

Trabajo Práctico N°4

Laboratorio 2

Hamer Thomas 2A

EXPLICACIÓN DEL PROYECTO

Este Proyecto simula ser un Sistema de Atención de Público para un Gimnasio llamado Metro Flex Gym, en el que puede venir cualquier tipo de Público para darse de Alta y comenzar a entrenar en el Gimnasio o Darse de Baja y dejar el mismo. También el Recepcionista podrá Modificar a estos Socios si ellos lo desean y podrá ver un Informe General del Gimnasio y Guardar estos datos en Distinto tipos de Archivos. El Socio podrá Elegir diferentes planes del Gimnasio y deberá dar todos sus datos para una Alta exitosa.

FUNCIONALIDAD

El Recepcionista deberá Seleccionar la Capacidad Máxima del Gimnasio para poder Cargar a sus Socios.



The screenshot shows a software window titled "Capacidad". Inside the window, there is a text prompt "Ingresa La Capacidad Maxima de Socios". Below this prompt is a numeric input field containing the value "15". To the right of the input field is a small vertical spinner control with up and down arrows. At the bottom of the window is a large rectangular button labeled "Aceptar".

El Gimnasio ya viene con una Lista de Socios, la cual será Cargada desde una Base de Datos.

The screenshot shows the MetroFlex Gym application interface. At the top, there's a header with the application name and window controls. Below the header, there's a summary section with the following information:

- Capacidad de Socios: 15
- Capacidad Libre: 9
- Total: \$ 15100

On the right side of the summary section, there's a digital clock showing 18:57:37 and a date showing 18/6/2022.

The main area displays a list of gym members with the following details:

ID	Nombre	Apellido	Sexo	DNI	Fecha de Ingreso	Estatus	Pase	Pago
1	Ronnie	Coleman	MASCULINO	24625100	4/6/2022	Activo	Musculacion	Pago: Cre
2	Jay	Cutler	MASCULINO	32782642	4/6/2022	Activo	Gympass	Pago: Efectivo
3	Tom	Platz	MASCULINO	17998120	4/6/2022	Activo	Pase: Libre	Pago: Debito
4	Chris	Bumstead	MASCULINO	35147890	4/6/2022	Activo	Pase: Libre	Pago: Debito
5	Larry	Wheels	MASCULINO	36529123	4/6/2022	Activo	Gympass	Pago: Efectivo
6	Noelia	Cvitanovic	FEMENINO	42897456	4/6/2022	Activo	Pase: Libre	Pago: Debito

At the bottom of the interface, there's a navigation bar with the following buttons:

- Añadir Socio
- Eliminar Socio
- Editar Socio
- Guardar Listado en XML
- Guardar Listado en Texto
- Informes
- Filtrar Datos

Below the 'Filtrar Datos' button, there's a text input field with the placeholder text 'Apellido/Nombre/Estatus/Pase'.

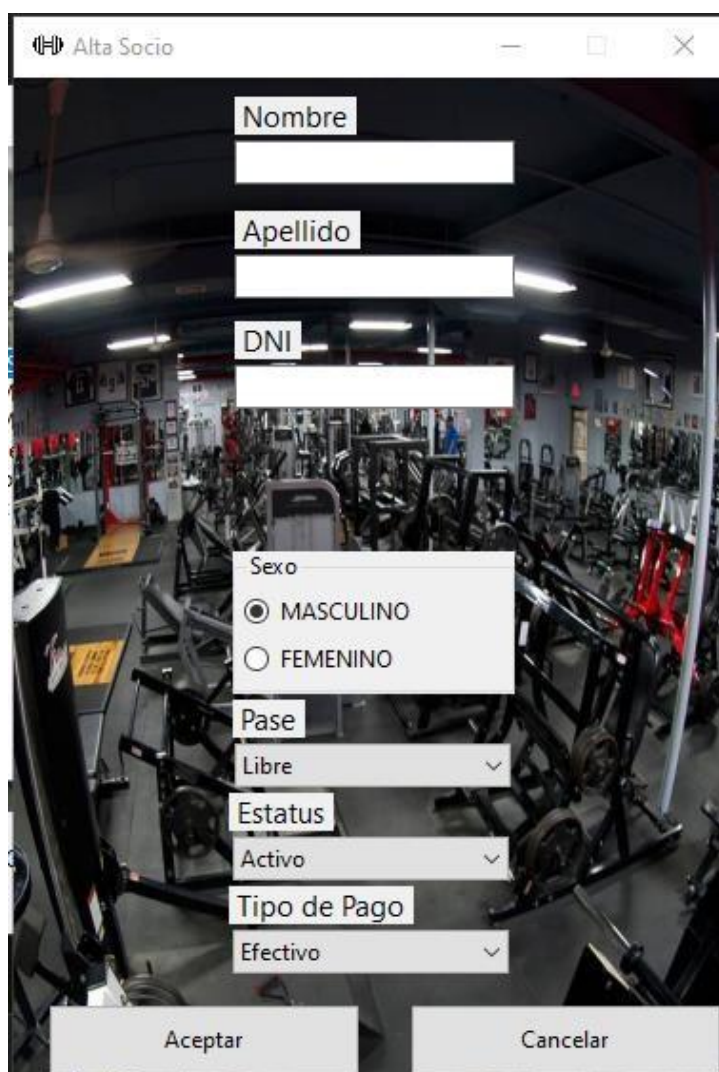
En Caso que el Usuario intente Editar o Eliminar a un Socio, que no Existe en la Base de Datos o La Pantalla de Socios este vacía, Le saldrá una Advertencia Informándole la Situación, también si el Usuario desea ver Informes del Gimnasio.

The image shows two warning dialog boxes side-by-side. Both have a title bar that says 'Warning' and a close button (X).

The left dialog box has a red circle with a white 'X' icon and the text 'El Listado Esta Vacio!!'. It has an 'Aceptar' button at the bottom.

The right dialog box has a yellow triangle with a black exclamation mark icon and the text 'No Hay Socios Ingresados!'. It also has an 'Aceptar' button at the bottom.

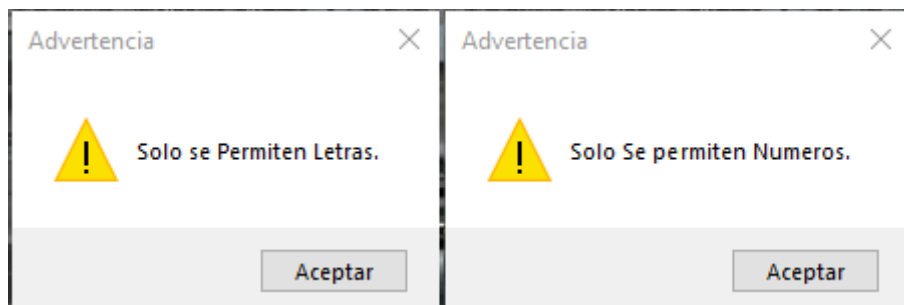
El Usuario del Proyecto Podrá dar el Alta de un Socio con sus datos.



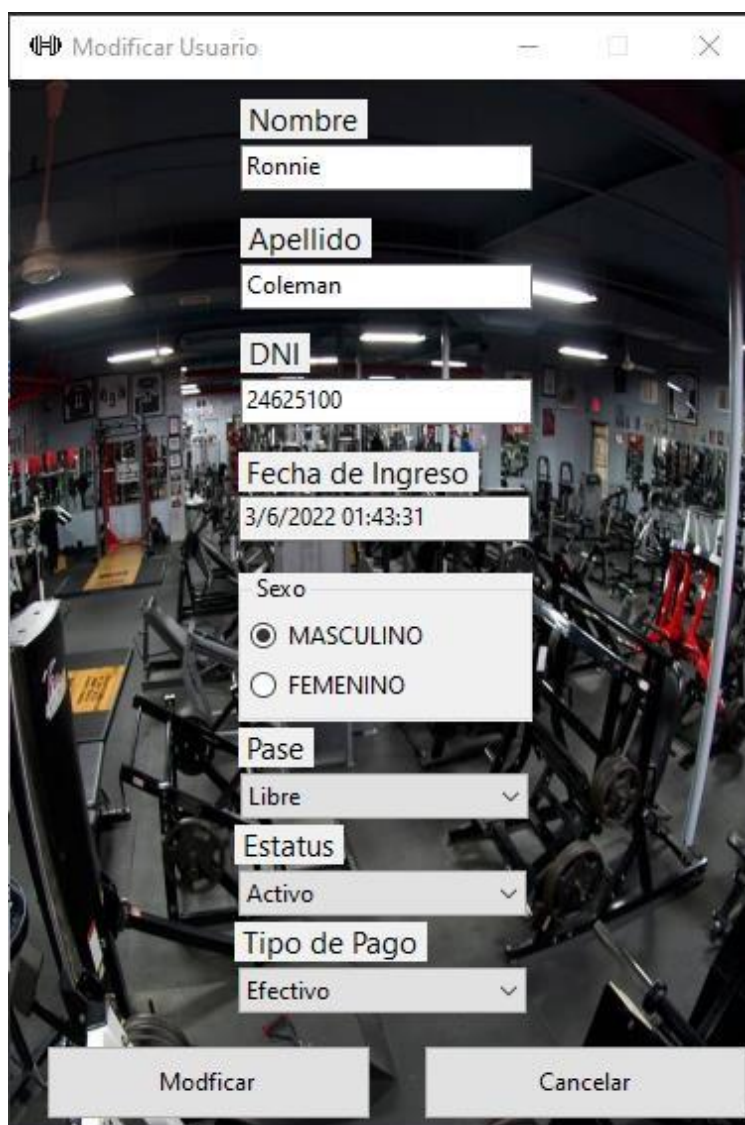
The image shows a software window titled "Alta Socio" (Member Registration) with a background image of a gym. The form contains the following fields and options:

- Nombre**: Text input field.
- Apellido**: Text input field.
- DNI**: Text input field.
- Sexo**: Radio button selection with options **MASCULINO** (selected) and **FEMENINO**.
- Pase**: Dropdown menu with the selected option **Libre**.
- Estatus**: Dropdown menu with the selected option **Activo**.
- Tipo de Pago**: Dropdown menu with the selected option **Efectivo**.
- Aceptar**: Button at the bottom left.
- Cancelar**: Button at the bottom right.

Si el Usuario de manera accidental ingresa un Numero en los Campos de Nombre y Apellido, Saldrá una Advertencia al Usuario informándole lo que debe ingresar, de la Misma manera pasa con el campo de DNI si se ingresa una Letra.



El Usuario podrá Modificar los datos de Cualquier Socio que desee.



Modificar Usuario

Nombre
Ronnie

Apellido
Coleman

DNI
24625100

Fecha de Ingreso
3/6/2022 01:43:31

Sexo
☒ MASCULINO
☐ FEMENINO

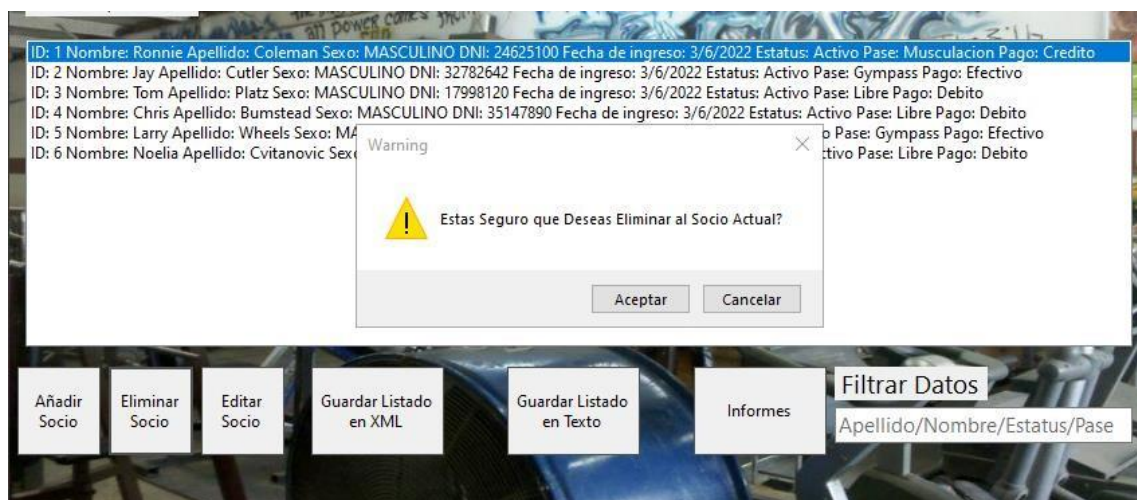
Pase
Libre

Estatus
Activo

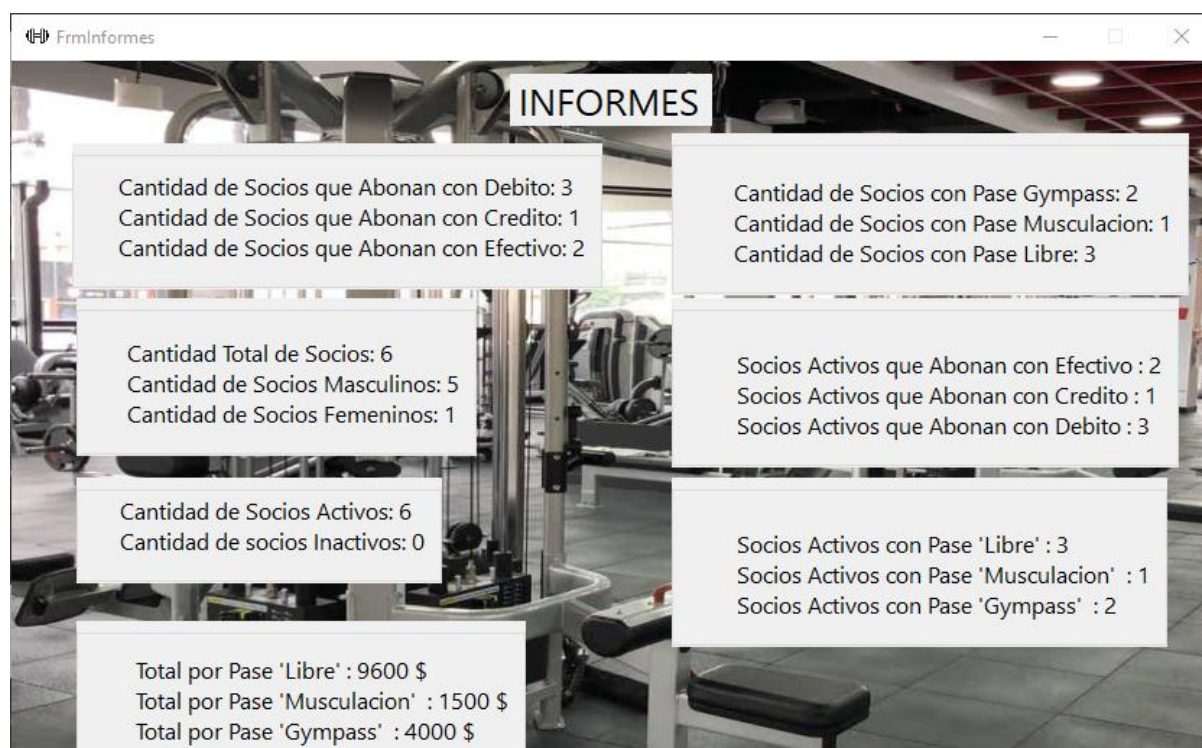
Tipo de Pago
Efectivo

Modificar Cancelar

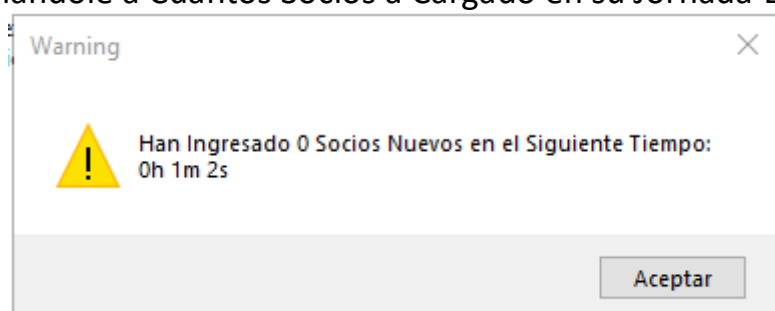
Si el Usuario desea Dar de Baja a un Socio le saldrá una Advertencia para confirmar su Elección.



El Usuario podrá ver un Informe de los Socios del Gimnasio.



Si el Usuario desea Salir de la Aplicación, le saldrá una Advertencia informándole a Cuantos Socios a Cargado en su Jornada Laboral.



El Usuario podrá Filtrar Socio para facilitar su Búsqueda, a través del Apellido/Nombre/Estatus o Pase.

ID: 3 Nombre: Tom Apellido: Platz Sexo: MASCULINO DNI: 17998120 Fecha de ingreso: 3/6/2022 Estatus: Activo Pase: Libre Pago: Debito

Añadir Socio	Eliminar Socio	Editar Socio	Guardar Listado en XML	Guardar Listado en Texto	Informes	Filtrar Datos
						Tom

Para que el Usuario pueda Utilizar la Aplicación deberá Utilizar la Base de Datos. El Usuario deberá ejecutar el Script de SQL.

Nombre	Fecha de modificación	Tipo	Tamaño
.vs	14/6/2022 17:23	Carpeta de archivos	
.vscode	14/6/2022 17:23	Carpeta de archivos	
Entidades	18/6/2022 16:45	Carpeta de archivos	
FormGimnasio	18/6/2022 18:10	Carpeta de archivos	
Hamer.Thomas.2A.TPFinal	14/6/2022 17:23	Carpeta de archivos	
Test	14/6/2022 17:23	Carpeta de archivos	
TestResults	18/6/2022 18:10	Carpeta de archivos	
TestUnitarios	14/6/2022 17:24	Carpeta de archivos	
Hamer.Thomas.2A.TPFinal.sln	3/6/2022 00:58	Visual Studio Solu...	3 KB
sociosGym.sql	18/6/2022 17:58	Microsoft SQL Ser...	5 KB

Deberá ingresar la Configuración Recomendada.

Connect to Database Engine

SQL Server

Server type: Database Engine

Server name: .

Authentication: Windows Authentication

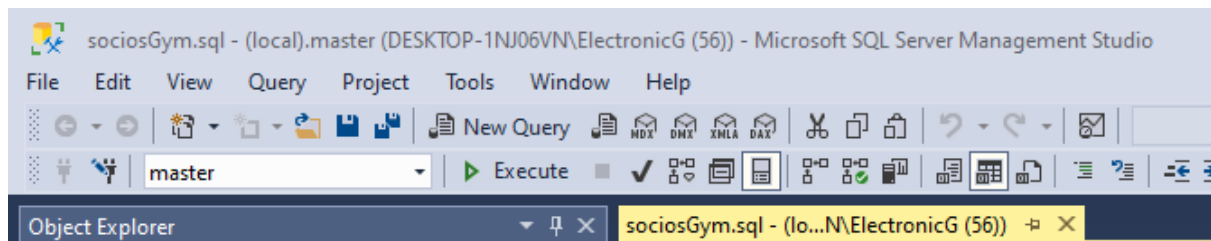
User name: DESKTOP-1NJ06VN\ElectronicG

Password:

☐ Remember password

Connect Cancel Help Options >>

Deberá Elegir la Opción de Execute y se Creara la Tabla la cual contiene a Dichos Socios con sus Respectivos Datos.



Clase 10 - Excepciones.

Implemento posibles Excepciones como por Ejemplo en caso que se sobrepase la Capacidad Máxima del Gimnasio.

```
        retorno = true;
    }
    else
    {
        throw new CapacidadMaximaException("No Hay Mas Lugares Disponibles!!!");
    }
    return retorno;
}
```

```
namespace Entidades
{
    8 referencias
    public class CapacidadMaximaException : Exception
    {
        #region Constructores
        /// <summary> Crea un Mensaje con la Excepcion.
        1 referencia
        public CapacidadMaximaException(String mensaje) : this(mensaje, null)
        {
        }

        /// <summary> Crea un Mensaje con la Excepcion y la Inner Exception.
        1 referencia
        public CapacidadMaximaException(String mensaje, Exception inner) : base(mensaje, inner)
        {
        }
        #endregion
    }
}
```

Clase 11 - Pruebas Unitarias.

Testeo los Principales Métodos del Proyecto y verifico que hagan lo correspondiente.

```
namespace TestUnitarios
{
    [TestClass]
    0 referencias
    public class TestGym
    {
        /// <summary> Valida que se puedan Agregar Socios Correctamente.
        [TestMethod]
        0 referencias
        public void AgregarSocios_Ok()...

        /// <summary> Valida que se Puedan Eliminar Socios Correctamente.
        [TestMethod]
        0 referencias
        public void EliminarSocios_Ok()...

        /// <summary> Valida que no se puedan Agregar Socios una vez Alcanzada La Capa ...
        [TestMethod]
        [ExpectedException(typeof(CapacidadMaximaException))]
        0 referencias
        public void AgregarSocios_Exception()...
    }
}
```

Clase 12 - Tipos Genéricos.

Implementado con Interfaces para Mayor practicidad.

```
namespace Entidades
{
    1 referencia
    public interface IAdministradorFiles<T>
    {
        2 referencias
        bool Exportar(T datos);
        2 referencias
        bool Importar(string ruta, out T datos);
    }
}
```


Clase 13 - Interfaces.

Implementada en la Serialización de Archivos.

```
namespace Entidades
{
    4 referencias
    public class Serializacion<T> : IAdministradorFiles<T>
        where T : class
    {
        private static string rutaArchivo;

        #region Metodos
        /// <summary>
        /// Arma la Ruta Donde se Guardara el Archivo.
        /// </summary>
        0 referencias
        static Serializacion()
        {
            string applicationData = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
            string nombreArchivo = "Socios.xml";
            rutaArchivo = Path.Combine(applicationData, nombreArchivo);
        }
        /// <summary>
        /// Exporta en XML Los Datos Pasados por Parametro.
        /// </summary>
        /// <param name="datos"></param>
        /// <returns></returns>
        2 referencias
    }
}
```

Clase 14 - Archivos y Serialización.

Utilizo Estos Principales Métodos para Serializar Archivos y poder Guardarlos tanto como XML o Txt.

```
private void ExportXml()
{
    serializador = new Serializacion<List<Socio>>();
    try
    {
        if (!string.IsNullOrEmpty(this.txtFiltro.Text))
        {
            this.gimnasio.lista = new List<Socio>(this.gimnasioFiltrado.lista);
        }

        if (serializador.Exportar(this.gimnasio.lista))
        {
            MessageBox.Show("El Archivo ha Sido Generado con Exito." +
                " El Mismo se Encuentra en el Escritorio.", "Atencion", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        }
        else
        {
            MessageBox.Show("Ocurrio un error.");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Warning", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

```

private void ExportTxt()
{
    SaveFileDialog saveFileDialog = new SaveFileDialog()
    {
        Filter = "TXT files|*.txt",
        Title = "Guardar Archivo",
        InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Desktop),
        DefaultExt = ".txt",
        CheckPathExists = true,
        CheckFileExists = false,
        FileName = "Listado socios"
    };

    if (saveFileDialog.ShowDialog() == DialogResult.OK)
    {
        try
        {
            StreamWriter sw = new StreamWriter(saveFileDialog.FileName);
            sw.WriteLine("Listado Generado El: " + this.lblFecha.Text + "\n\n");

            foreach (var item in lstSocios.Items)
            {
                sw.WriteLine(item.ToString());
            }
            sw.WriteLine("\n\nTotal Facturado: " + this.lblTotalFacturado.Text);
            sw.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Warning", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

```

```

public bool Exportar(T datos)
{
    bool retorno = false;
    try
    {
        if (rutaArchivo != null)
        {
            using (StreamWriter streamWriter = new StreamWriter(rutaArchivo))
            {
                XmlSerializer nuevoXml = new XmlSerializer(typeof(T));
                nuevoXml.Serialize(streamWriter, datos);
                retorno = true;
            }
        }
    }
    catch (Exception)
    {
        throw new Exception("Error al Querer Guardar El Archivo: " + rutaArchivo);
    }
    return retorno;
}

```

```

public bool Importar(string ruta, out T datos)
{
    bool retorno = false;
    datos = default;

    try
    {
        if (ruta != null)
        {
            using (StreamReader auxReader = new StreamReader(ruta))
            {
                XmlSerializer nuevoXml = new XmlSerializer(typeof(T));
                datos = (T)nuevoXml.Deserialize(auxReader);
                retorno = true;
            }
        }
    }
    catch (Exception e)
    {
        throw new Exception(e.Message);
    }
    return retorno;
}

```

Clase 16 - Conexión a Bases de Datos.

Utilizados para conectarse a la base de datos y poder aplicar un CRUD implementando su correspondiente interfaz.

SQLQuery1.sql - (lo...N\ElectronicG (69))

```

/***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP (1000) [id]
      ,[nombre]
      ,[apellido]
      ,[sexo]
      ,[pase]
      ,[pago]
      ,[estatus]
      ,[fecha_ingreso]
      ,[dni]
FROM [GimnasioMetroFlex].[dbo].[sociosGym]

```

100 %

Results Messages

	id	nombre	apellido	sexo	pase	pago	estatus	fecha_ingreso	dni
1	1	Ronnie	Coleman	MASCULINO	1	1	0	2022-06-04	24625100
2	2	Jay	Cutler	MASCULINO	0	0	0	2022-06-04	32782642
3	3	Tom	Platz	MASCULINO	2	2	0	2022-06-04	17998120
4	4	Chris	Bumstead	MASCULINO	2	2	0	2022-06-04	35147890
5	5	Lamy	Wheels	MASCULINO	0	0	0	2022-06-04	36529123
6	6	Noelia	Cvitanovic	FEMENINO	2	2	0	2022-06-04	42897456

```

1  using System.Collections.Generic;
2
3  namespace Entidades
4  {
5      1 referencia
6      public interface ICrud
7      {
8          2 referencias
9          List<Socio> ListarSocios();
10         1 referencia
11         Socio BuscarPorID(int id);
12         2 referencias
13         bool EliminarSocio(int id);
14         2 referencias
15         bool EditarSocio(Socio socio);
16         2 referencias
17         bool GuardarSocio(Socio socio);
18     }
19 }

```

Clase 17 - Delegados y Expresiones Lambda.

Utilizados en el Form de Informes para modificar los Labels con la información pertinente al momento de acceder a este Form.

```

/// <summary>
/// Obtiene la Cantidad de Lugares Libres.
/// </summary>
/// <returns>Devuelve la cantidad de lugares libres</returns>
1 referencia
private int ObtenerCantidadLugaresLibres() => this.Capacidad - this.lista.Count;

```

```

/// <summary>
/// Esta Función se Utiliza para Cargar el Formulario y Mostrar la Información en las Etiquetas.
/// </summary>
/// <param name="sender">El objeto que generó el evento.</param>
/// <param name="EventArgs"></param>
1 referencia
private void FrmInformes_Load(object sender, EventArgs e)
{
    this.invocarInformes += this.MostrarSociosPorGenero;
    this.invocarInformes += this.MostrarSociosPorPase;
    this.invocarInformes += this.MostrarSociosPorTipoPago;
    this.invocarInformes += this.MostrarSociosPorEstatus;
    this.invocarInformes += this.MostrarSociosActivosFormaDePago;
    this.invocarInformes += this.MostrarSociosActivosTiposDePase;
    this.invocarInformes += this.MostrarTotalPorTipoPase;
    this.invocarInformes.Invoke();
}

```

```

/// <summary>
/// Esta Metodo se utiliza para Darse de Baja de los Eventos a los que se Suscribio en el
/// Constructor del Formulario.
/// </summary>
/// <param name="sender">La fuente del evento.</param>
/// <param name="FormClosingEventArgs"></param>
1 referencia
private void FrmInformes_FormClosing(object sender, FormClosingEventArgs e)
{
    this.invocarInformes -= this.MostrarSociosPorGenero;
    this.invocarInformes -= this.MostrarSociosPorPase;
    this.invocarInformes -= this.MostrarSociosPorTipoPago;
    this.invocarInformes -= this.MostrarSociosPorEstatus;
    this.invocarInformes -= this.MostrarSociosActivosFormaDePago;
    this.invocarInformes -= this.MostrarSociosActivosTiposDePase;
    this.invocarInformes -= this.MostrarTotalPorTipoPase;
}

/// <summary>
/// Devuelve una Cadena con el Numero de Membresías Activas de Cada Tipo de Membresía.
/// </summary>
2 referencias
public void MostrarSociosActivosTiposDePase()
{
    this.lblSociosActivosPase.Text = informes.SociosActivosTipoDePase();
}

/// <summary>
/// Devuelve el Número de Socios Activos que Pagan en Efectivo.
/// </summary>
2 referencias
public void MostrarSociosActivosFormaDePago()
{
    this.lblActivosEfectivo.Text = informes.SociosActivosFormaDePago();
}

/// <summary>
/// Muestra el Número de Miembros que Tienen un Determinado Tipo de Pase.
/// </summary>
2 referencias
public void MostrarSociosPorPase()
{
    this.lblPase.Text = informes.SociosPorPase();
}

```

Clase 18 - Hilos.

Se utilizaron en el Form Principal para que, al Momento de Cerrar la Aplicación, Informe Cuantos Socios Nuevos Entraron en su Jornada Laboral.

```

#region Metodos
/// <summary>
/// Carga el Listado y Actualiza el Listbox.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 referencia
private void frmGym_Load(object sender, EventArgs e)
{
    socioDAO = new SocioDAO();
    gimnasio = new Gimnasio(this.EstablecerCapacidad(Capacidad));
    this.ActualizarListBox();
    this.ActualizarDatos();
    this.lblCapacidadSocios.Text += gimnasio.Capacidad.ToString();
    cancellationTokenSource = new CancellationTokenSource();
    cancellationToken = cancellationTokenSource.Token;
    inicio = DateTime.Now;
    this.task = Task.Run(() => SociosNuevosInicioYCierreAplicacion(), cancellationToken); ;
}

```



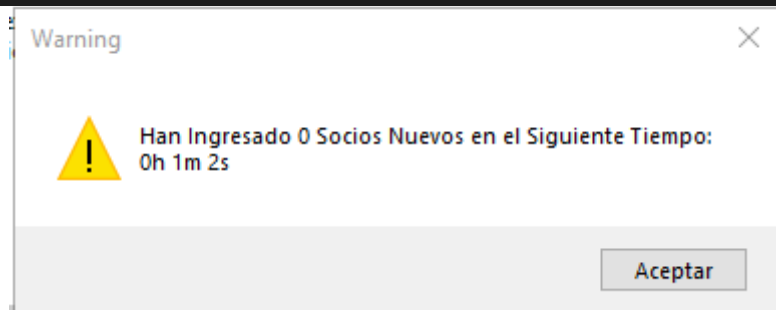
```

/// <summary>
/// Es una Funcion que se Ejecuta en un Hilo, y que Muestra un mensaje con la Cantidad de Nuevos Miembros que se han Agregado
/// a la Base de Datos en el Tiempo que la Aplicacion se ha estado Ejecutando.
/// </summary>
/// <returns>
/// El Metodo devuelve la diferencia Horaria entre la Hora de Inicio y la Hora de Finalizacion.
/// </returns>
1 referencia
private void SociosNuevosInicioYCierreAplicacion()
{
    DateTime dateTime = new DateTime();

    while (true)
    {
        if (this.cancellationToken.IsCancellationRequested)
        {
            fin = DateTime.Now;
            MessageBox.Show("Han Ingresado " + this.sociosNuevos + " Socios Nuevos en el Siguiete Tiempo: \n"
                + dateTime.ObtenerTiempo(inicio, fin).ToString("h'h 'm'm 's's'"),
                "Warning", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

            return;
        }
    }
}

```



Clase 19 - Eventos.

Utilizados en el Form de Informes para modificar los Labels con la información pertinente al momento de acceder a este Form

```

namespace FormGimnasio
{
    7 referencias
    public partial class FrmInformes : Form
    {
        #region Atributos
        private Gimnasio gimnasio;
        private Informes informes;
        public delegate void ManejarInformes();
        public event ManejarInformes invocarInformes;
        #endregion

        #region Constructores
        1 referencia
        public FrmInformes()
        {
            InitializeComponent();
        }
    }
}

```

```
/// <summary>
/// Esta Función se Utiliza para Cargar el Formulario y Mostrar la Información en las Etiquetas.
/// </summary>
/// <param name="sender">El objeto que generó el evento.</param>
/// <param name="EventArgs"></param>
```

1 referencia

```
private void FrmInformes_Load(object sender, EventArgs e)
{
    this.invocarInformes += this.MostrarSociosPorGenero;
    this.invocarInformes += this.MostrarSociosPorPase;
    this.invocarInformes += this.MostrarSociosPorTipoPago;
    this.invocarInformes += this.MostrarSociosPorEstatus;
    this.invocarInformes += this.MostrarSociosActivosFormaDePago;
    this.invocarInformes += this.MostrarSociosActivosTiposDePase;
    this.invocarInformes += this.MostrarTotalPorTipoPase;
    this.invocarInformes.Invoke();
}
```

```
/// <summary>
/// Devuelve una Cadena con el Numero de Membresías Activas de Cada Tipo de Membresía.
/// </summary>
```

2 referencias

```
public void MostrarSociosActivosTiposDePase()
{
    this.lblSociosActivosPase.Text = informes.SociosActivosTipoDePase();
}
```

```
/// <summary>
/// Devuelve el Número de Socios Activos que Pagan en Efectivo.
/// </summary>
```

2 referencias

```
public void MostrarSociosActivosFormaDePago()
{
    this.lblActivosEfectivo.Text = informes.SociosActivosFormaDePago();
}
```

```
/// <summary>
/// Muestra el Número de Miembros que Tienen un Determinado Tipo de Pase.
/// </summary>
```

2 referencias

```
public void MostrarSociosPorPase()
{
    this.lblPase.Text = informes.SociosPorPase();
}
```