



EXERCÍCIO #2

Enquanto você desenvolvia os modelos, o resto da equipe trabalhou nas outras camadas da API (repositórios, serviços, controllers), e ainda arrumaram tempo para escrever alguns testes unitários!

De qualquer forma, você ficou encumbido não só de **garantir a qualidade dos testes (que estejam passando e cobrindo requisitos adicionais)** mas também de **garantir que a cobertura de código atinja 100%**.

Requisitos adicionais:

- As equipes tem que ser retornadas em ordem alfabética por sigla do estado e depois pelo nome da própria equipe (ambas em ordem crescente)
- Ao criar uma equipe, validar se a sigla fornecida está associada a algum estado
- Ainda ao criar uma equipe, validar se já existe uma equipe com o mesmo nome para a mesma sigla
- Não permitir partidas cujo mandante e visitante sejam a mesma equipe
- **Todas as regras acima devem ser feitas na camada de serviço. No caso das validações, ao falharem devem retornar um ArgumentException na camada de serviço e um BadRequest na camada da controller**

Algumas diretivas para auxiliá-los:

1. O projeto já conta com as referências ao coverlet.collector e ao EF InMemory, necessárias para ter reports de cobertura e o repositório em memória, respectivamente. Ainda assim se tiverem problemas com o dotnet restore, podem adicionar os pacote explicitamente através dos comandos:
`dotnet add package coverlet.collector`
`dotnet add package Microsoft.EntityFrameworkCore.InMemory`
2. A solução fornecida está sem os modelos (feitos por você no Exercício #1) e por isso, naturalmente, **NÃO COMPILA**. Isto posto, os inclua na solução fazendo os ajustes necessários (alterar eventuais nomes de propriedades e/ou tipos de dados);



3. Lembre-se que nem todo código é passível de testes. O que pode (e deve) ser excluído da cobertura de código ([`ExcludeFromCodeCoverage`]):
 - a. Código que envolve configuração/inicialização do sistema, como `ZebrabetDbContext.cs`, `Program.cs` e `Startup.cs`
 - b. Camada de repositório (código testável, mas através de testes de integração, não unitários – que são o foco da atividade)
4. Melhorar a lógica dos testes (através de *Theories*, *Fluent Assertions*, *resources* etc.) existentes não é mandatório mas será considerado
5. O exercício será tido como bem sucedido quando as três condições abaixo forem atingidas:
 - a. Todos os testes passando com sucesso
 - b. Todos os requisitos adicionais atendidos
 - c. Cobertura de código de 100% (excluindo da cobertura **APENAS** o código pontuado no item 3.)

Em caso de dúvidas/dificuldades, envie um e-mail para eu@andrecassimiro.com.br 