Prueba Ingreso - Backend

Reglas Generales

Se debe cumplir con el análisis, diseño e implementación de la solución Backend en el framework propuesto en el correo anexo a este documento. Los temas que se evaluarán son:

- Arquitectura de la solución
- Programación orientada a objetos
- Consumo REST API
- Control de versiones
- Acceso a bases de datos

Se valorará positivamente el uso de los siguientes conceptos:

- Logs de aplicación
- Control de excepciones
- Inyección de dependencias
- Test unitarios
- Manejo de ORM en acceso a datos
- Nomenclatura estándar
    - Referencia C#
    - Referencia JAVA

La entrega debe ser realizada como una solución alojada en un repositorio de código de su elección, debe ser de acceso público para facilitar la revisión de la misma, adjuntar un **README.txt** con toda la información pertinente para poder hacer funcionar correctamente la solución o todos aquellos temas que no logró implementar de manera correcta.

Enunciado

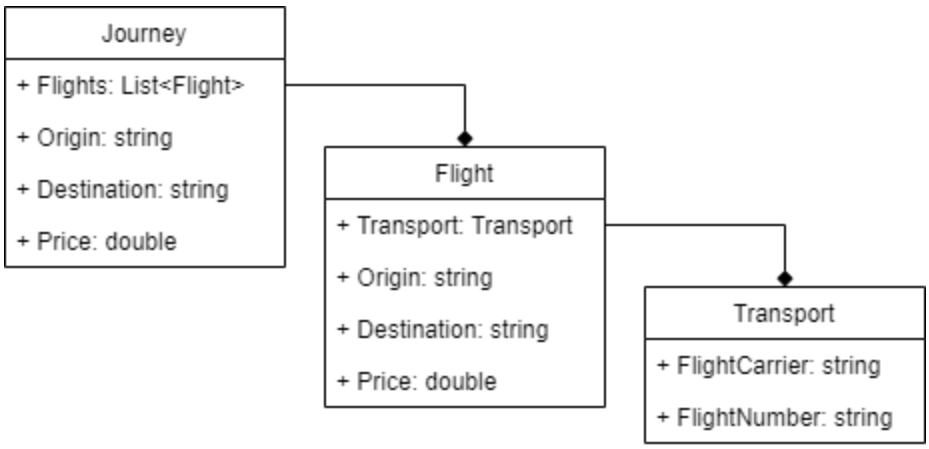
La empresa NEWSHORE AIR necesita una solución para poder conectar viajes a través del mundo, esta solución debe recibir como parámetros el origen y el destino de viaje del usuario, después el sistema debe consultar todos los vuelos asociados que tenga, y devolver la ruta de viaje al usuario si la ruta es posible o un mensaje avisando que la ruta no puede ser calculada, además al momento de calcular una ruta, esta debe ser guardada en algún sistema de persistencia para que si posteriormente se ingresan los mismos parámetros esta se pueda obtener fácilmente sin volver a calcularse.

Construya una solución API que pueda cumplir con el requerimiento anteriormente descrito, la solución debe tener al menos una división lógica de 3 capas.

- API
- Business
- DataAccess

Problema 1 – Modelado Clases

Como encargado de la solución deberá modelar un objeto estándar para trabajar con los vuelos en su aplicación, el modelo propuesto es el siguiente (diseño UML).



Problema 2 – Consumo REST API

Para poder calcular la ruta requerida por el usuario es necesario acceder a los vuelos asociados a NEWSHORE AIR, para este propósito se provee de una API que permita realizar la búsqueda de estos. Consuma la siguiente API (https://recruiting-api.newshore.es/api/flights) usando el método **GET** y use la clase **Flight** modelada en el **Problema 1** para mapear la respuesta.

Request	**Response**

Rutas únicas
https://recruiting-api.newshore.es/api/flights/0
 	
Rutas múltiples
https://recruiting-api.newshore.es/api/flights/1
 	
Rutas múltiples y de retorno
https://recruiting-api.newshore.es/api/flights/2
 	
Elija el nivel con el que desea trabajar, a mayor nivel el siguiente problema será más complejo de resolver, se valorará positivamente el uso de niveles altos.
 	
Tenga en cuenta que en cada consulta la respuesta es **estática** y no cambia en el tiempo.
	[
	{
	"DepartureStation": "MZL",
	"ArrivalStation": "JFK",
	"FlightCarrier": "AV",
	"FlightNumber": "8020",
	"Price": 1000.0,
	},
	{
	"DepartureStation": "JFK",
	"ArrivalStation": "BCN",
	"FlightCarrier": "AV",
	"FlightNumber": "8040",
	"Price": 1000.0,
	}
	]

Opcional: Optimizar la cantidad de peticiones que se hacen a esta API.

Problema 3 – Obtener Ruta

Se debe exponer un API en la cual el usuario pueda pasar sus parámetros de búsqueda. Exponga un API con un método **GET** que implemente el siguiente contrato.

Request	Response

{
"Origin": "MZL",
"Destination": "BCN"
}
	Utilice la clase **Journey** modelada en el **Problema 1** y cualquier otro campo que decida agregar para tener mas información del proceso de respuesta

Para calcular la respuesta debe utilizar como base los parámetros **Origin** y **Destination** que el usuario acaba de enviar, determine la lista de vuelos en orden para poder cumplir con dicha ruta, agréguelos al modelo respuesta y calcule el valor total **Price** de la ruta. Como ejemplo vea los siguientes parámetros y la respuesta esperada utilizando como base la respuesta del **Problema 2.**

Request	Response

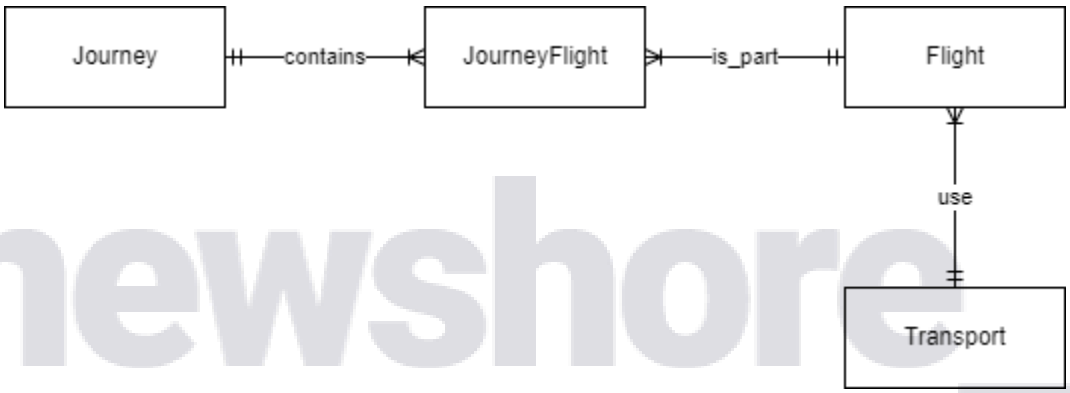
{
"Origin": "MZL",
"Destination": "BCN"
}
	{
	"Journey": {
	"Origin": "MZL",
	"Destination": "BCN",
	"Price": 2000.0,
	"Flights": [
	{
	"Origin": "MZL",
	"Destination": "JFK",
	"Price": 1000.0,
	"Transport": {
	"FlightCarrier": "AV",
	"FlightNumber": "8020"
	}
	},
	{
	"Origin": "JFK",
	"Destination": "BCN",
	"Price": 1000.0,
	"Transport": {
	"FlightCarrier": "AV",
	"FlightNumber": "8040"
	}
	}
	]
	}
	}

Para solucionar el problema debe buscar un vuelo cuyo **Origin** coincida con el de la consulta del usuario, al encontrarlo debe buscar entonces el vuelo siguiente a este, uno cuyo **Origin** coincida con el **Destination** del que acabo de encontrar, y esto sucesivamente hasta que pueda llegar al **Destination** de la consulta del usuario. Si no es posible calcular la ruta debe informar al usuario que su consulta no puede ser procesada.

Opcional: Agregar la posibilidad de configurar el número máximo de vuelos que puede tener una ruta.

Problema 4 – Persistencia/Acceso Datos

Por último, es necesario poder guardar y obtener rutas previamente consultadas. Al momento de recibir cualquier consulta verificar si la ruta ya fue calculada, si es de esta manera obtener la ruta de la BD, si no, hacer el calculo como en todo el flujo anterior del **Problema 2 y 3** y guardar el resultado en la BD, el modelo propuesto es el siguiente (modelo entidad-relación).



Opcional: En la creación de la base de datos incluir índices y constraints para los datos que considere necesarios.