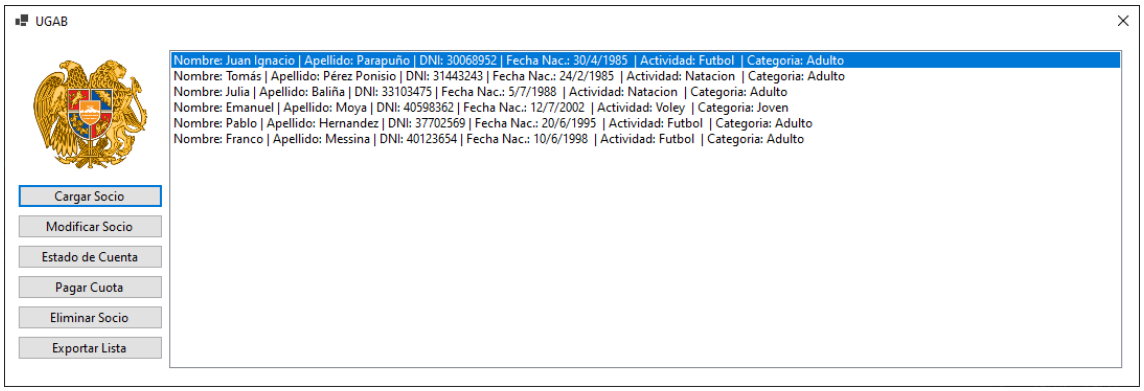
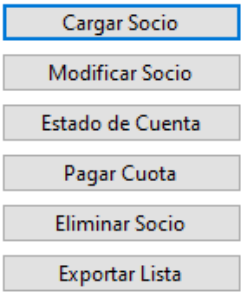
Tomás Pérez Ponisio, comisión 2E, primer cuatrimestre 2022

TP3 Final – Laboratorio 2: Gestor de socios del club UGAB

Siguiendo la consigna del TP el proyecto es un programa para gestionar los socios de un club. En el inicio se pueden ver listados los socios del club, en caso de que no haya ningún la lista estará vacía, y siempre se podrá cargar una lista de socios desde un archivo xml ubicándola en la carpeta correspondiente.

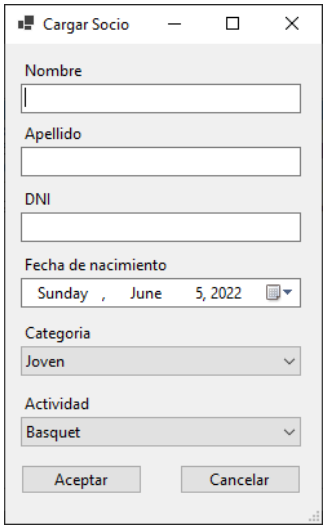


El menú tiene seis opciones con nombres descriptivos.



Pudiendo cargar un nuevo socio, modificar uno ya existente, ver el estado de cuenta de un socio (que cuotas pagó y si está al día), pagar una cuota del socio, eliminar un socio del club, y guardar los datos de la lista de socios para que la próxima vez que se abra el programa se puedan seguir gestionando los mismos datos.

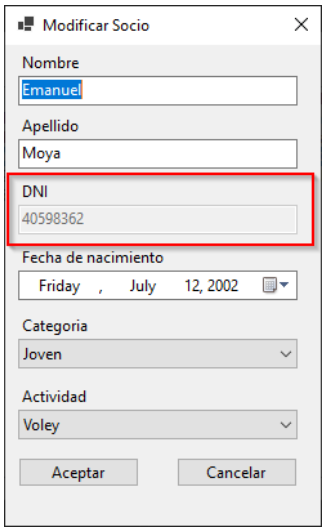
Cargar Socio



Aquí se puede cargar un nuevo socio al club ingresando todos los datos requeridos y en su correcto formato, por ejemplo, en el campo DNI solo se podrá ingresar números mayores a 0 y menores a 99999999.

El campo DNI será el identificador de cada socio, siendo este dato único e irrepetible dentro del club.

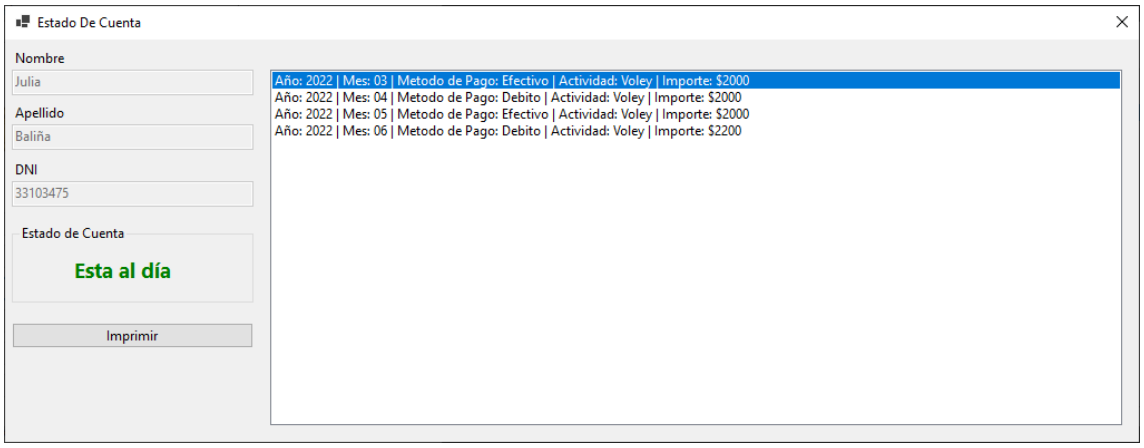
Modificar Socio

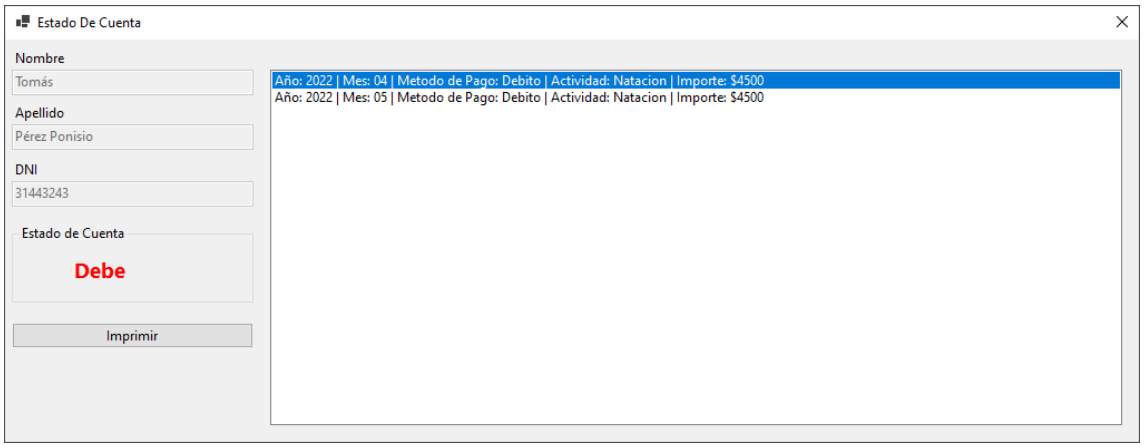


Con un socio seleccionado de la lista, se pueden modificar su datos menos el de DNI, si se quiere modificar un DNI, hay que eliminar el socio y crearlo de nuevo.

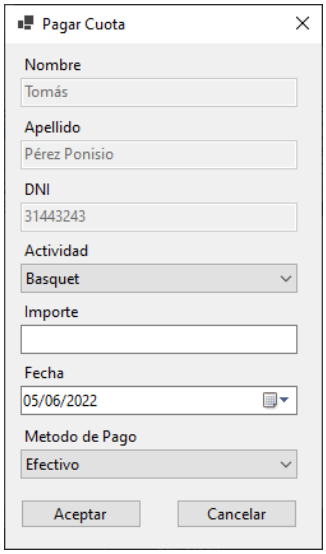
Estado de cuenta

Con un socio seleccionado se puede ver su estado de cuenta, esto es el historial de pagos de su cuota de socio y ver si debe o está al día. Un socio es deudor cuando no esta paga la cuota correspondiente al último mes. Esta información se puede imprimir, se genera un archivo de texto con los datos del socio, el historia de pagos y el estado de la cuenta.





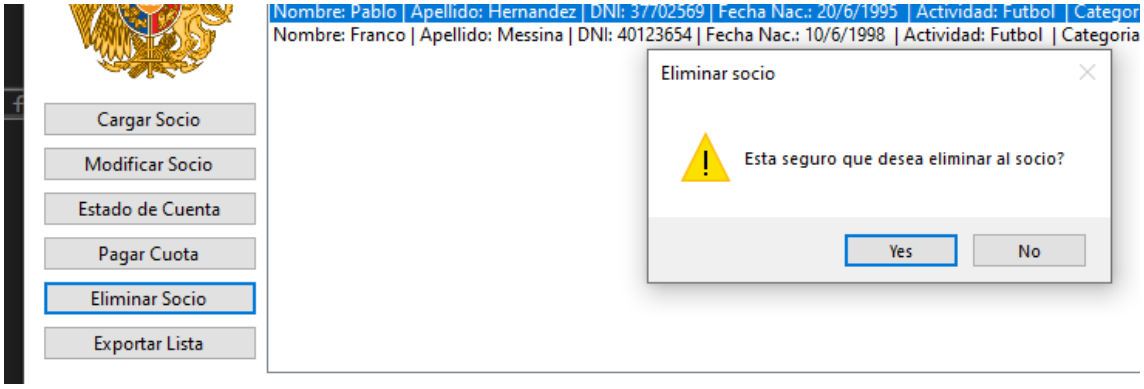
Pagar Cuota



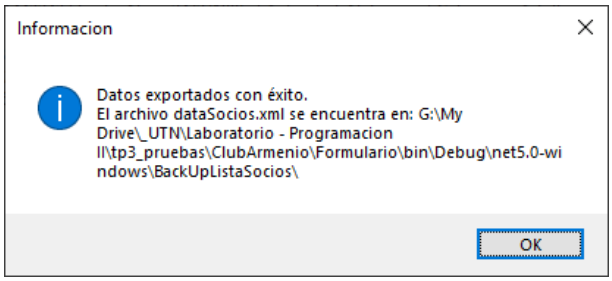
Aquí se puede pagar una cuota del socio seleccionado, se deberá ingresar el importe, fecha, la actividad y el método de pago y el registro se agregará automáticamente al historial de pagos (estado de cuenta) del socio. No se puede pagar dos veces el mismo mes, el sistema avisa con un mensaje.

Eliminar Socio

Con un socio seleccionado se puede eliminar del club, hay un pedido de confirmación previo.

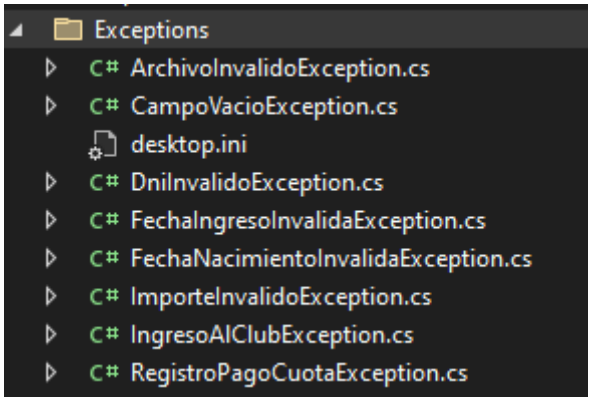


Exportar Lista



Exporta los datos del club en un archivo xml para que en el próximo inicio del programa se carguen estos mismos datos. Si al salir del programa no se exportó la lista previamente se perderán las últimas modificaciones.

Tema 10 – Excepciones



Las siguientes excepciones fueron implementadas en distintas secciones del proyecto.

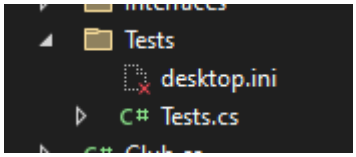
Por ejemplo, la *ArchivoInvalidoException* se dispara en todo lo que tenga que ver con manejo de archivos, en el *frmInicio* al leer o escribir el archivo con los datos de los socios, en *frmEstadoDeCuenta* al imprimir el archivo de texto.

```
17 references
public class ArchivoInvalidoException : Exception
{
    2 references
    public ArchivoInvalidoException(string message) : base(message)
    {
    }

    5 references
    public ArchivoInvalidoException(string message, Exception innerException) : base(message, innerException)
    {
    }
}
```

```
/// <summary>
/// Al cargar el formulario se busca si ya existe un archivo xml con datos de socios existentes, si lo encuentra se lee
/// y se crea un club con esa lista de socios encontrada. Si no hay archivo de datos se crea un club de cero.
/// Se actualiza la vista de la listbox.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 reference
private void frmInicio_Load(object sender, EventArgs e)
{
    try
    {
        if (File.Exists($"{serializer.RutaDeEscritura()}\\dataSocios.xml"))
        {
            this.club = new Club("Club Armenio", this.serializer.Leer("dataSocios.xml"));
        }
        else
        {
            this.club = new Club("Club Armenio");
        }
        this.ActualizaListaSocios();
    }
    catch (ArchivoInvalidoException ex)
    {
        MessageBox.Show(ex.Message, "Error al leer el archivo de datos de socios", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Tema 11 - Pruebas unitarias



Respecto de los test unitarios se testean algunas de las funcionalidades del programa, por ejemplo, el método *EliminarSocio* de la clase socio que retorna un string informando el resultado.

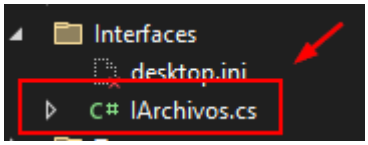
```
/// <summary>
/// Utilizando la sobrecarga del operador resta entre un club y un socio,
/// remueve el socio que recibe como paramtero de la lista del club
/// y retorna un mensaje en un string avisando si lo pudo remover o no
/// </summary>
/// <param name="socio"></param>
/// <returns>string</returns>
2 references | 1/1 passing
public string EliminarSocio(Socio socio)
{
    string mensaje = "El socio no se ha encontrado";
    if (this - socio)
    {
        mensaje = "Socio eliminado con exito";
    }
    return mensaje;
}
```

```
[TestMethod]
0 references
public void EliminarSocio_CuandoSeQuieraEliminarDeUnClubUnSocioQueNoExiste_DeberiaRetornarUnStringDiciendoQueElSocioNoSeHaEncontrado()
{
    //Arrange
    Club club = new Club("Test Club");
    Socio socio = new Socio("Juan", "Perez", 123, new DateTime(2000, 01, 01), Socio.EActividad.Basquet, Socio.ECategoria.Joven);
    string expected = "El socio no se ha encontrado";
    string actual;

    //Act
    actual = club.EliminarSocio(socio);

    //Assert
    Assert.AreEqual(expected, actual);
}
```

Tema 12 - Tipos genéricos y Tema 13 – Interfaces



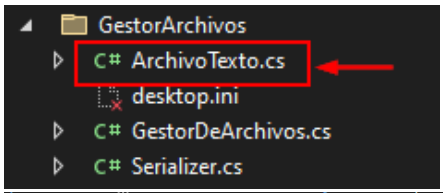
Por practicidad ambos se utilizan en la interfaz que usa el gestor de archivos

```
public interface IArchivos<T> where T : class
{
    #region Metodos
    3 references
    T Leer(string nombreArchivo);
    4 references
    void Escribir(string nombreArchivo, T elemento);
    #endregion
}
```

```
3 references
public class ArchivoTexto : GestorDeArchivo, IArchivos<string>
{
    #region Constructor
    1 reference
```

```
3 references
public class Serializer<T> : GestorDeArchivo, IArchivos<T> where T : class, new()
{
```

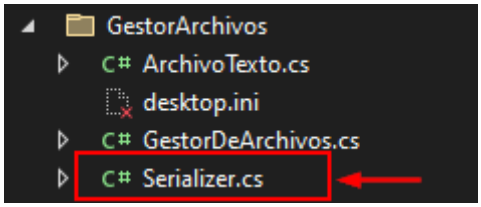
Tema 14 – Archivos



Archivos se usa en *frmEstadoDeCuenta* para imprimir un comprobante del estado de cuenta de un socio.

```
/// <summary>
/// Genera un documento de texto con los datos del socio, su historial de pago y si esta al dia con las cuotas o si debe el último mes.
/// Se abre el documento generado y se informa donde esta guardado. Si hay algun problema se arroja la exception correspondiente.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 reference
private void btnImprimir_Click(object sender, EventArgs e)
{
    try
    {
        ArchivoTexto archivo = new ArchivoTexto();
        string nombreDeArchivo;
        archivo.Escribir($"{socio.Nombre}{socio.Apellido}.txt", socio.Imprimir());
        nombreDeArchivo = $"{archivo.RutaDeEscritura()}{socio.Nombre}{socio.Apellido}.txt";
        ProcessStartInfo proceso = new ProcessStartInfo
        {
            FileName = nombreDeArchivo,
            UseShellExecute = true
        };
        Process.Start(proceso);
        MessageBox.Show($"Impresión exitosa.\nRecibo en: {archivo.RutaDeEscritura()}", "Informacion", MessageBoxButtons.OK, MessageBoxIcon
    }
    catch(ArchivoInvalidoException ex)
    {
        MessageBox.Show(ex.Message, "Error al escribir el archivo de texto", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Error al escribir el archivo de texto", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Tema 15 – Serialización



Usando un serializador en frmInicio para leer datos de socios desde un archivo xml y para exportar los datos del club a un archivo xml.

```
/// <summary>
/// Al cargar el formulario se busca si ya existe un archivo xml con datos de socios existentes, si lo encuentra se lee
/// y se crea un club con esa lista de socios encontrada. Si no hay archivo de datos se crea un club de cero.
/// Se actualiza la vista de la listbox.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 reference
private void frmInicio_Load(object sender, EventArgs e)
{
    try
    {
        if (File.Exists($"{serializer.RutaDeEscritura()}\\dataSocios.xml"))
        {
            this.club = new Club("Club Armenio", this.serializer.Leer("dataSocios.xml"));
        }
        else
        {
            this.club = new Club("Club Armenio");
        }
        this.ActualizaListaSocios();
    }
    catch (ArchivoInvalidoException ex)
    {
        MessageBox.Show(ex.Message, "Error al leer el archivo de datos de socios", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

```
/// <summary>
/// Exporta los datos de la lista de socios a un archivo "dataSocios.xml" en una carpeta llamada XML en el escritorio.
/// Si el archivo ya existe lo sobreescribe
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 reference
private void btnExportar_Click(object sender, EventArgs e)
{
    try
    {
        this.serializer.Escribir("dataSocios.xml", this.club.ListaSocios);
        MessageBox.Show($"Datos exportados con éxito.\nEl archivo dataSocios.xml se encuentra en: {this.serializer.RutaDeEscritura()
    }
    catch (ArchivoInvalidoException ex)
    {
        MessageBox.Show(ex.Message, "Error al escribir el archivo de datos de socios", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```