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

1. ~~Responda somente a 2 das seguintes questões identificando claramente a quais está a responder indicando para isso o número da questão a que está a responder~~!
   a) Porque é que de um modo geral não se devem apagar, fisicamente, registos numa base de dados utilizada por exemplo num site de comércio electrónico? Que alternativas de procedimento devemos utilizar neste tipo de situações? Justifique a sua resposta. (10%)
   R: De um modo geral, não se devem apagar registos de uma base de dados com uma elevada quantidade de relacionamentos (especialmente relacionamentos com obrigatoriedade) ou que de uma base de dados que deve persistir informação de forma "permanente" (de longa duração, p.e., um site de comércio eletrónico onde o utilizador poderá querer consultar as suas compras passadas). Isto porque, apagando esses registos, poderemos acabar em uma de duas situações. A primeira é a mais simples e consiste no facto de podermos precisar da informação associada àquele registo mais tarde. A segunda consiste no facto de que, apagando entradas de uma certa tabela que tenham a elas associadas entradas de outras tabelas através de um relacionamento de chave forasteira (foreign key), poderemos acabar com uma base de dados inconsistente ou até mesmo corrupta (especialmente no caso onde do lado da entrada dependente existe obrigatoriedade do relacionamento – causa exceções). Para contornarmos estas duas situações, ao invés de apagarmos as entradas da tabela deveremos colocar um novo atributo na mesma de modo a indicar-nos se aquela entrada ainda está "ativa" ou não.
   b) Por vezes em Websites faz-se a desnormalização de algumas tabelas da base de dados utilizada nesse Website. Qual a razão de ser desse procedimento? Justifique a sua resposta. (10%)
   R: Por vezes em Websites faz-se a desnormalização de algumas tabelas de base de dados de modo a aumentar a redundância dos dados guardadas nela o que por sua vez contribuirá para um desempenho superior ao que se verificaria caso não se fizesse a desnormalização porque o acesso a dados que de outra maneira implicaria um "query" complexo e com muitos "join"'s agora será mais simples.
   c) Ao contrário do previsto na teoria da Base de Dados, por vezes em algumas tabelas de uma base de dados de um website não se usam os ID de registos de outras tabelas dessa base de dados, chaves estrangeiras, mas os próprios dados. Qual a razão de ser desse procedimento? Justifique a sua resposta. (10%)
   R: A utilização de referências aos próprios dados provenientes de um relacionamento entre tabelas em contraste (ou a complementar) à utilização das chaves forasteiras chama-se a utilização de "Navigation Properties" e é muito comum no desenvolvimento de websites com recurso à Entity Framework Core (método Code-first) porque simplifica o acesso aos dados. Um exemplo prático poderá ser num sistema que tem entidades "Professor" e "Disciplina" onde um professor pode dar várias disciplinas mas uma disciplina só pode ser dada por um professor (e é obrigatório ser dado por um professor), se optarmos por utilizar navigation properties ao invés (ou a complementar) o uso de chaves forasteiras, sempre que quisermos aceder às disciplinas de um dado professor, poderemos simplesmente escrever:

```
val disciplines = professor.Disciplines;
```
ao invés de
```
val disciplines = _contextManager.Disciplines.Where(d => d.ProfessorId ==
professor.Id).ToList();
```

2. Considerando a implementação de Websites dos mais diversos tipos, responda às seguintes questões!
   a) Indique resumidamente o que se entende por Open Authorization e qual o interesse da sua utilização! (10%)
   R: Open Authorization é um mecanismo de delegação de acesso (de autênticação e autorização) no qual é utilizado um token de segurança em vez de um username e password de modo a evitar que seja necessária a partilha de dados pessoais do utilizador. É comumente utilizado na autênticação em aplicações com contas de redes sociais, entre outros, por exemplo, no caso de registo e autênticação em aplicações (que assim o permitam) com a conta da Google, Facebook, entre outros.
   b) Deverá ser utilizada em que situações? Justifique a sua resposta! (7,5%)
   R: Deve ser utilizada em situações em que não se encontra no interesse do utilizador partilhar os seus dados pessoais com a aplicação na qual se está a autenticar.
3. Considerando a necessidade de os Websites deverem ter uma componente forte no que à segurança diz respeito, ~~responda somente a 2 das seguintes questões (indique a quais das duas alíneas é que está a responder)!~~
   a) Indique as diferenças, justificando e contextualizando-as, entre os conceitos de Autênticação e de Autorização! É sempre necessário usar-se os dois? (10%)
   R: A autênticação é o processo (que ocorre continuamente, não só durante o login/registo) em que se prova a identidade do utilizador e pode ser feita com base em credenciais de acesso (username e password), tokens de segurança, entre outros. A autorização é o processo de garantir que um utilizador tem acesso ao conjunto de dados que está a pedir e pode ser feito com base em roles, autoridades, se o utilizador está autênticado ou não, entre outros. Existem situações em que só o authentication é utilizado e existem situações em que nem um nem o outro são utilizados mas o authorization nunca acontece sem previamente ter acontecido o authentication.
   b) Só é possível fazer-se a validação dos dados introduzidos pelos utilizadores, em sites através de formulários caso se tenha implementado um processo de autênticação/autorização! Comente esta afirmação, justificando o que referir! (10%)
   R: Esta afirmação é falsa, a validação dos dados introduzidos por utilizadores em formulários é (habitualmente) um processo aparte da autênticação/autorização e acontece com base nos critérios de validez dos dados introduzidos e dos seus respetivos valores (introduzidos pelo utilizador).
   c) Em sites desenvolvidos em ASP.Net MVC 5.0, descreva se e como é utilizado o Java Script e o porquê dessa utilização! Alternativamente e para as mesmas situações e contextos não se poderia utilizar por exemplo C#? Justifique a sua resposta! (10%)
   R: Em sites desenvolvidos em ASP.Net MVC 5.0, Java Script é utilizado para correr funções, validações e outros excertos de código localmente (na máquina do utilizador) tanto para situações em que o código diz mesmo respeito ao cliente do utilizador como também um modo de minimizar o número de pedidos que o utilizador faz ao servidor. C# não iria servir

para preencher este propósito porque C# não é uma linguagem que os browsers (aplicação-cliente que corre no PC do utilizador) modernos suportem.

4. Quanto à utilização de ASP.Net MVC 5.0 na construção de Websites com bases de dados, responda às seguintes questões!
   a) Que opções temos no ASP.Net MVC 5.0 para os workflows de utilização de base de dados relacionais na construção de um site usando essa framework? (7,5%)
   R: Atualmente temos duas opções para os workflows de utilização de base de dados relacionais, o "Code-first" e o "Database-first", antigamente também tinhamos acesso a um outro workflow denominado de "Model-first" mas foi descontinuado por apresentar vantagens semelhantes ao code-first mas um número de desvantagens superior, para além de apresentar alguma inflexibilidade em comparação com os outros dois métodos.
   b) Descreva sucintamente cada um deles e o porquê de existirem vários workflows. (10%)
   R: O code-first é a abordagem mais utilizada e consiste na construção de um modelo de dados em C#, a colocação de "DbSet"'s com as classes do modelo de dados correspondentes aqueles que nós queremos representar sob a forma de tabelas na base de dados no DbContext da aplicação e a aplicação consecutiva de migrações à DbContext sempre que se pretende "aumentar" a versão da base de dados, ou seja, sincronizar as tabelas e outros objetos da base de dados com o modelo de dados. O database-first é uma abordagem que atualmente cada vez mais se encontra em desuso e consiste na produção de um modelo de dados com base numa base de dados previamente existente de modo a permitir interfacar e aceder à base de dados de um modo semelhante ao método code-first. O database-first só deve ser utilizado em casos em que a base de dados contenha muitas tabelas, relacionamentos e outros dados relacionados à Schema da base de dados e em que a base de dados contenha uma quantidade elevada de dados. Isto porque, apesar do método code-first geralmente ser considerado melhor, no caso em que a base de dados já tenha uma elevada complexidade, é altamente provável que o programador não construa um modelo de dados exatamente igual à imagem da base de dados o que vai ter como consequência a impossibilidade de interfacagem com a mesma.
   c) Caso tivesse de construir um Website com ASP.Net MVC 5.0 qual dos workflows que referiu é que usaria e o porquê dessa escolha? (7,5%)
   R: Caso tivesse de construir um Website com ASP.Net MVC 5.0, o workflow pelo que eu optaria seria sempre o code-first porque presume-se que irei começar sem uma base de dados já existente e sem dados dos quais partir, para além disto existem inúmeras vantagens à abordagem code-first, como por exemplo a facilidade de adição de tabelas e atributos à base de dados e a deteção precoce de erros que possam vir a tornar-se maiores caso não sejam detetados a tempo.
5. Escolha a ou as opções que lhe parecem correctas:
   a) No caso da implementação de um website utilizando ASP.Net MVC 5.0 a utilização de HTML Helpers e de Razor só é necessária/possível se:
      - ~~É sempre necessário porque senão não seria possível utilizar-se nenhum SGBD?~~
      - ~~Depende de que browser vai ser utilizado para construir o Website!~~
      - Poderão não serem necessários, mas são quase sempre usados!
      - ~~A sua utilização é indiferente podendo serem usados ou não pois nem sempre se utiliza o Visual Studio no desenvolvimento de websites!~~

- ~~A sua utilização é possível somente se o website vai ser utilizado em Windows!~~
- A sua utilização implica a escrita de instruções mais compactas mas também mais complexas.
- Por causa da sua utilização é gerado código HTML!
- ~~Todas as opções indicadas estão correctas!~~
- ~~Todas as opções indicadas estão erradas!~~

b) Relativamente à resposta que deu na alínea a) desta pergunta, justifique a opção ou opções que selecionou! (10%)

- Poderão não serem necessários, mas são quase sempre usados!
  R: A utilização de HTML helpers e Razor permite-nos escrever instruções que em tempo de runtime gera HTML de modo a tornar a nossa página mais dinâmica, um bom exemplo disto é quando escrevemos @HTML.SelectListFor() que com base na coleção de dados que recebe, gera um select (em HTML) com um valor (value em HTML) para cada membro da coleção. Como é óbvio, num website muito simples sem a necessidade de geração de páginas estáticas através de processos dinâmicos, não precisamos de utilizar nem Razor (permite um Markup com código C# pelo meio) nem HTML helpers. O que acontece é que atualmente a grande maior parte dos websites já não é estático e acabamos sempre por precisar de utilizar frameworks ou ferramentas que nos permitam criar páginas com base nos dados que o modelo de dados (ou controlador no modelo MVC) nos fornece.
- A sua utilização implica a escrita de instruções mais compactas mas também mais complexas.
  R: No exemplo explicado acima, caso os dados fossem predefinidos (por exemplo no caso de uma lista constante de todos os valores possíveis num enum) seria possível escrevermos o código HTML todo à mão (isto é, se não quisessemos depois utilizar os valores selecionados no select list no controlador do modo tradicional) com os valores todos do enum colocados com o HTML tag <value></value> lá dentro, isto tornava o nosso código menos compacto e mais longo mas por outro lado tornaria o código mais simples de perceber e de escrever porque 1) olhando para o HTML saberíamos exatamente o que dali saía e 2) não teríamos de utilizar as várias técnicas de passagem de dados do controlador para a vista (como p.e. o viewbag/viewdata ou a utilização de um viewmodel).
- Por causa da sua utilização é gerado código HTML!
  R: Browsers modernos conseguem “entender” um conjunto de linguagens e ferramentas, como por exemplo HTML, CSS, JavaScript e com recurso a transpiladores como Babel (embutidos na maior parte dos browsers modernos, como é o caso do Google Chrome ou do Mozilla Firefox) até mesmo frameworks como o React, porém, este conjunto é limitado e não enquadra linguagens como C#. Sendo que o browser não “entende” C# out-of-the-box, a forma de conseguirmos apresentar o conteúdo de uma Razor view (página com código HTML misturado com C#) é através da transformação em runtime do Razor Markup para HTML. Isto verifica-se, por exemplo, no caso referido acima (SelectList).