Exame de P. Web da E.N. de 2021/2022

1. Considere o seguinte trecho de código num site desenvolvido em ASP.Net Core com C# 9:

```
…
var alunosExcelentes = db.Alunos.Where(a => a.Media >= 18)
        .OrderBy(a => a.Nome)

foreach (var aluno in alunosExcelentes)

        Console.WriteLine(aluno.Nome);

…
```

   Considerando que o código antes e depois deste trecho (assinalado por …) está correcto, este trecho de código estará também correcto? Se sim explicar o que faz; se não estiver correcto explicar o que não está e o porquê de não estar correcto, corrigir e mostrar o que faz após a correcção! Em qualquer dos casos, justifique detalhadamente a sua resposta! (15%)

   R: O trecho de código está correto. Atribui a alunosExcelentes a coleção de todos os alunos com média superior ou igual a 18 ordenado por nome de aluno. Após esta atribuição, itera por todos os alunos em alunosExcelentes e escreve o seu nome numa linha no ecrã.

2. Considerando a programação de Websites utilizando-se ASP.Net Core:
   a) Descreva detalhada e justificadamente a utilização de View Data, View Bag e Model Views salientando as vantagens/desvantagens de cada uma destas possibilidades e quando é que se devem utilizar, preferencialmente, cada uma delas, o porquê dessa utilização e também o porquê de se utilizar cada uma dessas possibilidades em vez de uma das outras! (15%)
      R: O View Data é um conjunto de mapeamentos de chave para valor, pode conter qualquer tipo de objeto (inclusive null) e habitualmente é utilizado para passar dados do controlador para a vista mas também pode ser usado para passar dados da vista. O View Bag é uma propriedade dinâmica resultante da abstração do View Data que permite ir buscar valores através do síntaxe ViewBag.ExemploChave enquanto que no ViewData só é possível através do síntaxe ViewData[“ExemploChave”]. O ViewBag é ligeiramente menos eficiente do que o ViewData. Tanto o ViewData como o ViewBag contêm atributos fracamente tipados o que é “error-prone” e “bug-prone”. Os View Models são modelos definidos pelo programador que servem para mostrar dados provenientes do modelo de dados nas vistas e tradicionalmente só contêm mesmo os dados necessários à apresentação de conteúdo nas vistas que a utilizem. View Models, sendo simplesmente classes, têm atributos fortemente tipados e por isso devem ser preferidos à ViewBag e ViewData. Normalmente os dados que colocamos no ViewBag e ViewData são de tamanho reduzido.
   b) Se tiver que se utilizar Model Views, qual é normalmente a base de partida para a definição de um Model View? Justifique a sua resposta. (10%)
      R: A base de partida para a definição de um View Model é a necessidade (a nível de apresentação) da vista que faça uso do mesmo. Se por exemplo numa dada vista

precisarmos de mostrar um título e uma lista de alunos, podemos definir uma ViewModel da forma seguinte:

```
class MyViewModel
{
    public string Title;
    public List<Aluno> Alunos;
    public MyViewModel(string title, List<Aluno> alunos)
    {
            this.Title = title;
            this.Alunos = alunos;
    }
}
```

Depois, quando a quisermos passar à vista, é só (no action method correspondente à vista no controlador) devolver a View(myViewModel), obviamente com o MyViewModel myViewModel instanciado.

3. Considerando a programação em C# 9:
   a) Refira o que se entende por Delegates e que vantagens considera que podem advir da sua utilização!(10%)
   R: Delegates são um tipo cuja instância permite guardar uma referência a um método (do mesmo tipo que o método) e a principal vantagem que nos traz é a possibilidade de passar referências a métodos a outros métodos que assim o esperem (devem ter uma variável do tipo do delegate no protótipo). Não é um conceito novo, a sua base já se encontra implementado em outras linguagens (como no caso do C/C++ - function pointers) mas delegates são altamente poderosos e permitem fazer coisas como definir métodos com comportamentos dinâmicos com base no(s) delegate(s) que receberem.
   b) Comente a seguinte afirmação, justificando devidamente o comentário que fizer! (15%)
   "É indiferente a utilização de Delegates do tipo referência ou do tipo valor, as circunstâncias da utilização de um destes tipos ou do outro em C# 9 depende do código que se estiver a desenvolver!"
   R: A meu ver a afirmação está incorreta porque a utilização de Delegates do tipo referência não é indiferente da utilização de Delegates do tipo valor. No primeiro caso, os argumentos do delegate podem ser alterados (ao alterar as suas referências) no método que a encapsule, no segundo não. Por outro lado, é verdade que as circunstâncias de utilização de um destes tipos ou do outro em C# 9 depende do código que se estiver a desenvolver, apesar disso não especificar nada sobre essas próprias circunstâncias.
4. No desenvolvimento de sites em ASP.Net Core com uma fonte de dados associada ao mesmo para persistência de dados:
   a) Refira, detalhando e justificando a sua resposta, se há necessidade e interesse em se utilizar Entity Framework Core! (10%)
   R: Não existe a necessidade de utilizar Entity Framework Core para interfacar com uma fonte de dados persistente em sites desenvolvidos com ASP.Net Core mas existe um grande interesse em assim o fazer porque Entity Framework Core disponibiliza features imprescindíveis ao desenvolvimento do site como por exemplo o acesso direto às entidades da base de dados através do DbContext.NomeDaEntidadeComSNoFim, a disponibilização do

LINQ (Language-Integrated Query – notações altamente poderosos (Query-Syntax e Extension Methods)) que nos permite escrever código C# que é transformado em SQL para ir buscar dados à base de dados, a disponibilização de métodos de utilização simples como DbContext.Add(InstânciaDeClasseQueMapeiaParaEntidadeNaBaseDeDados) que insere dados na base de dados e o DbContext.SaveChanges() que é semelhante ao Commit em SQL.

b) Como é que a Entity Framework Core pode ser utilizada e que opções da sua utilização podem ser consideradas aquando do inicio do desenvolvimento de um site, por exemplo se se trata de um site a ser desenvolvido de raiz ou se se trata de um site já existente e que vai ser objecto de uma actualização! (7,5%)
R: As opções de utilização do Entity Framework Core são nomeadamente duas (já foram três – com o "Model First", opção onde se partia de um diagrama de modelo e se autogerava a base de dados a partir daí), o "Code First" e o "Database First". O "Code First" consiste numa opção de utilização de base de dados onde se parte de um modelo de dados (código C# 9) e se gera e se vai moldando a base de dados através da aplicação de migrações (tecnicamente incorreto mas por outras palavras sincronizações da base de dados ao modelo de dados) à mesma através dos comandos Add-Migration <NomeMigração> e Update-Database. O "Database First" é uma outra opção de utilização da base de dados que está a cair cada vez mais em desuso onde se parte de uma base de dados previamente criada (tabelas e entradas das tabelas já definidas) e se gera e se vai moldando o modelo de dados (que aqui serve mais como um mapeamento da base de dados de modo a permitir o seu acesso a partir da aplicação através do Entity Framework Core) a partir da base de dados, neste modo para causar alterações na base de dados deverá ser feito exteriormente e com ferramentas exteriores. Normalmente deve-se sempre optar pela opção "Code First" a não ser que já se tenha uma base de dados com alta complexidade e com um número alto de entradas porque permite a alteração da base de dados de forma programática e dentro da aplicação e é menos "error-prone".

c) Refira também qual é o papel do modelo de dados desenvolvido em C# 9 quando se utiliza Entity Framework Core para se ter persistência de dados! (7,5%)
R: O papel principal do modelo de dados desenvolvido em C# 9 quando se utiliza Entity Framework Core é mapear as tabelas da base de dados para classes de modo a permitir que as entradas das mesmas tabelas sejam mapeadas para instâncias dessas mesmas classes. Assim, conseguimos trabalhar de forma programática e sem termos que fugir à orientação a objetos com os objetos existentes na base de dados. No workflow "Code First" de uso da base de dados também é esse modelo de dados que define as tabelas da base de dados. No workflow "Database First" de uso da base de dados é a base de dados que define o modelo de dados.

5. ~~Responda somente a 1 das seguintes questões identificando claramente a qual está a responder indicando para isso o número da questão a que está a responder~~: (10%)

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.
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.
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.