

Laboratorio 2

Trabajo Practico 4

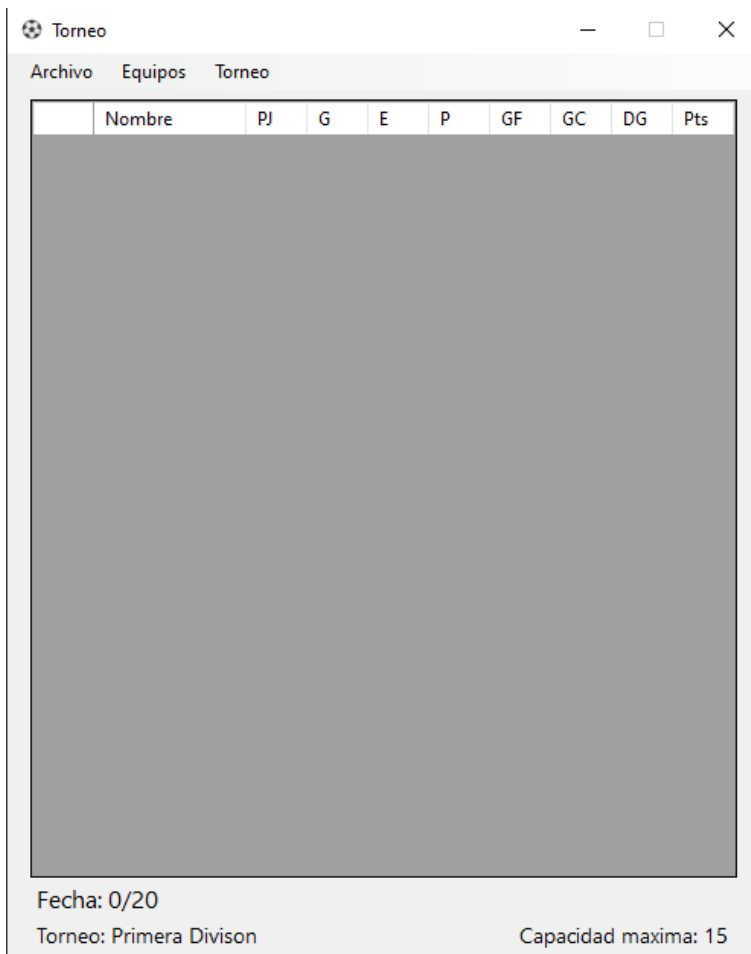
Alumno: Octavio Bill Zito

División: 2°E

Proyecto: Torneo de Futbol

Breve resumen:

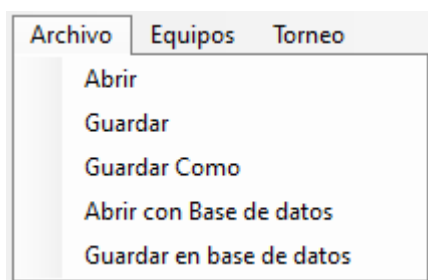
Al iniciar la aplicación nos encontramos con el siguiente form



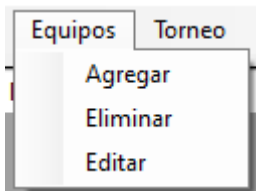
Nombre	PJ	G	E	P	GF	GC	DG	Pts
--------	----	---	---	---	----	----	----	-----

Fecha: 0/20
Torneo: Primera Divison
Capacidad maxima: 15

Haciendo click en la parte superior izquierda donde dice archivos podemos abrir, guardar, guardar como, abrir por la base de datos y guardar base de datos.



Al hacer click abrir nos abre una pestaña de Windows para abrir un archivo, que tiene que ser de formato txt, xml o json. Al hacer click en guardar, guardamos el archivo abierto, en caso que no haya un archivo abierto nos pedirá guardar en una ubicación, y guardar como nos abre una pestaña de Windows para guardar en la ubicación que deseamos. Abrir con Base de datos, trae la información de la base de datos y guardar, borra todo lo de la tabla y guarda lo que hay momentáneamente.

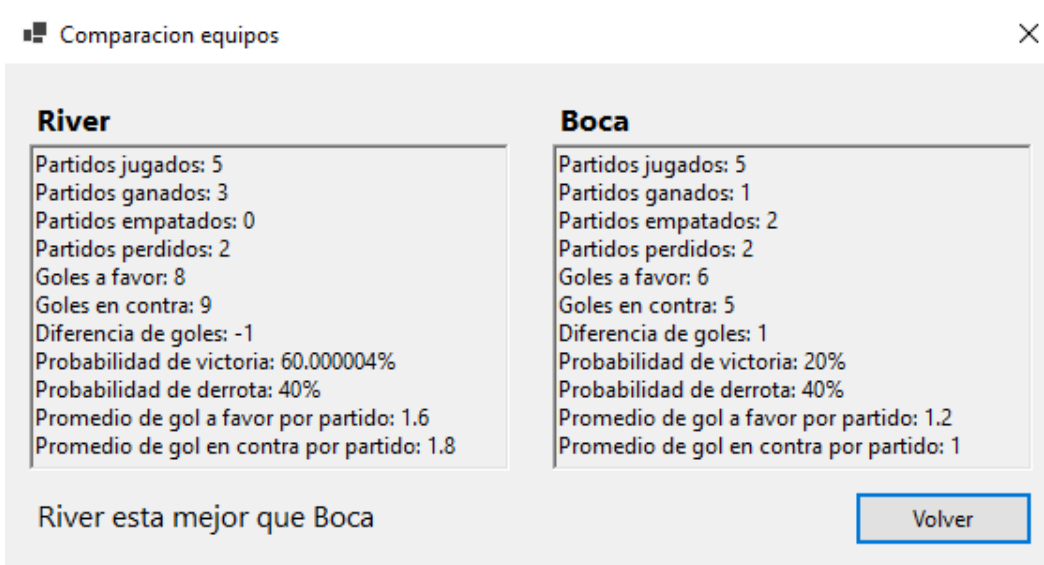


En la pestaña Equipos si hacemos click en agregar nos abre un nuevo form para agregar un equipo, si seleccionamos un equipo y hacemos click en eliminar lo borra y para editar también se necesita selecciona un equipo y abre un form para editar.



En la pestaña Torneo, si hacemos click en avanzar fechas y se encuentra más de 1 equipo y no llegamos a la fecha máxima simula una fecha a los equipos de la tabla.

Al seleccionar dos equipos con el botón Control y tocar en Comparar nos abre un form nuevo con los datos de cada equipo y sus estadísticas.

A screenshot of a software interface showing a form titled 'Comparacion equipos' (Team Comparison). The form displays statistics for two teams: River and Boca. The statistics include: Partidos jugados (Games played), Partidos ganados (Games won), Partidos empatados (Games drawn), Partidos perdidos (Games lost), Goles a favor (Goals for), Goles en contra (Goals against), Diferencia de goles (Goal difference), Probabilidad de victoria (Probability of victory), Probabilidad de derrota (Probability of defeat), Promedio de gol a favor por partido (Average goals for per game), and Promedio de gol en contra por partido (Average goals against per game). The form concludes with the statement 'River esta mejor que Boca' (River is better than Boca) and a 'Volver' (Return) button.

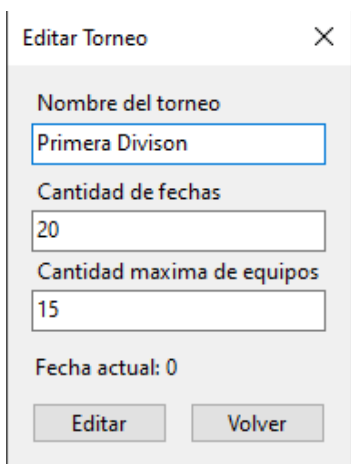
River	Boca
Partidos jugados: 5	Partidos jugados: 5
Partidos ganados: 3	Partidos ganados: 1
Partidos empatados: 0	Partidos empatados: 2
Partidos perdidos: 2	Partidos perdidos: 2
Goles a favor: 8	Goles a favor: 6
Goles en contra: 9	Goles en contra: 5
Diferencia de goles: -1	Diferencia de goles: 1
Probabilidad de victoria: 60.000004%	Probabilidad de victoria: 20%
Probabilidad de derrota: 40%	Probabilidad de derrota: 40%
Promedio de gol a favor por partido: 1.6	Promedio de gol a favor por partido: 1.2
Promedio de gol en contra por partido: 1.8	Promedio de gol en contra por partido: 1

River esta mejor que Boca

Volver

Resetear Torneo coloca todas las estadísticas del equipo en 0 y vuelve a la fecha 0

El botón info torneo abre un form para cambiar el nombre del torneo, la fecha máxima y la cantidad de equipos



Editar Torneo

Nombre del torneo
Primera Divison

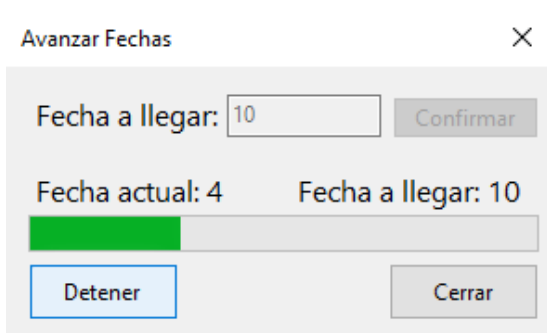
Cantidad de fechas
20

Cantidad maxima de equipos
15

Fecha actual: 0

Editar Volver

Y el botón avanzar hasta nos permite simular varias fechas, siempre y cuando no estemos en la fecha final o no haya más de 1 equipo



Avanzar Fechas

Fecha a llegar: 10 Confirmar

Fecha actual: 4 Fecha a llegar: 10

Detener Cerrar

Temas utilizados Parte 1:

Excepciones:

ExceptionXml
Class
→ Exception

Fields

clase : string

metodo : string

Properties

Clase { get; } : string

Metodo { get; } : string

Methods

ExceptionXml(string mensaje, string metodo, string clase)

ExceptionXml(string mensaje, string metodo, string clase, Exception innerException)

ExceptionTxt
Class
→ Exception

Fields

clase : string

metodo : string

Properties

Clase { get; } : string

Metodo { get; } : string

Methods

ExceptionTxt(string mensaje, string metodo, string clase)

ExceptionTxt(string mensaje, string metodo, string clase, Exception innerException)

ExceptionJson
Class
→ Exception

Fields

clase : string

metodo : string

Properties

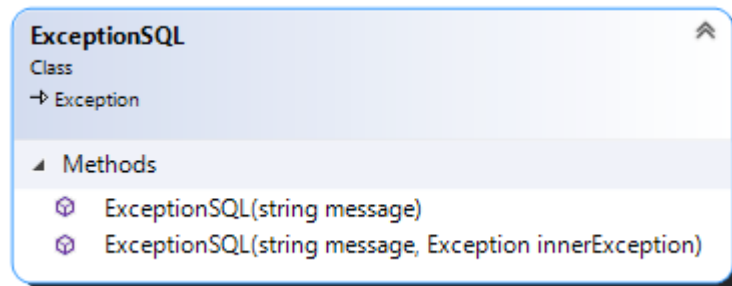
Clase { get; } : string

Metodo { get; } : string

Methods

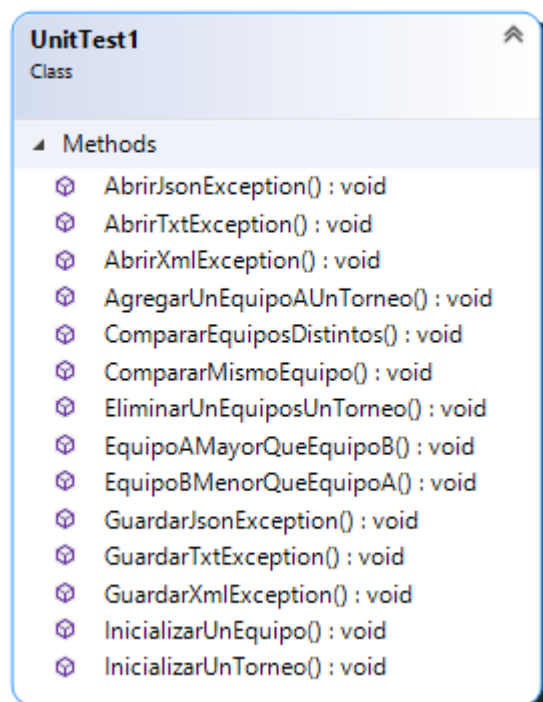
ExceptionJson(string mensaje, string metodo, string clase)

ExceptionJson(string mensaje, string metodo, string clase, Exception innerException)



Las exception de Json, Xml y Txt son utilizadas en la clase GestorDeArchivos, para capturar la excepcion lanzadas por el StreamReader, StreamWriter, XmlTextReader, XmlSerializer y JsonSerializer y relanzarlas con las excepciones propias con un mensaje para capturarla luego en los formularios.

Pruebas unitarias:

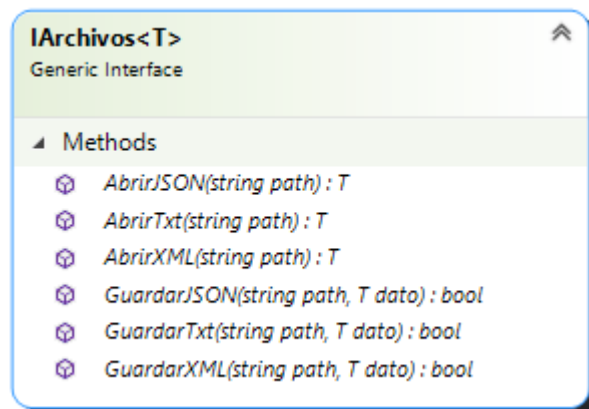


La clase UnitTest es de un proyecto de UnitTest en el cual posee varios métodos para comprobar el funcionamiento de las clases.

Tipos genéricos:

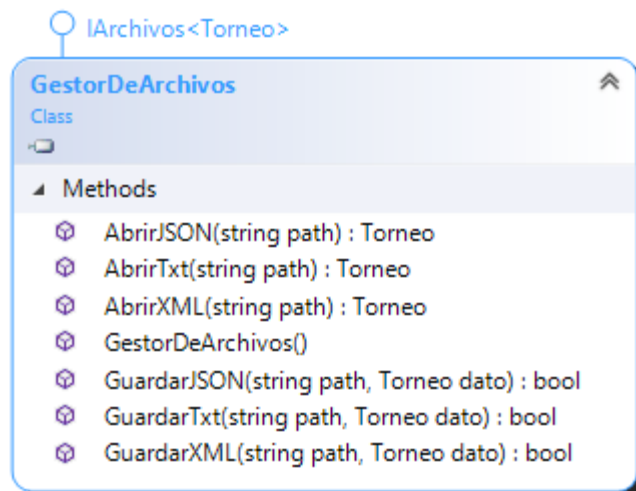
Usada en la interface IArchivos y aplicando su implementación en la clase GestorDeArchivos.

Interfaces:



La interface `IArchivos` posee un genérico `T` donde es obligatorio que sea de clase `Torneo`, posee 6 métodos para ser implementados en el `GestorDeArchivos`.

Archivos:

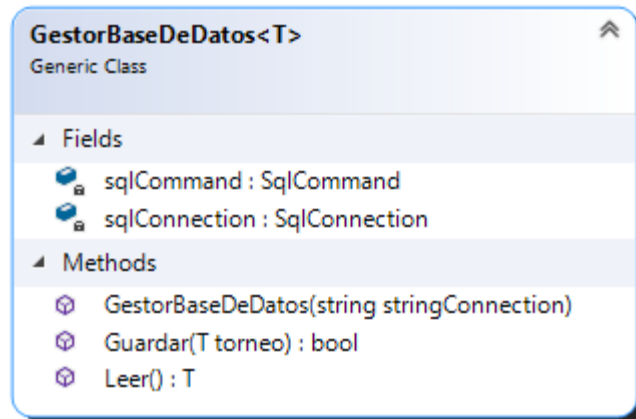


La clase `GestorDeArchivos` implementa la interfaz, esta clase abre y guarda archivos en formato `Json`, `Xml` y `Txt`, en caso de capturar alguna excepción esta es relanzada con las excepciones creadas.

Temas utilizados parte 2:

Base de datos:

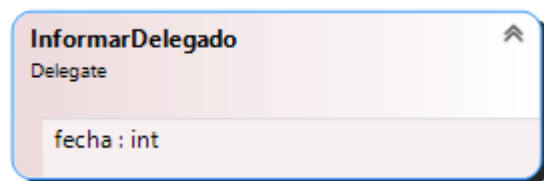
Cree una base de datos con el nombre equipos y una tabla llamada tablaEquipos, dejen el script en la carpeta principal del Trabajo Practico 4. La connection string se encuentra en el constructor del FormPrincipal al momento de instanciar la clase GestorBaseDeDatos<Torneo> con el genérico Torneo.



```
this.gdb = new GestorBaseDeDatos<Torneo>("Data Source=.;Initial Catalog=equipos;Integrated Security=true");
```

Línea 31 de la clase FormPrincipal.

Delegados:



Creo un delegado que retorna void y recibe un int con el nombre fecha. Es utilizado en la clase frmAvanzarFechas en el método Informar. Para llamar a la función informar.


```

public void informar(int fecha)
{
    if (this.InvokeRequired)
    {
        InformarDelegado del = new InformarDelegado(this.informar);
        object[] args = new object[] { fecha };
        this.Invoke(del, args);
    }
    else
    {
        this.lblFechaActual.Text = "Fecha actual: " + this.torneo.FechaActual.ToString();
        this.progressBar1.Value = fecha;
        this.frmPrincipal.ActualizarForm();
    }
}

```

También el evento, eventoInformar de Torneo es de tipo InformarDelegado.

Hilos:

En el frmAvanzarFechas si se cumplen ciertas condiciones crea un hilo que ejecute el método AvanzarHasta de la clase torneo pasándole por parámetro la fecha hasta que deseo llegar y un token para poder cancelarlo, este token es creado en el constructor de frmAvanzarFechas.

```

1 reference
private void AvanzarFechas()
{
    if (this.fechaLlegar > this.torneo.FechaActual && this.fechaLlegar <= this.torneo.CantidadDeFechas)
    {
        Task task = new Task(() => this.torneo.AvanzarHasta(this.fechaLlegar, tokenSource.Token));
        task.Start();
    }
}

```

Dentro del método AvanzarHasta se llama al método AvanzarTorneo y si retorna true pasamos a llamar al evento “eventoInformar” pasándole la fecha actual, luego se realiza un Thread.Sleep de 500 milisegundos y si la FechaActual no es igual a la fechaAvanzar y el token no fue cancelado sigue el proceso.

```

1 reference
public bool AvanzarHasta(int fechaAvanzar, CancellationToken token)
{
    do
    {
        if(this.AvanzarTorneo())
        {
            if (eventoInformar != null)
            {
                this.eventoInformar.Invoke(this.FechaActual);
            }
        }else
        {
            return false;
        }
        Thread.Sleep(500);
    } while (fechaAvanzar != this.FechaActual && !token.IsCancellationRequested);

    return true;
}

```

Eventos:

En la clase Torneo creo un evento del tipo InformarDelegado como se mostró previamente

```

public event InformarDelegado eventoInformar;

```

En el cual es llamado dentro del método AvanzarHasta de la misma clase, su implementación se encuentra en el frmAvanzarFechas.

```

this.torneo.eventoInformar += this.informar;

```

Suscribiéndole el método informar del formulario

```

public void informar(int fecha)
{
    if (this.InvokeRequired)
    {
        InformarDelegado del = new InformarDelegado(this.informar);
        object[] args = new object[] { fecha };
        this.Invoke(del, args);
    }
    else
    {
        this.lblFechaActual.Text = "Fecha actual: " + this.torneo.FechaActual.ToString();
        this.progressBar1.Value = fecha;
        this.frmPrincipal.ActualizarForm();
    }
}

```

Este primero necesita ser invocado al hilo principal, luego de eso modifica el valor de la barra de progreso, actualiza el datagrid y cambia el valor del label.

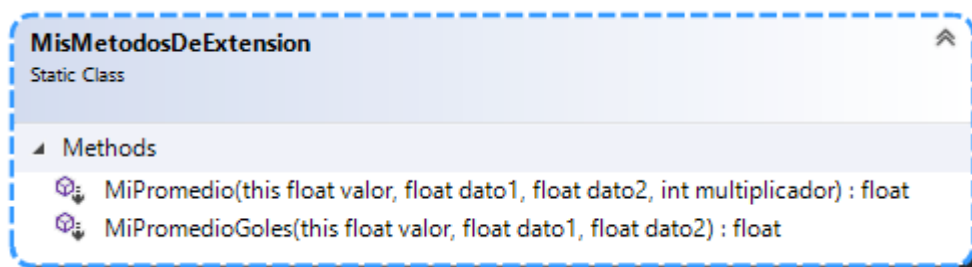
En el FormClosing de frmAvanzarFechas, si el hilo se encuentra ejecutándose se detiene y se le quita el manejador del evento

```
private void frmAvanzarFechas_FormClosing(object sender, FormClosingEventArgs e)
{
    this.tokenSource.Cancel();
    this.torneo.eventoInformar -= this.informar;
}
```

También al momento de hacer click en el botón detener de este mismo form se llama al método Cancel del tokenSource.

Métodos de extensión:

En la clase MisMetodosDeExtension que es static poseemos dos métodos para sacar promedios estos métodos son utilizados en FormComparar para sacar los promedios



```
probGanados = probGanados.MiPromedio((float)this.e1.Ganados, (float)this.e1.PartidosJugados, 100);
```

```
promGolesFavor = promGolesFavor.MiPromedioGoles((float)this.e1.GolesAFavor, (float)this.e1.PartidosJugados);
```