Época Especial 2020/2021

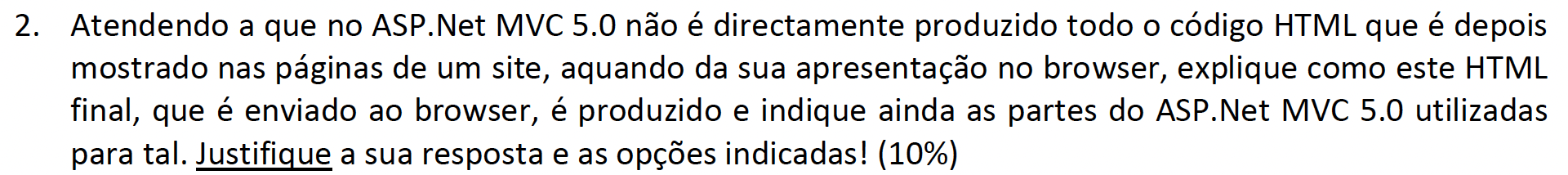
Um delegate em C# tem várias vantagens, sendo uma delas servir como ponteiro para um método podendo ter um nome diferente do método ou pode ser utilizado também como um método anónimo.

Exemplo:

public static void **ExampleMethod**(string value) { Console.WriteLine(value); }

public delegate void **DelegateFunction**(string value);

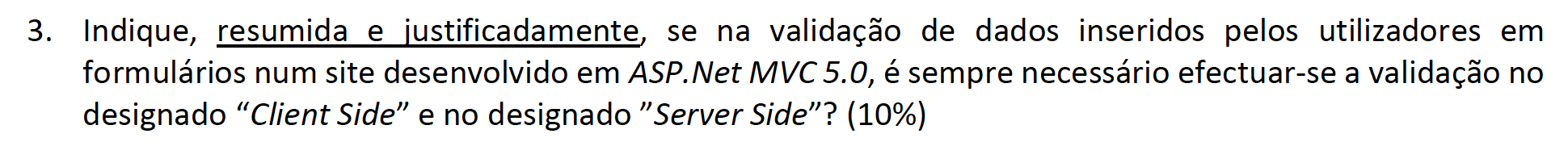
**DelegateFunction** del = **ExampleMethod**; // as funções devem ter parâmetros de entrada iguais.



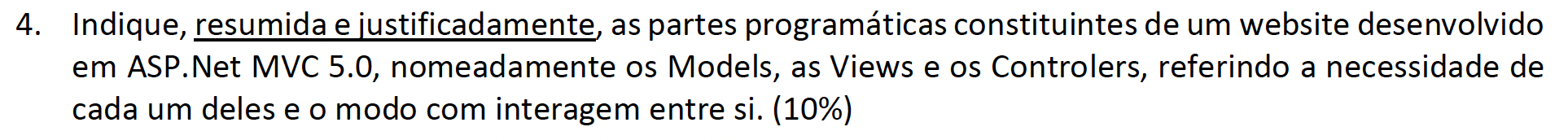
O HTML é produzido recorrendo a HTML Helpers, Tag Helpers e ao Razor.

Os HTML Helpers e Tag Helpers ajudam a compactar algum código HTML tendo apenas o necessário e posteriormente gera as tags necessárias para ser um HTML válido.

Sendo o Razor uma linguagem de marcação, permite obter um website dinâmico que recebe dados vindo dos controladores e mostrar os mesmos nas views.



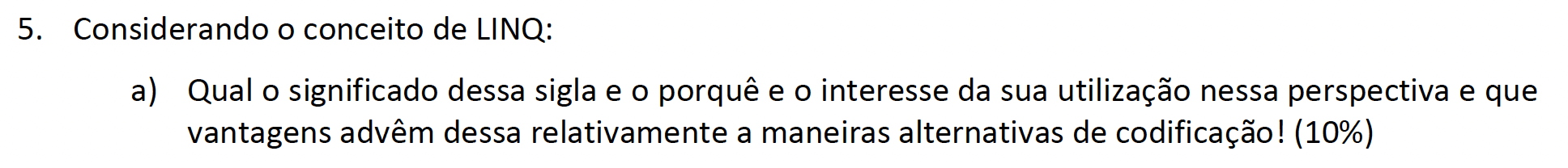
Não é sempre necessário fazer a validação Client-Side e Server-Side, no entanto é uma boa prática o mesmo ser feito visto que é possível fazer algumas validações Client-Side de inputs em formulários e voltar a verificar outras validações em Server-Side. Ou seja, no fundo as validações Client-Side apenas servem para UI/UX (parte visual), sempre que queremos fazer validações lógicas convém usar o Server-Side.

Model – Representam os modelos de domínio, onde é feita a definição das tabelas/classes, por exemplo numa abordagem Code First, em que definimos o nosso modelo para depois mapearmos para uma base de dados.

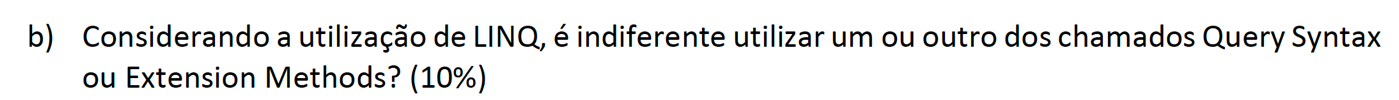
Usamos estes modelos para definir os objetos com que vamos trabalhar.

View – Responsável pela interface com o utilizador. Recebe dados vindos do controler, assim como também envia pedidos ao controler enviados pelo cliente. As views apenas enviam e recebem informação, não contendo qualquer lógica nelas.

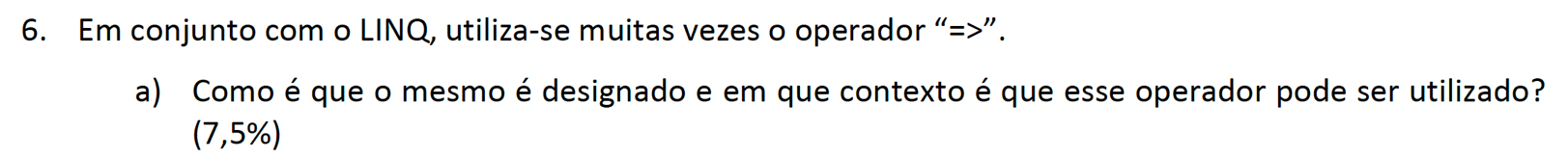
Controler – Recebem dados das views enviados pelos clientes e consoante o pedido acedem ao modelo de domínio para fazerem querys à base de dados. Os controlers posteriormente recebem o retorno da query e retornam para a view (que fez o pedido) o resultado.

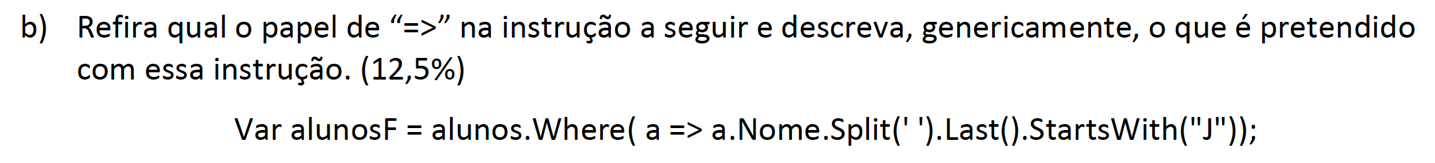


LINQ representa Language Integrated Query e é muito importante uma vez que em aplicações ASP.NET Core MVC normalmente usamos SQL Server e o LINQ efetua a tradução dos nossos modelos de domínio para as querys que fazemos à base de dados em SQL, uma vez que o SQL Server apenas reconhece a syntax SQL.



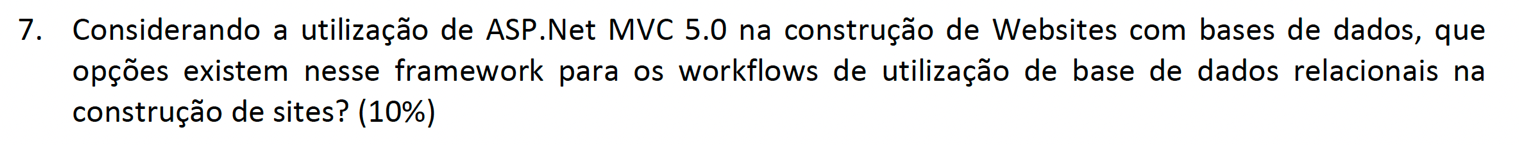
É indiferente a utilização de Query Syntax ou Extension Methods uma vez que o intuito de ambos é o mesmo, ou seja, comunicar com a base de dados. No entanto, os Extension Methods são tendencialmente mais simples de serem usados e são mais recentes.

O operador => tem o nome de Operador Lambda e o mesmo é bastante utilizado porque permite definir funções anónimas. Com recurso a este operador podemos aceder a atributos de um objeto sem ter de criar uma função para tal.



Na instrução, o operador lambda é utilizado para aceder ao atributo “Nome” do objeto sem recorrer a uma função para efetuar tal operação.

A instrução pesquisa numa lista de alunos (utilizando o Where), quais são os alunos cujo o ultimo nome (recorrendo ao Last e ao split, que separa as palavras por espaços em branco) começa com a letra “J”.

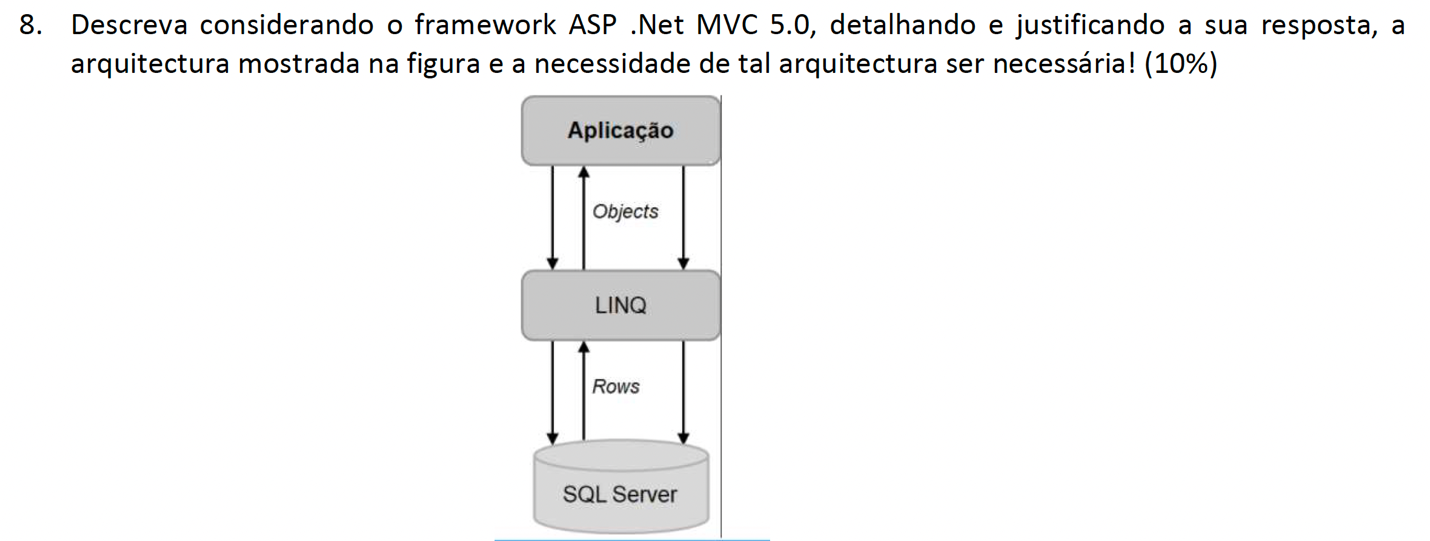


Existem dois workflows para a construção de website em ASP.NET MVC.

Code-First: Partimos do princípio que não temos nenhuma base de dados previamente criada e começamos por criar os modelos de domínio para depois ser possível aplicar as migrações fornecidas pelo Entity Framework e realizar update à base de dados para criar as nossas tabelas com base no modelo.

Normalmente esta é a maneira mais utilizada quando não temos uma base de dados previamente construída.

Database-First: Partimos de uma base de dados previamente construída que pode ou não ter alguns dados armazenados e pretendemos que os mesmos não se percam. Então para isso utilizamos este workflow juntamente com o Entity Framework que nos permite utilizas as tabelas já existentes e fazer um scaffolding das mesmas para os nossos modelos de domínio serem criados.

Este método é uma maneira simples de não apagarmos dados de uma base de dados já existente e continuarmos a utilizar a mesma sem ter de fazer nenhuma alteração física à mesma.

A arquitetura mostrada na imagem mostra uma arquitetura em que usamos o LINQ com intermediário para a comunicação entre os nossos modelos de domínio e o nosso servidor de base de dados. Ou seja, aqui vemos que para conseguirmos mapear entre os modelos e base de dados utilizamos o LINQ para fazer essa conversão onde o mesmo traduzir as queries feitas aos modelos praa queries SQL sendo que o SQL Server apenas entende Query Syntax SQL. Depois o LINQ transforma o retorno das queries de SQL em formato de objeto para ser possível utilizar este objeto em C# utilizando um paradigma de programação orientada a objetos.