



La Liga Torneo

PRO CS:GO

Liga online de Counter-Strike Global Offensive

Luis Andaur - 2E

Trabajo práctico N°4 (Entrega final)

----- Temas vistos entre la clase 10 y 20 -----

Laboratorio II



Descripción del proyecto

Breve resumen

El proyecto simula un torneo online de Counter-Strike. La metodología del torneo es una competencia individual , todos contra todos, en 9 (nueve) mapas con modalidades diferentes, dónde cada jugador tiene la posibilidad de sumar puntos de la siguiente manera:

- Kills (matar a otros jugadores) x1 punto por kill
- Bombas (bombas plantadas) x3 puntos por bomba
- Rehenes (rescate de rehenes) x3 puntos por rehen
- Headshot (disparo a la cabeza) x2 puntos por headshot

Pero se les resta por cada vez que mueren o los matan

- Muertes -x2 puntos por muerte.

El proyecto ya toma datos de jugadores registrados desde un archivo json, pero tambien cuenta con la posibilidad de agregar, editar y eliminar el jugador que desee.

Se simula el torneo y luego puede descargar en el formato que desee los reportes que se obtuvieron del mismo.

Sonido:

Se creó un boton para poder silenciar la aplicación.

Con sonido.



Sin sonido.



Home

Menu principal del sistema



Administrar Jugadores

Se visualiza a todos los jugadores, y también se agrega, edita o elimina a los mismos.

Simular Torneo

Simula el torneo completando las estadísticas de los jugadores con números random y sumando los puntos obtenidos.

Reportes

Reportes del torneo que se pueden guardar.

Salir

Sale y cierra la aplicación.

Administrar jugadores

AMB de la aplicación

Administrador de jugadores						
Nro. Jugador	Nombre	Edad	Genero	Nacionalidad	Especialidad	Primer Torneo
4567	Shir	36	Female	China	M	<input checked="" type="checkbox"/>
6354	Nils	38	Female	Brazil	S	<input checked="" type="checkbox"/>
594	Timmie	28	Female	Japan	XL	<input type="checkbox"/>
1937	Kare	20	Male	Czech Republic	L	<input checked="" type="checkbox"/>
726	Antone	47	Female	Brazil	XS	<input type="checkbox"/>
2456	Kevina	12	Male	Japan	XXL	<input checked="" type="checkbox"/>
4322	Eustacia	52	Male	Peru	XL	<input type="checkbox"/>
3345	Hillary	43	Male	Colombia	M	<input type="checkbox"/>
1111	Gordan	14	Female	China	S	<input checked="" type="checkbox"/>
2154	Kamilah	21	Female	France	XS	<input checked="" type="checkbox"/>
5847	Kristo	15	Female	Brazil	XXL	<input checked="" type="checkbox"/>
3847	Sancho	29	Female	China	L	<input type="checkbox"/>
3722	Nerissa	54	Male	Russia	L	<input checked="" type="checkbox"/>
736	Hetty	25	Female	Czech Republic	XXL	<input checked="" type="checkbox"/>
343	Ebba	17	Male	Panama	XXL	<input type="checkbox"/>
4234	Nonie	58	Female	Poland	XL	<input checked="" type="checkbox"/>
5522	Kessia	28	Male	Russia	XS	<input type="checkbox"/>
2091	Julietta	37	Male	United States	S	<input checked="" type="checkbox"/>
1112	Kinny	53	Male	China	S	<input type="checkbox"/>

Lista de todos los jugadores con sus atributos como nombre, edad, genero y nacionalidad y otros detallados a continuación:

Nro. Jugador: Nro único por jugador.

Especialidad: Especifica la modalidad donde el jugador es más hábil

"XS" = Cuerpo a cuerpo.

"S" = Corta distancia.

"M" = Media distancia.

"L" = Larga distancia.

"XL" = ExtraL.

"XXL" = Francotirador.

Primer Torneo: Se marca cuando es el primer torneo que juega.

Agregar

Abre el menu para completar los datos con un nuevo jugador

Editar

Selecciona un jugador y abre un menu con sus datos para que pueda modificarlos, menos el Nro de jugador.

Eliminar

Para eliminar sólo debe seleccionar un jugador y al presionar eliminar este será eliminado

Volver

Vuelve al menu principal/Home.



Simular Torneo

Simulador de todo el torneo

Simular Torneo



Simula la partida haciendo que los jugadores jueguen en los 9 mapas, asignandole de manera random las estadísticas para kills, muertes, bombas, rehenes, headshots y calculando la cantidad de puntos y sumandolos a cada jugador.

Reportes

Análisis de datos del torneo

Reportes

Reportes del torneo

Analisis de datos

*Participación total del torneo ----- 150		
*Categoría-----Female-----Male-----		
Genero:	74 49.33%	76 50.67%
Si:	39 52.70%	41 53.95%
No:	35 47.30%	35 46.05%
*Rango etario -----		
de 12 a 29:	22 29.73%	22 28.95%
de 30 a 49:	31 41.89%	31 40.79%
+ 50:	21 28.36%	23 30.26%
*Nacionalidad -----		
Latinos	7 9.46%	7 9.21%
No latinos	67 90.54%	69 90.79%
*Promedios -----		
Kills:	9327 62.180	9887 65.380
Muertes:	9328 62.187	10199 67.993
Headshots:	1947 12.980	2145 14.300
Bombas:	322 02.147	315 02.100
Rehenes:	234 01.560	248 01.653

Volver

Exportar análisis de datos

TorneoPRO : CS GO

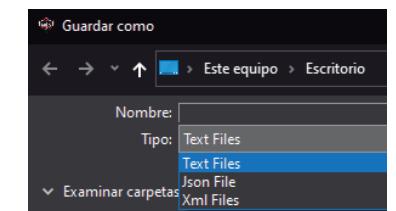
EASport

U\$D1000000

Filtros

Campeón	Posiciones	Más Killers
Más mancos	Primer Torneo	Bombardeos
Rescatistas	XXL Headshots	El menor

Exportar



Reportes resultantes del torneo que se pueden guardar en la extensión que desee el usuario y tambien puede elegir la ubicación donde desee guardar sus reportes generados.



Temas utilizados

----- *entre clase 10 y 15* -----

Excepciones
Pruebas Unitarias
Tipos Genéricos
Interfaces
Archivos
Serialización

Excepciones

Ejemplo de implementación

Implementado en varias partes del sistema para controlar posibles incidentes y evitar que el programa deje de funcionar, por ejemplo una de las implementadas es:

```
8 referencias
public class Exception_NroJugadorYaExiste : Exception
{
    /// <summary>
    /// Crea una excepcion con un mensaje
    /// </summary>
    /// <param name="message">Mensaje de la excepcion</param>
    1 referencia
    public Exception_NroJugadorYaExiste(string message) : this(message, null)
    {

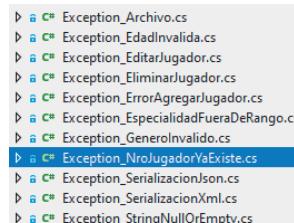
    }

    /// <summary>
    /// Crea una excepcion con un mensaje y un innerException
    /// </summary>
    /// <param name="message">Mensaje de la excepcion</param>
    /// <param name="innerException">innerException de la excepcion</param>
    1 referencia
    public Exception_NroJugadorYaExiste(string message, Exception innerException) : base(message, innerException)
    {
    }
}
```

Dicha excepcion se lanzará cuando el nro. de jugador ingresado coincida con uno existente. Por ejemplo se implementa en el metodo AgregarJugador()

```
1 referencia
public static void AgregarJugador(string nroJugador, string nombre, string edad, string ge
{
    if (!string.IsNullOrEmpty(nroJugador) && !string.IsNullOrEmpty(nombre) && !string.IsNullOrEmpty(edad))
    {
        int.TryParse(nroJugador, out int auxNroJugador);
        if (ExisteJugador(auxNroJugador))
        {
            throw new Exception_NroJugadorYaExiste("Error el nro de jugador ya existe");
        }
        int.TryParse(edad, out int auxEdad);

        listaJugadores.Add(new Jugador(auxNroJugador, nombre, auxEdad, genero, nacionalida
    }
}
```

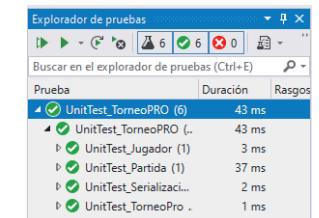
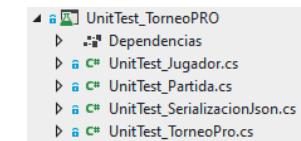


Pruebas unitarias

Ejemplo de implementación

En el proyecto 'UnitTest_TorneoPRO' hay test unitarios, donde se testean métodos que hagan lo correspondiente.

Se puede observar que el 100% de los test pasan correctamente las pruebas.



Un ejemplo de implementación es en la deserialización de un archivo json, dónde si no encuentra el archivo enviado por parámetro, lanza una excepcion del tipo Exception_SerializacionJson.

```
[TestClass]
0 referencias
public class UnitTest_SerializacionJson
{
    [TestMethod]
    [ExpectedException(typeof(Exception_SerializacionJson))]
    0 referencias
    public void TestLeer_RecibeNombreArchivo_ReturnaExceptionSerializacionJson()
    {
        string archivo = "noExiste.json";

        SerializacionJson<List<Jugador>>.Leer(archivo);
    }
}
```

Tipos genéricos

Ejemplo de implementación

Implementados en archivos y serialización por un tema de practicidad.



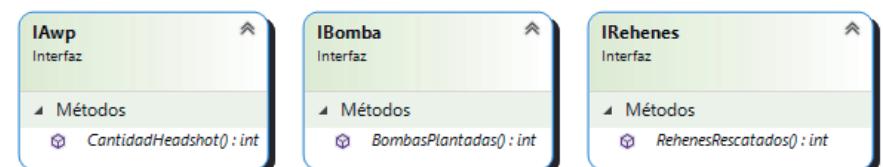
Se puede apreciar su uso en el método GuardarComo():

```
2 referencias
private void GuardarComo(Jugador tipo)
{
    this.saveFile.ShowDialog();
    if (this.saveFile.FileName != String.Empty)
    {
        if (Path.GetExtension(this.saveFile.FileName) == ".txt")
        {
            try
            {
                Archivo<Jugador>.Escribir(tipo, this.saveFile.FileName);
            }
        }
        -----
        try
        {
            SerializacionJson<Jugador>.Escribir(tipo, this.saveFile.FileName);
        }
        -----
        try
        {
            SerializacionXml<Jugador>.Escribir(tipo, this.saveFile.FileName);
        }
    }
}
```

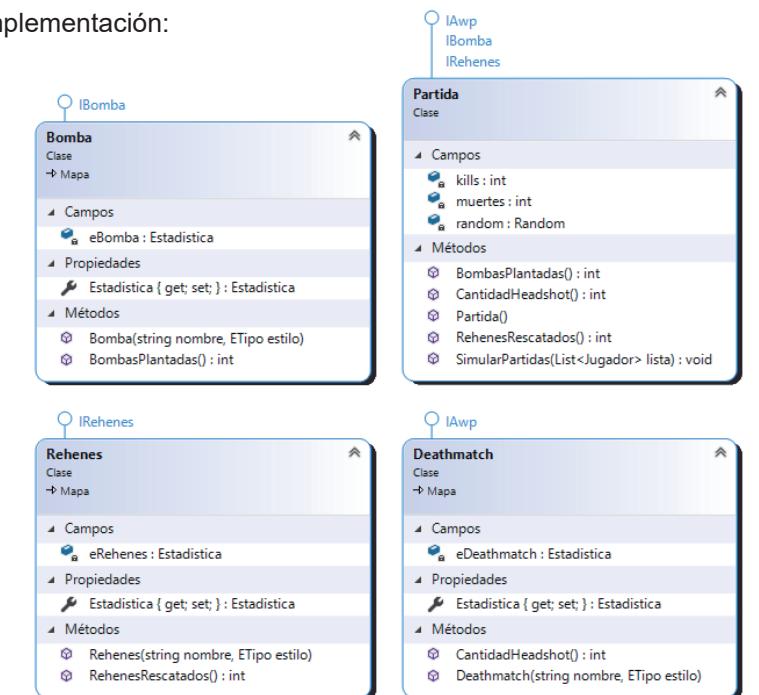
Interfaces

Ejemplo de implementación

Se crearon 3 interfaces con métodos que permiten implementarlos en clases heredadas, por ejemplo Bomba, Deathshot y Rehenes que heredan de la clase Mapa, y otra clase que nada tiene que ver con estas como lo es Partida, pudiendo darles funcionalidades particulares dependiendo la necesidad.



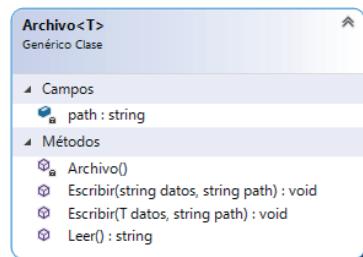
Implementación:



Archivos

Ejemplo de implementación

Se utilizan para escribir en txt los reportes generados en el torneo.



Se puede apreciar su uso en el método GuardarComo():

```

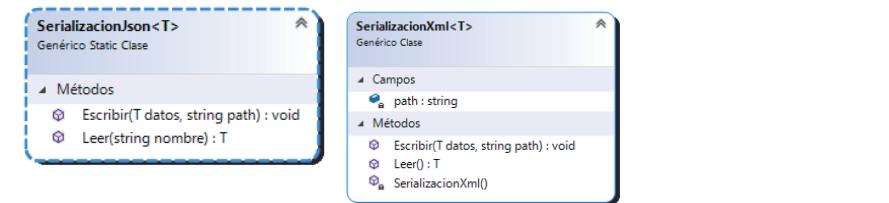
2 referencias
private void GuardarComo(Jugador tipo)
{
    this.saveFile.ShowDialog();
    if (this.saveFile.FileName != String.Empty)
    {
        if (Path.GetExtension(this.saveFile.FileName) == ".txt")
        {
            try
            {
                Archivo<Jugador>.Escribir(tipo, this.saveFile.FileName);
            }
        }
    }
}

```

Serialización

Ejemplo de implementación

Se utilizan para serializar y deserializar archivos y listas de los reportes generados en el torneo.



SerializacionXml se puede apreciar su uso en el método GuardarComo():

```

try
{
    SerializacionXml<Jugador>.Escribir(tipo, this.saveFile.FileName);
}

```

SerializacionJson se puede apreciar su uso deserializando la lista inicial con los jugadores del torneo en el método static CargarJugadores() de la clase static TorneoPro.

```

private static void CargarJugadores()
{
    listaJugadores = SerializacionJson<List<Jugador>>.Leer("listaJugadores.json");
    if (listaJugadores==null)
    {
        throw new Exception_SerializacionJson("No se deserializo la lista inicial");
    }
}

```



Temas utilizados

----- *entre clase 16 y 20* -----

SQL
Delegados
Hilos
Eventos
Métodos de extensión
Expresiones Lambda

SQL

Conexión a base de datos

En la clase **ConexiónDB** se conecta a la base de datos para traer el registro de jugadores inscriptos.

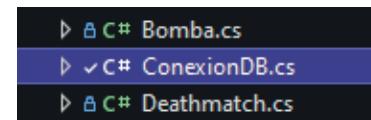
Datos necesarios:

Script: En este se encuentra la información necesaria para crear la base de datos completa y se encuentra dentro de la carpeta TP4 con el nombre script.sql.

ConnectionString: Se encuentra en el constructor de la clase **ConexiónDB**, línea 23.

```
21     static ConexionDB()
22     {
23         conexion = new SqlConnection(@"Server=DESKTOP-MACHMM\SQLEXPRESS;Database=TorneoCSGO;Trusted_Connection=True;");
```

Nombre base de datos: Mi base de datos se llama **TorneoCSGO** con una tabla llamada **Jugadores**.



Delegados

Ejemplo de implementación

Hago uso del delegado Simulacion que retorna un bool y recibe una lista de jugadores. Se le agregan 3 manejadores los cuales hacen que se simule el torneo completo.

```
public delegate bool Simulacion(List<Jugador> lista);
```

```
simulador += MapaBomba;
simulador += MapaDeathmatch;
simulador += MapaRehenes;
```

Se lo invoca desde el Frm_Home.

```
fullPartidas.simulador.Invoke(TorneoPro.ListaJugadores);
```

Tambien se utiliza el DelegadoActualizar con un evento.

```
public delegate void DelegadoActualizar(int carga);
```

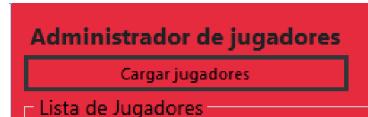
```
public static event DelegadoActualizar EventoActualizar;
```

Hilos

Ejemplo de implementación

En el botón cargar jugadores se crea un hilo secundario que conecta a la base de datos y comienza la carga de jugadores.

```
private void btn_CargarJugadorex_Click(object sender, EventArgs e)
{
    try
    {
        dgv_ListaJugadores.Rows.Clear();
        Task.Run(CargarJugadores);
        Frm_CargarJugadores cargando = new Frm_CargarJugadores();
        cargando.ShowDialog();
    }
}
```



Una vez que comienza la carga de jugadores podemos dejar cargando hasta el total o podemos cancelar en la cantidad deseada.



Para esto se creo un CancellationTokenSource el cual se utiliza para cancelar la carga de jugadores.

```
private CancellationTokenSource tokenSource;
```

Eventos

Ejemplo de implementación

Se utiliza un evento del tipo DelegadoActualizar para que vaya actualizando el número de jugadores cargados mientras los va leyendo desde la base de datos.

```
public static event DelegadoActualizar EventoActualizar;

ConexionDB.EventoActualizar += this.ActualizarProgressBar;

if (EventoActualizar is not null)
{
    ConexionDB.EventoActualizar.Invoke(auxLista.Count);
}
```

Se le agrega el manejador ActualizarProgressBar el cual va actualizando los valores a la medida que se van obteniendo.

```
private void ActualizarProgressBar(int valor)
{
    if (this.prb_Progreso.InvokeRequired)
    {
        DelegadoActualizar del = new DelegadoActualizar(this.ActualizarProgressBar);
        object[] arg = new object[] { valor };
        this.prb_Progreso.Invoke(del, arg);
    }
    else
    {
        this.prb_Progreso.Minimum = 0;
        this.prb_Progreso.Maximum = ConexionDB.totalColumnas;
        this.prb_Progreso.Value = valor;
        this.lbl_JugadoresCargados.Text = valor.ToString();
        lbl_CantidadTotal.Text = ConexionDB.totalColumnas.ToString();
        BotonEnable();
    }
}
```

Metodos de extensión

Ejemplo de implementación

Se crea la clase MisMetodosExtension donde posee dos métodos para darle formato específico al los datos tipo string dónde se utiliza en la clase Estadística para darle formato al análisis de datos.

```
public static class MisMetodosExtension
{
    8 referencias
    public static string MiFormato(this string dato, string titulo, int num1, float por1, int num2, float por2)
    {
        return String.Format("{0,-20}{1,5} | {2,5:0.00}% ||{3,5} | {4,5:0.00}%", titulo, num1, por1, num2, por2);
    }

    5 referencias
    public static string MiFormato2(this string dato, string titulo, int num1, float por1, int num2, float por2)
    {
        return String.Format("{0,-20}{1,5} | {2,5:0.000} ||{3,5} | {4,5:0.000}", titulo, num1, por1, num2, por2);
    }
}
```

"Participacion total del torneo ----- 150
"Categoria-----Female--- ---Male---
Genero: 74 49.33% 76 50.67%
"Primer torneo -----
Sí: 39 52.70% 41 53.95%
No: 35 47.30% 35 46.05%
"Rango etario -----
de 12 a 29: 22 29.73% 22 28.95%
de 30 a 49: 31 41.89% 31 40.79%
+ 50: 21 28.38% 23 30.26%
"Nacionalidad -----
Latinos 7 9.46% 7 9.21%
No latinos 67 90.54% 69 90.79%
"Promedios -----
Kills: 9327 62.180 9807 65.380
Muertes: 9328 62.187 10199 67.993
Headshots: 1947 12.980 2145 14.300
Bombas: 322 02.147 315 02.100
Rehenes: 234 01.560 248 01.653

Expresiones Lambda

Ejemplo de implementación

Se utiliza en la clase Jugador como recurso para ordenar una lista de forma descendente por diferentes tipos.

```
auxlista2 = auxlista2.OrderByDescending(x => x.Estadistica.Bombas).ToList();

auxlista2 = auxlista2.OrderByDescending(x => x.Estadistica.Muerte).ToList();

auxlista2 = auxlista2.OrderByDescending(x => x.Estadistica.Kills).ToList();
```