Delegados y Eventos:

Declaracion de delegado y evento en la clase Alumno

```
public delegate void NuevoAlumno();
99+ referencias
public class Alumno : Persona
{
    private ESector sector;
    private ETurno turno;
    private EOrientacion orientacion;
    public static event NuevoAlumno Created;
    public static event NuevoAlumno Failed;
```

Evento invocado en método ingresar alumno:

```
public static void IngresarAlumno(Alumno alumno)
{
    try
    {
        if (alumno.Nombre.IsName() && alumno.Apellido.IsName())
        {
            DataBase.InsertToDB(alumno);
            Created.Invoke();
        }
        else
        {
            Failed.Invoke();
        }
    }
    catch (InvalidExtensionException)
    {
        throw new InvalidExtensionException();
    }
    catch (Exception)
    {
        throw new Exception();
    }

}
```

Utilizacion de evento en FrmIngresarAlumno:

```
1 referencia
private void btnIngresar_Click(object sender, EventArgs e)
{
    try
    {
        Alumno.Created += FillDataGrid;
        Alumno.Created += () => MessageBox.Show("Ingresado con Exito!", "Aviso!");
        Alumno.Failed += () => MessageBox.Show("Error al ingresar Alumno!", "Aviso!");
        Alumno.IngresarAlumno(new Alumno(txtNombre.Text, txtApellido.Text, int.Parse(txtEdad.Text), (EGenero)cmbGenero.SelectedItem,
                    (ESector)cmbSector.SelectedItem, (ETurno)cmbTurno.SelectedItem, (EOrientacion)cmbOrientacion.SelectedItem));
    }
    catch (InvalidExtensionException ex)
    {
        MessageBox.Show(ex.StackTrace, ex.Message);
    }
    catch (FormatException ex)
    {
        MessageBox.Show(ex.StackTrace, ex.Message);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.StackTrace, ex.Message);
    }
    finally
    {
        Alumno.Created -= FillDataGrid;
        Alumno.Created -= () => MessageBox.Show("Ingresado con Exito!", "Aviso!");
        Alumno.Failed -= () => MessageBox.Show("Error al ingresar Alumno!", "Aviso!");
    }
}
```

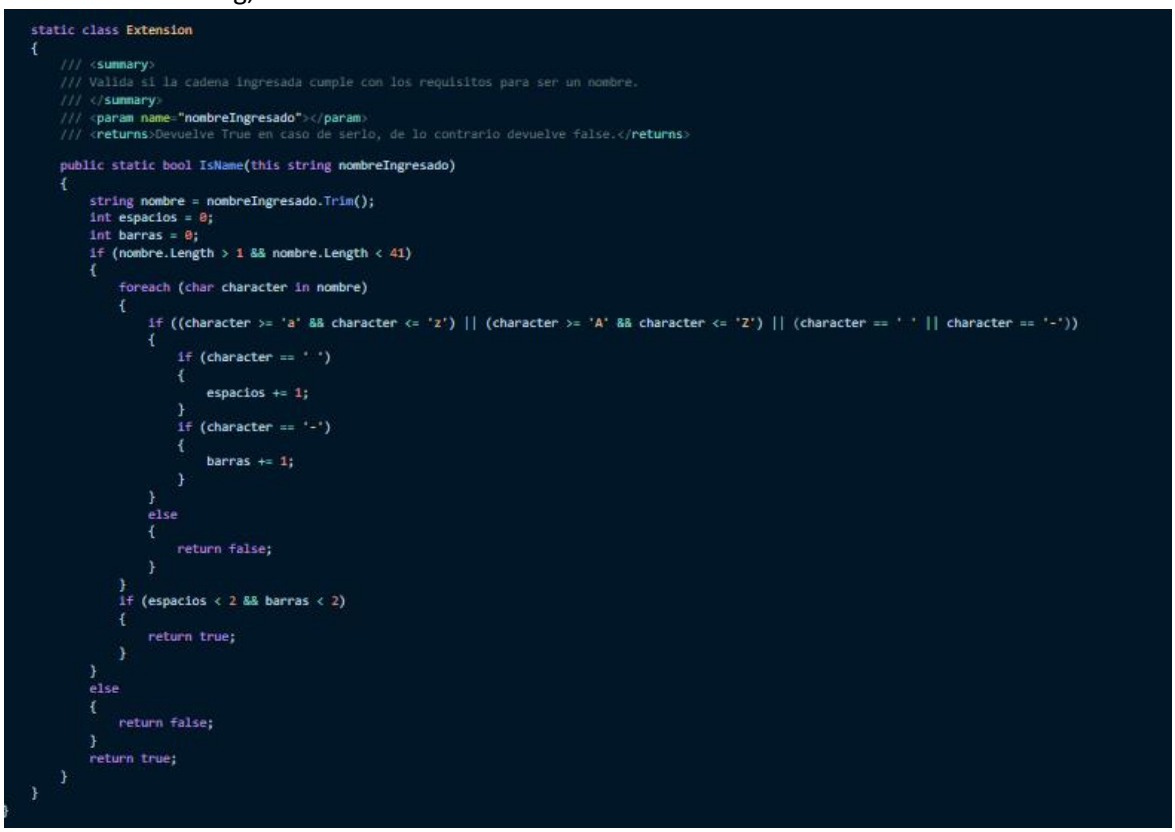
Threads:

Utilizo hilos en el formulario principal a la hora de cargar por primera vez los datos, y sin permitir realizar un análisis de datos hasta que esta tarea termine, pero si permitiendo que se acceda al apartado de ingresar alumno.

```
private async void Form1_Load(object sender, EventArgs e)
{
    this.pictureBox1.Visible = false;
    this.picLoading.SizeMode = PictureBoxSizeMode.StretchImage;
    this.picLoading.Image = Image.FromFile(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "loading.gif"));
    this.lblInicioText.Text = "Cargando datos de Alumnos.....";
    await CargarDatos();
}

/// <summary>
/// Realiza la carga de datos en un hilo paralelo, permitiendo continuar con otras tareas.
/// </summary>
/// <returns></returns>
1 referencia
private async Task CargarDatos()
{
    try
    {
        await Task.Run(() => {
            alumnos = DataBase.ImportFromDB();
            Thread.Sleep(5000);
            if (btnData.InvokeRequired)
            {
                btnData.BeginInvoke((MethodInvoker)delegate ()
                {
                    this.Refresh();
                    lblInicioText.Text = "Datos Cargados con Exito.";
                });
            }
            else
            {

                this.Refresh();
                lblInicioText.Text = "Datos Cargados con Exito.";

            }
            Thread.Sleep(1000);
        });
        this.picLoading.Visible = false;
        this.pictureBox1.Visible = true;
        this.btnData.Enabled = true;
        this.lblInicioText.Text = "Bienvenido, Que desea realizar?";
    }
    catch (InvalidExtensionException ex)
    {
        MessageBox.Show(ex.Message, ex.StackTrace);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, ex.StackTrace);
    }

}
```

Metodos de extensión:

Utilizados con String, sumándole una validación:



```
static class Extension
{
    /// <summary>
    /// Valida si la cadena ingresada cumple con los requisitos para ser un nombre.
    /// </summary>
    /// <param name="nombreIngresado"></param>
    /// <returns>Devuelve True en caso de serlo, de lo contrario devuelve false.</returns>

    public static bool IsName(this string nombreIngresado)
    {
        string nombre = nombreIngresado.Trim();
        int espacios = 0;
        int barras = 0;
        if (nombre.Length > 1 && nombre.Length < 41)
        {
            foreach (char character in nombre)
            {
                if ((character >= 'a' && character <= 'z') || (character >= 'A' && character <= 'Z') || (character == ' ' || character == '-'))
                {
                    if (character == ' ')
                    {
                        espacios += 1;
                    }
                    if (character == '-')
                    {
                        barras += 1;
                    }
                }
                else
                {
                    return false;
                }
            }
            if (espacios < 2 && barras < 2)
            {
                return true;
            }
        }
        else
        {
            return false;
        }
        return true;
    }
}
```