# Trabajo práctico 4 – Laboratorio II

## Jorge Enrique García Torrebilla 2E

Está aplicación está enfocada en la suscripción, gestión, y análisis de pólizas de seguro. Se utilizaron 2 tipos de seguros: Seguro de vida y seguro para vehículos.

Cuenta con un ABM y varios botones para filtrar la lista que se muestra en el dataGridView con la opción de exportar al formato deseado (XML o Json).

El análisis datos se encuentra en el botón "Estadísticas" en la que se muestran los porcentajes y promedios dependiendo de las variables de las pólizas.

## **Excepciones:**

Se utilizaron 2 excepciones personalizadas implementadas en varias partes del código para controlar los errores.

Clase ArchivoException: para controlar la serialización de los archivos y poder mostrar un mensaje personalizado del error.

```
public bool Exportar(string nombreArchivo, string info)
{
    string rutaDefinitiva = ruta + nombreArchivo;

    try
    {
        using (StreamWriter escritor = new StreamWriter(rutaDefinitiva))
        {
            escritor.WriteLine(info);
        }
        return true;
    }
    catch (Exception)
    {
        throw new ArchivoException($"Hubo un problema para importar la información de: {rutaDefinitiva}");
    }
}
```

Clase PolizaException: se usó para controlar el seteo de los datos cuando se da de alta o se modifica una póliza.

```
if (value > 200000)
{
    this.sumaAsegurada = value;
}
else
{
    throw new PolizaException("Algo ha salido mal: La suma asegurada no es válida. Aseguramos por un mínimo de 200.000$.");
}
```

#### Pruebas unitarias:

Las pruebas unitarias están el proyecto "TestingTP".

Se testaron si se lanzaban las excepciones en los casos de error, la importación de archivos XML y el que el cálculo del costo de una póliza de vehículo sea el esperado.

```
[TestMethod]
②|Oreferencias
public void TestImportarXML()
{
    XML<List<PolizaVida>> vidaXml = new XML<List<PolizaVida>>();
    XML<List<PolizaVehiculo>> vehiculosXml = new XML<List<PolizaVehiculo>>();
    List<PolizaVehiculo> lista2 = vehiculosXml.Importar("Polizas_Vehiculo.xml");
    List<PolizaVida> lista = vidaXml.Importar("Polizas_Vida.xml");
    Assert.IsTrue(lista.Count == 20);
    Assert.IsTrue(lista2.Count == 20);
}
```

## Tipos genéricos:

Se implemento en archivos para poder escribir y leer objetos de cualquier tipo y también se usaron métodos genéricos en la aplicación, por ejemplo:

```
6 referencias
private void Filtrar<T>(List<T> lista, Predicate<T> predicate) where T: Poliza
{
    manejador.DataSource = null;
    manejador.DataSource = lista.FindAll(predicate);
    dgvPolizas.DataSource = manejador.DataSource;
    manejador.ResetBindings(false);
}
```

```
1referencia
private void btnNoFumadores_Click(object sender, EventArgs e)
{
    this.lblTitulo.Text = "NoFumadores";
    Filtrar(Suscripciones.PolizasVida, (x) => x.Fumador == false);
}
```

#### Interfaces:

Se usaron 3 interfaces: IArchivo, IInformacion, IFumador

**IArchivo:** Implementada por las clases TXT, XML y JSON. Con sus métodos Importar y Exportar.

**Ilnformacion:** Implementada en las clases Poliza y Calculos. En Poliza se informan y poder exportar todos los datos de la póliza y en Calculos se informan todos los datos estadísticos recopilados.

**IFumador:** Implementada en la clase PolizaVida. Con sus métodos para calcular las tasas en los seguros de vida distintos a los criterios de los seguros para vehículos.

```
/// <summary>
/// Método que recopila todas la estadísticas del seguro y las retorna en un string.
/// </summary>
/// <returns></returns>
3referencias
public string Informacion()
{
    StringBuilder sb = new StringBuilder();
    StringBuilder sb2 = new StringBuilder();

    sb.AppendLine("- Ánalisis de datos -\n");
    sb.AppendLine($"Fecha: {DateTime.Now.ToString("D")}\n");
    sb.AppendLine($"Total de pólizas: {totalPolizas} - {Porcentaje(totalPolizas, totalPolizas, AppendLine(AnalisisSumaAsegurada());
    sb.AppendLine(AnalisisSexo());
    sb.AppendLine(AnalisisEdad());
    sb.AppendLine(AnalisisFumadores());
    sb.AppendLine(AnalisisFumadores());
    sb.AppendLine(AnalisisPorTipoVehiculo());
    sb.AppendLine(AnalisisAntiguedad());
    return sb.ToString();
}
```

```
public virtual string Informacion()
{
    StringBuilder sb = new StringBuilder();

    sb.AppendLine($"Fecha: {DateTime.Now..ToString("dd/MM/yyyy")}\n");
    sb.AppendLine($"Información del asegurado\n");
    sb.AppendLine($"DNI: {this.dni}");
    sb.AppendLine($"Nombre: {this.nombre}");
    sb.AppendLine($"Apellido: {this.apellido}");
    sb.AppendLine($"Edad: {this.edad}");
    sb.AppendLine($"Sexo: {this.sexo}");
    sb.Append($"Suma asegurada anual: {this.sumaAsegurada}");

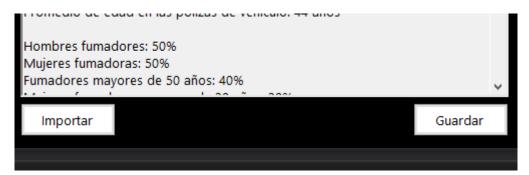
    return sb.ToString();
}
```

#### **Archivos:**

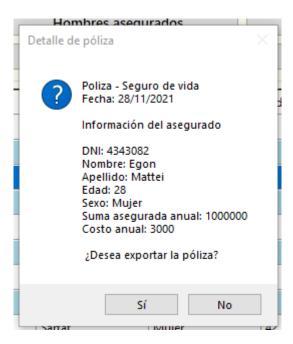
Se usaron archivos en todo el programa. Al iniciar la aplicación se importan 40 pólizas de dos archivos XML. Luego se tiene la opción de poder exportar la lista que se muestra en el DataGridView a formato JSON o XML.



Se pueden guardar las estadísticas actuales en formato txt e importar estadísticas que se encuentren en la carpeta de la aplicación "EstadisticasTXT" para mostrarlas en el RichTextBox.



Al hacer doble clic en una fila del dataGridView se da la opción de exportar la póliza seleccionada.



### Base de datos:

Se implemento el uso de base datos para el ABM y para cargar las pólizas al iniciar la aplicación.

Ejemplo de uso:

```
public static List<PolizaVida> TraerPolizasVida()
    try
        List<PolizaVida> polizas = new List<PolizaVida>();
         if (conexion.State != ConnectionState.Open)
             conexion.Open();
        comando.CommandText = $"SELECT * FROM VIDA";
        lectorBD = comando.ExecuteReader();
        while (lectorBD.Read())
             polizas.Add(new PolizaVida(lectorBD["nombre"].ToString(),
                                             lectorBD["apellido"].ToString(),
                                             int.Parse(lectorBD["dni"].ToString()),
                                             (ESexo)int.Parse(lectorBD["sexo"].ToString()),
                                             int.Parse(lectorBD["edad"].ToString()),
double.Parse(lectorBD["sumaAsegurada"].ToString())
bool.Parse(lectorBD["fumador"].ToString())));
         lectorBD.Close();
        return polizas;
    catch (Exception)
         throw new Exception("Error al leer las polizas de vida desde la base de datos.");
```

## Delegados y lambda:

Se uso un delegado en la clase Poliza para implementar un evento.

```
public delegate void CambioDeSumaAsegurada(string mensaje);
```

Se usaron expresiones lambda en gran parte del código, en especial en la clase Calculos:

```
1 referencia
private string AnalisisPorTipoVehiculo()
{
    StringBuilder sb = new StringBuilder();
    int totalMoto = Suscripciones.PolizasVehiculos.FindAll((x) => x.TipoVehiculo == ETipo.Moto).Count();
    int totalCamiones = Suscripciones.PolizasVehiculos.FindAll((x) => x.TipoVehiculo == ETipo.Camion).Count();
    int mujeresMoto = Suscripciones.PolizasVehiculos.FindAll((x) => x.TipoVehiculo == ETipo.Moto && x.Sexo == ESexo.Mujer).Count();
    int hombreMoto = Suscripciones.PolizasVehiculos.FindAll((x) => x.TipoVehiculo == ETipo.Moto && x.Sexo == ESexo.Hombre).Count();
    int hombreMoto = Suscripciones.PolizasVehiculos.FindAll((x) => x.TipoVehiculo == ETipo.Moto && x.Sexo == ESexo.Hombre).Count();
    int hombreAuto = Suscripciones.PolizasVehiculos.FindAll((x) => x.TipoVehiculo == ETipo.Auto && x.Sexo == ESexo.Hombre).Count();
    int mujeresAuto = Suscripciones.PolizasVehiculos.FindAll((x) => x.TipoVehiculo == ETipo.Auto && x.Sexo == ESexo.Hombre).Count();
    sb.AppendLine($"Porcentaje de mujeres con seguro de moto: {Porcentaje(totalMoto, mujeresMoto)}%");
    sb.AppendLine($"Porcentaje de hombres con seguro de moto: {Porcentaje(totalMoto, hombreMoto)}%");
    sb.AppendLine($"Porcentaje de hombres con seguro de auto: {Porcentaje(totalAuto, mujeresAuto)}%");
    sb.AppendLine($"Porcentaje de pólizas para camiones: {Porcentaje(totalAuto, hombreMuto)}%");
    sb.AppendLine($"Porcentaje de pólizas para camiones: {Porcentaje(totalPolizasVehiculos, totalCamiones)}%");
    sb.AppendLine($"Porcentaje de pólizas para auto: {Porcentaje(totalPolizasVehiculos, totalAuto)}%");
    schappendLine($"Porcentaje de pólizas para auto: {Porcentaje(totalPolizasVehiculos, totalAuto)}%");
    schappendLine($"Porcent
```

#### Hilos:

Se implemento al cargar la información inicial de la base datos en el formulario principal

```
Inferencia
private async void FrmPrincipal_Load(object sender, EventArgs e)
{
    try
    {
        await Task.Run(() =>
        {
             Thread.Sleep(5000);
            CargarListas();
        });
        this.lblTitulo.Text = "TodasLasPolizas";
        this.lblTitulo.Visible = false;
    }
    catch (ArchivoException exception)
    {
        MessageBox.Show(exception.Message, "Error al cargar pólizas", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    catch (Exception exception)
    {
        MessageBox.Show(exception.Message, "Error al cargar pólizas", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

También al abrir el formulario de estadísticas simulando la recopilación y el cálculo de datos antes de mostrarlos.

```
1referencia
private void FrmEstadisticas_Load(object sender, EventArgs e)
{
    Task.Run(() => { Thread.Sleep(5000); CargarEstadisticas(); });
}

1referencia
private void CargarEstadisticas()
{
    if (this.rtxtContenido.InvokeRequired)
    {
        this.rtxtContenido.BeginInvoke((MethodInvoker)delegate ())
        {
            this.rtxtContenido.Text = calculos.Informacion();
            this.btnExportar.Enabled = true;
        });
    }
}
```

#### **Eventos:**

Se implemento con el evento CostoModificado usando el delegado CambioDeSumaAsegurada.

El evento se invoca cuando se modifica la suma asegurada de una póliza. El evento notifica el porcentaje de cambio dependiendo si aumento o disminuyo la suma asegurada.

El manejador se asocia en el formulario FrmGestionPoliza.

```
if (value > this.sumaAsegurada && this.sumaAsegurada > 0)
   if (CostoModificado is not null)
        CostoModificado.Invoke($"{nombre} {apellido} aumento un {Calculos.Porcentaje(sumaAsegurada, value) - 100}% su suma asegurada! E
else if(value < this.sumaAsegurada)
   if (CostoModificado is not null)
        CostoModificado.Invoke($"{nombre} {apellido} disminuyo un {Math.Abs(Calculos.Porcentaje(sumaAsegurada, value) - 100)}% su s
this.sumaAsegurada = value;</pre>
```

## Métodos de extensión:

Se usaron 4 métodos de extensión implementados en la clase estática Extensiones. Ejemplo de uso:

```
public static bool SoloLetras(this string texto)
{
    foreach (char caracter in texto.ToCharArray())
    {
        if (char.IsDigit(caracter))
        {
            return false;
        }
    }
    return true;
}
```