Correcciones Prueba de ingreso FullStack

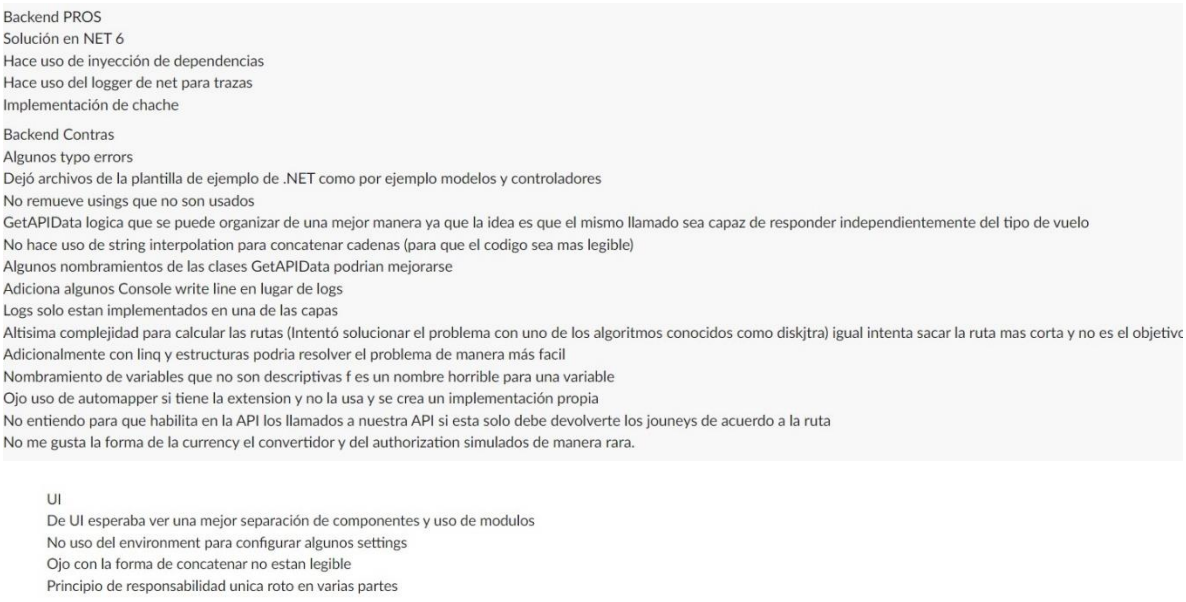
Andrés Felipe Hernández

https://felipedelosh.github.io/AFE-PORTFOLIO/

Newshore Prueba_Ingreso_Fullstack_2022

2023

A los 18 días del mes de septiembre se me informa sobre múltiples errores y correcciones con la prueba anterior. Es mi deber ofrecer al lector una disculpa debido a que aprendí los conceptos sobre la marcha. Soy FullStack Ruby On Rails + Python.



BACKEND

https://github.com/felipedelosh/_newShoreAIR

Algunos type errors

Hay un mal uso del var en múltiples pates del código

Variable	***PATH***	***Tipo correcto***
`var nodeA`	newShoreAIR\Business\Availability\AvailabilityBusiness.cs	`string`
`var nodeB`	newShoreAIR\Business\Availability\AvailabilityBusiness.cs	`string`
`var price`	newShoreAIR\Business\Availability\AvailabilityBusiness.cs	double
`var sizeOfResponse`	newShoreAIR\Business\Availability\AvailabilityBusiness.cs	int
`var sizeData`	newShoreAIR\Business\Availability\AvailabilityBusiness.cs	Int

var tempPrice	newShoreAIR\Business\Availability\AvailabilityBusiness.cs	double
var lastUrl	newShoreAIR\Business\Availability\AvailabilityBusiness.cs	string
var result	newShoreAIR\Business\Availability\AvailabilityBusiness.cs	string
var flightsResponse	newShoreAIR\Business\Availability\AvailabilityBusiness.cs	List<GetJsonFlightResponse>
var i	newShoreAIR\Business\Availability\AvailabilityBusiness.cs	GetJsonFlightResponse
var flightResponse	newShoreAIR\Business\Mapper\APIResponseFlights.cs	List<Flight>
var flight	newShoreAIR\Business\Mapper\APIResponseFlights.cs	GetJsonFlightResponse
var transport	newShoreAIR\Business\Mapper\Flight_Transport.cs	Transport
var response	newShoreAIR\Helper\GetAPIData.cs	string
var client	newShoreAIR\Helper\GetAPIData.cs	RestClient
var request	newShoreAIR\Helper\GetAPIData.cs	RestRequest
var request	newShoreAIR\newShoreAPI\Controllers\JourneyController.cs	ModelRequesFlights
var fligtsData	newShoreAIR\newShoreAPI\Controllers\JourneyController.cs	Journey
var endData	newShoreAIR\newShoreAPI\Controllers\JourneyController.cs	string

Dejo archivos de la plantilla de ejemplo de .net como por ejemplo modelos y controladores

Al momento de crear un API en Visual Studio Existe una plantilla de ejemplo “WeatherForecast” dicha plantilla se usó solo como motivo de prueba (Especie de PING) (No los eliminé en primera instancia debido a que no había creado un controlador de vuelos. Y olvide hacerlo después.) es de mi total conocimiento que es una mala práctica de programación por ello serán removidas:

Archivo	***Path***
WeatherForecastController	newShoreAIR.newShoreAPI.Controllers. WeatherForecastController.cs
WeatherForecast	newShoreAIR.newShoreAPI. newShoreAIR.newShoreAPI.cs

No remueve usings que no son usados

Los using NO UTILIZADOS y removidos son:

USING	***PATH***
`using System.Linq.Expressions;`	newShoreAIR\Business\Availability\AvailabilityBusiness.cs
`using System.Xml.Linq;`	newShoreAIR\Business\Availability\AvailabilityBusiness.cs
`using System.Linq;`	newShoreAIR\Business\Mapper\APIResponseFlights.cs
`using System.Text;`	newShoreAIR\Business\Mapper\APIResponseFlights.cs
`using System.Threading.Tasks;`	newShoreAIR\Business\Mapper\APIResponseFlights.cs
`using AutoMapper;`	newShoreAIR\Business\Mapper\APIResponseFlights.cs
`using System.Collections.Generic;`	newShoreAIR\Business\Mapper\Flight_Transport.cs
`using System.Linq;`	newShoreAIR\Business\Mapper\Flight_Transport.cs
`using System.Text;`	newShoreAIR\Business\Mapper\Flight_Transport.cs
`using System.Threading.Tasks;`	newShoreAIR\Business\Mapper\Flight_Transport.cs
`using System.Linq;`	newShoreAIR\Helper\Cache\CacheController.cs
`using System.Security.Policy;`	newShoreAIR\Helper\Cache\CacheController.cs
`using System.Text;`	newShoreAIR\Helper\Cache\CacheController.cs
`using System.Threading.Tasks;`	newShoreAIR\Helper\Cache\CacheController.cs
`using System.Collections.Generic;`	newShoreAIR\Helper\Cache\CacheItem.cs
`using System.Linq;`	newShoreAIR\Helper\Cache\CacheItem.cs
`using System.Text;`	newShoreAIR\Helper\Cache\CacheItem.cs
`using System.Threading.Tasks;`	newShoreAIR\Helper\Cache\CacheItem.cs
`using System;`	newShoreAIR\Helper\RoutesCalculator\Edge.cs
`using System.Collections.Generic;`	newShoreAIR\Helper\RoutesCalculator\Edge.cs
`using System.Linq;`	newShoreAIR\Helper\RoutesCalculator\Edge.cs
`using System.Text;`	newShoreAIR\Helper\RoutesCalculator\Edge.cs
`using System.Threading.Tasks;`	newShoreAIR\Helper\RoutesCalculator\Edge.cs
`using System;`	newShoreAIR\Helper\RoutesCalculator\Graph.cs
`using System.Linq;`	newShoreAIR\Helper\RoutesCalculator\Graph.cs
`using System.Text;`	newShoreAIR\Helper\RoutesCalculator\Graph.cs
`using System.Threading.Tasks;`	newShoreAIR\Helper\RoutesCalculator\Graph.cs
`using System.Xml.Linq;`	newShoreAIR\Helper\RoutesCalculator\Graph.cs
`using System.Collections;`	newShoreAIR\Helper\RoutesCalculator\ShortestPathFinder.cs
`using System.Text;`	newShoreAIR\Helper\RoutesCalculator\ShortestPathFinder.cs

using System.Threading.Tasks;	newShoreAIR\Helper\RoutesCalculator\ShortestPathFinder.cs
using System.Web.Configuration;	newShoreAIR\Helper\RoutesCalculator\ShortestPathFinder.cs
using System;	newShoreAIR\Helper\Authentication.cs
using System.Collections.Generic;	newShoreAIR\Helper\Authentication.cs
using System.Linq;	newShoreAIR\Helper\Authentication.cs
using System.Text;	newShoreAIR\Helper\Authentication.cs
using System.Threading.Tasks;	newShoreAIR\Helper\Authentication.cs
using System.Linq;	newShoreAIR\Helper\CurriencesConverter.cs
using System.Text;	newShoreAIR\Helper\CurriencesConverter.cs
using System.Threading.Tasks;	newShoreAIR\Helper\CurriencesConverter.cs
using System.Collections.Generic;	newShoreAIR\Helper\GetAPIData.cs
using System.Linq;	newShoreAIR\Helper\GetAPIData.cs
using System.Text;	newShoreAIR\Helper\GetAPIData.cs
using System.Threading.Tasks;	newShoreAIR\Helper\GetAPIData.cs
using System;	newShoreAIR\Models\Contracts\IAvailability.cs
using System.Collections.Generic;	newShoreAIR\Models\Contracts\IAvailability.cs
using System.Linq;	newShoreAIR\Models\Contracts\IAvailability.cs
using System.Text;	newShoreAIR\Models\Contracts\IAvailability.cs
using System.Threading.Tasks;	newShoreAIR\Models\Contracts\IAvailability.cs
using System;	newShoreAIR\Models\Contracts\IMap.cs
using System.Collections.Generic;	newShoreAIR\Models\Contracts\IMap.cs
using System.Linq;	newShoreAIR\Models\Contracts\IMap.cs
using System.Text;	newShoreAIR\Models\Contracts\IMap.cs
using System.Threading.Tasks;	newShoreAIR\Models\Contracts\IMap.cs
…	…
…	…

…

Se usa en todas las otras clases CONTROL + K + E para eliminar todos los using no utilizados.

GetAPIData lógica que se puede organizar de una mejor manera ya que la idea es que el mismo llamado se capaz de responder independientemente del tipo de vuelo.

GetAPIData se creó con el fin de consumir un API externa, Por ello procedo a dejarla con una única responsabilidad: Consumir API externa.

Para simplificar todo en una llamada procedo a realizar el consumo de los vuelos del API de newShore en: newShoreAIR\Business\Availability\AvailabilityBusiness.cs en el método GetJourney.

Con esto el mismo llamado es capaz de responder.

No hace uso de String interpolación para concatenar cadenas (Para que el código sea más legible).

Se remplaza los concatenados por: $”x {<CODE>}”;

Antes	PATH	Después
`string _url = url + "/" + v;`	newShoreAIR\Helper\GetAPIData.cs	`$"{url}/{v}";`

…

Algunos nombramientos de las clases GetAPIData podrían mejorarse

Procede a cambiarse todos los var por su clase correspondiente.

Adiciona algunos Console Write line en lugar de los logs

Dichos Console Writeline eran informativos (Supervisar procesos por consola). Olvidé retirarlos. Por ello procedo a borrarlos de los siguientes lugares

newShoreAIR\Business\Availability\AvailabilityBusiness.cs

newShoreAIR\Helper\GetAPIData.cs

Logs solo están implementados en una de las capas

Realmente los LOGS solo deberían implementarse en la capa de API + bussines + Helpers

Solo se hace la inyección de dependencias.

//Registrar en newShoreAIR\newShoreAPI\IOC\AutofacBusinessModule.cs

```
private readonly ILogger<ClaseCorrespondiente> _logger;
```

…

```
Constructor (Entra)
{
      Asigna;
}
```

Estos son las clases donde se implementan los loggers

Mediante inyección de dependencias (newShoreAIR\newShoreAPI\IOC\AutofacBusinessModule.cs)

newShoreAIR\Business\Availability\AvailabilityBusiness.cs

newShoreAIR\Helper\RoutesCalculator\RouteCalculator.cs

newShoreAIR\Helper\GetAPIData.cs

newShoreAIR\Helper\CurriencesConverter.cs

Altisima complejidad para calcular rutas (Intentó solucionar el problemas con uno de los algoritmos conocidos como diskjtra) igual intenta sacar la ruta más corta y no es el objetivo.

Procedo a eliminar el uso de grafos. Y recorrer LINQ recursivamente.

Adicionalmente con linq y estructuras podría resolver el problema de manera más fácil.

Procedo a eliminar el uso de grafos e importar LINQ en el archivo:

newShoreAIR\Helper\RoutesCalculator\RouteCalculator.cs

“Se soluciona de manera recursiva”

Nombramiento de variables que no son descriptivas f es un nombre horrible para una variable.

En el archivo: newShoreAIR\Business\Availability\AvailabilityBusiness.cs en el método para buscar el vuelo en cache se necesita un temporal de Flight que bautice por error como f se porcede a cambiar el nombre por tempFlight. Adicionalmente se necesita un Transport temporal procedo a renombrar como tempTransport

Uso de automapper si tienen la extención y no la usa y se crea una implementación propia.

Debido a que solucione el problema con grafos no tuve la necesidad de hacer un mapper, por ello procedo a realizar la respuesta usando mapper.

En el archivo: newShoreAIR\Business\Availability\AvailabilityBusiness.cs

Se importa, se inicializa en el constructor y se mapea en la llamada GET

No entiendo para que habilita en la API los llamados a nuestra api si está solo debe devolverte los journeys deacuerdo a la ruta

Procedo a eliminar los llamados a la API:

https://localhost:7036/api/v1/Journey/getFlightsV0

https://localhost:7036/api/v1/Journey/getFlightsV1

https://localhost:7036/api/v1/Journey/getFlightsV2

La exposición de esas API deriva a que quería informar al usuario todos los vuelos disponibles para rellenar los campos en el FRONTEND, procedo a cambiar la lógica y exponer el siguiente ENDPOINT para que el FRONTEND tenga la lista de vuelos disponibles:

https://localhost:7036/api/v1/Journey/getAllFlights

No me gusta la forma de la currency el convertidor y authortization simulados de manera rara

Actualización del servicio de curriences:

Lo que se pretendía simular era un llamado a:

https://docs.openexchangerates.org/reference/api-introduction

Luego guardar las tazas de cambio en un diccionario (Dolarizado), cuando el usuario consulte un vuelo en una denominación X se le entregará el valor registrado. En caso contrario retornaría -9999. (Simula menos infinito) Por lo cual se detecta un error de moneda y no se realiza la conversión. Y se le manda al usuario un mensaje en el JSON.

//Procedo a hacerlo real.

Para la validación de TOKEN se hace mediante librería.

FRONTEND

https://github.com/felipedelosh/_newShoreAIRFRONTEND

“Es mi deber pedir disculpas al lector, debido a que solo tuve un par de horas para aprender angular e implementar el desarrollo correspondiente.”

De UI esperaba ver una mejor separación de componentes y uso de módulos.

Debido a que implemente un SPA y también por términos de tiempo no separé todos los componentes por ello se procede a hacer los siguientes cambios:

Se crea el componente HEADER, FOOTER en una nueva carpeta SHARED (debido a que son componentes transversales)

No uso el Enviroment para configurar algunos settings

De hecho si se usó, existía antes en: src\environment.ts y se usa en los servicios en la parte de import y obtener la ruta del api.

Para mayor claridad procedo a moverlo a la carpeta: src\environments\environment.ts y organizo el import en src\app\service\api.service.ts

Ojo con la forma de concatenar no es tan legible

Se cambian todos los concatenados a String formateado.

Principio de responsabilidad única roto en varias partes

Se procede a eliminar el método getData.

Particionar responsabilidades: En todos los métodos que consumen servicios

Además de renombrar variables para mayor legibilidad.

Recordar instalar el Pluggin para poder consumir el API

https://chrome.google.com/webstore/detail/moesif-origin-cors-change/digfbfaphojjndkpccljibejjbppifbc