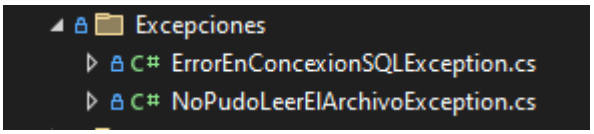
Descripción de la Aplicación

Terapias Horus es un gestor de consultas para terapias alternativas, se puede cargar nuevos pacientas y generar consultas que deben de tener un paciente asignado y una práctica asignas. SE pueden generar consultas cuanto lleva gastado cada paciente o el total de los ingreso generados.

Unidad 10 - Excepciones



Se generaron 2 excepciones para el programa, una para conocer is hay errores en la conexión a la base de datos y otra en caso de no poder encontrar el archivo para leer

Unidad 11 - Pruebas Unitarias

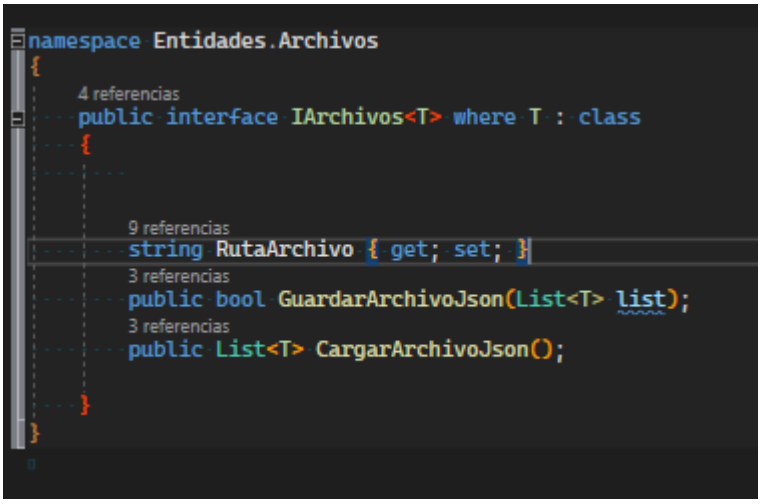
```
[TestMethod]
0 referencias
public void AlInvocarElMetodoGetMaxIdDePaciente_SiTieneDatos_SeObtieneElUltimoIdDeLaLista()
{
    //Arrange
    List<Consulta> consultas = new List<Consulta>
    {
        new Consulta(1, "123", 1, 1),
        new Consulta(2, "123", 2, 2),
        new Consulta(3, "123", 3, 1),
        new Consulta(4, "123", 4, 2)
    };
    List<Practica> practicas = new List<Practica>();
    Practica p1 = new Practica();
    p1.IdPractica = 1;
    p1.PracticaNombre = "1";
    p1.Precio = 10;
    Practica p2 = new Practica();
    p2.IdPractica = 2;
    p2.PracticaNombre = "2";
    p2.Precio = 25;
    practicas.Add(p1);
    practicas.Add(p2);
    //Act
    int resultado = Consulta.PrecioTotalGastado(consultas, practicas);
    //Assert
    Assert.AreEqual(70 , resultado);
}
```

Implementado en 2 funciones de la clase Consulta.

Unidad 12 - Generics

implementadas junto a Interfaces

Unidad 13 - Interfaces



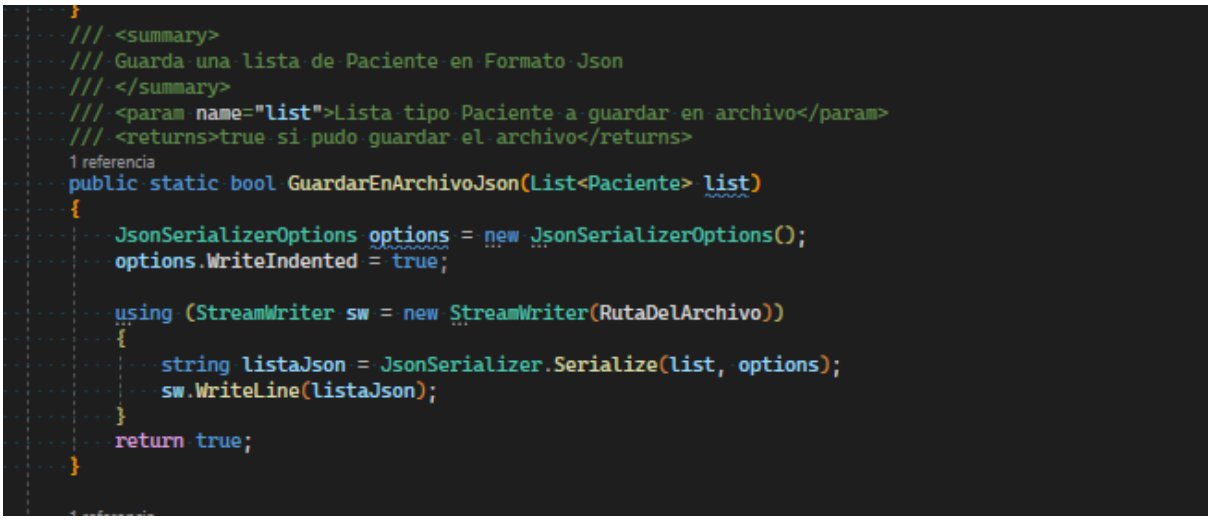
```
namespace Entidades.Archivos
{
    4 referencias
    public interface IArchivos<T> where T : class
    {


        9 referencias
        string RutaArchivo { get; set; }
        3 referencias
        public bool GuardarArchivoJson(List<T> list);
        3 referencias
        public List<T> CargarArchivoJson();

    }
}
```

Se implemento una interfaz para que los distintos tipos de clase puedan guardar archivos

Unidad 14 - Archivos y Serialización

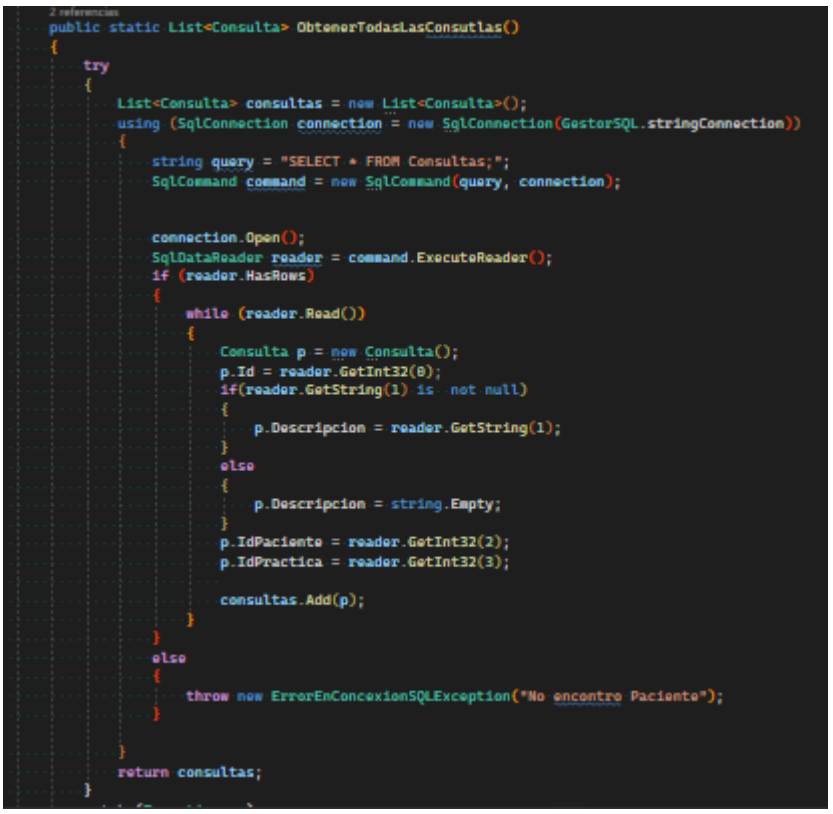


```
    /// <summary>
    /// Guarda una lista de Paciente en Formato Json
    /// </summary>
    /// <param name="list">Lista tipo Paciente a guardar en archivo</param>
    /// <returns>true si pudo guardar el archivo</returns>
    1 referencia
    public static bool GuardarEnArchivoJson(List<Paciente> list)
    {
        JsonSerializerOptions options = new JsonSerializerOptions();
        options.WriteIndented = true;

        using (StreamWriter sw = new StreamWriter(RutaDelArchivo))
        {
            string listaJson = JsonSerializer.Serialize(list, options);
            sw.WriteLine(listaJson);
        }
        return true;
    }
```

Se pueden serializar listas de las clases y guardarlas en archivos de tipo .json

Unidad 15 - SQL y Conección a BDD



```
2 referencias
public static List<Consulta> ObtenerTodasLasConsutlas()
{
    try
    {
        List<Consulta> consultas = new List<Consulta>();
        using (SqlConnection connection = new SqlConnection(GestorSQL.stringConnection))
        {
            string query = "SELECT * FROM Consultas;";
            SqlCommand command = new SqlCommand(query, connection);


            connection.Open();
            SqlDataReader reader = command.ExecuteReader();
            if (reader.HasRows)
            {
                while (reader.Read())
                {
                    Consulta p = new Consulta();
                    p.Id = reader.GetInt32(0);
                    if(reader.GetString(1) is not null)
                    {
                        p.Descripcion = reader.GetString(1);
                    }
                    else
                    {
                        p.Descripcion = string.Empty;
                    }
                    p.IdPaciente = reader.GetInt32(2);
                    p.IdPractica = reader.GetInt32(3);

                    consultas.Add(p);
                }
            }
            else
            {
                throw new ErrorEnConcexionSQLException("No encontro Paciente");
            }

        }
        return consultas;
    }
```

Se implementa en archivos SQL para cada clase, los distintos CRUD la tabla de la correspondiente de la BDD
Recordar recuperar el archivo dentro de git llamado DBs_Backup en la carpeta principa.

Unidad 16 - Delegados y Expresiones Lambda

Se utiliza en el evento

Unidad 17 - Programación Multihilo y Concurrencia

Se utiliza en el evento

Unidad 18 - Eventos

```
//Evento
public static event DelegadoGuardarArchivo OnGuardarArchivoEnJson;
```

```
1 referencia
public static bool GuardarEnArchivoJson(List<Paciente> list)
{
    if(list is not null)
    {
        Thread.Sleep(1000);
        JsonSerializerOptions options = new JsonSerializerOptions();
        options.WriteIndented = true;

        using (StreamWriter sw = new StreamWriter(RutaDelArchivo))
        {
            string listaJson = JsonSerializer.Serialize(list, options);
            sw.WriteLine(listaJson);
        }
        Paciente.OnGuardarArchivoEnJson.Invoke("Se Guardó correctamente");
        return true;
    }
    Paciente.OnGuardarArchivoEnJson.Invoke("No se pudo guardar el archivo");
    return false;

}
```

```
1 referencia
public FrmPacientes(List<Paciente> listaPacientes)
{
    InitializeComponent();
    this.listaPacientes = listaPacientes;
    Paciente.OnGuardarArchivoEnJson += this.MostrarGuardadoOK;

}
```

Unidad 20 - Métodos de Extensión