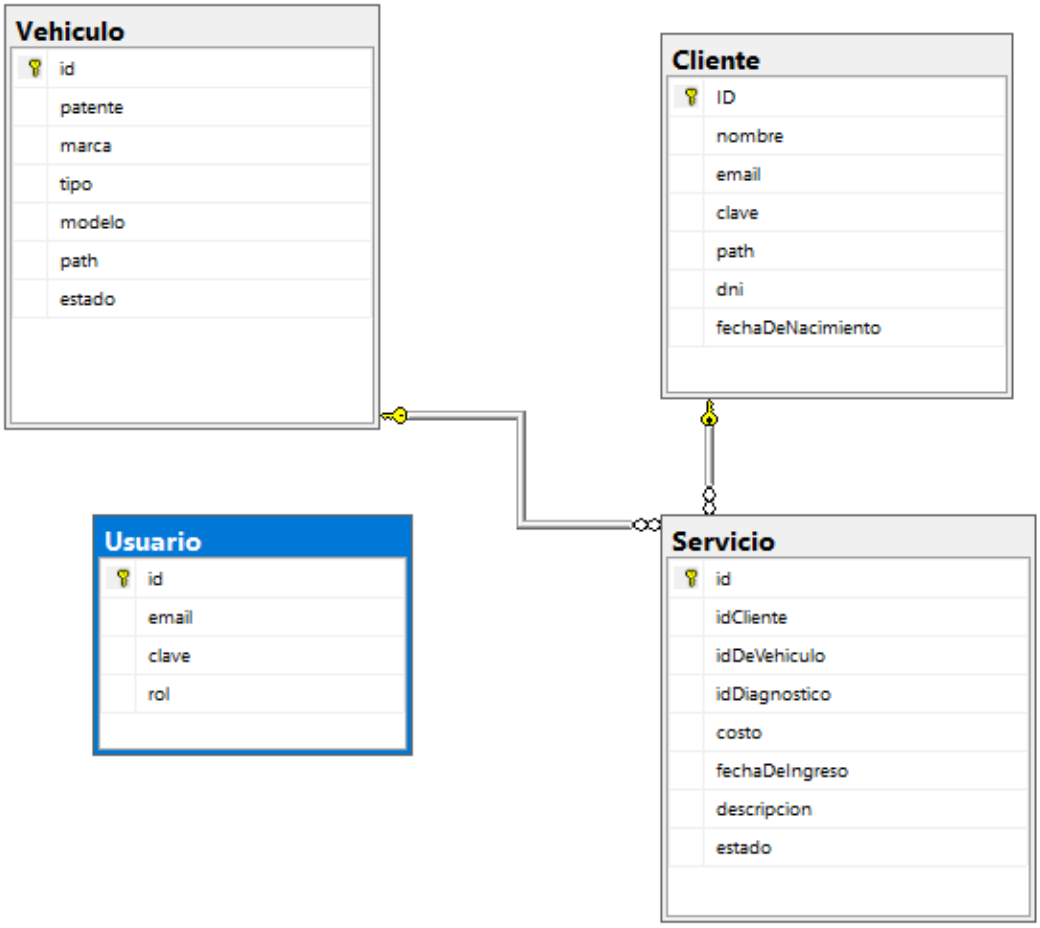
Ejercicio Integrador 2

El proyecto se trata de un **taller mecánico** donde permite el ingreso a 2 tipos de usuarios (Cliente, Personal) ,donde permite dar de alta,modificar,eliminar o cargar y guardar de archivo de clientes o de servicios y atender distintos servicios.

Entidades:



2 tipos de usuarios:

Cliente

● Puede iniciar sesión o registrarse (Dando de datos como el dni,Fecha de naciomiento, nombre)

● Puede dar de alta,modificar,eliminar servicios, dando los datos del vehículo y una descripción del problema,

● Puede cargar y guardar Archivos con datos de servicios.

● Puede ver la lista de vehículos registrados

Personal

- Puede ver la lista de clientes registrados y dar de alta,modificar,eliminar.
- Puede Atender Servicios indicando que Diagnostico tiene y el precio a pagar
- Puede ver la lista de vehículos registrados

Temas Implementados

Excepciones

Cree Dos clases de Excepciones una para la carga de datos atreves de la base de datos y otra para la carga de datos atreves de archivos

```
16 referencias
public class ConeccionBaseDeDatosException : Exception
{
    0 referencias
    public ConeccionBaseDeDatosException(string message) : this(message, null)
    {
    }

    9 referencias
    public ConeccionBaseDeDatosException(string message, Exception innerException) : base(message, innerException)
    {
    }

}
```

ConneccionABaseDeDatosExeption: Donde le indico si sucedió algún error a la hora de carga los datos.

Para Esto lo que hice fue a la hora de cargar y guardar los archivos de la base de datos utilice un tryCatch para que atrape la excepción que se produjo y lanzo una nueva excepción del tipo de la clase

```
{
    List<Cliente> list = null;

    try
    {
        coneccionSql.Open();
        comando.CommandText = $"Select * From Cliente";

        using (SqlDataReader dataReader = comando.ExecuteReader())
        {
            list = new List<Cliente>();
            while (dataReader.Read())
            {
                list.Add(ObtenerUnElemento(dataReader));
            }
        }
    }
    catch (Exception e)
    {
        throw new ConeccionBaseDeDatosException("Ocurrio un problema al intentar obtener los archivos de la base de datos", e.InnerException)
    }
    finally
    {
        if (coneccionSql.State == ConnectionState.Open)
        {
            coneccionSql.Close();
        }
    }
```

Y a la hora de cargar los datos en la aplicación , llamo función con un tryCatch y en caso de que suceda alguna excepción le muestro un messageBox de error con el mensaje de la excepción lanzada y cierro la aplicación

```
try
{
    unNegocio = Negocio.CargarBaseDeDatos("Taller Mecanico");
}
catch (Exception ex)
{
    FrmMenuPrincipal.InformarError("Error Base De Datos", ex.Message);
    this.Close();
}
```

La otra excepción que cree fue

```
3 referencias
public class JsonFileException : Exception
{
    2 referencias
    public JsonFileException(string message, Exception innerException) : base(message, innerException)
    {
    }
}
```

JsonFileException: Donde le indico si sucedió algún error a la hora de carga los datos.

Para Esto lo que hice fue a la hora de cargar y guardar el archivos de datos utilice un tryCatch para que atrape la excepción que se produjo y lanzo una nueva excepción del tipo de la clase

```
1 referencia
public static List<T> LeerArchivo(string path)
{
    List<T> listaDeClientes = default;

    if (ValidarPath(path) == true)
    {
        try
        {
            using (StreamReader sr = new StreamReader(path))
            {
                string text = sr.ReadToEnd();
                listaDeClientes = JsonSerializer.Deserialize<List<T>>(text);
            }
        }
        catch (Exception e)
        {
            throw new JsonFileException($"Ocurrio en error al intentar Leer el archivo de {typeof(T).Name}", e);
        }
```

Y a la hora de cargar los datos en la aplicación , llamo función con un tryCatch y en caso de que suceda alguna excepción le muestro un messageBox de error con el mensaje de la excepción lanzada

```
1 referencia
private void BtnCargar_Click(object sender, EventArgs e)
{
    List<T> unLista;
    if ((this.openFileDialog = AbrirArchivo("Cargar archivo", "Archivo Json (*.Json)|*.Json",
     Environment.GetFolderPath(Environment.SpecialFolder.Desktop))) is not null)
    {
        try
        {
            if ((unLista = JsonFile<T>.LeerArchivo(this.openFileDialog.FileName)) is not null)
            {
                ActualizarDataGried(dgtvList, listGeneric);
                this.OnInformar("Abrir Archivo", "Se cargo el archivo correctamente");
            }
        }
        catch (Exception ex)
        {
            OnInformarError("Abrir Archivo", ex.Message);
        }
    }
}
```

Pruebas unitarias

Cree dos clases de puebas unitarios , dende pruebo alguna funcionalidades de la Clase Cliente , Usuario, Vehiculo

ClienteTest

```
[TestClass]
0 referencias
public class ClienteTest
{
    [TestMethod]
    [DataRow("      ")]
    [DataRow(",,,,,,,,,,,,,,,,,Lkfkfk,,,,,,,,,,,,,,,,,+,,,")]
    [DataRow("9621233l")]
    0 referencias
    public void ValidarNombre_CunadoNoEsValida(string nombre)
    {
        Assert.IsFalse(Cliente.ValidarNombre(nombre));
    }

    [TestMethod]
    [DataRow(" mauro A")]
    [DataRow(",,,,,,,,,,,,,,,,,Lkfkfk,,,,,,,,,,,,,,,,,,,,")]
    [DataRow("pep pe pe")]
    [DataRow("lope")]
    0 referencias
    public void ValidarNombre_CunadoEsValida(string nombre)
    {
        Assert.IsTrue(Cliente.ValidarNombre(nombre));
    }
```

Donde hice dos método el cual recibe un string y prueba el método ValidarNombre de la clase cliente. Uno de ellos prueba el caso en que el método del cliente debería devolver true y otro en casos debería devolver false

Para esto A un método le paso atreves de un dataRow el string con las condiciones que debe cumplir para que sea un nombre y se lo paso al método validarNombre que debería devolver true

```
[TestMethod]
[DataRow(" mauro A")]
[DataRow("pepe")]
[DataRow("pep pe pe")]
[DataRow("lope")]
0 referencias
public void ValidarNombre_CunadoEsValida_DeberiaDebolverTrue(string nombre)
{
    Assert.IsTrue(Cliente.ValidarNombre(nombre));
}
```

En otro método le paso el string que no cumple con las condiciones que debería retornar false

```
[TestMethod]
[DataRow("      ")]
[DataRow(",,,,,,,,,,,,,,,,,Lkfkfk,,,,,,,,,,,,,,,,,+,,,")]
[DataRow("9621233l")]
0 referencias
public void ValidarNombre_CunadoNoEsValida_DeberiaDebolverFalse(string nombre)
{
    Assert.IsFalse(Cliente.ValidarNombre(nombre));
}
```

Lo Mismo hice a la hora de probar el método **validarDni**

```
[TestMethod]
[DataRow("hola56")]
[DataRow(",,,5552,,,,")]
[DataRow("22 966 3394")]
0 referencias
public void ValidarDni_CunadoNoEsValida_DeberiaDebolverFalse(string dni)
{
    Assert.IsFalse(Cliente.ValidarDni(dni));
}

[TestMethod]
[DataRow("22,966,333")]
[DataRow("22-966-333")]
[DataRow("22.966.332")]
[DataRow("11.278.666")]
0 referencias
public void ValidarDni_CunadoEsValida_DeberiaDebolverTrue(string dni)
{
    Assert.IsTrue(Cliente.ValidarDni(dni));
}
```

Y lo mismo hice con la clase **UsuarioTest**

```
[TestMethod]
[DataRow("pepe@gmail.com")]
[DataRow("mario@gmail.com")]
[DataRow("koko@gmail.com")]
0 referencias
public void ValidarEmail_CuandoElEmailEsValido_DeberiaRetornarTrue(string email)
{
    Assert.IsTrue(Usuario.ValidarEmail(email));
}

[TestMethod]
[DataRow("jajaja*jajjajaj")]
[DataRow("..............")]
[DataRow("1111111@111111111")]
0 referencias
public void ValidarEmail_CuandoElEmailEsNoValido_DeberiaRetornarFalse(string email)
{
    Assert.IsFalse(Usuario.ValidarEmail(email));
}
```

VehiculoTest

```
/// <summary>
/// Verifica en que caso el metodo ValidarPatente deberia Retornar false
/// </summary>
/// <param name="patente"></param>
[TestMethod]
[DataRow("Hjajaajajajja")]
[DataRow(null)]
[DataRow("6S5*25A")]
0 | 0 referencias
public void ValidarPatente_CunadoNoValido_DeberiaRetornarFalse(string patente)
{
    Assert.IsFalse(Vehiculo.ValidarPatente(patente));
}

/// <summary>
/// Verifica en que caso el metodo ValidarPatente deberia Retornar True
/// </summary>
/// <param name="patente"></param>
[TestMethod]
[DataRow("6S525A")]
[DataRow("6S5 25A")]
[DataRow("HoalMe")]
0 | 0 referencias
public void ValidarPatente_CunadoLaPatenteEsValida_DeberiaRetornarTrue(string patente)
{
    Assert.IsTrue(Vehiculo.ValidarPatente(patente));
}
```

Generics

Hice un Formulario genérico que muestra los datos de un elemento de una clase cualquiera

```
18 referencias
public partial class FrmMostrar<T> : Form
{
    Type unTipo;
    T element;
    List<PropertyInfo> propertiesGets;
    Predicate<PropertyInfo> predicate;
    1 referencia
    public FrmMostrar(T element,string path = null, string titulo = "Perfil")
    {
        InitializeComponent();
        this.unTipo = typeof(T);
        this.element = element;
        this.path = path;
        this.Titulo = titulo;
    }
    5 referencias
    public FrmMostrar(T element, Predicate<PropertyInfo> predicate, string path = null, string titulo = "Perfil") : this(element, path, titulo)
    {
        this.predicate = predicate;
    }
    1 referencia
```

En esta clase lo que hago es pedirle el elemento que quiere mostrar atreves del constructor
Después obtengo las propiedades get del tipo genérico que me pasaron

```
this.propertiesGets = ObtenerPropiedades(unTipo.GetProperties(), unaPropiedad => unaPropiedad is not null && unaPropiedad.CanRead == true);
```

```
2 referencias
private static List<PropertyInfo> ObtenerPropiedades(PropertyInfo[] propertyInfos, Predicate<PropertyInfo> predicate)
{
    List<PropertyInfo> result = new List<PropertyInfo>();

    if (predicate is not null && propertyInfos is not null)
    {
        foreach (PropertyInfo unProperty in propertyInfos)
        {
            if (predicate.Invoke(unProperty) == true)
            {
                result.Add(unProperty);
            }
        }
    }

    return result;
}
```

Obtengo los valores de las propiedades del elemento.

En un método lo que hago es crear un label donde guardo el nombre de esas propiedades y los valores

```
/// <summary>
/// Crea un Label y gueada los datos de la propiedad
/// </summary>
/// <param name="unaPropiedad">datos a guardar</param>
/// <returns></returns>
1 referencia
private Control CrearUnControl(PropertyInfo unaPropiedad)
{
    Label unControl = default;
    if (unaPropiedad is not null)
    {
        unControl = new Label();
        unControl.AutoSize = true;
        unControl.Font = new Font("Arial", 12.5F, FontStyle.Regular, GraphicsUnit.Point);
        unControl.Name = $"lbl{unaPropiedad.Name}";
        unControl.Text = $"{unaPropiedad.Name}: {unaPropiedad.GetValue(element)}";
    }

    return unControl;
}
```

Y Los labels que cree los guardo dentro de flowLayoutPanel, ya que este panel los acomoda automáticamente

```
/// <summary>
/// Recorre la lista de PropertyInfo y la llma una funcion que crea controles y ese control creado lo guarda dentro de una lista de controles
/// </summary>
/// <param name="listaDeControles">la lista donde se va guardar los controles creados</param>
/// <param name="unaLista">la lista que contiene las propiedades que se quiren guardar</param>
/// <returns>(true) si se pudo agragar los controles , (false) si no se pudo</returns>
1 referencia
private bool CrearControles(Control.ControlCollection listaDeControles, List<PropertyInfo> unaLista)
{
    bool result = default;

    if (unaLista is not null && listaDeControles is not null &&
     unaLista.Count > 0)
    {
        result = true;
        foreach (PropertyInfo unProperty in unaLista)
        {
            listaDeControles.Add(CrearUnControl(unProperty));
        }
    }

    return result;
}
```

También Generics lo Implemente

En esta clase

```
/// <summary>
/// Permite Manejar los datos de una lista atravez de un abm , cosultar datos de esta lista , filtar elementos , etc
/// </summary>
/// <typeparam name="T">el tipo de la clase de abm que se va generar</typeparam>
12 referencias
public abstract partial class FrmListar<T> : Form, IAbm<T>
    where T : class
{
    protected T element;
    private int indexRow;
    protected List<T> listGeneric;
    private SaveFileDialog saveFileDialog;
    private OpenFileDialog openFileDialog;
    protected event Action<string, string> InformarError;
    protected event Action<string, string> Informar;
    protected event Func<string, List<T>, List<T>> Buscador;
    protected event Func<List<T>, string, List<T>> Filtrar;
    4 referencias
    public FrmListar(List<T> listGeneric)
    {
        InitializeComponent();
        this.listGeneric = listGeneric;
        this.Load += FrmListar_Load;
        indexRow = -1;
    }
    6 referencias
    private T this[int index]
```

```
6 referencias
public abstract bool Alta();

/// <summary>
/// Guarda los datos de la lista en el dataGriedView
/// </summary>
/// <param name="dgtv">el dataGriedView donde se van a guardar los datos</param>
/// <param name="lista">la lista con los elementos a guardar</param>
14 referencias
public abstract void ActualizarDataGried(DataGridView dgtv, List<T> lista);

/// <summary>
/// Agrega columnas al Data dataGriedView
/// </summary>
/// <param name="dgtvList"></param>
/// <param name="listGeneric"></param>
6 referencias
public abstract void AgregarColumnasDataGried(DataGridView dgtvList);
6 referencias
public abstract bool Baja(T element);
7 referencias
public abstract bool Mostrar(T element);
6 referencias
public abstract bool Modificacion(T element);
```

Ejemplo de uso (FrmServicios),(FrmClientes),(FrmVehiculos)

```
10 referencias
public override void ActualizarDataGried(DataGridView dgtv, List<Servicio> lista)
{

    if(dgtv is not null && lista is not null)
    {
        dgtv.Rows.Clear();
        foreach (Servicio unServicio in lista)
        {
            dgtv.Rows.Add(unServicio.FechaDeIngreso, unServicio.UnVehiculo.Patente, unServicio.Problema,
                unServicio.CotizacionStr,unServicio.Diagnistico);
        }
    }
}

3 referencias
public override void AgregarColumnasDataGried(DataGridView dgtvList)
{
    dgtvList.Columns.Add("colFecha", "Fecha De Ingreso");
    dgtvList.Columns.Add("colUnVehiculo", "Patente");
    dgtvList.Columns.Add("colProblema", "Problema");
    dgtvList.Columns.Add("colCosto", "Costo");
    dgtvList.Columns.Add("colDiagnostico", "Diagnostico");
}
```

Interfaces

Realice dos interfaces genéricas

IAbm

```
/// <summary>
/// Esta interfaz contiene las fucionalidades para realizar un Abm
/// </summary>
/// <typeparam name="T"></typeparam>
1 referencia
public interface IAbm<T> where T : class
{
    /// <summary>
    /// Permite dar de alta un elemento del tipo pasado por parametro
    /// </summary>
    /// <returns>(false) en caso de uqe no se haya podido dar de alta,(true) de caso contratio</returns>
    6 referencias
    public bool Alta();

    /// <summary>
    /// Permite Modificacar un elemento del tipo pasado por parametro
    /// </summary>
    /// <returns>(false) en caso de uqe no se haya podido dar de Modificacion,(true) de caso contratio</returns>
    6 referencias
    public bool Modificacion(T element);

    /// <summary>
    /// Permite Eliminar un elemento del tipo pasado por parametro
    /// </summary>
    /// <returns>(false) en caso de uqe no se haya podido dar de Modificacion,(true) de caso contratio</returns>
    6 referencias
    public bool Baja(T element);

    /// <summary>
    /// Permite Mostrar los datos de un elemento del tipo pasado por parametro
    /// </summary>
    /// <returns>(false) en caso de uqe no se haya podido dar de Modificacion,(true) de caso contratio</returns>
    7 referencias
    public bool Mostrar(T element);
}
```

Esta la implemento en una clase de formulario donde realizo el abm de servicios y cliente (FrmListar),(FrmServicio),(FrmClientes)

Otra Interface es IConeccionABaseDeDatos

```
/// <summary>
/// Esta interfaz contiene las funcionalidades para el menejo de datos atravez de la base de datos
/// </summary>
/// <typeparam name="T"></typeparam>
4 referencias
public interface IConeccionABaseDeDatos<T>
    where T : class
{
    /// <summary>
    /// Lee los datos de la base de datos y los guarda dentro de una lista
    /// </summary>
    /// <returns>(List<T>) la lista con los datos</returns>
    /// <exception cref="ConeccionBaseDeDatosException"></exception>
    10 referencias
    public List<T> Leer();

    /// <summary>
    /// Permite obtener un elemento del tipo indicado obteniendo los dadoas que guarda el SqlDataRead
    /// </summary>
    /// <param name="dataReader">Dataread con los Datos</param>
    /// <returns>(null) en caso de que no se pueda lee los datos</returns>
    10 referencias
    public T ObtenerUnElemento(SqlDataReader dataRead);

    /// <summary>
    /// Agrega los Datos De la lista en la base de datos
    /// </summary>
    /// <param name="list">(List<T>) los datos a guardar</param>
    /// <returns>(true) si se pudo gurdar los datos, (false) de caso contrario</returns>
    8 referencias
    public bool Agregar(List<T> lista);
```

```
    /// <summary>
    /// Agrega un unElemento a la base de datos
    /// </summary>
    /// <param name="unElemento">el unElemento a agregar</param>
    /// <returns>(true) si lo pudo agragar,(false) si no lo pudo agragar</returns>
    /// <exception cref="ConeccionBaseDeDatosException"></exception>
    8 referencias
    public bool Agregar(T unElemento);
}
```

Esta interfaz contiene las funcionalidades para el manejo de datos atreves de la base de datos

Donde la implemento en las clases que se encargan del manejo de datos (ClienteDao), (UsuarioDao), (VehiculoDao),(ServicioDao)

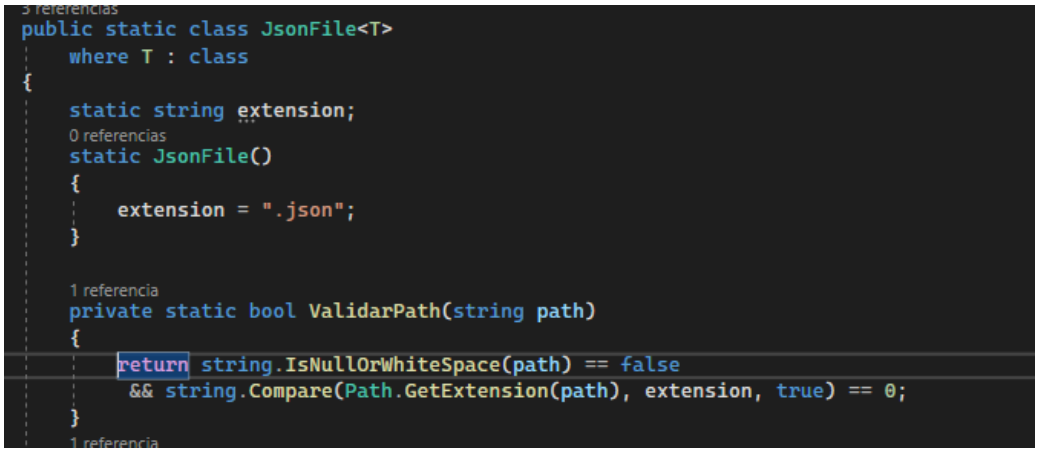
```
2 referencias
public bool Agregar(List<Cliente> list)
{
    bool estado;
    estado = false;
    if (list is not null && list.Count > 0)
    {
        estado = true;
        foreach (Cliente element in list)
        {
            try
            {
                Agregar(element);
            }
            catch (ConeccionBaseDeDatosException)
            {
                throw;
            }

        }
    }

    return estado;
}
```

Archivos Y Serializacion

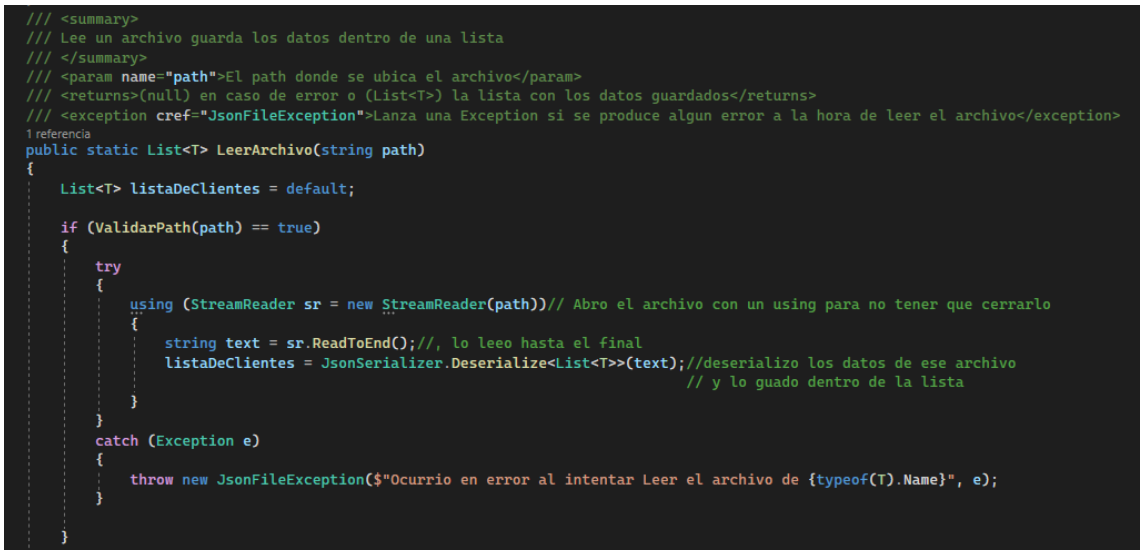
Implente una clase estatica llamada (JsonFile<T>) , la cual es genérica y contiene métodos que se encargan de cargar y serializar los datos en archivo json



```
3 referencias
public static class JsonFile<T>
    where T : class
{
    static string extension;
    0 referencias
    static JsonFile()
    {
        extension = ".json";
    }

    1 referencia
    private static bool ValidarPath(string path)
    {
        return string.IsNullOrWhiteSpace(path) == false
         && string.Compare(Path.GetExtension(path), extension, true) == 0;
    }
    1 referencia
```

A la hora de leer lo que hago es



```
/// <summary>
/// Lee un archivo guarda los datos dentro de una lista
/// </summary>
/// <param name="path">El path donde se ubica el archivo</param>
/// <returns>(null) en caso de error o (List<T>) la lista con los datos guardados</returns>
/// <exception cref="JsonFileException">Lanza una Exception si se produce algun error a la hora de leer el archivo</exception>
1 referencia
public static List<T> LeerArchivo(string path)
{
    List<T> listaDeClientes = default;

    if (ValidarPath(path) == true)
    {
        try
        {
            using (StreamReader sr = new StreamReader(path))// Abro el archivo con un using para no tener que cerrarlo
            {
                string text = sr.ReadToEnd();//, lo leeo hasta el final
                listaDeClientes = JsonSerializer.Deserialize<List<T>>(text);//deserializo los datos de ese archivo
                                                                            // y lo guado dentro de la lista
            }
        }
        catch (Exception e)
        {
            throw new JsonFileException($"Ocurrio en error al intentar Leer el archivo de {typeof(T).Name}", e);
        }

    }
```

A la hora De Guardar

```
/// <summary>
/// Guarda los datos de la lista dentro de un archivo
/// </summary>
/// <param name="path">el path donde se va a guardar el archivo</param>
/// <param name="list">la lista con los datos que se quiere guardar</param>
/// <returns>(true) si se pudo guardar los datos ,(false) de caso contrario</returns>
/// <exception cref="JsonFileException">Lanza una Exception si se produce algun error a la hora de Guardar el archivo</exception>
1 referencia
public static bool GuardarArchivo(string path, List<T> list)
{
    bool result = false;

    if (!string.IsNullOrWhiteSpace(path))
    {
        try
        {
            using (StreamWriter sw = new StreamWriter(path))// Creo o Abro el archivo con un using para no tener que cerrarlo
            {
                string texto = JsonSerializer.Serialize(list);//Seliazo los datos de la lista a json y los guardo
                sw.WriteLine(texto);//y los escrivo dentro del archivo
                result = true;
            }
        }
        catch (Exception e)
        {
            throw new JsonFileException($"Ocurrio un error al intentar Serializar el archivo de {typeof(T).Name}", e);
        }
    }
```

Y a la hora de usarlo, lo uso en (FrmListar)

```
private void BtnCargar_Click(object sender, EventArgs e)
{
    List<T> unLista;
    if ((this.openFileDialog = AbrirArchivo("Cargar archivo", "Archivo Json (*.Json)|*.Json",
     Environment.GetFolderPath(Environment.SpecialFolder.Desktop))) is not null)
    {  ///Le Pido al usurio que seleccione que archio quiere leer
        try
        {
            if ((unLista = JsonFile<T>.LeerArchivo(this.openFileDialog.FileName)) is not null)// le paso el path al metodo
                //que lee el archivo
            {
                ActualizarDataGried(dgtvList, listGeneric);
                this.OnInformar("Abrir Archivo", "Se cargo el archivo correctamente");
            }
        }
        catch (Exception ex)
        {
            OnInformarError("Abrir Archivo", ex.Message);
        }
    }
}
```

```
1 referencia
private void BtnGuardarArchivo_Click(object sender, EventArgs e)
{
    if ((this.saveFileDialog = GuardarArchivo("Guardar archivo", "Archivo Json (*.Json)|*.Json",
        Environment.GetFolderPath(Environment.SpecialFolder.Desktop))) is not null)
    {
        ///Le Pido al usurio que seleccione donde quiere guardar el arhivo
        try
        {
            JsonFile<T>.GuardarArchivo(this.saveFileDialog.FileName, listGeneric);//Le paso el path y lo guardo
            OnInformar("Guardar archivo", "los datos se Guardaron correctamente ");
        }
```

Conexión a bases de datos

Para la conexión de base de datos lo que hice fue crear 4 clases que cada una se encarga de los datos de una clase en concreto.

Y la hora de manejar los datos lo que hago es leerlos de la base de datos cuando se inicia la aplicación y una vez que se cierra la aplicación guardo los datos que se actualizaron

Guardar Datos

```
/// <summary>
/// Guarda los datos de la instancia en la base de datos
/// </summary>
/// <returns>(false) en caso de que no haya podido guardar los datos , (true) si pudo gurdar los datos</returns>
1 referencia
public bool GuardarBaseDeDatos()
{
    bool estado;
    estado = false;
    try
    {
        new ServicioDao().Agregar(listaDeServicio);
        new VehiculoDao().Agregar(listaDeVehiculos);
        new ClienteDao().Agregar(this.Clientes);
        new UsuarioDao().Agregar(this.listaDeUsuarios);
        estado = true;
    }
    catch (ConeccionBaseDeDatosException)
    {
        throw;
    }

    return estado;
}
```

Carga los datos

```
/// <summary>
/// Carga los datos de la base de datos y los guada dentro de una variable de tipo negocio y la devuelve
/// </summary>
/// <param name="nombre"></param>
/// <returns>(NULL) en caso de que no pudo cargar los elementos,(Negocio) de caso contrario</returns>
1 referencia
public static Negocio CargarBaseDeDatos(string nombre)
{
    Negocio unNegocio = default;
    try
    {
        unNegocio = new Negocio(nombre,new UsuarioDao().Leer(), new ClienteDao().Leer(),
            new VehiculoDao().Leer(), new ServicioDao().Leer());
    }
    catch (Exception)
    {
        throw;
    }

    return unNegocio;
}
```

Implemente un método que agrega un elemento a la base de datos

```
public bool Agregar(Cliente unElemento)
{
    bool estado;
    estado = false;
    try
    {
        comando.Parameters.Clear();// limpio la lista de parametros
        comando.Parameters.AddWithValue("@email", unElemento.Email);
        comando.Parameters.AddWithValue("@clave", unElemento.Clave);
        comando.Parameters.AddWithValue("@nombre", unElemento.Nombre);
        comando.Parameters.AddWithValue("@fechaDeNacimiento", unElemento.FechaDeNacimiento.ToString("d/MM/yy"));
        comando.Parameters.AddWithValue("@dni", unElemento.Dni);
        comando.Parameters.Add(SetValueSqlParameter("@path", unElemento.Path));
        coneccionSql.Open();//abro la coneccion a la base de datos
        comando.CommandText = "INSERT INTO Cliente(nombre,email,clave,dni,path,fechaDeNacimiento) " +
            "Values(@nombre,@email,@clave,@dni,@path,@fechaDeNacimiento)";//escrivo el comando que se va ejecutar para agregar un cliente

        if (comando.ExecuteNonQuery() == 1)//Ejecuto el comando y verifico si se pudo ejecutar
        {
            estado = true;
        }
    }
```

```
if (coneccionSql.State == ConnectionState.Open)//verifico si la conneccion a la bas de datos esta abierta
{
    coneccionSql.Close();//Cierro la coneccion
}
```

Implemente un método que lee los elementos de la base de datos

```
try
{
    coneccionSql.Open();
    comando.CommandText = $"Select * From Cliente";//Escribo el comando para traemrme la tabla de clientes con todos los datos

    using (SqlDataReader dataReader = comando.ExecuteReader())//Ejecuto el comando y los guardo dento de SqlDataReader
    {
        list = new List<Cliente>();
        while (dataReader.Read())//leeo los datos que se almacenaron
        {
            list.Add(ObtenerUnElemento(dataReader));
        }
    }
}
catch (Exception e)
{
    throw new ConeccionBaseDeDatosException("Ocurrio un problema al intentar obtener los archivos de la base de datos", e.InnerEx
}
```

Y un método que obtener los datos guardados en el SqlDataRead

```
/// <summary>
/// Permite obtener un elemento del tipo cliente obteniendo los dadoas que guarda el SqlDataRead
/// </summary>
/// <param name="dataReader">Dataread con los Datos Cliente</param>
/// <returns>(null) en caso de que no se pueda lee los datos</returns>
3 referencias
public Cliente ObtenerUnElemento(SqlDataReader dataReader)
{
    return new Cliente(Convert.ToInt32(dataReader["ID"]), Convert.ToString(dataReader["nombre"]), Convert.ToString(dataReader["dni"]),
        Convert.ToDateTime(dataReader["fechaDeNacimiento"]),Convert.ToString(dataReader["email"]), Convert.ToString(dataReader["clave"]));
}
```

Lo mismo hice con las demás 3 clases

Task

Implemente un formulario donde le permito al personal atender servicios (FrmTareas)

Cuando atiendo un Servicio le pido que le de diagnóstico y un costo y una vez ingresado simulo un arreglo del vehículo y este arreglo lo simulo atreves de un ProgressBar y para permitir que avance el ProgressBar , lo hago en un hilo secundario y le permito al usuario mientras se esta arreglando el vehículo siga interactuando con la aplicación.

```
/// <summary>
/// Activa el ProgressBar var en un hilo secundario
/// </summary>
/// <param name="barra"></param>
/// <param name="unServicio"></param>
/// <returns></returns>
1 referencia
public bool ActivarProgessBar(ProgressBar barra, Servicio unServicio)
{
    bool estado;
    estado = false;
    if (unServicio is not null && (tarea is null || tarea.IsCompleted == true || cancellationTokenSource.IsCancellationRequested == true))
    {
        estado = true;
        cancellationTokenSource = new CancellationTokenSource();
        tarea = Task.Run(() => AvanzarProgessBar(barra, unServicio), cancellationTokenSource.Token);

    }
    return estado;
}
```

```
/// <summary>
/// Incrementar ProgressBar hasta que llegue al final o hasta que se cancele el hilo
/// </summary>
/// <param name="barra">la barra que queremos incrementar</param>
/// <param name="unServicio"></param>
1 referencia
private void AvanzarProgessBar(ProgressBar barra, Servicio unServicio)
{
    Random random = new Random();

    if (barra is not null && unServicio is not null)
    {
        do
        {
            Thread.Sleep(random.Next(100, 1000));
            IncrementarProgessBar(barra);
        } while (barra.Value < barra.Maximum && cancellationTokenSource.IsCancellationRequested == false);

        ActualizarBoton(this.btnIniciarServicio);
        cancellationTokenSource = null;
        if (barra.Value >= barra.Maximum && this.unNegocio - unServicio)
        {
            FrmMenuPrincipal.Informar("Se Compelto el servicio", "Se Compelto el servicio correctamente");
        }
        this.ActualizarDataGriedView(base.dgtvList, unNegocio.ServiciosEnProcesos);
        InicializarProsgerBar(barra);
    }
}
```

```
/// <summary>
/// Incrementa el ProgressBar y verifica si esta el hilo principal o en otro
/// </summary>
/// <param name="barra">barra de progeso a incrementar</param>
2 referencias
private void IncrementarProgessBar(ProgressBar barra)
{
    if (InvokeRequired)//Verifica si esta en un hilo distinto al principal
    {
        Invoke(() => IncrementarProgessBar(barra));//Vuelve a invocar la funcion dentro de otro hilo
    }
    else
    {
        barra.Increment(1);//incrementa la barra
    }
}
```

Tambien le permito cancelar el arreglo

```
/// <summary>
/// Cuando se presiona el boton cancela el servicio
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 referencia
private void BtnCancelarServicio_Click(object sender, EventArgs e)
{
    if(cancellationTokenSource is not null)
    {
        cancellationTokenSource.Cancel();
        btnCancelarServicio.Enabled = false;
    }
}
```

Eventos

Utilizo este tema en (FrmListar<T>),(FrmAltaDeCliente),(FrmAltaServicio)

(FrmListar<T>**)**

```
/// <summary>
/// Informa si sucecedio un error en la aplicacion y muestra un mensaje indicando que error sucedio
/// </summary>
protected event Action<string, string> InformarError;
/// <summary>
/// Informa si sucecedio algun suceso en la aplicacion y muestra un mensaje indicando que suceso sucedio
/// </summary>
protected event Action<string, string> Informar;

/// /// <summary>
///Permite buscar un elemento en la lista
/// </summary>
/// <param name="mensaje">El texto que se escribio en el text box</param>
/// <param name="unaLista">la lista donde se va a busca el texto escrito</param>
/// <returns>la lista con los elementos que concide con el texto ingresado</returns>
protected event Func<string, List<T>, List<T>> Buscador;

/// <summary>
/// Permite filtrar elementos de una lista
/// </summary>
/// <param name="lista">la lista donde se va a busca elementos por el filtro pedido</param>
/// <param name="nombreDelFiltro">el filtro ingresado</param>
/// <returns>la lista con los elementos que concide con el filtro ingresado</returns>
protected event Func<List<T>, string, List<T>> Filtrar;
```

Y estos eventos los llamo atreves de un método

```
/// <summary>
/// Permite invocar al evento InformarError pasandole los parametros , verificando que los parametros pasados sean validos y que el evento
/// este referenciado a un metodo
/// </summary>
/// <param name="titulo"></param>
/// <param name="mensaje"></param>
/// <returns>(false) si se cumplieron las condiciones ,(true) si se se pudo inv</returns>
3 referencias
private bool ManejadorInformarError(string titulo, string mensaje)
{
    bool estado;
    estado = false;
    if (this.InformarError is not null)
    {
        this.InformarError(titulo, mensaje);
        estado = true;
    }

    return estado;
}
```

Donde lo uso

```
1 referencia
private void BtnCargar_Click(object sender, EventArgs e)
{
    List<T> unLista;
    if ((this.openFileDialog = AbrirArchivo("Cargar archivo", "Archivo Json (*.Json)|*.Json",
      Environment.GetFolderPath(Environment.SpecialFolder.Desktop))) is not null)
    {
        try
        {
            if ((unLista = JsonFile<T>.LeerArchivo(this.openFileDialog.FileName)) is not null)
            {
                ActualizarDataGried(dgtvList, listGeneric);
                this.ManejadorInformar("Abrir Archivo", "Se cargo el archivo correctamente");
            }
        }
        catch (Exception ex)
        {
            ManejadorInformarError("Abrir Archivo", ex.Message);
        }
    }
}
```

(FrmAltaServicio**)**

```
/// <summary>
/// Se Activa cuando se da de alta correctamente un servicio
/// </summary>
/// <param name="unServicio">los datos del servicio que se ingresaron</param>
/// <param name="unVehiculo">los datos del vehiculo que se ingresaro</param>
public event Action<Servicio, Vehiculo> OnSeIngesaronDatos;
```

Y a la hora de invocarlo hice lo mismo

```
/// <summary>
/// Permite invocar al evento seIngesaronDatos pasandole los parametros
/// , verificando que los parametros pasados sean validos y que el evento
/// este referenciado a un metodo
/// </summary>
/// <param name="titulo"></param>
/// <param name="mensaje"></param>
/// <returns>(false) si se cumplieron las condiciones ,(true) si se se pudo invocar al metodo</returns>
1 referencia
private void ManejadorSeIngesaronDatos(Servicio unServicio, Vehiculo unVehiculo)
{
    if (OnSeIngesaronDatos is not null)
    {
        OnSeIngesaronDatos.Invoke(unServicio, unVehiculo);
    }
}
```

Lo mismo hice en el **(** FrmAltaDeCliente**).**

Métodos de extensión

Hice una clase (`StringExtended`)

```
/// <summary>
/// Esta clase extiende la clase string , agragandole funcionalidades que manejan los caracteres del string
/// </summary>
0 referencias
public static class StringExtended
{
    /// <summary>
    /// Verifica si el string contiene los elementos del array de char pasado por parametro
    /// y los Borra los caracteres
    /// </summary>
    /// <param name="texto"></param>
    /// <param name="charsABorrar"></param>
    /// <returns>(string) el string con los caracteres modificados , (NULL) en caso de que los parametros no sean validos</returns>
    5 referencias
    public static string BorrarCaracteres(this string texto, char[] charsABorrar)
    {
        string result = new string(texto);
        if (charsABorrar is not null && charsABorrar.Length > 0)
        {
            foreach (char caracterDeLaLista in charsABorrar)
            {
                if (texto.Contains(caracterDeLaLista) == true)
                {
                    result = result.Replace(caracterDeLaLista.ToString(), "");
                }
            }
        }
se encontraron problemas.
```

```
/// <summary>
/// Verifica si el string cumple con el criterio pasado por parametro
/// </summary>
/// <param name="texto"></param>
/// <param name="criterio"></param>
/// <returns>(true) si cumple con el criterio,(false) de caso contrario</returns>
4 referencias
private static bool VerificarString(this string texto, Predicate<char> criterio)
{
    bool result;
    result = false;


    if (texto is not null && texto.Length > 0 && criterio is not null)
    {
        result = true;
        foreach (char caracter in texto)
        {
            if (criterio(caracter) == false)
            {
                result = false;
                break;
            }
        }
    }

    return result;
}
```

```
/// <summary>
/// Verifica si el string pasado por parametro contiene algun caracter que no sea una letra
/// </summary>
/// <param name="texto"></param>
/// <returns>(true) si contiene solo letras,(false) de caso contrario</returns>
2 referencias
public static bool isLetter(this string texto)
{
    return texto.VerificarString(char.IsLetter);
}
```

Delegados

(StringExtended)

```
/// <summary>
/// Verifica si el string cumple con el criterio pasado por parametro
/// </summary>
/// <param name="texto"></param>
/// <param name="criterio">el criterio que va a cumplir</param>
/// <returns>(true) si cumple con el criterio,(false) de caso contrario</returns>
4 referencias
private static bool VerificarString(this string texto, Predicate<char> criterio)
{
    bool result;
    result = false;


    if (texto is not null && texto.Length > 0 && criterio is not null)
    {
        result = true;
        foreach (char caracter in texto)
        {
            if (criterio(caracter) == false)
            {
                result = false;
                break;
            }
        }
    }

    return result;
}
```

(FrmMenuPrincipal),

```
/// <summary>
/// Muestra un Control informando que criterio debe cumplir el elemento pasado por parametro
/// </summary>
/// <param name="unControl">el control que va a mostrar el error</param>
/// <param name="mensaje">Mensaje informando que criterio debe cumplir el control</param>
/// <param name="predicate">el metodo que va a derminar si se cumplieron las criterio , que debe
/// <param name="element">el elemento a verificar</param>
/// retornar (True) en caso que se cumpla o (false) de caso contrario</param>
/// <returns>(false) en caso de que el elemento no se cumpla con la criterio de el metodo pasado por parametro,
/// de lo contrario devueve (true)</returns>
17 referencias
public static bool ActivarControlError<T>(Control unControl, string msgError, Predicate<T> predicate, T element)
{
    bool estado;
    estado = false;
    if (unControl is not null && predicate is not null)
    {
        unControl.Visible = true;
        unControl.Text = msgError;
        if ((estado = predicate.Invoke(element)) == true)
        {
            estado = true;
            unControl.Visible = false;
        }
    }
```

Como lo invoco (FrmAltaCliente)

```
= FrmMenuPrincipal.ActivarControlError(lb_Fallas, "No se aceptan valores vacios", FrmMenuPrincipal.DetectarTextBoxVacio, this.Controls) == true
 FrmMenuPrincipal.ActivarControlError(lb_Fallas,"el dni debe tener como min 6 y max 8 numeros", Cliente.ValidarDni, this.txtDni.Text)
FrmMenuPrincipal.ActivarControlError(lb_Fallas, "La fecha no es valida", Cliente.ValidarFechaDeNacimiento, this.DateFechaDeNacimiento.Value) ==
FrmMenuPrincipal.ActivarControlError(lb_Fallas,"el Nombre Debe Contener solo letras", Cliente.ValidarNombre, this.txtNombre.Text) == true
FrmMenuPrincipal.ActivarControlError(lb_Fallas,"el Email Debe tener como min 8 caracteres", Cliente.ValidarEmail, this.txtEmail.Text)
FrmMenuPrincipal.ActivarControlError(lb_Fallas,"el Clave Debe tener como min 8 caracteres", Cliente.ValidarContracenia, this.txtClave.Text);

esult == true)
```

(FrmMostrar)

```
/// <summary>
/// crea una la lista de propiedades guardando los elementos que cumplen con el criterio pasado por parametro
/// </summary>
/// <param name="propertyInfos">ela lista de propiedades</param>
/// <param name="predicate">el Criterio a cumplir</param>
/// <returns>la lista de propiedades con los elementos que cumplen con el criterio pasado por parametro </returns>
2 referencias
private static List<PropertyInfo> ObtenerPropiedades(PropertyInfo[] propertyInfos, Predicate<PropertyInfo> predicate)
{
    List<PropertyInfo> result = new List<PropertyInfo>();

    if (predicate is not null && propertyInfos is not null)
    {
        foreach (PropertyInfo unProperty in propertyInfos)
        {
            if (predicate.Invoke(unProperty) == true)
            {
                result.Add(unProperty);
            }
        }
    }

    return result;
}
```