Programa: Competencia

La funcionalidad que pretende cumplir este programa es para una organización la cual organiza juegos según 3 categorías que son: Ajedrez, Carreras y Quemados. Se acuerda un límite de puntos el cual va a determinar el ganador de la categoría. El programa nos permite llevar el registro de los juegos que se van llevando a cabo, nos otorga estadística de los mismos y por ultimo permite exportar una lista con los juegos que el usuario seleccione para importarla si luego es necesario.

Temas utilizados e implementación:

Excepciones

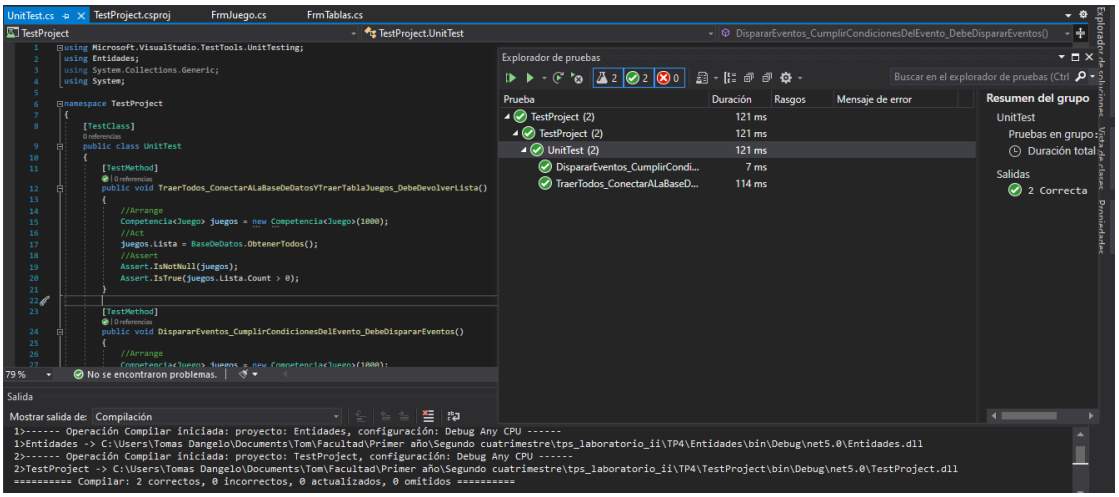
Para el uso de este programa cree 2 excepciones propias que se encuentran en las clases "*PuntosExcedidosException*" y "*ParametrosInvalidosException*". Estas las utilizo reiteradas veces en el programa para cubrir posibles excepciones. Provocadas por ejemplo por la inserción de parámetros que no corresponden. Ejemplo en la línea 54:

```
43          1 referencia
44          private void btn_Add_Click(object sender, EventArgs e)
45          {
46              try
47              {
48                  if(txt_Rojos.Text.SoloNumeros() && txt_Verdes.Text.SoloNumeros() && txt_Puntos.Text.SoloNumeros() && txt_Duracion.Text.SoloNumeros() && cmb_equipos
49                  {
50                      this.juego = new Juego(0, (Equipo)cmb_equipos.SelectedItem,Convert.ToInt32(txt_Rojos.Text), Convert.ToInt32(txt_Verdes.Text), Convert.ToInt32(t
51                      this.DialogResult = DialogResult.OK;
52                  }
53                  else
54                  {
55                      throw new ParametrosInvalidosException();
56                  }
57              }
58              catch (Exception ex)
59              {
60                  MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
61              }
62          }
```

También se utilizan varias estructuras try-catch a lo largo del programa para capturar posibles excepciones propias del sistema.

Pruebas unitarias

Se utilizaron pruebas unitarias para verificar el correcto funcionamiento de las funciones más importantes del programa. Estas se encuentran en el proyecto "TestProject" dentro de la solución.



Tipos genéricos

Se utiliza la clase genérica Competencia<T> para guardar el registro de los distintos juegos.

```
namespace Entidades
{
    36 referencias
    public class Competencia<T> where T : Juego
    {
        protected List<T> juegos;
        protected int puntosMax;

        public delegate void Evento(object sender, EventArgs e);
        public event Evento EventoFinalizado;
        public event Evento EventoReporte;

        2 referencias
        public Competencia()
        {
            juegos = new List<T>();
        }
```

Interfaces y archivos

Se hace uso de la interfaz ISerializable que permite serializar Juegos en formato Json. Esta obliga a implementar un serializador y un deserializador.

```
namespace Entidades
{
    1 referencia
    interface ISerializable
    {
        1 referencia
        bool SerializarAJson(string ruta, List<Juego> juego);
        1 referencia
        List<Juego> DeserializarAJson(string ruta);
    }
}
```

```
namespace Entidades
{
    56 referencias
    public class Juego : ISerializable
    {
        protected int codigo;
        protected Equipo ganador;
        protected int participantesRojos;
        protected int participantesVerdes;
        protected int puntos;
        protected double minutos;
        0 referencias
        public Juego()
        {
```

Base de datos

Se utiliza una base de datos para un almacenamiento consistente de los mismos, la aplicación va a crear, leer y eliminar los juegos haciendo uso de la misma. Hay una clase llamada “BaseDeDatos” que se encarga de la comunicación con la misma.

```
namespace Entidades
{
    19 referencias
    public class BaseDeDatos
    {
        private static string cadena_conexion;
        private static SqlConnection conexion;
        private static SqlCommand comando;

        0 referencias
        static BaseDeDatos()
        {
            comando = new SqlCommand();
            comando.CommandType = CommandType.Text;
            cadena_conexion = @"Data Source=PC-ASROCK;Database=Competencia;Trusted_Connection=True;
            conexion = new SqlConnection(BaseDeDatos.cadena_conexion);
            comando.Connection = conexion;
        }

        5 referencias | 1/1 pasando
        public static List<Juego> ObtenerTodos()
        {
            List<Juego> lista = new List<Juego>();

            try
            {
                conexion.Open();
```

Delegados, eventos y expresiones lambda

Se utiliza un evento para dar a saber que se finalizó un juego o se excedió el tiempo que debería durar la competencia. Esto tiene un manejador que se ejecuta las acciones necesarias si se disparó dicho evento. También se utiliza delegados de tipo <Action> en los hilos. Las expresiones lambda se utilizan para las propiedades de juego.

```
13
14          public delegate void Evento(object sender, EventArgs e);
15          public event Evento EventoFinalizado;
16          public event Evento EventoReporte;
17
```

```
public static void VerificarEstado(Competencia<T> c)
{
    double total = 0;
    foreach (T item in c.Lista)
    {
        total += item.Puntos;
    }
    if (total >= c.puntosMax)
    {
        if (c.EventoFinalizado != null)
        {
            c.EventoFinalizado(c, EventArgs.Empty);
        }
    }
    if(c.DuracionTotal >= 360)
    {
        if (c.EventoReporte != null)
        {
            c.EventoReporte(c, EventArgs.Empty);
        }
    }
}
```

```
        Resultados<Juego> resultados = new Resultados<Juego>();
        carrera.EventoFinalizado += Competencia_EventoFinalizado;
        quemados.EventoFinalizado += Competencia_EventoFinalizado;
        ajedrez.EventoFinalizado += Competencia_EventoFinalizado;
        carrera.EventoReporte += Competencia_EventoReporte;
        quemados.EventoReporte += Competencia_EventoReporte;
        ajedrez.EventoReporte += Competencia_EventoReporte;

        Actualizar_Listas();
    }


    3 referencias
    private void Competencia_EventoFinalizado(object sender, EventArgs e)
    {
        string tipo = sender.GetType().ToString();
        switch (tipo)
```

Hilos

Se utiliza un hilo distinto para exportar la información en archivos.

```
97      1 referencia
        private void btn_ExportarTipo_Click(object sender, EventArgs e)
98      {
99          dgv_juegos.SelectAll();
100         Task hiloExportar = Task.Run(Exportar);
101         Task.WaitAll(hiloExportar);
102         dgv_juegos.ClearSelection();
103     }
```

Metodos de extensión

Se extiende la clase string para hacer validaciones en la entrada de datos del frmJuegos, los métodos extendidos son SoloNumeros() y SoloChar()

```
7   namespace Entidades
8   {
        0 referencias
9       public static class StringExtendido
10      {
11          /// <summary>
12          /// Metodo de extensión para verificar que un texto solo tenga numeros
13          /// </summary>
14          /// <param name="str"></param>
15          /// <returns></returns>
            4 referencias
16          public static bool SoloNumeros(this string str)
17          {
18              foreach (char c in str)
19              {
20                  if (!char.IsDigit(c) && c != '.' && c != ',')
21                  {
22                      return false;
23                  }
24              }
25
26              return true;
27          }
28          /// <summary>
29          /// Metodo de extensión para verificar que un texto solo tenga letras
30          /// </summary>
31          /// <param name="str"></param>
32          /// <returns></returns>
            0 referencias
33          public static bool SoloCaracteres(this string str)
```