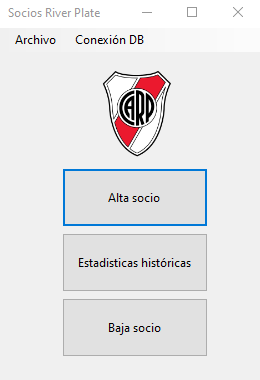
# TP4 Laboratorio 2

Alumno: Mato Ezequiel Esteban

División: 2° D

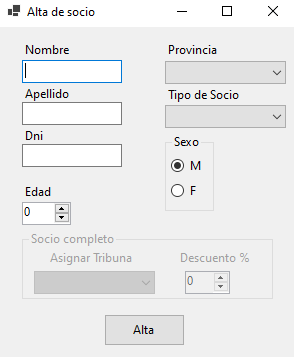
# **Visualizador de Socios**



El proyecto simula a atención de un club en donde se podrán dar de altas socios, tanto simple o completos, los cuales obtienes unos beneficios extras.

Se podrá dar de alta un socio nuevo, bajas lógicas, asignar tribunas y descuentos, guardar la información y poder visualizar los datos actuales del club.

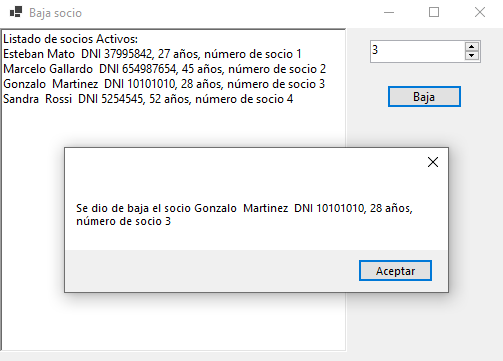
## Alta socio



Desde el botón **Alta socio**,podremos acceder al menú de alta e ingresar un nuevo socio a la nómina de socios.

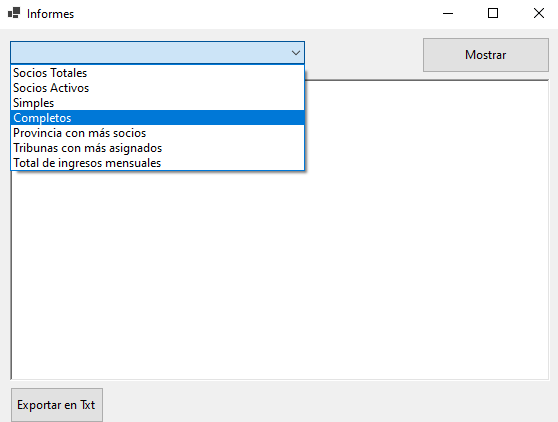
Se ingresarán los datos requeridos, previa validación, y se deberá seleccionar si es un socio *Simple* o *Completo*, el cual tiene como agregado la opción de asignarle una tribuna y una cantidad de descuento al socio.

## Baja Socio



Ingresando a **Baja Socio**, accederemos al menú de baja donde podremos visualizar los socios activos actualmente y seleccionar, por *número de socio*, el que se desee dar de baja.

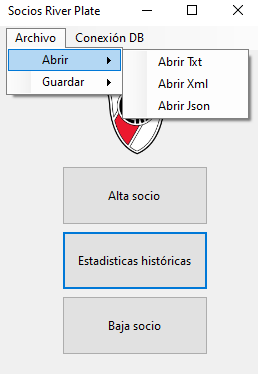
## Estadísticas Históricas



Desde la opción **Estadísticas históricas** podremos imprimir diferentes informes para visualizar por pantalla, se deberá seleccionar y clickear en **Mostrar** para ver la información.

El botón situado en la esquina inferior izquierda **Exportar en Txt**, generara un archivo .txt en el escritorio con la información seleccionada actualmente.

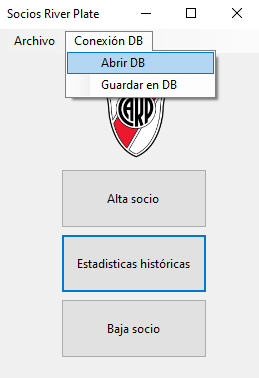
## Menú archivo



Podremos exportar e importar documentación en el formato deseado.

En el formato .txt, se guardará la información y al abrirla solamente se visualizará en un MessageBox lo contenido en el archivo .txt

## Menú Conexión DB

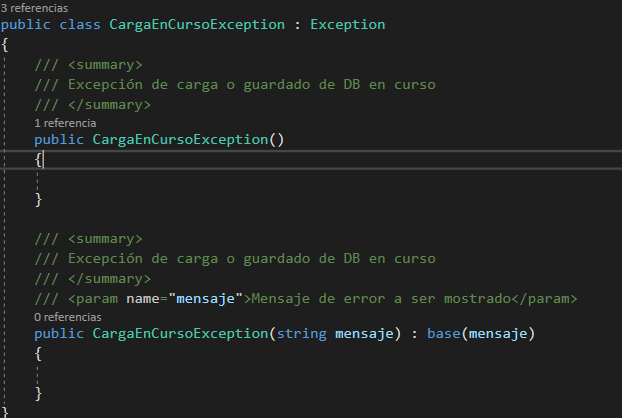


Por último, desde este menú se podrá realizar la conexión a la base de datos, en donde podremos volcar la información y leerla nuevamente para la persistencia de datos.

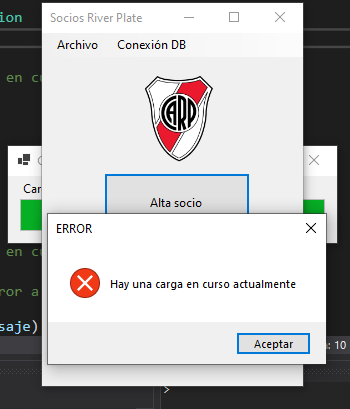
Cada vez que guardemos en la DB, se estarán agregando nuevamente todos los socios listados.

# Temas requeridos para el trabajo práctico

## **Excepciones**

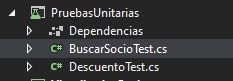


Se controlaron las diferentes excepciones que pudieran interrumpir la ejecución de la aplicación. Por ejemplo, una de ellas es cuando ya hay una carga en curso de la información de desde la DB, y se intenta nuevamente guardar o cargar la información. Donde ocurrirá una excepción y se controla informando el siguiente error.



## **Test Unitarios**

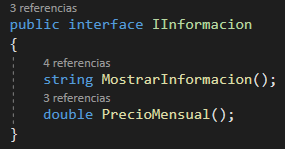
Se realizaron 2 Test Unitario en donde se probó el correcto funcionamiento del método **BuscarSocio** y la extensión **DesconstarPorcentaje**



## **Interfaces**

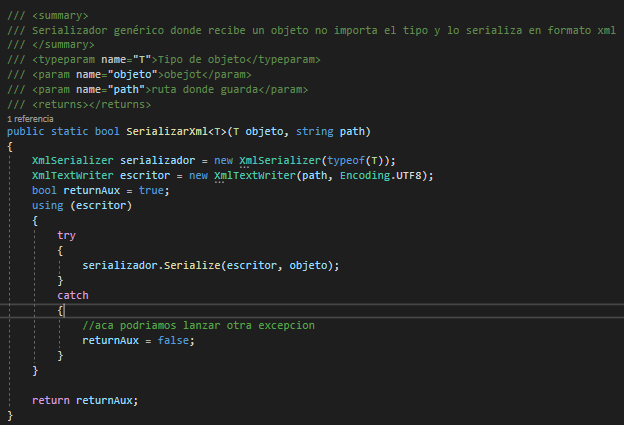
Se utilizaron interfaces para mostrar información requerida en los socios, por ejemplo.

La implementan tanto socios **Simple** como **Completo**



## **Tipos Genéricos y Archivos**

Por practicidad se utilizaron tipo genéricos y archivos en conjunto para poder Serializar y Deserealizar la información, y así aprovechar las funcionalidades, por ejemplo



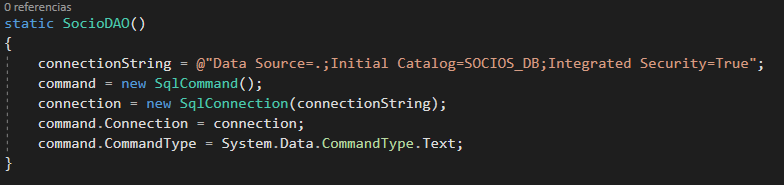
También para poder escribir en .txt los informes requeridos, esto se puede encontrar en la clase **Archivadora.cs**

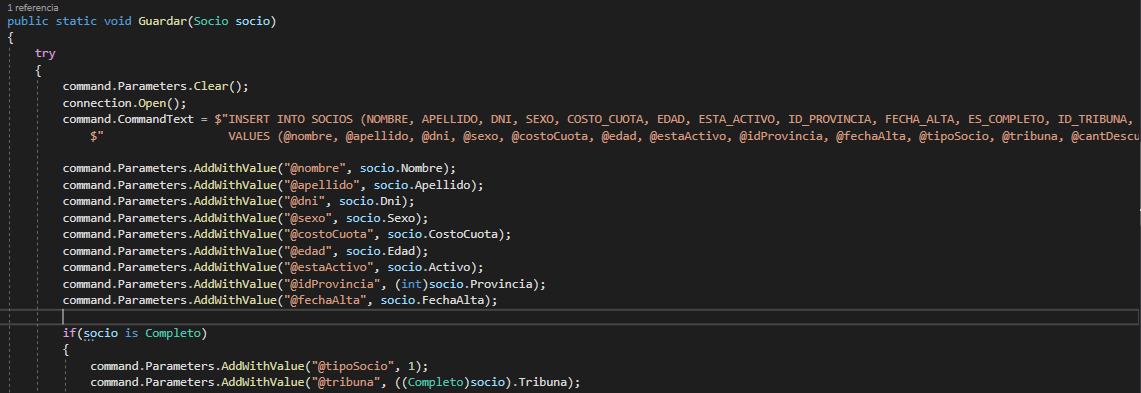


## **Conexión a Base de Datos**

En la clase **SocioDAO.cs** se encontrará la implementación del manejo de base de datos.

Tenemos un método **Leer** y **Guardar** los cuales nos permitirán y manejando la información.

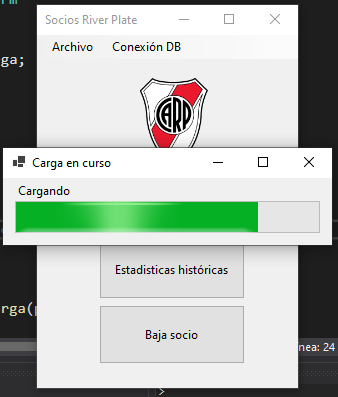




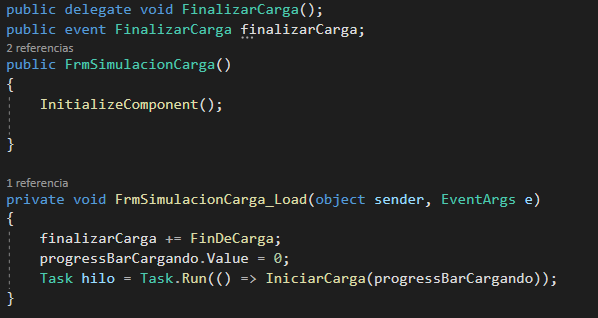


## **Delegados, Expresiones Lambda, Hilos y Eventos**

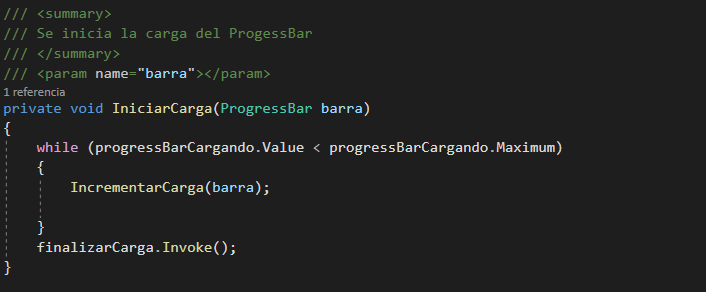
Por tema de practicidad y a modo solo visual para a aplicación, se incorporó un Progress Bar el cual simulara un proceso de carga al momento de realizar interacciones con la Base de Datos, tanto al cargar como al guardar.



En un hilo secundario se realizó la ejecución de un formulario que va a ir incrementando cada segundo, un 20% de la barra de progreso, y mientras poder seguir utilizando la aplicación.

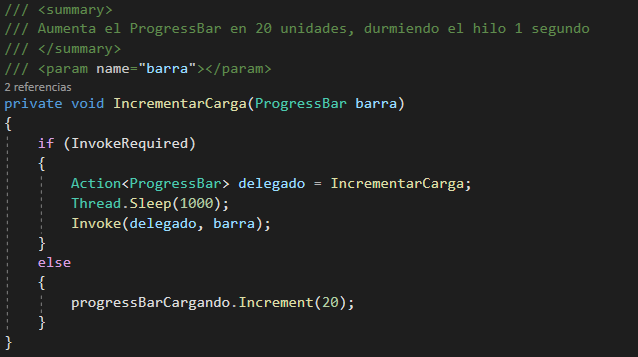


Con un delegado referenciamos el método FinDeCarga, el cual se disparará cuando se termine de realizar la carga, informando el fin y pasando a cerrar el formulario.



Iniciar carga se correrá en otro hilo aparte para no bloquear completamente el formulario de carga y así poder visualizarlo correctamente.

Podemos observamos el **Invoke** que se realiza al momento de finalizar la carga.

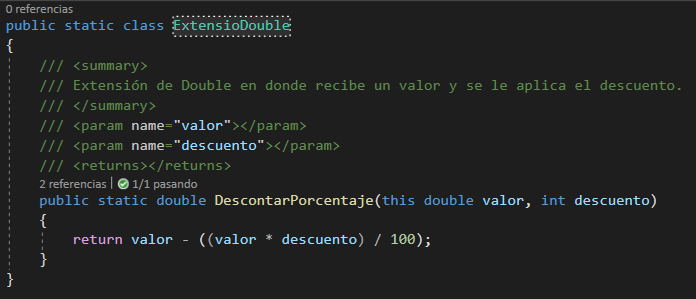


Como el proceso de carga se realiza en otro hilo, nos vemos obligados a Invocar, a través de un delegado, la modificación que se quiera realizar en el formulario principal, ya que de manera contraria no sería posible.

Se realiza a través de un delegado y pasando los argumentos en array, en este caso al ser uno solo no sería necesario.

Se utilizó expresiones lambda por practicidad y comodidad en esta sección del código.

## **Métodos de extensión**



Utilizado para extender Double y así poder aplicar descuento en porcentajes.

