# TP FINAL LABORATORIO II

# FRANCO TOBIAS MARTINEZ

La idea de mi programa se basa en una encuesta sobre el rating televisivo, la principal función del mismo se basa en mostrar los datos en porcentajes sobre diversos filtros disponibles, ya sea género (masculino, femenino, no binario), rango etario u una edad especifica, y por canal en específico u canales totales.

Además, se creó un ABM de personas para que cualquiera pueda votar e influir en los resultados finales, a su vez se creo un sistema de login para administradores, cuales tienen la capacidad de agregar canales a la encuesta, modificar nombre del canal y eliminarlo, así como también realizar un ABM de otros administradores, y exportar toda la información del sistema en XML.

Dichos resultados se exportan en formato .txt.

*Temas Aplicados*

*Clase 15 y Clase 16 – SQL*

Establezco la conexión a la base de datos en la clase “ConnectionDB”.

```
13 references
public class ConnectionDB
{
    private SqlConnection connection;
    private SqlCommand command;
    private SqlDataReader reader;

    6 references
    public ConnectionDB()
    {
        Connection = new SqlConnection(@"Data Source = localhost\SQLEXPRESS;Database = TP4_Final_FrancoTobiasMartinez_2E;Trusted_Connection = True;");
        command = new SqlCommand();

        Command.CommandType = CommandType.Text;
        Command.Connection = Connection;
    }

    55 references
    public SqlConnection Connection { get => connection; set => connection = value; }
    83 references
    public SqlCommand Command { get => command; set => command = value; }
    68 references
    public SqlDataReader Reader { get => reader; set => reader = value; }
}
}
```

También lo utilizo para el ABM de personas, canales y administradores y para traerme los datos de la base de datos.

```
/// </summary>
/// <param name="obj"></param>
/// <returns></returns>
2 references
public bool AddToList(Persona obj)
{
    try
    {
        connectionDB.Command.CommandText = $"INSERT INTO TablePersonas(DNI,Nombre,Apellido,IDCanal,Edad,Genero) " +
                                           $"Values(@DNI,@Nombre,@Apellido,@IDCanal,@Edad,@Genero)";
        connectionDB.Command.Parameters.Clear();
        connectionDB.Command.Parameters.AddWithValue("@DNI", obj.Dni);
        connectionDB.Command.Parameters.AddWithValue("@Nombre", obj.Nombre);
        connectionDB.Command.Parameters.AddWithValue("@Apellido", obj.Apellido);
        connectionDB.Command.Parameters.AddWithValue("@IDCanal", obj.IdCanal);
        connectionDB.Command.Parameters.AddWithValue("@Edad", obj.Edad);
        connectionDB.Command.Parameters.AddWithValue("@Genero", obj.Genero);

        if (connectionDB.Connection.State != ConnectionState.Open)
        {
            connectionDB.Connection.Open();
        }

        connectionDB.Command.ExecuteNonQuery();

    }
    catch (Exception)
    {

        throw;
    }
    finally
    {
        connectionDB.Reader.Close();
        connectionDB.Connection.Close();

    }
    return true;
}
```

(Ejemplo agregar personas a la base de datos)

A su vez lo utilizo para hacer ciertos filtros

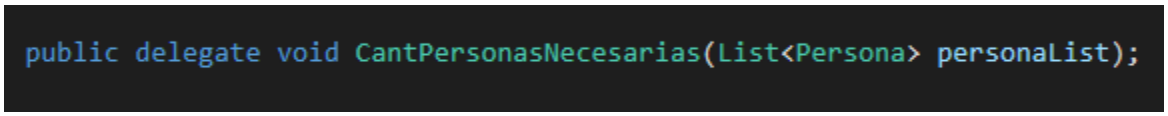
```
/// <returns></returns>
3 references
public static string PorcentajesTotalesPorCanal()
{
    try
    {
        string queryCantVotos = $"SELECT COUNT(TablePersonas.ID) AS 'VotosTotales' FROM TablePersonas";
        string queryVotosPorCanal = $"SELECT COUNT(TablePersonas.IDCanal) as 'CantVotos', TableCanales.Nombre from TablePersonas " +
                                    $"JOIN TableCanales on TablePersonas.IDCanal = TableCanales.ID " +
                                    $"GROUP BY TableCanales.Nombre ORDER BY 'CantVotos' DESC";
        GetVotosTotales(queryCantVotos);
        GetVotosPorCanal(queryVotosPorCanal);

        arrayPorcentajeVotos = new float[arrayVotosPorCanal.Length];
        resultado.Clear();

        SacarPorcentaje(arrayVotosPorCanal, arrayPorcentajeVotos, votosTotales);

        resultado.Append("Votos Totales Registrados: " + votosTotales);
        resultado.AppendLine();
        resultado.AppendLine();
        resultado.AppendLine(ObtenerResultados(arrayVotosPorCanal, arrayPorcentajeVotos));

        return resultado.ToString();
    }
    catch (Exception)
    {

        throw;
    }
}
```

*Clase 17- Expresiones Lambda y* Delegados

Expresion lambda utilizada para recorrer dos listas e ir incrementando 1 en 1 el valor de arrayVotosPorCanal

```
personalist.ForEach(persona =>
{
    int x = 0;
    canalList.ForEach(canal =>
    {
        if (Canal.GetNameById(persona.IdCanal) == canal.Nombre && persona.Edad == edadAFiltrar)
        {
            arrayVotosPorCanal[x] += 1;
        }
        x++;
    });
});
```

Delegado Personalizado para la utilización de un evento

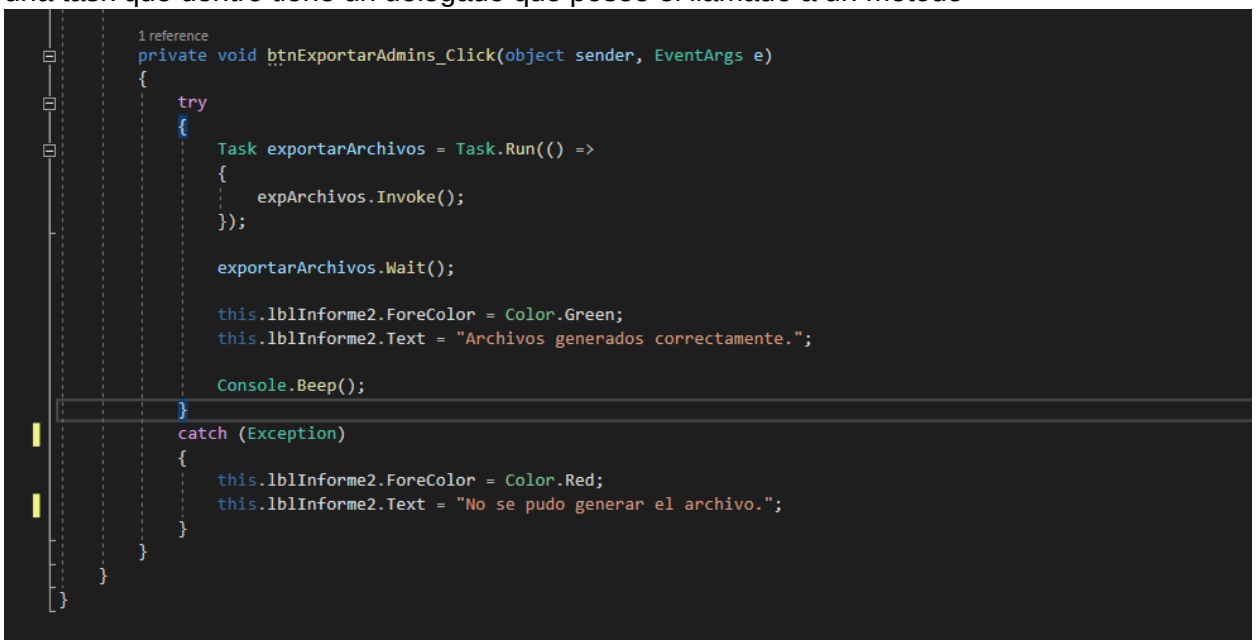


```
public delegate void CantPersonasNecesarias(List<Persona> personaList);
```

Clase 18 – Hilos

Utilizo hilos en varias partes, este es un ejemplo.

En este caso lo utilizo para guardar los datos del sistema en archivos XML, entonces lanzo una task que dentro tiene un delegado que posee el llamado a un método



```
1 reference
private void btnExportarAdmins_Click(object sender, EventArgs e)
{
    try
    {
        Task exportarArchivos = Task.Run(() =>
        {
            expArchivos.Invoke();
        });

        exportarArchivos.Wait();

        this.lblInforme2.ForeColor = Color.Green;
        this.lblInforme2.Text = "Archivos generados correctamente.";

        Console.Beep();
    }
    catch (Exception)
    {
        this.lblInforme2.ForeColor = Color.Red;
        this.lblInforme2.Text = "No se pudo generar el archivo.";
    }
}
}
```

El cual lanza otras tres task para escribir los datos de cada clase en hilos diferentes para ahorrar tiempo y se hagan simultáneamente.

```
        }

        1 reference
        public void ExportarArchivos() {
            try
            {
                Task.Run(Canal.ExportarArchivo);
                Task.Run(Admin.ExportarArchivo);
                Task.Run(Persona.ExportarArchivo);

            }
            catch (Exception)
            {
                throw;
            }
        }

    }
```

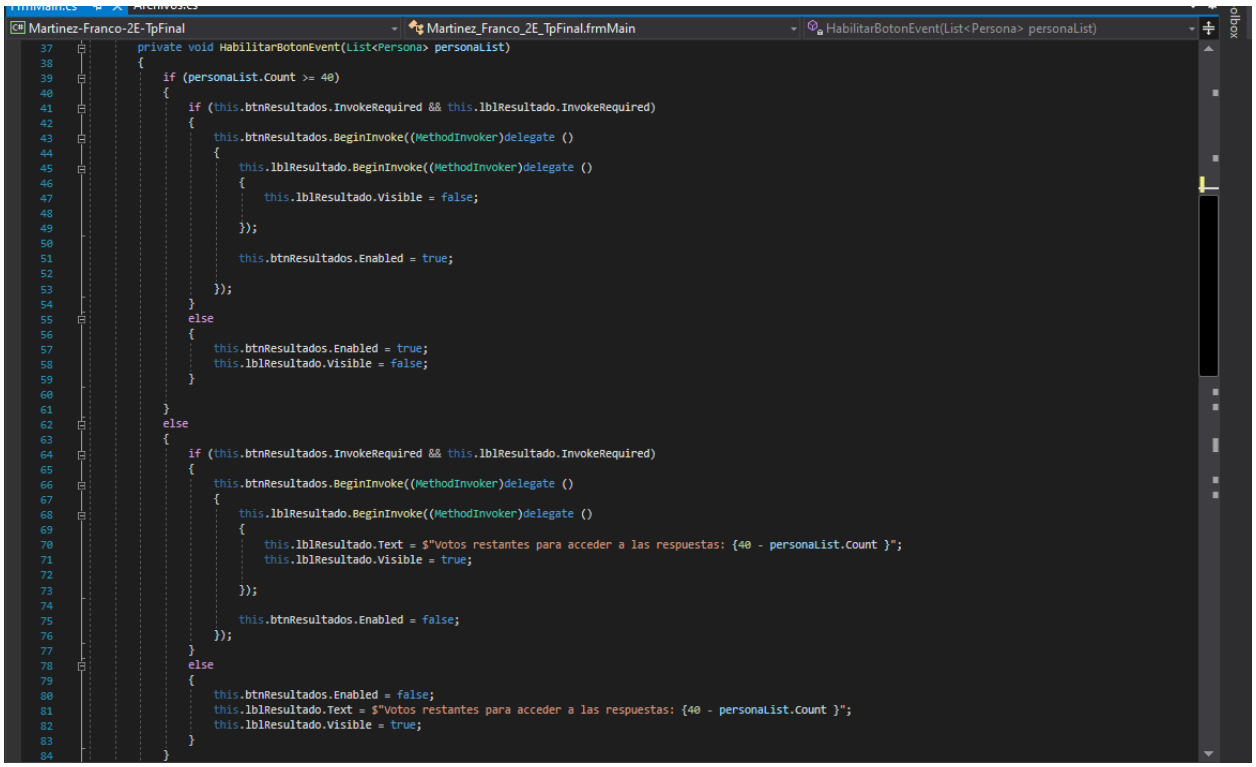
Clase 19 – Eventos

Eventos lo utilizo de manera que no se puedan ver los resultados de los votos hasta que se hayan registrado como mínimo 40 personas deshabilitando el botón “Resultados”.

Lo hago en un Thread aparte ya que quiero que este evento se ejecute a la par del Thread principal y no se detendrá hasta que el programa finalice debido a que le permito al usuario eliminar o agregar votos por lo que la cantidad de personas varia.

```
    {
        1 reference
        public frmMain()
        {

            InitializeComponent();

            Task.Run(() => PersonasNecesarias());
            Thread.Sleep(2000);

        }

        //emisor
        1 reference
        private void PersonasNecesarias()
        {
            while (true)
            {
                Persona persona = new Persona();
                persona.CantPersonasNecesarias += HabilitarBotonEvent;
                Thread.Sleep(1000);
                persona.ContarPeronas();

            }
        }
```

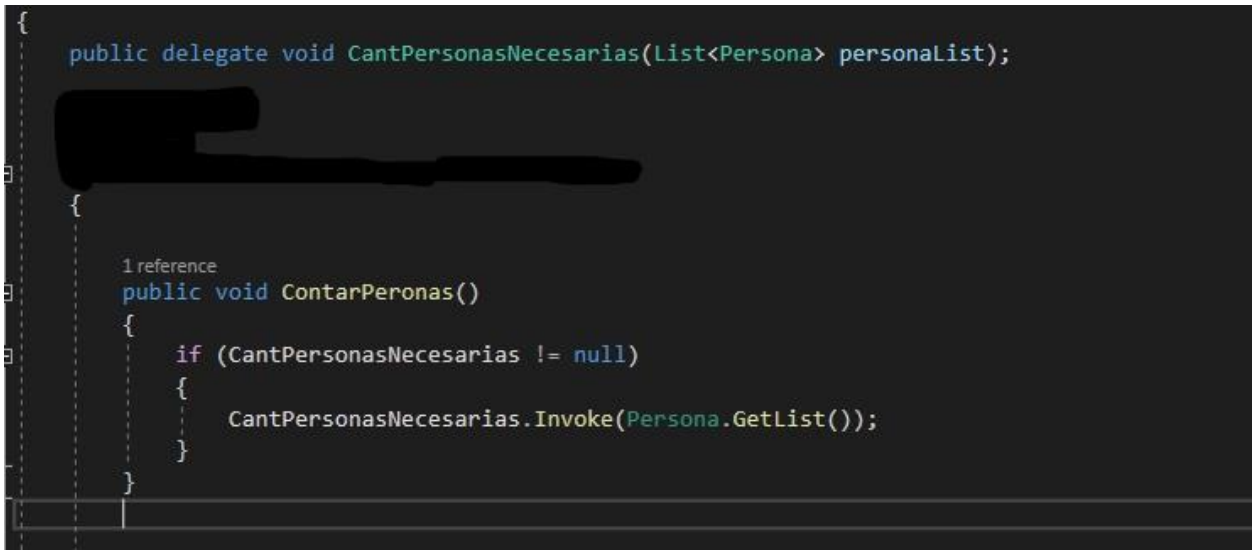
Verifico la cantidad de personas en personalist para habilitar o deshabilitar el botón “Resultado”, y mostrar un label de la cantidad de personas que falta para poder acceder a los resultados.




```
private void HabilitarBotonEvent(List<Persona> personaList)
{
    if (personaList.Count >= 40)
    {
        if (this.btnResultados.InvokeRequired && this.lblResultado.InvokeRequired)
        {
            this.btnResultados.BeginInvoke((MethodInvoker)delegate ()
            {
                this.lblResultado.BeginInvoke((MethodInvoker)delegate ()
                {
                    this.lblResultado.Visible = false;

                });

                this.btnResultados.Enabled = true;

            });
        }
        else
        {
            this.btnResultados.Enabled = true;
            this.lblResultado.Visible = false;
        }

    }
    else
    {
        if (this.btnResultados.InvokeRequired && this.lblResultado.InvokeRequired)
        {
            this.btnResultados.BeginInvoke((MethodInvoker)delegate ()
            {
                this.lblResultado.BeginInvoke((MethodInvoker)delegate ()
                {
                    this.lblResultado.Text = $"Votos restantes para acceder a las respuestas: {40 - personaList.Count }";
                    this.lblResultado.Visible = true;

                });

                this.btnResultados.Enabled = false;
            });
        }
        else
        {
            this.btnResultados.Enabled = false;
            this.lblResultado.Text = $"Votos restantes para acceder a las respuestas: {40 - personaList.Count }";
            this.lblResultado.Visible = true;
        }
    }
```

El manejador del Evento está ubicado en FrmMain.cs,

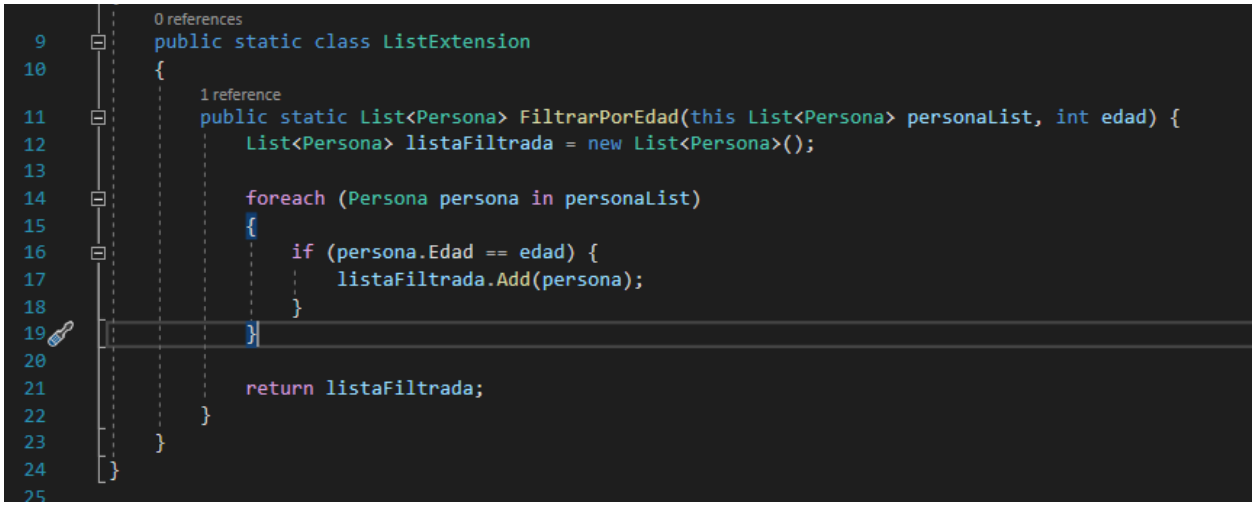
Delegado y su método para hacer el invoke



```
{
    public delegate void CantPersonasNecesarias(List<Persona> personaList);

    [redacted]

    {

        1 reference
        public void ContarPeronas()
        {
            if (CantPersonasNecesarias != null)
            {
                CantPersonasNecesarias.Invoke(Persona.GetList());
            }
        }
```

Clase 20 – Método de extensión

Este método de extensión pertenece a la colección List, lo que hace es que me retorna ya una lista filtrada por edad. Lo utilizo para sacar los resultados en los filtros.



```
0 references
public static class ListExtension
{
    1 reference
    public static List<Persona> FiltrarPorEdad(this List<Persona> personaList, int edad) {
        List<Persona> listaFiltrada = new List<Persona>();

        foreach (Persona persona in personaList)
        {
            if (persona.Edad == edad) {
                listaFiltrada.Add(persona);
            }
        }

        return listaFiltrada;
    }
}
}
```