El programa consiste en un análisis de datos de un grupo dinámico de alumnos, comienza cargando datos de un archivo XML, los cuales son de serializados y asignados a una lista que vamos a utilizar para filtrar por diferentes valores (Genero, Sector, Turno y Orientación), podemos ingresar mas alumnos que se acoplen a dicha filtración.

En el apartado de filtración podemos elegir si mostrar la filtración en un archivo CSV, en la pantalla o en ambos, y sumado a esto podemos elegir guardar esa filtración en un archivo XML y/o JSON. Las filtraciones pueden contemplar mas de un parámetro a la vez, esto quiere decir que podemos filtrar por un genero y una orientación al mismo.

Temas:

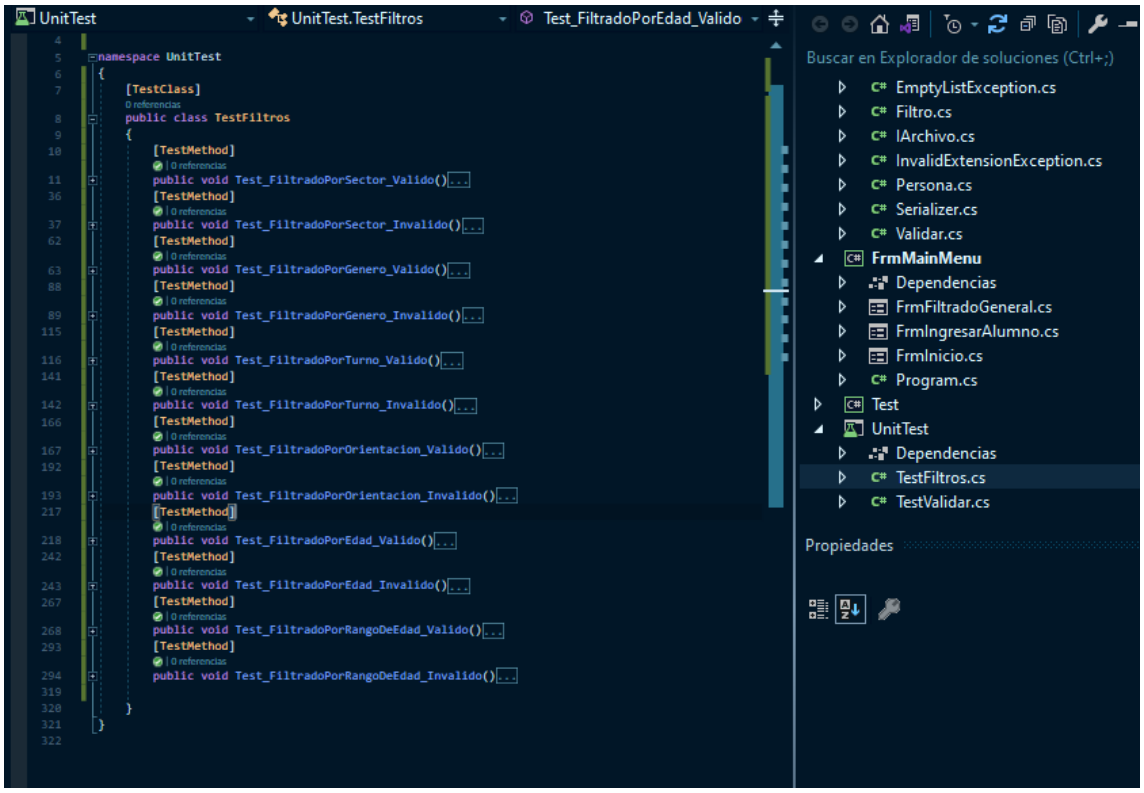
#Excepciones:

Las excepciones se utilizan en varias partes del código, algunos ejemplos:

```
{
    if (this.tipo == ETipo.XML)
    {
        if (Path.GetExtension(path) == ".xml")
        {
            using (XmlTextWriter tw = new XmlTextWriter(path, Encoding.UTF8))
            {
                tw.Formatting = Formatting.Indented;
                XmlSerializer xmlSer = new XmlSerializer(typeof(T));
                xmlSer.Serialize(tw, dato);
            }
        }
        else
        {
            throw new InvalidExtensionException("Extension invalida para tipo de archivo XML.");
        }
    }
    else...

}
/// <summary> Lee un dato de un archivo json o xml.
5 referencias
public T Leer(string path)
{
    T objeto = new T();
    if (this.tipo == ETipo.XML)
    {
        if (Path.GetExtension(path) == ".xml")
        {
            try...
        }
        else { throw new InvalidExtensionException("Extension invalida para tipo de archivo XML."); }
    }
    else
    {
        if (Path.GetExtension(path) == ".json")
        {

            try
            {
                string json = File.ReadAllText(path);
                objeto = JsonSerializer.Deserialize<T>(json);
                return objeto;
            }
            catch (Exception)
            {

                throw;
            }
        }
        else { throw new InvalidExtensionException("Extension invalida para tipo de archivo XML."); }

    }
}
```

#Tests Unitarios:

Los tests unitarios se encuentran en el proyecto UnitTest.



#Interfaces y Generics:

Se aplica interfaces en la Clase IArchivo, la cual es generica, dicha clase se utiliza en ArchivoText y Serializer, los cuales adoptan los métodos Leer y Escribir, Serializer permite elegir entre serializar y deserializar en formato XML o JSON.

#Archivos y Serialización:

Se utilizan en la clase ArchivoText y Serializer:

```
public void Escribir(T dato, string path)
{
    if (this.tipo == ETipo.XML)
    {
        if (Path.GetExtension(path) == ".xml")
        {
            using (XmlTextWriter tw = new XmlTextWriter(path, Encoding.UTF8))
            {
                tw.Formatting = Formatting.Indented;
                XmlSerializer xmlSer = new XmlSerializer(typeof(T));
                xmlSer.Serialize(tw, dato);
            }
        }
        else
        {
            throw new InvalidExtensionException("Extension invalida para tipo de archivo XML.");
        }
    }
    else...

}
/// <summary> Lee un dato de un archivo json o xml.
5 referencias
public T Leer(string path)
{
    T objeto = new T();
    if (this.tipo == ETipo.XML)
    {
        if (Path.GetExtension(path) == ".xml")
        {
            try...
        }
        else { throw new InvalidExtensionException("Extension invalida para tipo de archivo XML."); }
    }
    else
    {
        if (Path.GetExtension(path) == ".json")
        {

            try
            {
                string json = File.ReadAllText(path);
                objeto = JsonSerializer.Deserialize<T>(json);
                return objeto;
            }
            catch (Exception)
            {

                throw;
            }
        }
        else { throw new InvalidExtensionException("Extension invalida para tipo de archivo XML."); }

    }
}
```

```
namespace Entidades
{
    6 referencias
    public class ArchivoText : IArchivo<string>
    {
        /// <summary> Escribe un archivo de texto en una ruta indicada
        4 referencias
        public void Escribir(string dato, string path)
        {
            try
            {
                using (StreamWriter sw = new StreamWriter(path))
                {
                    sw.Write(dato);
                }
            }
            catch (Exception)
            {
                throw new Exception("Error al escribir el archivo.");
    namespace Entidades
        /// <summary> Lee un archivo de texto en la ruta indicada por
        2 referencias
        public string Leer(string path)
        {
            string aux = string.Empty;
            try
            {
                using (StreamReader sr = new StreamReader(path))
                {
                    while (!sr.EndOfStream)
                    {
                        aux += $"{sr.ReadLine()}";
                    }
                }
            }
            catch (Exception)
            {
                throw new Exception("Error al leer el archivo");
            }
            return aux;
        }
    }
```