivoBinario = new FileStream(nombreArchivo, FileMode.Open, FileAccess.Read))
        using (BinaryReader lectorBinario = new BinaryReader(archivoBinario))
        {
            while (lectorBinario.BaseStream.Position < lectorBinario.BaseStream.Length)
            {
                // Leer los datos en el mismo orden en que fueron escritos
                string nombre = lectorBinario.ReadString();
                string apellidoPaterno = lectorBinario.ReadString();
                string apellidoMaterno = lectorBinario.ReadString();
                string sexo = lectorBinario.ReadString();
                int edad = lectorBinario.ReadInt32();
                // DNI debe leerse en la misma posición en que fue escrito
                string dni = lectorBinario.ReadString();
                int fotoLength = lectorBinario.ReadInt32();
                lectorBinario.ReadBytes(fotoLength); // Foto

                if (dni == dniBuscado)
                    return true; // Si encontramos el DNI, retornamos true.
            }
        }
        return false; // No se encontró el DNI.
    }
```

FileMode.Open abre el archivo,
FileAccess.Read, lo abre para lectura

```
public bool GuardarPersona(persona persona)
{
    try
    {
        using (FileStream archivoBinario = new FileStream(nombreArchivo, FileMode.Append))
        using (BinaryWriter EscritorBinario = new BinaryWriter(archivoBinario))
        {
            EscritorBinario.Write(persona.Nombre);
            EscritorBinario.Write(persona.ApellidoPaterno);
            EscritorBinario.Write(persona.ApellidoMaterno);
            EscritorBinario.Write(persona.Sexo);
            EscritorBinario.Write(persona.Edad);
            EscritorBinario.Write(persona.Dni);
            EscritorBinario.Write(persona.Foto.Length);
            EscritorBinario.Write(persona.Foto);
        }
        return true;
    }
    catch (Exception ex)
    {
        return false;
    }
}
```

FileMode.Append abre el archivo para agregar información en él, si no existe, lo crea **FileMode.Create**, crea el archivo, si existe, lo reescribe

```
public persona BuscarPersonaPorDni(string dniBuscado)
{
    if (!File.Exists(nombreArchivo))
    {
        return null; // Retorna null si no hay archivo
    }
    using (FileStream archivoBinario = new FileStream(nombreArchivo, FileMode.Open, FileAccess.Read))
    using (BinaryReader lectorBinario = new BinaryReader(archivoBinario))
    {
        while (lectorBinario.BaseStream.Position < lectorBinario.BaseStream.Length)
        {
            // Leer los datos en el mismo orden en que fueron escritos
            string nombre = lectorBinario.ReadString();
            string apellidoPaterno = lectorBinario.ReadString();
            string apellidoMaterno = lectorBinario.ReadString();
            string sexo = lectorBinario.ReadString();
            int edad = lectorBinario.ReadInt32();
            string dni = lectorBinario.ReadString();
            int fotoLength = lectorBinario.ReadInt32();
            byte[] foto = lectorBinario.ReadBytes(fotoLength); // Leer la foto

            if (dni == dniBuscado)
            {
                // Si el DNI coincide, devuelve un objeto Persona con los datos encontrados
                return new persona
                {
                    Nombre = nombre,
                    ApellidoPaterno = apellidoPaterno,
                    ApellidoMaterno = apellidoMaterno,
                    Sexo = sexo,
                    Edad = edad,
                    Dni = dni,
                    Foto = foto
                };
            }
        }
        return null;
    }
}
```

La clase persona adema de los atributos, tiene los métodos:

- `public bool DniExiste(string dniBuscado)`, devuelve true si el dni ya esta registrado en el archivo, y devuelve false si no lo encuentra registrado.
- `public bool GuardarPersona(persona persona)`, devuelve true si se ha guardado la información de la persona, de lo contrario devuelve false.
- `public persona BuscarPersonaPorDni(string dniBuscado)`, devuelve un objeto persona cargado de información en caso se encuentre los datos de algún registro asociado al dni enviado como parámetro al método, si en caso no existe, devuelve null.

En el formulario

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Linq.Expressions;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace binario
{
    3 referencias
    public partial class Form1: Form
    {
        private persona objPersona = new persona();
        1 referencia
        public Form1()
        {
            InitializeComponent();
        }
        1 referencia
        private void Form1_Load(object sender, EventArgs e)
        {

            cargarComboEdad();
        }
        1 referencia
        private void cargarComboEdad()
        {
            cmbEdad.Items.Add(" ");
            for (int i = 1; i <= 100; i++)
            {
                cmbEdad.Items.Add(i);
            }
        }
```

```
private bool validarNombre()
{
    if(txtNombre.Text.Trim().Length < 3)
        return false;
    else
        return true;
}
1 referencia
private bool validarApellidoPaterno()
{
    if (txtApellidoPaterno.Text.Trim().Length < 3)
        return false;
    else
        return true;
}
1 referencia
private bool validarApellidoMaterno()
{
    if (txtApellidoMaterno.Text.Trim().Length < 3)
        return false;
    else
        return true;
}

1 referencia
private bool validarSexo()
{
    if (string.IsNullOrEmpty(cmbSexo.Text))
        return false;
    else
        return true;
}
1 referencia
private bool validarEdad()
{
    if (string.IsNullOrEmpty(cmbEdad.Text))
        return false;
    else
        return true;
}
```

```
private bool validarDni()
{
    if (txtDni.Text.Trim().Length != 8)
        return false;
    else
    {
        if (Regex.IsMatch(txtDni.Text.Trim(), @"^\d{8}$"))
            return true;
        else
            return false;
    }
}
1 referencia
private bool validarFoto()
{
    if (pbFoto.Image == null)
        return false;
    else
        return true;
}

3 referencias
private void limpiarControles()
{
    txtNombre.Clear();
    txtApellidoPaterno.Clear();
    txtApellidoMaterno.Clear();
    txtDni.Clear();
    cmbSexo.SelectedIndex = 0;
    cmbEdad.SelectedIndex = 0;
    pbFoto.Image = null;
    btnCargarImagen.ForeColor = Color.Black;
}
private void btnCargarImagen_Click(object sender, EventArgs e)
{
    OpenFileDialog archivoFoto = new OpenFileDialog();
    archivoFoto.Filter = "Archivos de Imagenes (*.png, *.jpg)|*.png; *.jpg";

    if (archivoFoto.ShowDialog() == DialogResult.OK)
    {
        pbFoto.Image = System.Drawing.Image.FromFile(archivoFoto.FileName);
    }

}
```

```
private void btnGuardar_Click(object sender, EventArgs e)
{
    persona nuevaPersona = new persona();
    if (!validarNombre())
    {
        MessageBox.Show("Debe ingresar un nombre (min 3 caracteres)",
                        "Error en nombre",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
        txtNombre.Clear();
        txtNombre.Focus();
        return;
    }

    if (!validarApellidoPaterno())
    {
        MessageBox.Show("Debe ingresar el apellido paterno (min 3 caracteres)",
                        "Error en apellido paterno",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
        txtApellidoPaterno.Clear();
        txtApellidoPaterno.Focus();
        return;
    }

    if (!validarApellidoMaterno())
    {
        MessageBox.Show("Debe ingresar el apellido materno (min 3 caracteres)",
                        "Error en apellido materno",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
        txtApellidoMaterno.Clear();
        txtApellidoMaterno.Focus();
        return;
    }
```

```
if (!validarSexo())
{
    MessageBox.Show("Debe seleccionar el sexo de la persona",
                    "Error en sexo de la persona",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
    return;
}

if (!validarEdad())
{
    MessageBox.Show("Debe seleccionar la edad de la persona",
                    "Error en edad de la persona",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
    return;
}

if (!validarDni())
{
    MessageBox.Show("Debe ingresar el dni (8 digitos)",
                    "Error en edad de la persona",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
    txtDni.Focus();
    return;
}
if (!validarFoto())
{
    MessageBox.Show("Debe cargar la foto de la persona",
                   "Error en foto",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
    btnCargarImagen.ForeColor = Color.Red;
    return;
}
```

```
// si pasa la validacion, hay que verificar que el dni no este
// registrado para otra persona
if (objPersona.DniExiste(txtDni.Text.Trim()))
{
    MessageBox.Show("El DNI ingresado, ya se encuentra registrado",
                    "MENSAJE",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
    txtDni.Focus();
    return;
}
else
{
    //guardamos el archivo
    objPersona.Nombre = txtNombre.Text.Trim().ToLower();
    objPersona.ApellidoPaterno = txtApellidoPaterno.Text.Trim().ToLower();
    objPersona.ApellidoMaterno = txtApellidoMaterno.Text.Trim().ToLower();
    objPersona.Dni = txtDni.Text.Trim();
    objPersona.Sexo = cmbSexo.SelectedItem.ToString();
    objPersona.Edad = int.Parse(cmbEdad.SelectedItem.ToString());
    using (MemoryStream ms = new MemoryStream())
    {
        pbFoto.Image.Save(ms, System.Drawing.Imaging.ImageFormat.Png); // Guardar como PNG
        objPersona.Foto = ms.ToArray();
    }
    if (objPersona.GuardarPersona(objPersona))
    {
        //ok
        MessageBox.Show("Los datos de la persona se almacenaron correctamente",
                    "MENSAJE",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information);
        limpiarControles();
    }
    else
    {
        //no se pudo guardar
        MessageBox.Show("No se pudo guardar los datos",
                    "MENSAJE",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
        this.Close();
    }
}
}
```

```
        private void btnBuscar_Click(object sender, EventArgs e)
        {
            if (!validarDni())
            {
                MessageBox.Show("Debe ingresar un dni (8 digitos)",
                                "Ingrese un DNI para buscar",
                                MessageBoxButtons.OK,
                                MessageBoxIcon.Error);
                limpiarControles();
                txtDni.Focus();
                return;
            }
            else
            {
                objPersona = objPersona.BuscarPersonaPorDni(txtDni.Text.Trim());
                if (objPersona == null)
                {
                    //no existe o no hay archivo
                    MessageBox.Show("No existe el DNI ingresado, o ha ocurrido un error en el archivo",
                                "NO EXISTE LA PERSONA",
                                MessageBoxButtons.OK,
                                MessageBoxIcon.Error);
                    limpiarControles();
                    return;
                }
                else
                {
                    //cargamos el objeto
                    txtNombre.Text = objPersona.Nombre;
                    txtApellidoPaterno.Text = objPersona.ApellidoPaterno;
                    txtApellidoMaterno.Text = objPersona.ApellidoMaterno;
                    txtDni.Text = objPersona.Dni;
                    cmbEdad.SelectedItem = objPersona.Edad;
                    cmbSexo.SelectedItem = objPersona.Sexo;
                    using (MemoryStream flujo = new MemoryStream(objPersona.Foto))
                    {
                        // Convertir byte[] en Image y asignarla al PictureBox
                        pbFoto.Image = Image.FromStream(flujo);
                    }
                }
            }
        }
    }
}
```

Prueba el código

Practica lo aprendido

Sistema de Gestión de Biblioteca con Multilistas y Archivos Binarios

Se desea desarrollar un sistema para la gestión de una biblioteca que almacene información sobre autores y los libros que han escrito. Para ello, se debe utilizar la estructura de multilistas y almacenar los datos en archivos binarios.

1.Estructura de Datos:

1. Cada autor tiene:
    1. Código de autor (string)
    2. Nombre completo (string)
    3. Nacionalidad (string)
    4. Lista de libros escritos (multilista)
2. Cada libro tiene:
    1. Código de libro (string)
    2. Título (string)
    3. Año de publicación (int)
    4. Género (string)

2. Funciones Principales:

- Registrar un autor en el archivo binario.
- Registrar un libro y asociarlo a un autor específico.
- Buscar un autor por su código y mostrar sus datos junto con los libros que ha escrito.
- Buscar un libro por su código y mostrar la información del autor correspondiente.
- Eliminar un libro de la lista de un autor y del archivo binario.
- Eliminar un autor, asegurándose de eliminar también todos sus libros.
- Ordenar los libros de cada autor por año de publicación antes de guardarlos.

Consideraciones Técnicas:

- Se debe usar archivos binarios para almacenar la información de autores y libros.
- Se utilizará una estructura de multilistas, donde cada autor tendrá una lista enlazada de libros.
- Se debe validar que no se dupliquen códigos de autores o libros.

PREGRADO

Ingeniería de Sistemas de Información

Escuela de Ingeniería de Sistemas y Computación | Facultad de Ingeniería

UPC
Universidad Peruana
de Ciencias Aplicadas

Prolongación Primavera 2390,
Monterrico, Santiago de Surco
Lima 33 - Perú
T 511 313 3333
https://www.upc.edu.pe

exígete, innova

LAUREATE
INTERNATIONAL
UNIVERSITIES