

Prueba Ingreso - Fullstack

Reglas Generales

Se debe cumplir con el análisis, diseño e implementación de la solución Backend y Frontend 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
- Manejo de estilos en cascada CSS

Se valorará positivamente el uso de los siguientes conceptos:

- Logs de aplicación
- Control de excepciones
- Inyección de dependencias
- Test unitarios
- Uso de states manager
- Uso de interceptors
- Uso de injection tokens
- Nomenclatura estándar
 - [Referencia C#](#)
 - [Referencia JAVA](#)
 - [Referencia Angular](#)

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 web 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.

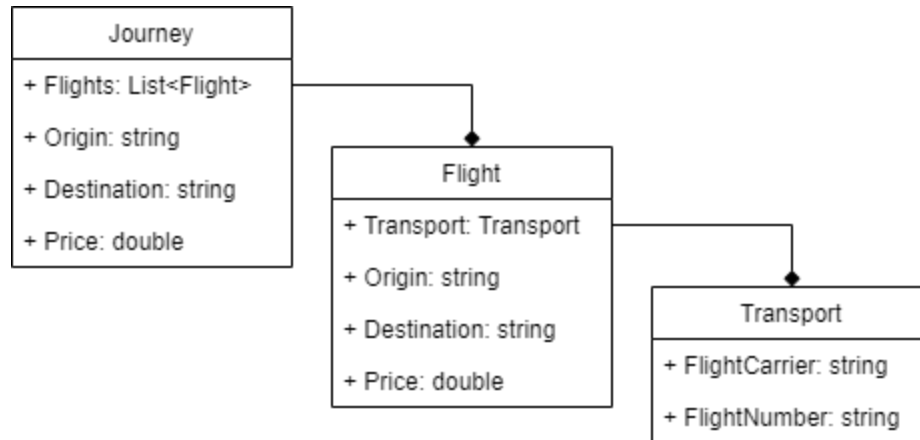
Construya una solución API que pueda cumplir con el requerimiento anteriormente descrito, la solución debe tener al menos una división lógica de 3 capas.

- API
- Business
- DataAccess

Construya una solución web que pueda cumplir con el requerimiento anteriormente descrito, consumiendo la solución API que usted haga.

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	<pre>[{ "DepartureStation": "MZL", "ArrivalStation": "JFK", "FlightCarrier": "AV", "FlightNumber": "8020", "Price": 1000.0, }, { "DepartureStation": "JFK", "ArrivalStation": "BCN", "FlightCarrier": "AV", "FlightNumber": "8040", "Price": 1000.0, }]</pre>
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.	

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

Problema 3 – Obtener Ruta Parte 1

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
<pre>{ "Origin": "MZL", "Destination": "BCN" }</pre>	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
<pre>{ "Origin": "MZL", "Destination": "BCN" }</pre>	<pre>{ "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" } }] } }</pre>

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 – Obtener Ruta Parte 2

Se debe facilitar un formulario web donde el usuario pueda ingresar los campos **Origin** y **Destination**, dicho formulario debe cumplir las siguientes reglas.

- Los campos no pueden contener el mismo valor.
- Los campos solo deben permitir 3 caracteres exactamente.
- Los campos deben tener una estrategia para que el input siempre se registre en mayúscula.

Cuando el usuario ingrese los campos correctamente, debe existir un botón que se encargue de consumir el API expuesta en el **Problema 3** y mostrar el resultado en pantalla.

Problema 5 – Selección Moneda

Por último, después de obtener la respuesta en el **Problema 4** el usuario debe tener la posibilidad de ver el valor de la ruta en la moneda que seleccione, por defecto se asume que los valores devueltos en el **Problema 2** se encuentran en **USD**. Adicione al menos dos formatos más de moneda y la posibilidad de que al obtener la ruta de vuelo pueda cambiar esta y ver los valores equivalentes.

Nota: Recuerde que la API devuelve un valor genérico en **USD**, por lo tanto, la estrategia de conversión queda a decisión del desarrollador.