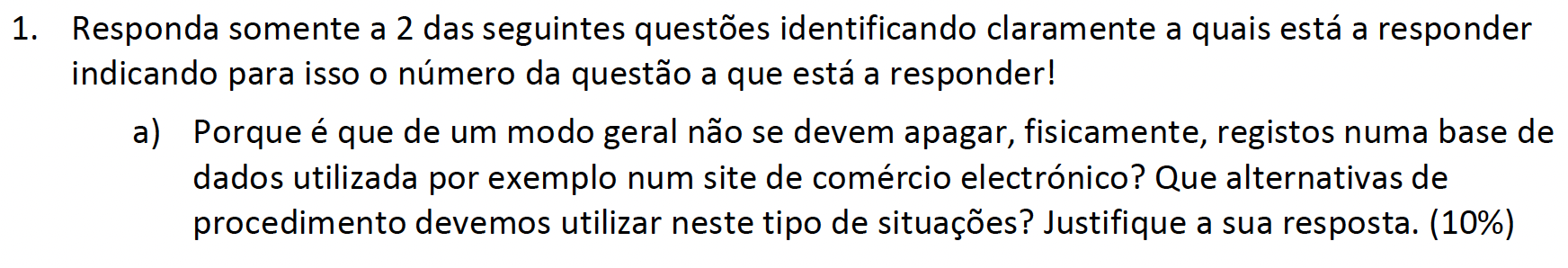
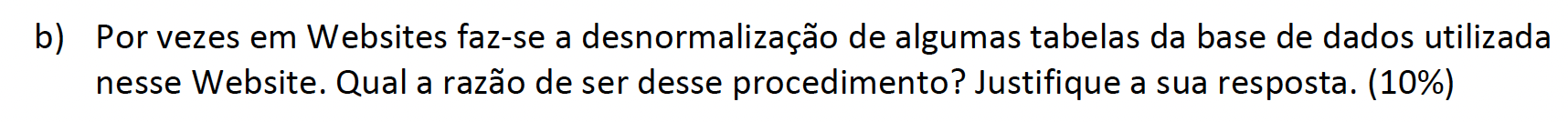
Época Normal 2020/2021



De um modo geral não se devem apagar fisicamente registos de uma base de dados, porque esses dados já estão a ser utilizados por utilizadores ou até mesmo pelo gestor da plataforma, logo uma alternativa que teríamos era usar um script que contenha essa informação que queríamos apagar e para executar esse script efetuávamos uma migração.

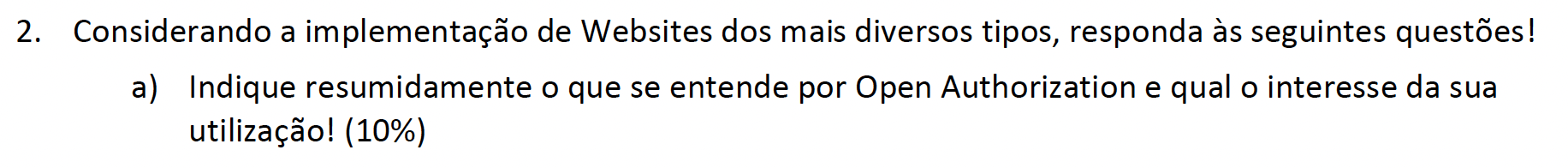


A desnormalização é importante porque tenta otimizar o desempenho de leitura de uma base de dados, adicionando assim dados redundantes. Como normalmente as consultas requerem a utilização de JOINS entre tabelas para obter dados, isto acaba por comprometer o desempenho da base de doas, então para contornar este problema desnormaliza-se a base de dados juntamente com os dados apenas numa única tabela. Por exemplo na criação de uma view na base de dados em que essa view representa uma query com vários JOINS.

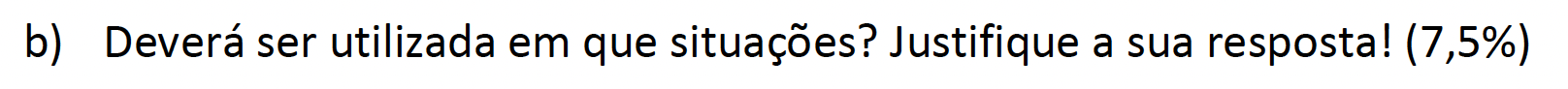
Uma imagem com texto

Descrição gerada automaticamente

A razão desse procedimento é para facilitar aos clientes, uma vez que por exemplo, se a loja com o id=1 tiver o nome Zara, é muito mais conveniente ao cliente saber o nome da loja do que o seu id.



Open Authorization é um tipo de autenticação token-based que permite que as organizações partilhem informação através de terceiros (Facebook , Instagram, Twitter, etc)



Deve ser utilizada em situações em que a organização atual confie nos terceiros e esses terceiros sejam confiáveis de modo a existir uma autenticação externa credível e fiável.

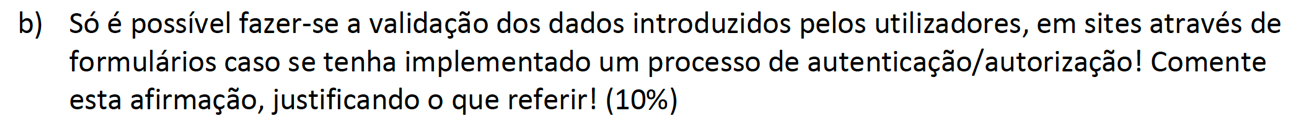
Uma imagem com texto

Descrição gerada automaticamente

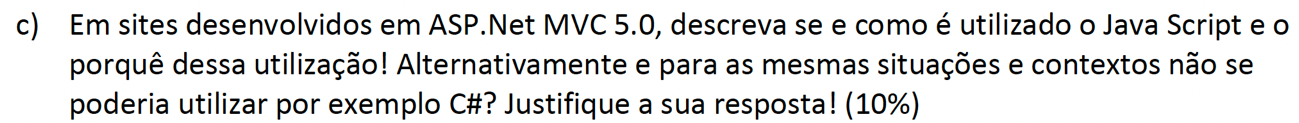
Autenticação: Processo de login, por exemplo, acesso a um site.

Autorização: Verificar se existe ou não permissão para um determinado utilizador (que já está previamente autenticado) aceder a um certo recurso.

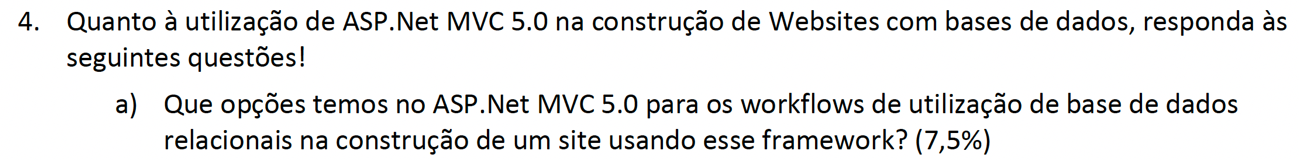
Não é necessário usar sempre as duas mas ao utilizar uma delas implica a utilização da outras na maioria dos casos.



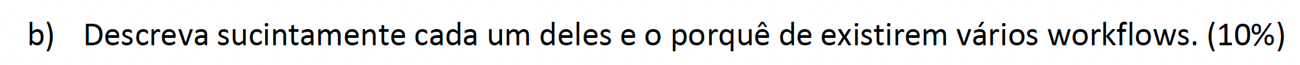
É uma afirmação falsa uma vez que podemos ter um formulário para um cliente submeter um email com alguns conselhos e para isso temos de ter alguns campos como o mail do mesmo, o seu numero de telemóvel e afins para validar esses campos não precisamos de autenticação ou autorização, apenas precisamos de validar se o email introduzido é valido e se o numero de telemóvel também é um número válido.



Em sites desenvolvidos em ASP.NET MV 5.0, o JavaScript é utlizado Client-Side ou seja, sempre que queremos gerar alguma coisa que seja visual para o utilizador, neste caso se quisermos utilizador C# teríamos de fazer uso do Blazor que faz com que seja possível executar código C# em Cliente-Side.



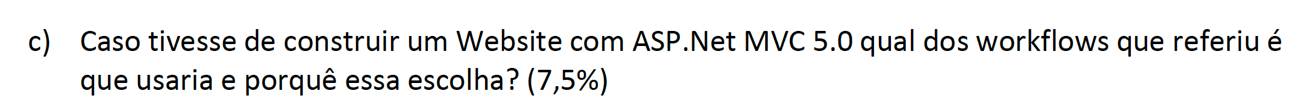
Temos dois tipos de workflows de utilização de base de dados relacionais: Database-First e Code-First.



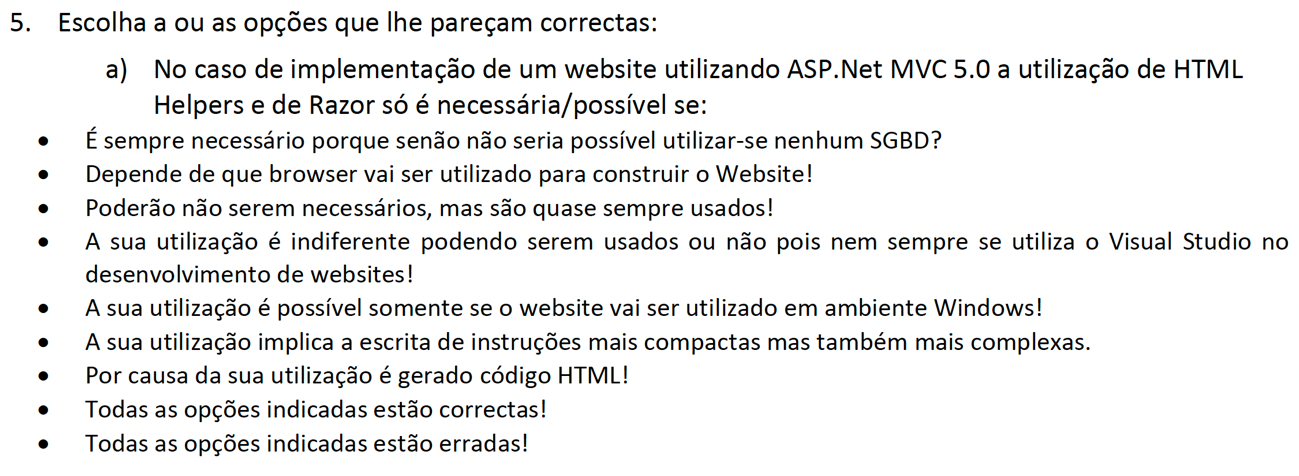
Database-First: é bastante útil quando já temos na nossa posse uma base de dados e esta já contém alguns dados que não queremos modificar. Se optarmos por este workflow, a partir da base de dados já existente, podemos criar os nossos modelos de domínio usando o Entity Framework Core com a ajuda do Scaffolding que pega nas tabelas e transforma em modelos de domínio tendo então uma maneira de comunicar entre modelos e tabelas da base de dados.

Code-First: é o workflow normalmente utilizado aquando do início da criação de uma aplicação web uma vez que partimos sem base de dados criada e a mesma é criada aplicando migração (que o Entity Framework fornece) usando os nossos modelos criados. Ou seja, neste caso a base de dados vai ser criada consoante os nossos modelos podendo as tabelas serem alterados aplicando migrações à base de dados.

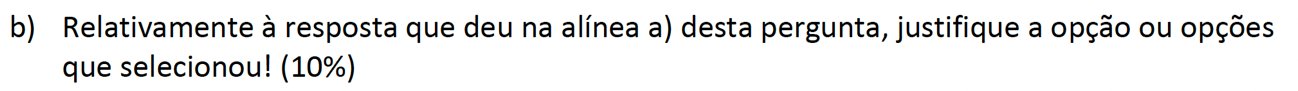
É importante existirem vários workflows porque como foi descrito, um workflow é útil se já tivermos uma base de dados e o outro é útil se não tivermos base de dados, logo temos um workflow para cada caso.



Caso tivesse de construir um website ASP.NET MVC 5.0 usaria o Code-First, porque como descrito anteriormente, parto do principio que não tinha base dados criada previamente e essa iria ser moldado consoante a criação dos meus modelos de domínio, podendo sempre alterar quando achasse necessário aplicando as migrações que o EF fornece.



* Poderão não ser necessárias, mas são quase sempre usados!
* Por causa da sua utilização é gerado código HTML!
* A sua utilização implica a escrita de instrução mais compactas mas também mais complexas.



Não é obrigatório utilizar HTML Helpers nem Tag Helpers, no entanto eles existem para nos dar uma maior facilidade de interagir entre Views e Controllers.

Ao utilizarmos HTML Helpers e Tag Helpers, estes ajudam a criar e a renderizar elementos de HTML em ficheiros Razor, logo geram código HTML.

Ao ser usado os HTML Helpers e os Tag Helpers ficamos com a possibilidade de criar instruções que geram mais do que uma tag em html por exemplo o que faz com que fiquemos com uma instrução compacta, no entanto não percebemos o que a mesma faz. Ela pode tornar-se também complexa..