

TP N°2 (2°D)

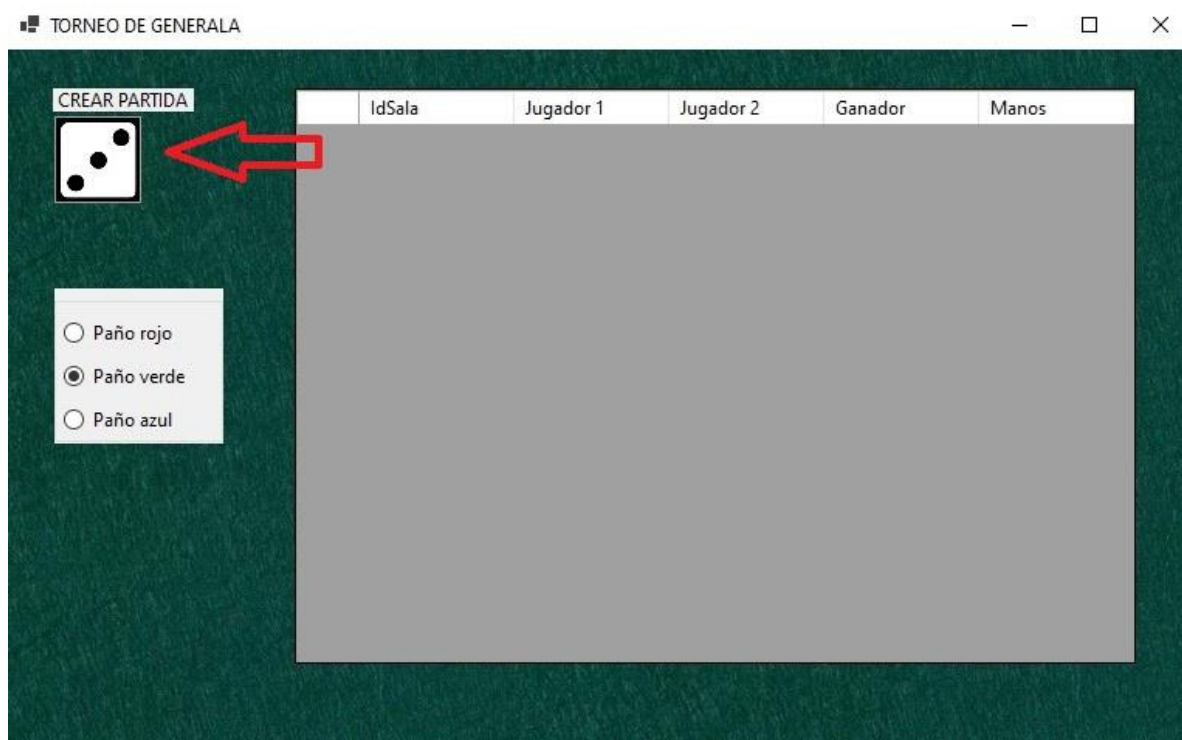
GENERALA

Sosa Maximiliano

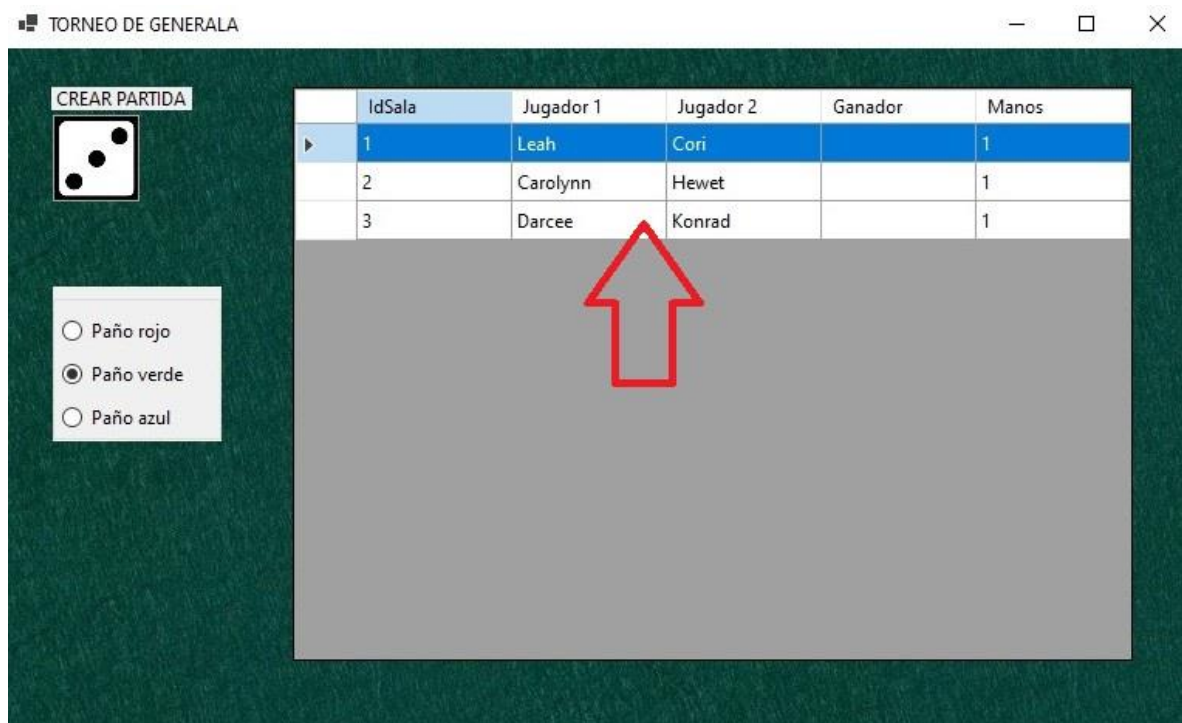


La aplicación inicia con la ventana de partidas.

Las partidas se inician con el botón **CREAR PARTIDA**.



Las partidas iniciadas se visualizan en la tabla de partidas



Para ver el desarrollo de una partida se hace doble click sobre la fila

MANO: 2

Dal 0

DADOS:
3 3 3 1 4
NO HAY JUEGO
PUNTAJE: 0

DADOS:
6 3 2 4 2
NO HAY JUEGO
PUNTAJE: 0

Burr 0

DADOS:
3 4 6 5 5
NO HAY JUEGO
PUNTAJE: 0

DADOS:
3 4 5 3 6
NO HAY JUEGO
PUNTAJE: 0

Jugador 2	Ganador	Manos
Cori	Leah	6
Hewet	Carolynn	4
Konrad	Konrad	5
Betty	Lottie	1
Burr		1
Dal		
Rodi		1

Para cancelar una partida en marcha se presiona el botón rojo

PARTIDA CANCELADA

Leon 20

DADOS:
2 5 1 1 1
NO HAY JUEGO
PUNTAJE: 20

Burr 0

DADOS:
3 2 6 2 2
NO HAY JUEGO
PUNTAJE: 0

DADOS:
3 3 1 4 2
NO HAY JUEGO
PUNTAJE: 0

DADOS:
5 1 2 1 5
NO HAY JUEGO
PUNTAJE: 0

Jugador 2	Ganador	Manos
Leah		6
Carolynn		4
Konrad		5
Lottie		1
PARTIDA CANC...		3
Rodi		4
PARTIDA CANC...		1

Gana el jugador que llega a ≥ 100 puntos.



Creé una clase **ParticipantesDB** que se conecta con la base de datos y trae a través del método **ObtenerJugadores** una lista de jugadores.

```

7
8 namespace Entidades
9 {
10     4 referencias
11     public class ParticipantesDB
12     {
13         string connectionString;
14         SqlConnection connection;
15         SqlCommand command;
16
17     1 referencia
18     public List<Jugador> ObtenerJugadores()
19     {
20         List<Jugador> participantes = new List<Jugador>();
21
22         try
23         {
24             connectionString = "Server = .; Database = SegundoTP; Trusted_Connection = True; Encrypt = False;";
25             connection = new SqlConnection(connectionString);
26
27             command = new SqlCommand();
28
29             connection.Open();
30
31             command.Connection = connection;
32
33             command.CommandType = System.Data.CommandType.Text;
34
35             command.CommandText = "SELECT * FROM Participantes3";
36
37             SqlDataReader reader = command.ExecuteReader();
38
39             while (reader.Read())
40             {
41                 int id = reader.GetInt32(0);
42                 string nombre = reader.GetString(1);
43                 string apellido = reader.GetString(2);

```

DELEGADOS

En la clase Delegados declaro tres delegados propios.

```
7 namespace Entidades
8 {
9     3 referencias
10     public static class Delegado
11     {
12         public delegate void Tirar(Jugador jugador1, Jugador jugador2);
13         public delegate bool Ganador(Jugador jugador);
14         public delegate void MostrarJuego(string texto, string texto2);
15     }
16 }
17
```

Les asigno los métodos.

```
16 public partial class FrmPartida : Form
17 {
18     Sala sala;
19     public int mano;
20     Delegado.Ganador DGanador;
21     Delegado.MostrarJuego DMostrarJuego;
22     public Action<string> cancelada;
23     public int idForm;
24
25     1 referencia
26     public FrmPartida()
27     {
28         InitializeComponent();
29
30         mano = 0;
31         DMostrarJuego = ActualizarDatos;
32
33         sala = new Sala(DMostrarJuego);
34         sala.IdSala = FrmPrincipal.idFormPartida;
35         idForm = FrmPrincipal.idFormPartida;
36         Torneo.Salas.Add(sala);
37         cancelada = Mostrar;
38         sala.PartidaTerminada += cancelada;
39         DGanador = Sala.HayGanador;
40     }
41 }
```

EXCEPCIONES, INTERFACES, GENERICS, SERIALIZACION

Tengo dos clases **SERIALIZADORAS** que implementan una **INTERFAZ GENÉRICA** que contiene dos firmas de métodos que son **Escribir** y **Leer**.

Se lanzan **EXCEPCIONES** en caso de que no pueda escribir en el archivo o si la ruta de lectura no existe.

```
8
9 namespace Entidades
10 {
11     5 referencias
12     public class SerializadoraJSON<T> : ISerializadora<T> where T : class
13     {
14         public static string ruta;
15
16         2 referencias
17         public SerializadoraJSON()
18         {
19             ruta = Environment.GetFolderPath(Environment.SpecialFolder.Desktop) + @"Sosa.Maximiliano.TP2-Serializacion";
20         }
21
22         2 referencias
23         public void Escribir(T objeto, string nombreArchivo)
24         {
25             try
26             {
27                 string rutaCompleta = ruta + @"\" + nombreArchivo + ".json";
28
29                 if (!Directory.Exists(ruta))
30                 {
31                     Directory.CreateDirectory(ruta);
32                 }
33
34                 string objetoJson = JsonSerializer.Serialize(objeto);
35
36                 try
37                 {
38                     File.AppendAllText(rutaCompleta, objetoJson);
39                 }
40                 catch (Exception ex)
41                 {
42                     throw new Exception($"Error al intentar leer o escribir el archivo. {ex.Message}");
43                 }
44             }
45             catch (PathTooLongException)
46             {
47                 throw;
48             }
49             catch (IOException)
50             {
51                 throw;
52             }
53             catch (Exception ex)
54             {
55                 throw;
56             }
57         }
58
59         2 referencias
60         public T Leer(string archivo)
61         {
62             T datos = default;
63
64             string rutaCompleta = ruta + @"\" + archivo + ".json";
65
66             if (!Directory.Exists(ruta))
67             {
68                 throw new Exception("La ruta no existe");
69             }
70
71             string archivoJson = File.ReadAllText(rutaCompleta);
72             datos = JsonSerializer.Deserialize<T>(archivoJson);
73             return datos;
74         }
75     }
76 }
77
78
79
80
81
```


UNIT TESTING

Hay una clase de test unitarios aplicados a varios métodos de diferentes clases.

```
1 namespace Entidades.Test
2 {
3     [TestClass]
4     public class ReglasDeberia
5     {
6
7         [DataRow(1, 2, 3, 4, 5)]
8         [DataRow(1, 3, 5, 2, 4)]
9         [DataRow(5, 4, 3, 2, 1)]
10        [TestMethod]
11        public void ValidarEscaleraMenor(int dado1, int dado2, int dado3, int dado4, int dado5)
12        {
13            List<int> dados = new List<int>();
14
15            dados.Add(dado1);
16            dados.Add(dado2);
17            dados.Add(dado3);
18            dados.Add(dado4);
19            dados.Add(dado5);
20
21            bool resultadoEsperado = true;
22            bool resultado = Reglas.EscaleraMenor(dados);
23            Assert.AreEqual(resultadoEsperado, resultado);
24        }
25
26        [DataRow(2, 3, 4, 5, 6)]
27        [DataRow(2, 4, 6, 3, 5)]
28        [DataRow(6, 5, 4, 3, 2)]
29        [TestMethod]
30        public void ValidarEscaleraMayor(int dado1, int dado2, int dado3, int dado4, int dado5)
31        {
32            List<int> dados = new List<int>();
33
34            dados.Add(dado1);
35            dados.Add(dado2);
36            dados.Add(dado3);
37            dados.Add(dado4);
38            dados.Add(dado5);
39        }
40    }
41 }
```

EVENTOS

En la clase **Sala** declaro un evento y lo utilizo para enviar el mensaje de “**PARTIDA CANCELADA**” cuando se presiona el botón.

(Funciona, pero no sé si está bien aplicado.)

```
17 private string ganador = string.Empty;
18 private int mano;
19
20 public event Action<string> partidaTerminada;
21
22 public CancellationTokenSource cts;
23 public CancellationToken ct;
```

```

79
80     if (ct.IsCancellationRequested)
81     {
82         partidaTerminada?.Invoke("PARTIDA CANCELADA");
83     }
84

```

TASK

En el constructor de la clase **Sala** ejecuto una tarea que hace que ni bien se instancie una sala se inicie una partida en otro hilo.

```

25     1 referencia
26     public Sala(MostrarJuego DMostrar)
27     {
28         dados = new List<int>();
29         for (int i = 0; i < 5; i++)
30         {
31             dados.Add(rand.Next(1, 7));
32         }
33         cts = new CancellationTokenSource();
34         ct = cts.Token;
35
36         this.jugador1 = Torneo.Participantes[rand.Next(0, 100)];
37         this.jugador2 = Torneo.Participantes[rand.Next(0, 100)];
38
39         Task task = Task.Run(() => { this.JugarPartida(DMostrar, ct); });
40
41     }

```