14

Mãos na massa: Implementando a busca de aeroportos

Chegou a hora de colocar a mão na massa!

1) Copie o arquivo **geo** disponível no download nesse link (https://caelum-online-public.s3.amazonaws.com/712-dotnet-mongodb-parte2/04/base-mongodb-localizacoes-aeroportos-para-importar.zip) e descompacte em alguma pasta do seu computador.

2) Copie o caminho onde a **pasta** está localizada, abra o Prompt de Comando e entre na pasta:

```
cd CAMINHO_DA_PASTA
```

3) Após isso, execute o seguinte comando:

```
mongorestore -d geo geo
```

4) Utilize site **json2csharp** para gerar uma classe automaticamente a partir de um registro da base de dados que você acabou de criar, e crie a classe **Aeroporto** em seu projeto.

5) Adicione as referências do **MongoDB.Bson**, **MongoDB.Serialization.Attributes** e **MongoDB.Driver.GeoJsonObjectModel** à classe **Aeroporto**.

6) Copie o que foi gerado no site, mas apenas o conteúdo da classe **RootObject** e cole dentro da classe **Aeroporto**.

7) Adicione a representação do ID interno que todo registro no Mongo deve ter.

8) Modifique o tipo de **loc** para `GeoJsonPoint<GeoJson2DGeographicCoordinates>`

9) Crie a classe **conectandoMongoDBGeo** e adicione as referências **MongoDB.Bson** e **MongoDB.Driver**

10) Copie as 5 variáveis do exemplo **conectandoMongoDB** feito anteriormente e modifique-as da seguinte forma:

```
public const string STATUS_DE_CONEXAO = "mongodb://localhost:27017";
public const string NOME_DA_BASE = "geo";
public const string NOME_DA_COLECAO = "airport";

private static readonly IMongoClient _cliente;
private static readonly IMongoDatabase _BaseDeDados;
```

11) Copie os métodos da classe **conectandoMongoDB** dos exemplos que foram feitos anteriormente e cole na classe **conectandoMongoDBGeo** do projeto atual.

12) Modifique o nome para que seja igual ao nome da classe, para resolver o erro que surgiu.

13) Altere o nome da coleção para `Airports` , ajustando também o seu tipo e o seu retorno.

14) Crie a classe **buscaAeroportos**, copie a estrutura da classe **listandoDocumentos** feita anteriormente, e cole nessa nova classe.

15) Adicione as referências à **MongoDB.Bson** e **MongoDB.Driver**.

16) Altere a variável `conexaoBiblioteca` para `conexaoAeroporto`, e modifique o tipo da instância, de `conectandoMongoDB` para `conectandoMongoDBGeo`.

17) Adicione a referência à **MongoDB.Driver.GeoJsonObjectModel** e crie uma instância de `GeoJson2DGeographicCoordinates`, com o nome `ponto`, passando para ele a longitude **-118.325258** e a latitude **34.103212**.

18) Crie uma instância de `GeoJsonPoint`, com o nome `localizacao`, passando o ponto como parâmetro:

```
var localizacao = new GeoJsonPoint<GeoJson2DGeographicCoordinates>(ponto);
```

19) Crie um filtro de aeroporto e crie também uma condição para buscar os pontos dentro de um raio de busca de 100000 metros.

20) Busque os aeroportos na coleção, baseado na condição feita no passo anterior e salve-os na variável `listaAeroportos`.

21) Modifique o `foreach` para **listaAeroportos** e altere o formato do JSON para **Aeroporto**.