Introdução

Atualmente utilizamos diversas APIs para buscar voos, e após receber os resultados é necessário fazer o agrupamento entre as idas e voltas.

Nesse teste você irá desenvolver uma API que faz esse agrupamento de voos.

Disponibilizamos uma API onde você irá consultar os vôos a serem agrupados. No total são 15 voos, categorizados em 2 tipos de tarifa.

Requisitos

Será necessário utilizar algum desses frameworks para desenvolver o projeto:

- Laravel (https://laravel.com/)
- Lumen (https://lumen.laravel.com/)

Entregas

- Será necessário entregar todo o código gerado no teste
- Como é uma API você também pode anexar sua rota.
- Entregue uma documentação com todos os passos para executar seu projeto.
- Disponibilize o código e a documentação em um link no github, ou em um arquivo zip para download.

[opcional]: Utilizar o GitHub para fazer a entrega é um diferencial
[opcional]: Utilizar o Swagger, Postman ou algo similar para documentar sua rota é um diferencial
[opcional]: Disponibilizar o teste na internet, para que possa ser testado via navegador ou postman é um diferencial

Pontos de Avaliação

- Toda a estrutura da sua API; (Rota, HTTP response, HTTP Status, etc..)
- Nomenclatura e padronização das suas variáveis, funções, classes;
- Separação de responsabilidades;
- Lógica e otimização de processamento;

Instruções

1) Realize a consulta dos voos via API

- **URL API**: http://prova.123milhas.net/api/flights

Descrição dos principais campos:

Campo	Descrição
id	Identificador único do voo
cia	Companhia aérea responsável pelo voo
fare	Tipo de tarifa (Fator determinante no agrupamento)
flightNumber	Número do voo
departureDate	Data de saída do voo
arrivalDate	Data de chegada do voo
origin	Identificador do aeroporto de saída
destination	Identificador do aeroporto de chegada
price	Preço do voo (Fator determinante no agrupamento)
outbound	Determina se o voo é ida (Fator determinante no agrupamento)
inbound	Determina se o voo é volta (Fator determinante no agrupamento)

Postman Collection: https://www.getpostman.com/collections/4ad483786de4106d9ef1

2) Faça o agrupamento dos voos.

Regras de agrupamento:

- Deve-se gerar grupos com uma ou mais opções de ida e volta;
- Dentro de um mesmo grupo não podem ter voos de tarifas diferentes;
- Ao formar um grupo é necessário criar um identificador único;
- Todo grupo deve ter um preço total;

O que é um grupo?

O grupo é um conjunto de voos ida + volta, que tem o mesmo tipo de tarifa.

Exemplos:
Considere a seguinte lista de voos para entender a base do agrupamento.

Voos[1]:
Id: 1 | R$ 100,00 | Tarifa: 1AF | Ida
Id: 2 | R$ 200,00 | Tarifa: 1AF | Volta
Id: 3 | R$ 100,00 | Tarifa: 1AF | Ida
Id: 4 | R$ 250,00 | Tarifa: 1AF | Volta
Id: 5 | R$ 500,00 | Tarifa: 2DA | Ida
Id: 6 | R$ 800,00 | Tarifa: 2DA | Volta

Grupos:
Id 1 (Valor Total: R$ 300,00 | Idas: [Voo 1, Voo 3] | Voltas: [Voo 2])
Id 2 (Valor Total: R$ 350,00 | Idas: [Voo 1, Voo 3] | Voltas: [Voo 4])
Id 3 (Valor Total: R$ 1300,00 | Idas: [Voo 5] | Voltas: [Voo 6])

Utilizando os voos acima será possível montar 3 grupos.

Perceba que um grupo pode ter mais de um voo de ida / volta, desde que o valor total do grupo não sofra alterações.

OBS: Aqui listamos apenas os itens determinantes para montar um grupo, os voos tem mais informações.[1]

3) Faça a ordenação dos resultados

- Por padrão faça a ordenação dos grupos pelo menor preço.

continua...

4) Resultado

Retorne um json com os resultados no seguinte formato:

```
{
   “flights”: // retorne aqui os voos consultados na api em prova.123milhas.net
   “groups”: [
      {
         “uniqueId”: // id unico do grupo
         “totalPrice”: // preço total do grupo
         “outbound”: [ // voo(s) de ida
            {
               “id”
            },...
         ]
         “inbound”: [ // voo(s) de volta
            {
               “id”
            },...
         ]
      },...
   ],
   “totalGroups”: // quantidade total de grupos
   “totalFlights”:  // quantidade total de voos únicos
   “cheapestPrice”: // preço do grupo mais barato
   “cheapestGroup”: // id único do grupo mais barato
}
```