

TP4 – Laboratorio 2

Avalos Leandro Sebastián – 2A.

Buena vida Seguros

20 de junio del 2022



Introducción a cómo funciona la aplicación

El proyecto simulara la atención al cliente de un empleado de una aseguradora de automóviles. El empleado podrá dar de alta a nuevos clientes, modificarlos de ser necesario y realizar una baja lógica de algún cliente que ya no quiera pertenecer a la aseguradora.

También podrá ver los datos cargados en sistema de las distintas sucursales de la aseguradora y en caso de necesitar agregar una sucursal al sistema el empleado podrá hacerlo cargando sus datos.

Por último, se podrá listar a los clientes que estén activos en la compañía y también cuenta con un Historial de actividades para llevar un seguimiento de cuando se dio de alta un cliente, se lo modifico, se dio de baja o si se cargó alguna sucursal en el sistema.

En el sistema se encuentran cargados algunos clientes y sucursales para poder probar las funcionalidades.

Dar de alta un cliente

Alta de Cliente

Alta de Cliente

Nombre

Apellido

Año del Vehículo

DNI

Numero de telefono

Marca del vehículo

Domicilio

Edad

18

Modelo del vehículo

Coberturas

☒ Seguro Basico

☐ Seguro Intermedio

☐ Seguro Contra Todo

Precio: 3000\$

Coberturas:

-> Tiene cobertura por choque

-> Tiene cobertura por robo

-> Tiene cobertura por granizo

-> No tiene cobertura por incendio parcial

-> No tiene cobertura de parabrisas y luneta

-> No tiene cobertura de reposicion de airbag

-> No tiene cobertura de seguro del conductor

-> No tiene cobertura por accidente total

-> No tiene cobertura por incendio total

Aceptar

Cancelar

En este apartado el empleado podrá dar de alta un potencial cliente pidiendo algunos datos de la persona y del vehículo que posee, así como también le da la oportunidad de escoger el plan que desea para asegurar su automóvil.

Modificar Cliente

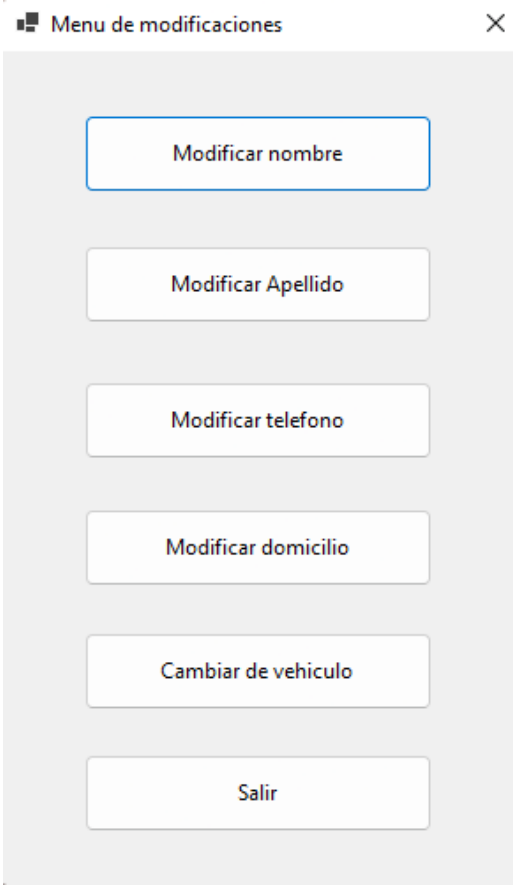
Chequeo de numero de cliente

Por favor ingrese el numero de cliente

Aceptar

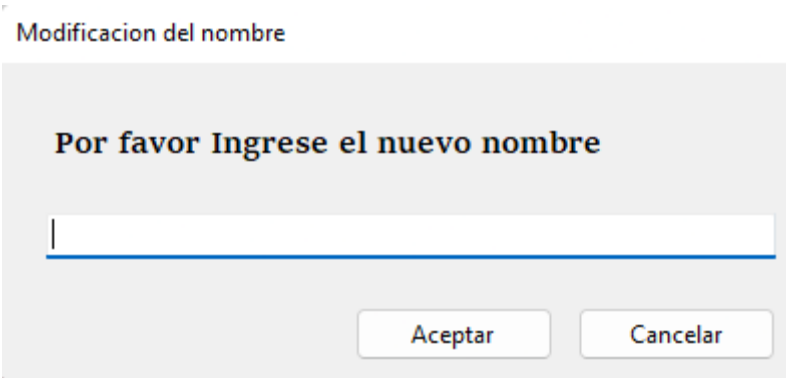
Cancelar

Cuando ingresamos a la opción de modificar lo primero que ocurre es que se nos pregunta el numero de cliente del usuario para corroborar que se encuentra en el sistema. De ser un cliente valido se procede al siguiente menú.



A screenshot of a software dialog box titled "Menu de modificaciones" with a close button (X) in the top right corner. The dialog box has a light gray background and contains six white rectangular buttons with black text, arranged vertically. The buttons are: "Modificar nombre", "Modificar Apellido", "Modificar telefono", "Modificar domicilio", "Cambiar de vehiculo", and "Salir".

Una vez en este menú podremos elegir que campo desea modificar el cliente.



A screenshot of a software dialog box titled "Modificacion del nombre". The dialog box has a light gray background and contains the text "Por favor Ingrese el nuevo nombre" in bold black font. Below the text is a white text input field with a blue border. At the bottom of the dialog box are two white buttons with black text: "Aceptar" and "Cancelar".

Si elige la opción de modificar nombre, apellido, teléfono o domicilio aparecerá esta pequeña ventana solicitando el dato correspondiente.

Nuevo Vehículo

Nuevo vehículo

Año del vehículo Marca del vehículo Modelo del vehículo

Coberturas

☒ Seguro Basico ☐ Seguro Intermedio ☐ Seguro Contra Todo

Precio: 3000\$

Coberturas:

- > Tiene cobertura por choque
- > Tiene cobertura por robo
- > Tiene cobertura por granizo
- > No tiene cobertura por incendio parcial
- > No tiene cobertura de parabrisas y luneta
- > No tiene cobertura de reposicion de airbag
- > No tiene cobertura de seguro del conductor
- > No tiene cobertura por accidente total
- > No tiene cobertura por incendio total

Aceptar Cancelar

Si se elige la opción de cambio de vehículo, aparecerá esta ventana solicitando los datos del nuevo vehículo y permitirá al cliente elegir un nuevo seguro para su auto.

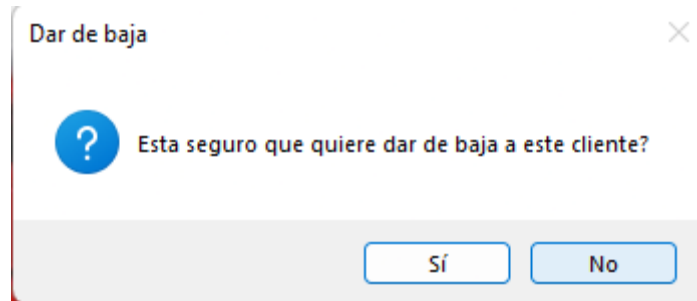
Dar de baja un cliente

Chequeo de numero de cliente

Por favor ingrese el numero de cliente

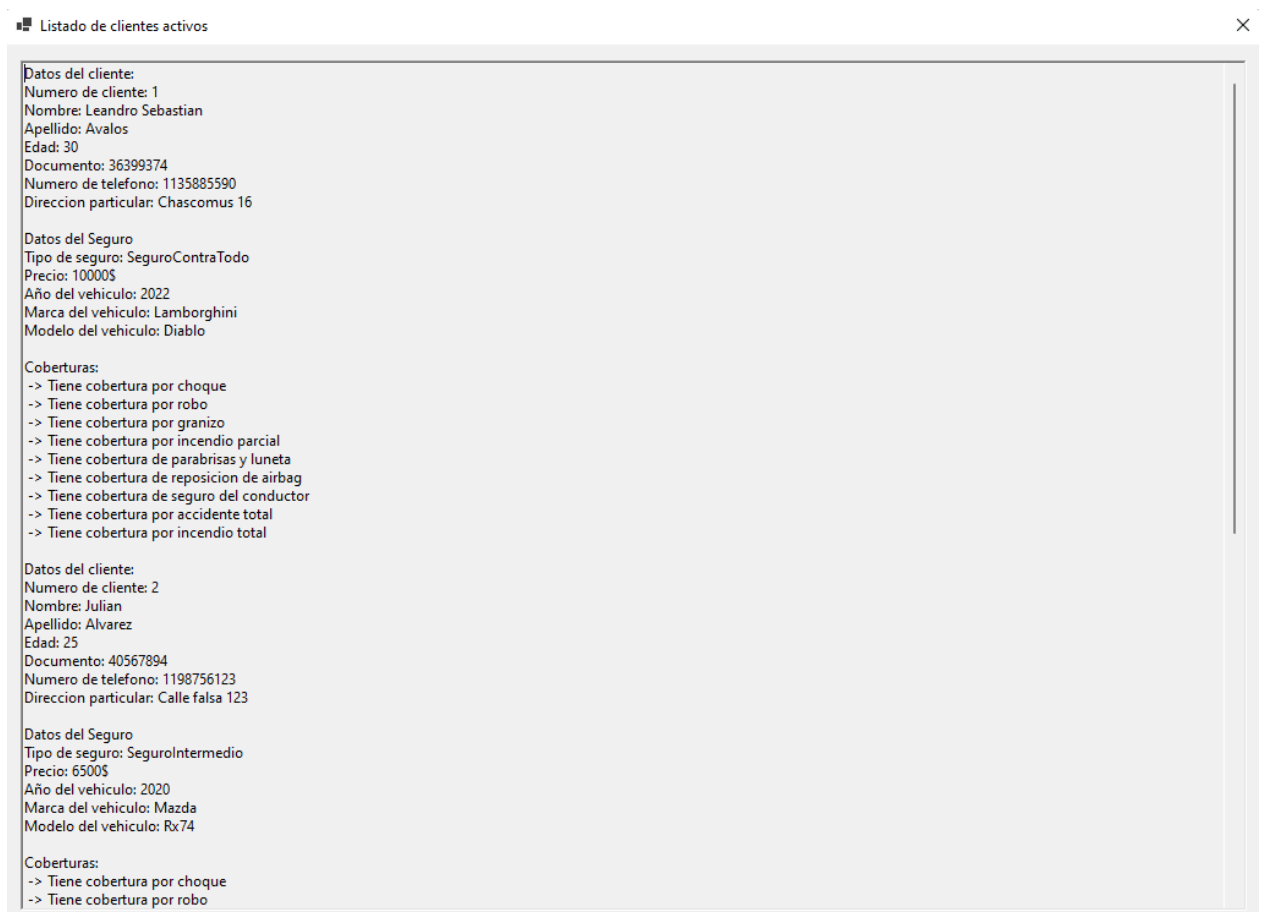
Aceptar Cancelar

Nuevamente se solicitara el numero de cliente para corroborar que sea un cliente en el sistema.



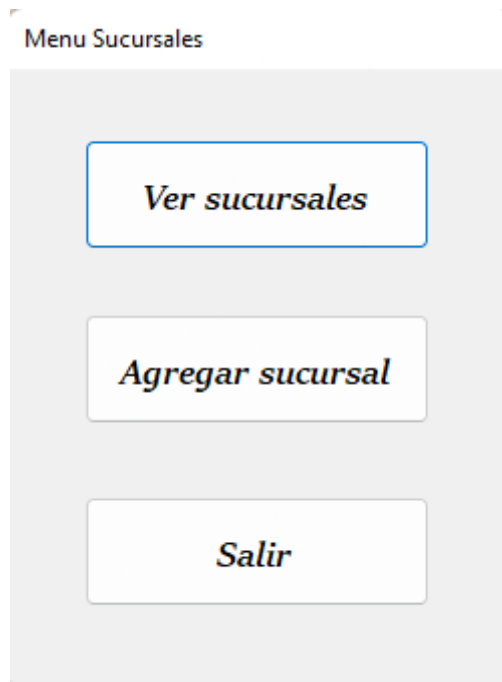
Una vez corroborado que el cliente esta en el sistema se pregunta si se esta seguro de querer darlo de baja. Si se presiona que si el cliente pasa a estar inactivo aplicándosele una baja lógica.

Cientes activos

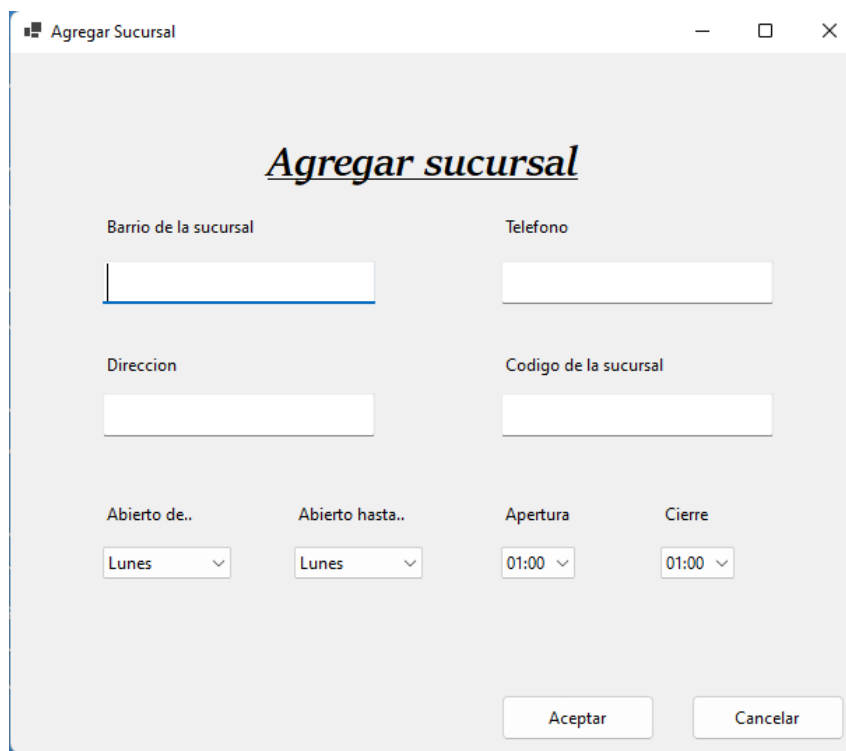


En este apartado podremos ver un cuadro de texto mostrando únicamente a los clientes que estén activos en el sistema.

Sucursales



Aquí se puede ver sucursales cargadas en sistema o agregar una nueva sucursal al sistema.

A screenshot of a software window titled "Agregar Sucursal". The window has a light gray background and contains a form with several input fields and dropdown menus. The title "Agregar sucursal" is centered at the top in a large, bold, italicized font. Below the title, there are four input fields arranged in a 2x2 grid: "Barrio de la sucursal", "Telefono", "Direccion", and "Codigo de la sucursal". Below these, there are four dropdown menus arranged in a 2x2 grid: "Abierto de..", "Abierto hasta..", "Apertura", and "Cierre". The "Abierto de.." and "Abierto hasta.." dropdowns are currently set to "Lunes". The "Apertura" and "Cierre" dropdowns are currently set to "01:00". At the bottom right of the window, there are two buttons: "Aceptar" and "Cancelar".

Historial de actividades

Historial de actividades

```
Se dio de alta un cliente en el sistema el domingo, 5 de junio de 2022 23:39
Se dio de alta un cliente en el sistema el domingo, 5 de junio de 2022 23:44
Se dio de alta un cliente en el sistema el domingo, 5 de junio de 2022 23:46
Se agrego una sucursal al sistema el domingo, 5 de junio de 2022 23:54
```

En este apartado se puede hacer un seguimiento de los movimientos realizados por el empleado.

Temas utilizados para la aplicación

- Excepciones
- Test Unitarios
- Tipos Genéricos
 - Interfaces
 - Archivos
- Serialización

NOTA: REVISAR EL APARTADO DE ARCHIVO PARA SABER LA RUTA DONDE SE GUARDAN LOS ARCHIVOS TXT Y XML.

Excepciones

Implementadas en varios sectores de los distintos formularios para controlar posibles incidentes que puedan detener la ejecución del programa. A continuación, dejo dos imágenes las cuales una es una excepción propia y otra es una donde fue implementada.

9 references

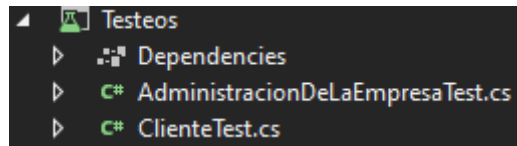
```
public class CamposVaciosException : Exception
{
    /// <summary>
    /// Constructor que instancia excepciones de tipo CamposVaciosException.
    /// </summary>
    /// <param name="mensaje">Mensaje que se lanzara cuando ocurra la excepcion.</param>
    3 references
    public CamposVaciosException(string mensaje) :this(mensaje, null)
    {
        ..
    }

    /// <summary>
    /// Constructor que instancia excepciones de tipo CamposVaciosException.
    /// </summary>
    /// <param name="mensaje">Mensaje que se lanzara cuando ocurra la excepcion.</param>
    /// <param name="innerException">Proporciona información sobre la excepcion interna.</param>
    1 reference
    public CamposVaciosException(string mensaje, Exception innerException) :base(mensaje,innerException)
    {
        ..
    }
}
```

1 reference

```
private void btnAceptar_Click(object sender, EventArgs e)
{
    try
    {
        if (!chequearCamposVacios())
        {
            int proximoNumeroDeCliente = int.Parse(GestionDeArchivosTxt.Txt_Reader($"{AppDomain.CurrentDomain.BaseDirectory}
            unaAdministracion += AltaDelCliente(proximoNumeroDeCliente);
            GestionDeArchivosSerializados<List<Cliente>>.Xml_Serializer($"{AppDomain.CurrentDomain.BaseDirectory}" + @"Clien
            GestionDeArchivosTxt.Txt_Writer($"{AppDomain.CurrentDomain.BaseDirectory}" + @"HistorialDeActividades.txt", mens
            proximoNumeroDeCliente += 1;
            GestionDeArchivosTxt.Txt_Writer($"{AppDomain.CurrentDomain.BaseDirectory}" + @"NumeroCliente.txt", proximoNumero
            MessageBox.Show("Se dio de alta al cliente exitosamente!", "Alta exitosa!!", MessageBoxButtons.OK, MessageBoxIcon
            this.Close();
        }
    }
    catch (CamposVaciosException ex)
    {
        MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    catch (AgregarClienteException ex)
    {
        MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    catch (ArchivosException ex)
    {
        MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Test unitarios



En el apartado de testeos hay test para algunas de las clases en donde se testea que ciertos métodos cumplan su función correspondiente.

```
[TestMethod]
0 references
public void BuscarClientePorNumeroCliente_SiElClienteSeEncuentraEnLaLista_DeberiaDevolverElIndice()
{
    //Arrange
    AdministracionDeLaEmpresa unaAdmin = new AdministracionDeLaEmpresa();
    Cliente clienteUno = new Cliente(1, 36399374, "Sergio", "Perez", 1164987456, "Calle falsa 123", 25,
    Cliente clienteDos = new Cliente(2, 40569874, "Ramon", "Falcon", 1123456897, "Falsa calle 321", 30,
    unaAdmin += clienteUno;
    unaAdmin += clienteDos;
    int expected = 1;
    int actual;

    //Act
    actual = unaAdmin.BuscarClientePorNumeroCliente("2");

    //Assert
    Assert.AreEqual(expected, actual);
}
```

Tipos genéricos

```
10 references
public static class GestionDeArchivosSerializados<T> where T : class
{
    /// <summary>
    /// Metodo que serializa un archivo en formato xml.
    /// </summary>
    /// <param name="ruta">Ruta donde se creara o guardara el archivo con formato xml.</param>
    /// <param name="datoASerializar">Dato que sera guardado en el archivo con formato xml.</param>
    7 references
    public static void Xml_Serializer(string ruta, T datoASerializar)
    {
        try
        {
            using (StreamWriter streamWriter = new StreamWriter(ruta))
            {
                XmlSerializer xml = new XmlSerializer(typeof(T));
                xml.Serialize(streamWriter, datoASerializar);
            }
        }
        catch (Exception)
        {
            throw new ArchivosException("Ocurrio un error al realizar la serializacion");
        }
    }
}
```

Implementado en la clase que gestiona la serialización y deserialización de XML y TxT. Recibe una lista objetos de tipo Cliente para ser trabajados como.

```
case "apellido":
    if (!string.IsNullOrEmpty(txtDatoAPedir.Text) && FrmChequeoDeDatos.SonLetras(txtDatoAPedir.Text))
    {
        unaAdministracion.ListaDeClientes[indice].Apellido = txtDatoAPedir.Text;
        MessageBox.Show("Apellido reemplazado con exito", "Actualizacion exitosa", MessageBoxButtons.OK, MessageBoxIcon.Information);
        GestionDeArchivosSerializados<List<Cliente>>.Xml_Serializer($"{AppDomain.CurrentDomain.BaseDirectory}" + @"Clientes.xml");
    }
}
```

Interfaces

```
2 references
public interface IDatosRequeridos
{
    /// <summary>
    /// Propiedad de lectura y escritura que debera implementarse por la interfaz.
    /// </summary>
    4 references
    double NumeroTelefonico { get; set; }

    /// <summary>
    /// Propiedad de lectura y escritura que debera implementarse por la interfaz.
    /// </summary>
    4 references
    string DireccionParticular { get; set; }

    /// <summary>
    /// Metodo para mostrar informacion que debera implementarse por la interfaz.
    /// </summary>
    /// <returns>Devolvera una string con datos.</returns>
    5 references
    string Mostrar();
}
```

Implementadas en las Clases Cliente y Sucursales ya que son clases que no tienen que ver entre ellas pero sin embargo tienen funcionalidades en común.

```
[XmlInclude(typeof(SeguroBasico))]
[XmlInclude(typeof(SeguroIntermedio))]
[XmlInclude(typeof(SeguroContraTodo))]
52 references
public class Cliente : IDatosRequeridos
{
    private int numeroDeCliente;
    private int documento;
    private string nombre;
    private string apellido;
    private double numeroTelefonico;
    private string direccionParticular;
    private int edad;
    private bool enServicio;
    Servicios servicio;
}
```

```

public class Sucursales : IDatosRequeridos
{
    private string barrio;
    private double numeroTelefonico;
    private string direccionParticular;
    private string codigoDeSecursal;
    private string abiertoDe;
    private string abiertoHasta;
    private string apertura;
    private string cierre;
}

```

Archivos

NOTA: TANTO LOS ARCHIVOS TXT COMO LOS ARCHIVOS XML SE ENCUENTRAN EN LA SIGUIENTE RUTA:

```

Txt_Reader($"{AppDomain.CurrentDomain.BaseDirectory}

```

La cual hace referencia a la carpeta donde se ejecuta el programa, en el debug de Windows form.

```

\\Avalos.LeandroSebastian.2A.TP3\WindowsForms\bin\Debug

```

ref	4/6/2022 23:29	Carpeta de archivos	
Cientes.xml	5/6/2022 23:51	XML Document	4 KB
Entidades.dll	6/6/2022 05:07	Extensión de la ap...	19 KB
Entidades.pdb	6/6/2022 05:07	Program Debug D...	19 KB
HistorialDeActividades	5/6/2022 23:54	Documento de te...	1 KB
NumeroCliente	5/6/2022 23:51	Documento de te...	1 KB
SeguroBasico	31/5/2022 20:24	Documento de te...	1 KB
SeguroContraTodo	31/5/2022 20:33	Documento de te...	1 KB
SeguroIntermedio	31/5/2022 20:32	Documento de te...	1 KB
Sucursales	5/6/2022 23:54	JSON File	1 KB
WindowsForms.deps	5/6/2022 14:57	JSON File	1 KB
WindowsForms.dll	6/6/2022 05:07	Extensión de la ap...	457 KB
WindowsForms	6/6/2022 05:07	Aplicación	123 KB
WindowsForms.pdb	6/6/2022 05:07	Program Debug D...	25 KB
WindowsForms.runtimeconfig.dev	4/6/2022 23:29	JSON File	1 KB
WindowsForms.runtimeconfig	4/6/2022 23:29	JSON File	1 KB

```

/// <summary>
/// Metodo que guarda datos en un archivo con formato txt.
/// </summary>
/// <param name="ruta">Ruta donde se creara o guardara el archivo con formato txt.</param>
/// <param name="datoAGuardar">Dato a guardar en el archivo con formato txt.</param>
/// <param name="concatena">Criterio recibido para saber si tiene que concatenar el text
9 references
public static void Txt_Writer(string ruta, string datoAGuardar, bool concatena)
{
    try
    {
        using (StreamWriter streamWriter = new StreamWriter(ruta,concatena))
        {
            streamWriter.WriteLine(datoAGuardar);
        }
    }
    catch (Exception)
    {
        throw new ArchivosException("Ocurrio un error al guardar el archivo");
    }
}

```

Utilizo archivos para cargar en un RichTextBox los distintos tipos de coberturas que poseen los seguros, también son utilizados para hacer el seguimiento de actividades que luego serán plasmadas en el apartado de Historial de actividades y para guardar el número de cliente de la última persona ingresada para que luego se auto asigne a la siguiente persona.

Coberturas

☒ Seguro Basico
 ☐ Seguro Intermedio
 ☐ Seguro Contra Todo

Precio: 3000\$

Coberturas:

- > Tiene cobertura por choque
- > Tiene cobertura por robo
- > Tiene cobertura por granizo
- > No tiene cobertura por incendio parcial
- > No tiene cobertura de parabrisas y luneta
- > No tiene cobertura de reposicion de airbag
- > No tiene cobertura de seguro del conductor
- > No tiene cobertura por accidente total
- > No tiene cobertura por incendio total

```
GestionDeArchivosTxt.Txt_Writer($"{AppDomain.CurrentDomain.BaseDirectory}" + @"HistorialDeActividades.txt", mensaje, true);
proximoNumeroDeCliente += 1;
GestionDeArchivosTxt.Txt_Writer($"{AppDomain.CurrentDomain.BaseDirectory}" + @"NumeroCliente.txt", proximoNumeroDeCliente.T
MessageBox.Show("Se dio de alta al cliente exitosamente!", "Alta exitosa!!", MessageBoxButtons.OK, MessageBoxIcon.Exclamati
this.Close();
```

Serialización

NOTA: REVISAR EL APARTADO DE ARCHIVO PARA SABER LA RUTA DONDE SE GUARDAN LOS ARCHIVOS TXT Y XML.

```
10 references
public static class GestionDeArchivosSerializados<T> where T : class
{
    /// <summary>
    /// Metodo que serializa un archivo en formato xml.
    /// </summary>
    /// <param name="ruta">Ruta donde se creara o guardara el archivo con formato xml.</param>
    /// <param name="datoASerializar">Dato que sera guardado en el archivo con formato xml.</param>
    7 references
    public static void Xml_Serializer(string ruta, T datoASerializar)
    {
        try
        {
            using (StreamWriter streamWriter = new StreamWriter(ruta))
            {
                XmlSerializer xml = new XmlSerializer(typeof(T));
                xml.Serialize(streamWriter, datoASerializar);
            }
        }
        catch (Exception)
        {
            throw new ArchivosException("Ocurrio un error al realizar la serializacion");
        }
    }
}
```

Utilizado para serializar y de serializar listas de tipo Cliente y Sucursal. La lista de tipo Cliente es trabajada en XML.

```
if (File.Exists($"{AppDomain.CurrentDomain.BaseDirectory}" + @"Clientes.xml"))
{
    this.unaAdministracion.ListaDeClientes = GestionDeArchivos<List<Cliente>>.Xml_Reader($"{AppDomain.CurrentDomain.BaseDirectory}" + @"Clientes.xml");
}
```


Utilizo la de serialización XML en el form central para cargar la lista con los datos siempre y cuando los archivos existan.

```
if (frmChequeoDeDatos.DialogResult == DialogResult.OK && unaAdministracion.ListaDeClientes[frmChequeoDeDatos.Indice].EnServicio == true)
{
    DialogResult chequeo = MessageBox.Show("Esta seguro que quiere dar de baja a este cliente?", "Dar de baja", MessageBoxButtons.YesNo,
    if (chequeo == DialogResult.Yes)
    {
        try
        {
            this.unaAdministracion.ListaDeClientes[frmChequeoDeDatos.Indice].EnServicio = false;
            GestionDeArchivosSerializados<List<Cliente>>.Xml_Serializer($"{AppDomain.CurrentDomain.BaseDirectory}" + @"Clientes.xml", una
            GestionDeArchivosTxt.Txt_Writer($"{AppDomain.CurrentDomain.BaseDirectory}" + @"HistorialDeActividades.txt", mensaje,true);
```

Y en esta imagen es utilizada la serialización para cuando se da de baja un cliente que la lista se actualice con la baja de esa persona.

Nuevos Temas utilizados para la aplicación

- SQL
- Delegados y Expresiones Lambda
- Programación Multi-Hilo y concurrencia
- Eventos
- Métodos de Extensión

NOTA:EL ARCHIVO .SQL SE ENCUENTRA DENTRO DE LA CARPETA DE LA SOLUCION.

SQL

Utilizado para traer la información de la base de datos de las sucursales ya existentes en el sistema o creadas. También es utilizado al momento de dar de alta una sucursal para que directamente se guarde en su base de datos correspondiente.

```
public static class GestionDeArchivosSQL
{
    /// <summary>
    /// Metodo para traer la informacion contenida en una base de datos
    /// </summary>
    /// <returns>Retorna una lista de sucursales tomada de la base de datos</returns>
    1 reference
    public static List<Sucursales> Leer()
    {
        List<Sucursales> sucursales = new List<Sucursales>();
        try
        {
            string query = "select * from Sucursales";
            using (SqlConnection connection = new SqlConnection("Server=.;Database=Sucursales;Trusted_Connection=True;"))
            {
                SqlCommand cmd = new SqlCommand(query, connection);
                connection.Open();
                SqlDataReader reader = cmd.ExecuteReader();

                while (reader.Read())
                {
                    string barrio = reader.GetString(0);
                    double numeroTelefonico = (double)reader.GetDecimal(1);
                    string direccionParticular = reader.GetString(2);
                    string codigoDeSecursal = reader.GetString(3);
                    string abiertoDe = reader.GetString(4);
                    string abiertoHasta = reader.GetString(5);
                    string apertura = reader.GetString(6);
                    string cierre = reader.GetString(7);
                    Sucursales sucursal = new Sucursales(barrio, numeroTelefonico, direccionParticular, codigoDeSecursal, abiertoDe, abiertoHasta, apertura,
                    sucursales.Add(sucursal);
                }
            }
        }
    }
}
```

```

1 reference
public void MostrarSucursales()
{
    this.Text = "Listado de sucursales";
    List<Sucursales> sucursales = GestionDeArchivosSQL.Leer_SQL();
    if(sucursales.Count > 0)
    {
        foreach (Sucursales unaSucursal in sucursales)
        {
            rtbMostrarDatos.Text += unaSucursal.Mostrar();
        }
    }
    else
    {
        rtbMostrarDatos.Text = "No hay sucursales para mostrar";
    }
}

```

```

/// <summary>
/// Metodo que instancia una sucursal para agregarla a la lista de sucursales, serializa en formato json la sucursal añ
/// </summary>
1 reference
private void AgregarSucursal()
{
    Sucursales unaSucursal = new Sucursales(txtBarrioDeLaSucursal.Text, double.Parse(txtTelefono.Text), txtDireccion.Te
        (string)cmbAbiertoDe.SelectedItem, (string)cmbAbiertoHasta.SelectedItem, (string)cmbHorarioApertura.SelectedItem

    string mensaje = $"Se agrego una sucursal al sistema el {DateTime.Now:f}";

    try
    {
        unaSucursal.Guardar_SQL();
        GestionDeArchivos<string>.Txt_Writer($"{AppDomain.CurrentDomain.BaseDirectory}" + @"HistorialDeActividades.txt"
        MessageBox.Show("Se agrego la sucursal al sistema exitosamente!", "Sucursal adherida exitosamente!!", MessageBo
    }
    catch (ArchivosException ex)
    {
        MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    catch (Exception)
    {
        MessageBox.Show("Ops... Ocurrio un error", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
}

```

Programación Multi-Hilo y concurrencia

Utilizado para correr en segundo plano la barra de progreso en el formulario principal, la cual simula la carga de datos de la base de datos de las sucursales.

```
/// <summary>
/// Metodo para iniciar la Task que corra en segundo plano la progress bar
/// </summary>
1 reference
private void IniciarHilo()
{
    Task task = new Task(() => ArranqueDeProceso(progressBar, lblBarraDeProgreso));
    task.Start();
}

/// <summary>
/// Metodo que mientras la progress bar no este en su valor maximo llamara a el metodo
/// y esto lo realiza en un intervalo random de 0,1ms a 1 segundo
/// </summary>
/// <param name="barraProgresiva">Barra de progreso enviada por parametro</param>
/// <param name="tituloBarra">Label que pertenece al progreso de la barra de progreso
1 reference
private void ArranqueDeProceso(ProgressBar barraProgresiva, Label tituloBarra)
{
    while (barraProgresiva.Value < barraProgresiva.Maximum)
    {
        Thread.Sleep(random.Next(100, 1000));
        IncrementoDeBarraProgreso(barraProgresiva, tituloBarra);
    }
}
```

```
2 references
private void IncrementoDeBarraProgreso(ProgressBar barraProgresiva, Label tituloBarra)
{
    if (this.InvokeRequired)
    {
        delegadoHilos = IncrementoDeBarraProgreso;
        object[] parametros = new object[] { barraProgresiva, tituloBarra };
        this.Invoke(delegadoHilos, parametros);
    }
    else
    {
        barraProgresiva.Increment(1);
        tituloBarra.Text = $"Progreso: {barraProgresiva.Value}%";

        EventoApagarYPrender.Invoke(barraProgresiva);
    }
}
```

Delegados y expresiones lambda

El delegado es utilizado en el subproceso de la barra de progreso para realizar la recursividad de la función que incrementa el progreso de la barra.

```
public delegate void DelegadoHilos(ProgressBar barraProgresiva, Label tituloBarra);
private DelegadoHilos delegadoHilos;
```

```
private void IncrementoDeBarraProgreso(ProgressBar barraProgresiva, Label tituloBarra)
{
    if (this.InvokeRequired)
    {
        delegadoHilos = IncrementoDeBarraProgreso;
        object[] parametros = new object[] { barraProgresiva, tituloBarra };
        this.Invoke(delegadoHilos,parametros);
    }
    else
    {
        barraProgresiva.Increment(1);
        tituloBarra.Text = $"Progreso: {barraProgresiva.Value}%";

        EventoApagarYPrender.Invoke(barraProgresiva);
    }
}
```

Lambda es utilizado para arrancar el proceso de la barra de progreso en la tarea que inicia el sub proceso.

```
private void IniciarHilo()
{
    Task task = new Task(() => ArranqueDeProceso(progressBar, lblBarraDeProgreso));
    task.Start();
}
```

Eventos

En la aplicación se utilizan dos eventos. Uno que mientras la barra de progreso no esté al máximo desactiva el botón de ver o dar de alta sucursales y cuando la barra se encuentra al máximo vuelve a habilitar el botón y envía un mensaje diciendo que la base de datos se cargó exitosamente y otro evento para enviar una notificación cuando se quiere modificar o dar de baja una persona que avisa que la persona se encuentra en el sistema y se puede modificar sus datos o dar de baja.

```
public delegate void DelegadoApagadoEncendido(ProgressBar barraProgresiva);  
public event DelegadoApagadoEncendido EventoApagarYPrender;
```

```
private void IncrementoDeBarraProgreso(ProgressBar barraProgresiva, Label tituloBarra)  
{  
    if (this.InvokeRequired)  
    {  
        delegadoHilos = IncrementoDeBarraProgreso;  
        object[] parametros = new object[] { barraProgresiva, tituloBarra };  
        this.Invoke(delegadoHilos,parametros);  
    }  
    else  
    {  
        barraProgresiva.Increment(1);  
        tituloBarra.Text = $"Progreso: {barraProgresiva.Value}%";  
  
        EventoApagarYPrender.Invoke(barraProgresiva);  
    }  
}
```

Método utilizada por el evento:

```
public void ApagarYPrenderBoton(ProgressBar barraProgresiva)
{
    if(barraProgresiva.Value < barraProgresiva.Maximum)
    {
        btnSucursales.Enabled = false;
    }
    else if(barraProgresiva.Value == barraProgresiva.Maximum)
    {
        btnSucursales.Enabled = true;
        MessageBox.Show("Se han terminado de cargar los archivos de las sucursales.\nYa puede acceder al boton de sucursales.");
        barraProgresiva.Visible = false;
        lblBarraDeProgreso.Visible = false;
    }
}
```

```
public delegate void DelegadoMensajeID();
public static event DelegadoMensajeID EventoMensajeID;
```

```
/// <summary>
/// Constructor estatico para añadir al EventoMensajeID el metodo EventoMensajeID
/// </summary>
0 references
static FrmChequeoDeDatos()
{
    AdministracionDeLaEmpresa.EventoMensajeID += EventoMensajeID;
}
```

```
public int BuscarClientePorNumeroCliente(string numeroCliente)
{
    int indice = -1;
    int contador = 0;

    foreach (Cliente unCliente in listaDeClientes)
    {
        if (unCliente == int.Parse(numeroCliente))
        {
            indice = contador;
            EventoMensajeID.Invoke();
            break;
        }
        contador++;
    }
    return indice;
}
```

Método utilizada por el segundo evento para notificar que la persona esta en el sistema y se puede modificar o dar de baja.

```
public static void EventoMensajeID()
{
    MessageBox.Show("El cliente se encuentra en el sistema, proceda a modificarlo", "Acceder a modificar",
```

Métodos de extensión

Utilizado para extender el método Guardar_SQL a la clase Sucursales.

```
public static void Guardar_SQL(this Sucursales sucursal)
{
    string query = "insert into Sucursales (barrio, numeroTel
        "values (@barrio, @numeroTelefonico, @direccionPartic
    SqlConnection connection = null;

    try
```

Utilizo una instancia de Sucursal para llamar al método de Guardar_SQL y poder dar de alta esa sucursal.

```

private void AgregarSucursal()
{
    Sucursales unaSucursal = new Sucursales(txtBarrioDeLaSucursal,
        (string)cmbAbiertoDe.SelectedItem, (string)cmbAbiertoHa

    string mensaje = $"Se agrego una sucursal al sistema el {Da

    try
    {
        unaSucursal.Guardar_SQL();
        GestionDeArchivos<string>.Txt_Writer($"{AppDomain.Curre
        MessageBox.Show("Se agrego la sucursal al sistema exito

    }
    catch (ArchivosException ex)
    {
        MessageBox.Show(ex.Message, "Error", MessageBoxButtons.

    }
    catch (Exception)
    {
        MessageBox.Show("Ops... Ocurrio un error", "Error", Mes

    }
}

```