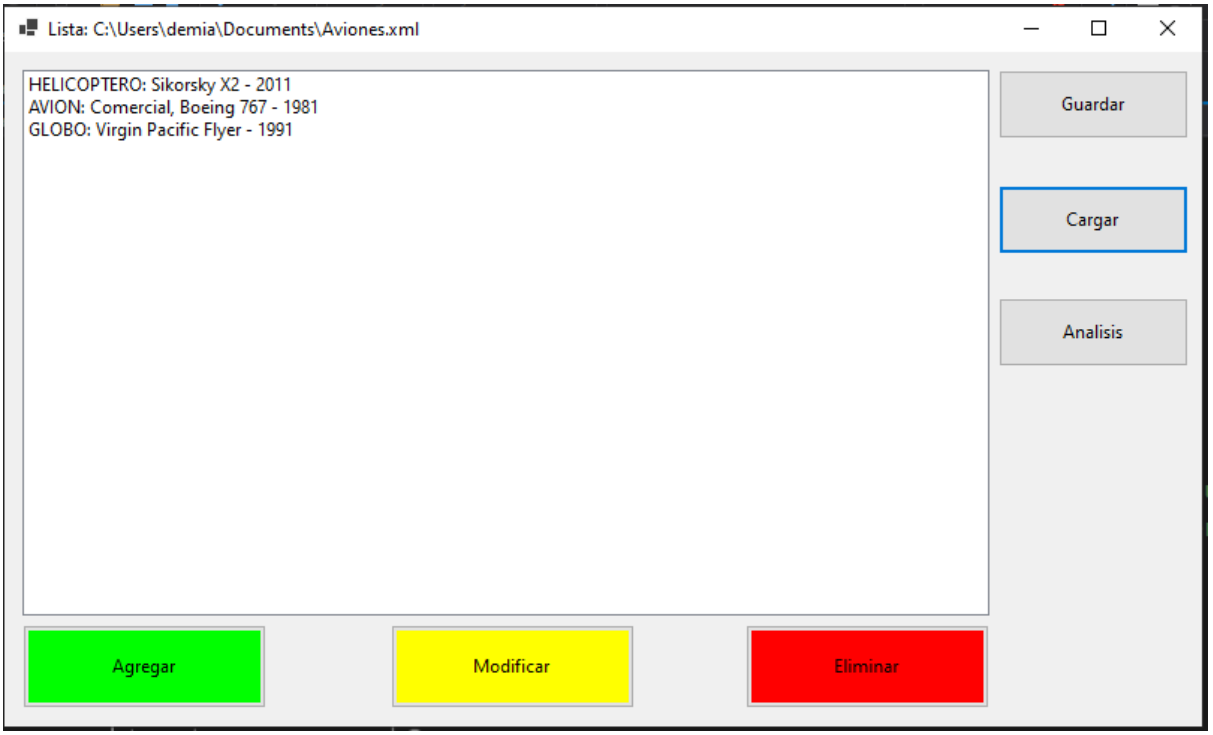
TP3, Panello Fausto



La función pretendida en este TP consiste en la creación de una lista de aeronaves, y luego poder comparar los datos de cada una de las mismas entre sí a medida que se van agregando nuevos elementos, además del promedio de los datos de cada tipo de aeronave. Subí también en el repositorio de GitHub un XML que hice anteriormente con datos realistas de varias aeronaves para que pueda ya cargarse y comparar.

A continuación, listo los temas de las clases 10 a 15 que están en el código:

10 - Excepciones: Utilizo excepciones, por ejemplo, en la parte de la serialización de XML, y también cuando intento convertir la variable escrita por el usuario de año y rotores de un helicóptero a string. (FormAgregarModificar.cs), (Lista.cs).

11 - Pruebas Unitarias: Creé un proyecto de UnitTest probando las funcionalidades de agregar y remover aeronaves de una lista. También verifiqué la igualdad de la sobrecarga de operadores en este unitTest. Lo hice de tipo Xunit porque estoy usando .NET 5.0, y no podía crear un proyecto de UnitTest normal en .NET 5.0.

12 - Tipos Genéricos: La clase Lista es completamente genérica, y se puede usar para cualquier tipo de dato que se quiera almacenar en una lista. (Lista.cs).

13 - Interfaces: Usé dos clases que dependen de una interfaz; la clase genérica Lista, en la que fuerzo a Lista a tener un Agregar y Remover, y también la clase Aeronave, en la que especifico todas las variables que tiene que tener una Aeronave. (Ilista.cs, IAeronave.cs).

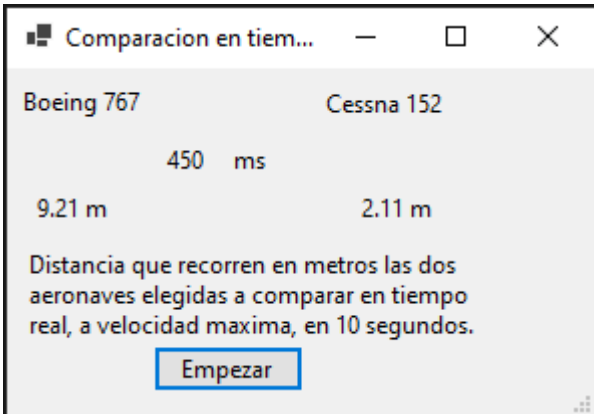
14 - Archivos y Serialización: En las funciones Save y Load de la clase Lista está toda la lógica para poder guardar una Lista dado un path, y para poder cargar una,

respectivamente. Arriba de cada una de estas funciones, comenté cómo funcionan detalladamente. (Lista.cs).

16 - Conexiones a bases de datos: La clase AccesoDatos proporciona todas las funciones necesarias para poder realizar conexiones con las bases de datos y las aeronaves. Hay diversas llamadas en el código a esas funciones para poder agregar, modificar, o eliminar de la base de datos. Además, adjunte un script para poder crear la database y las tablas necesarias para que funcione. (AccesoDatos.cs, FormAeronaves.cs, FormAgregarModificar.cs).

17 - Delegados: En el FormCarrera, creé un delegado para luego poder instanciarlo en el hilo, y crear la funcionalidad que tiene el mismo en este momento. (FormCarrera.cs).

18 - Hilos: En el FormCarrera, creé un hilo para poder realizar dos cuentas a la vez y en tiempo real de la distancia que podrían recorrer dos aeronaves dadas en 10 segundos, y poder compararlas. (FormCarrera.cs).



19 - Eventos: En FormCarrera, relaciono el evento btnEmpezar.Click con un Handler o Manejador para poder luego llamarlo y que corra el hilo. (FormCarrera.cs).

20 - Métodos de extensión: Creé un tipo de Exception llamado ListaLlenaException, que a su vez tiene un método de extensión ListaLlenaExceptionExtension, con una función Informar que recibe ListaLlenaException, así extendiendo la clase.

Notas: Eliminé la clase extra que había creado en el UnitTest que era el error que me habían marcado.

Agregué algunos datos como “% de Aeronaves de más de 1000 kg”, o “% de Aeronaves que rompen la barrera del sonido”, además de la nueva comparación que permite ver cuánto recorrieron dos aeronaves dadas en 10 segundos.