Exame de P. Web da E.ES. de 2020/2021

1. Considerando a utilização de Delegates em C# refira, justificando a sua resposta, as vantagens na utilização deste tipo de elemento! (10%)
   R: Delegates são (simplificando ao máximo) tipos que nos permitem enquanto programadores referenciar métodos (muito à semelhança de ponteiros para funções em C/C++), para além de nos permitirem invocar o método a partir de uma variável local, também nos possibilitam a passagem de referências a métodos a outros métodos (no caso em que o segundo método tem um tipo que seja delegate num dos seus parâmetros). Delegates podem ser instanciados a partir de métodos com nome ou a partir de métodos anónimos. O C# fornece-nos três principais tipos de delegates para nos facilitar a vida, os action, func e predicate; action são delegates com um número qualquer de parâmetros (de quaisquer tipos) que não devolvem resultados, func são delegates com um número qualquer superior ou igual a um de parâmetros (de quaisquer tipos) que devolvem resultados do tipo igual ao tipo do último type-parameter que lhes é passado e predicates são delegates com um parâmetro (de qualquer tipo) que devolvem um resultado do tipo booleano (true/false). As principais vantagens na utilização deste tipo de elemento são a possibilidade de escrever métodos (que recebam um parâmetro do tipo delegate) que tenham comportamentos diferentes dependendo do método que lhes é passado. São muito utilizados em queries LINQ com recurso aos Extension Methods (por exemplo no Enumerable<T>.Where(Predicate<T> p) ou no IQueryable<T>.Where(Predicate<T> p)) sob a forma de Lambda Expressions (expressões do tipo (a, b, ...) => {return ...}), mas não só.
2. Atendendo a que no ASP.Net MVC 5.0 não é directamente produzido todo o código HTML que é depois mostrado nas páginas de um site, aquando da sua apresentação no browser, explique como este HTML final, que é enviado ao browser, é produzido e indique ainda as partes do ASP.Net MVC 5.0 utilizadas para tal. Justifique a sua resposta e as opções indicadas! (10%)
   R: No ASP.Net MVC 5.0, tradicionalmente não se escreve todo o código HTML que é mostrado nas páginas de um site porque existem elementos que possamos querer apresentar de forma dinâmica (por exemplo listas que podem ter um número variável de entradas), para possibilitar a escrita de código dinâmico (que dependa dos dados fornecidos à vista) que depois possa ser transformado em HTML, recorremos ao Razor (Markup proveniente da mistura de C# com HTML). As Razor pages (nome dado às páginas escritas em Razor) são um conceito que só existe do lado do servidor, aquando de um pedido de apresentação de uma página pelo cliente, o servidor transforma o Razor em HTML puro e o servidor devolve o produto final.
3. Indique, resumida e justificadamente, se na validação de dados inseridos pelos utilizadores em formulários num site desenvolvido em ASP.Net MVC 5.0, é sempre necessário efectuar-se a validação no designado “Client Side” e no designado ”Server Side”? (10%)
   R: Na validação de dados inseridos pelos utilizadores em formulários num site desenvolvido em ASP.Net MVC 5.0, não é sempre necessário efetuar-se a validação no “Client Side” e no “Server Side”. É pratica comum fazer-se sempre a validação do lado do servidor porque se não for feita a validação do lado do servidor (em casos em que os dados tenham obrigatoriamente que seguir certos formatos), um simples pedido do cliente poderá causar danos graves do lado do servidor (como por exemplo, o fecho inesperado do servidor com causa numa excessão – que

obviamente não deve acontecer num Website porque é um serviço partilhado por vários utilizadores - ou a persistência de dados incoerentes na base de dados). É considerado (na maior parte dos casos) fazer a validação do lado do Cliente para 1) apresentar logo uma mensagem de erro adequada em caso de dados inválidos e 2) impedir o envio de pedidos desnecessários ao servidor (já identificamos que os dados estão inválidos e como tal o servidor iria responder-nos com uma mensagem de erro, é escusado obrigar o processamento dos dados no servidor se sabemos à priori que os dados não cumprem o critério definido). Praticamente nunca se verificam situações em que a validação só é feita do lado do cliente (é considerado muito mau hábito) porque se a validação falhar no cliente ou se algum utilizador malicioso decidir fazer modificações no código da aplicação local, o servidor poderá ficar num estado inconsistente.

4. Indique, resumida e justificadamente, as partes programáticas constituintes de um website desenvolvido em ASP.Net MVC 5.0, nomeadamente os Models, as Views e os Controlers, referindo a necessidade de cada um deles e o modo com interagem entre si. (10%)
   R: As partes programáticas constituintes de um website desenvolvido em ASP.Net MVC 5.0 podem ser muitas ou poucas, dependendo do website que se esteja a desenvolver mas assumindo um website com pouca complexidade técnica, as partes mais recorrentes e importantes são nomeadamente as Models – classes do modelo de dados que podem ser específicos para representar objetos reais e/ou entidades na base de dados (Entity framework), Views – ficheiros de Markup (nomeadamente Razor) que podem caracterizar páginas ou partes de uma página do website (os view partials), onde é misturado HTML com C# e os Controllers – classes com action methods (os métodos públicos dos controladores são geralmente action methods que devolvem ações assíncronas ao invés de resultados provenientes de tarefas síncronas de modo a optimizar a utilização da thread de receção e invocação de pedidos do servidor) responsáveis pelo controlo e pelas ações desencadeadas pelos pedidos HTTP. Existem também outras partes programáticas como View Models (modelos que só servem para apresentar dados ao utilizador e habitualmente guardam dados provenientes de modelos/tabelas da base dados), o startup.cs, o program.cs, o DbContext (no caso de aplicações que recorram a Entity), entre outros.
5. Considerando o conceito de LINQ:
   a) Qual o significado dessa sigla e o porquê e o interesse da sua utilização nessa perspectiva e que vantagens advêm dessa relativamente a maneiras alternativas de codificação! (10%)
      R: LINQ advém de Language-Integrated Query o que por sua vez vem do facto de ser um modo de realizar queries na base de dados dentro do código da aplicação em C#. Não só torna possível a realização de queries no código principal da aplicação (tradicionalmente nos controladores) como também facilita muito! LINQ tem um síntaxe (tanto no Query Syntax como nos Extension Methods) muito mais familiar a um programador orientado a objetos, enquanto que para escrever SQL é preciso sair totalmente desse paradigma. LINQ tem ainda a vantagem de poder ser aplicado não só a entradas das tabelas da base dados como também pode ser aplicado a coleções que sejam “filhos” de Enumerable.
   b) Considerando a utilização de LINQ, é indiferente utilizar um ou outro dos chamados Query Syntax ou Extension Methods? (10%)
      R: Apesar de na maior parte dos casos, conseguirmos resultados idênticos recorrendo às duas formas de utilização (conseguimos resultados diferentes com Extension Methods devido à sua maior flexibilidade) do LINQ, não é indifierente utilizar um ou o outro, primeiro

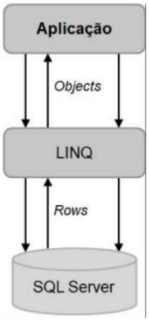
porque são sintaticamente muito diferentes, enquanto que os Extension Methods se aproximam muito mais de programação orientada a objetos, o query syntax aproxima-se muito mais a linguagens de query, como o SQL, segundo, porque os Extension Methods por estarem constantemente a evoluir (e o query syntax não, a Microsoft suporta mais os Extension Methods e a maior parte dos programadores também aderem mais aos mesmos) são muito mais flexíveis (é possível fazer mais coisas com Extension Methods do que com Query Syntax).

6. Em conjunto com o LINQ, utiliza-se muitas vezes o operador "=>".
   a) Como é que o mesmo é designado e em que contexto é que esse operador pode ser utilizado? (7,5%)
   R: O operador representado por "=>" denomina-se de Lambda Operator e utiliza-se na construção de Lambda Expressions, um tipo de expressão que surgiu nas versões mais atuais do C# com o qual conseguimos escrever métodos anónimos com um síntaxe muito mais compacto e legível.
   b) Refira qual o papel de "=>" na instrução a seguir e descreva, genericamente, o que é pretendido com essa instrução. (12,5%)

   ```
   Var alunosF = alunos.Where( a => a.Nome.Split(' ').Last().StartsWith("J"));
   ```

   R: O papel do Lambda Operator na instrução mostrada acima é permitir que seja aplicado uma operação a cada um dos alunos, sendo a operação a verificação do último nome do aluno começar com J (devolve um valor booleano – true/false). A instrução na sua totalidade permite obter todos os alunos que têm o último nome a começar com J.

7. Considerando a utilização de ASP.Net MVC 5.0 na construção de Websites com bases de dados, que opções existem nesse framework para os workflows de utilização de base de dados relacionais na construção de sites? (10%)
   R: Na construção de Websites com bases de dados e recorrendo a ASP.Net MVC 5.0, as opções para o workflow de utilização de base de dados que existem na mesma são a "Code First" e a "Database First". O "Code First" caracteriza-se por ser um método no qual partimos de um modelo de dados (código) escrito em C# e da qual geramos e vamos atualizando a base de dados (através da aplicação (Update-Database) de migrações consecutivas (Add-Migration <NomeMigração>) – "atualizações" ou "sincronizações" da base de dados). O "Database First" caracteriza-se por ser um método no qual partimos de uma base de dados e da qual vamos gerando um modelo de dados (classes C#) que servem para interfacar entre os objetos da base de dados e a aplicação. O "Code First" é o método preferido dos programadores por nunca obrigar o programador a sair do paradigma de orientação a objetos e também porque é considerado mais simples e menos "bug-prone".
8. Descreva considerando o framework ASP .Net MVC 5.0, detalhando e justificando a sua resposta, a arquitectura mostrada na figura e a necessidade de tal arquitectura ser necessária! (10%)



R: A arquitetura mostrada na figura acima contempla a relação entre aplicações ASP.Net MVC 5.0, LINQ e o servidor da base de dados. Esta relação pode ser visto como um mapeamento de entradas da base de dados para objetos (instâncias de classes do modelo de dados) na aplicação. Esta arquitetura é necessária para que seja possível interagir com a base de dados da aplicação de forma simples.