Programação Visual
Trabalho de Laboratório nº 4

Objetivo	MVC – Introdução. Aplicações básicas com definição de regras de encaminhamento e secções.
Programa	Pretende-se criar o *site* da papelaria da EST para a venda *online* de artigos.
Regras	Implementar o código necessário e testar no fim de cada nível. Use as convenções de codificação adotadas para a linguagem C# e para o modelo MVC.

Descrição

Nível 1

- Crie uma nova aplicação **ASP.NET Core Web Application**.
  Dê-lhe o nome "**ESTaPapelaria**" e selecione o *template* **ASP.NET Core Web Application (Model-View-Controller).**
- Nesta aplicação pretendemos ter uma página de entrada no site apenas com uma imagem que inclui um *link* que nos leva à página principal que irá ser criada. Sendo assim, apague a ação **`Privacy`** e todas as vistas do controlador **Home.** Crie agora uma nova vista para a ação **`Index`**, esta vista **não usa uma página de *layout***. Altere a **View** criada para que mostre a imagem disponibilizada com este enunciado.
- A página principal, que ainda não foi criada irá usar o *layout* que também é fornecido com este enunciado. Para isso, coloque na pasta **shared** das vistas a página **_ESTLayout.cshtml** fornecida.
- Crie agora um novo controlador **`PapelariaController`** usando o *template* **Empty MVC Controller** e defina uma vista para a ação **`Index`** desse controlador. Neste caso, use como *layout* a página **_ESTLayout.cshtml** que adicionou.
- Altere agora a **View** da ação **`Index`** do controlador **Papelaria** que deverá mostrar o texto "**EST – a – Papelaria**".
- Na página de entrada do site, inclua um *link* associado à imagem da papelaria que leve para a ação **`Index`** do controlador que acabou de definir.
- Compile o projeto e visualize o resultado obtido. Redimensione o tamanho da página para que fique com as dimensões de um smartphone e visualize o resultado.

Programação Visual
Trabalho de Laboratório nº 4

Nível 2

- Para o modelo, crie uma nova classe **`Artigo`** que será usada para representar os artigos divulgados e vendidos no *site*. Defina para a classe criada as seguintes propriedades implícitas: <**`Id`**, **`Nome`**, **`Descricao`**, **`Preco`**, **`Desconto`**, **`EmPromocao`**>.
- Para efeitos de teste, crie na classe **`Program`** uma lista pública e estática de artigos com o nome **`BDArtigos`** e preencha-a com 3 itens: <**2**, "**Esferográfica EST**", "**Esferográfica de gel**", **1.99**, **0.10**, **true**>, <**5**, "**Pen EST**", "**Pen de 16GB**", **7.90**, **0.15**, **false**> e <**6**, "**Clip EST**", "**Clips coloridos**", **2.99**, **0.15**, **true**>.
- Crie um controlador **Artigos** (*template* **empty**) e nessa classe defina como atributo uma lista de artigos que deverá ir buscar a informação à lista definida antes:

```
private List<Artigo> artigos = Program.BDArtigos;
```

- Crie a ação **`Listar`** dentro do controlador **Artigos**. Associe a esta ação uma nova **view** com base no *template* **List** e no modelo **Artigo**, usando como página de *layout*, *a página* **_ESTLayout.cshtml** adicionada antes. Teste.
- Para que possa fornecer um *footer* diferente nas páginas que criou, substitua o elemento **<footer>** por uma **secção** com o nome **Footer** utilizando o *html helper* **`RenderSection`**. Parametrize-a como não obrigatória (**`required: false`**).
- Coloque na **View Index** do controlador **Papelaria** uma "*@section Footer*" contendo o texto: "**Morada: EST Setúbal - Estefanilha - Telefone: 265 790 000**" e na **View Listar** uma secção idêntica o texto "**Veja as nossas promoções**".
- Para facilitar a criação de páginas no futuro, pode alterar a página **_ViewStart** na pasta das **Views** substituindo, na definição do *layout* por omissão, a página `"_Layout"` por `"~/Views/Shared/_ESTLayout.cshtml"`.

Nível 3

- Acrescente agora uma ação **`promocoes`** ao controlador **Artigos**. Esta ação deve filtrar a lista de artigos e mostrar apenas os artigos em promoção. Use LINQ para a filtragem. Teste e verifique que a opção de menu para as promoções ficou a funcionar.
- No *footer* da vista "**listar.cshtml**" coloque um link na palavra "promoções" para a ação anterior.
- Para simplificar o acesso ao site, crie agora as seguintes regras:
  - O acesso à ação **index** do controlador **Papelaria** deve fazer-se com o URL: "nome-do-site/**pap**".
  - O acesso à página com a lista de artigos deve fazer-se com o URL: "nome-do-site/**pap/artigos**".
  - O acesso à página com as promoções deve fazer-se com o URL: "nome-do-site/**pap/promocoes**".

Programação Visual
Trabalho de Laboratório nº 4

Nível 4

- Para que o utilizador possa mais facilmente encontrar os artigos que procura quando seleciona a opção **Listar** do menu que está no *site*, inclua na vista associada um *form* que recebe o texto da filtragem. Este texto deve ser enviado de volta ao servidor, para uma ação **`Listar`** do mesmo controlador, usando agora o método **Post**. A ação que recebe o texto deve usá-lo para filtrar o nome dos artigos. Neste caso, a lista dos artigos que contiverem o texto recebido deve ser enviada de volta para a mesma vista **Listar**.

Nível 5

- Pretende-se agora poder editar um artigo da papelaria. De facto, a vista **Listar** gerada já inclui um *link* para uma ação **`Edit`** que ainda não está criada. Sendo assim, comece por alterar o link substituindo-o por:

```
@Html.ActionLink("Editar", "Editar", new { id=item.Id })
```

- A alteração efetuada implica que se irá chamar a ação **`Editar`** e que esta irá receber o valor do **Id** do artigo como parâmetro. Sendo assim crie a ação **`Editar(int? id)`** dentro do controlador **`ArtigosController`**. Dentro da ação deve procurar, na lista de artigos, o artigo com o **id** recebido e passar esse artigo para a vista. Crie depois a vista com base no *template* **Edit** e no modelo **`Artigo`**. Teste e verifique que a página é mostrada com os dados do artigo quando se seleciona o *link* do mesmo na página **Listar**.
- Para completar a edição do artigo, deve guardar as alterações efetuadas diretamente no artigo respetivo da coleção de artigos. Neste caso, o botão **Save** da vista de edição vai chamar novamente a ação **`Editar`** mas agora usa o verbo **Post** o que leva a que seja chamado um método diferente no controlador. Sendo assim, crie agora, no controlador dos artigos, a ação **`Editar(int id, Artigo artigo)`** e decore a ação com o atributo **`[HttpPost].`** Dentro da ação, procure o artigo com o **`id`** recebido e altere toda a informação desse artigo com os dados recebidos do utilizador que vêm dentro do parâmetro **`artigo`**. Depois chame a vista com o artigo alterado. Atenção, não mude o valor do **`Id`** no artigo da coleção. Para segurança, pode mesmo apagar a edição desta propriedade na vista associada.
  Nota: Nesta aplicação não se preocupe em verificar os dados recebidos dado que apenas se pretende demonstrar o funcionamento da ação de editar.

Desafio

- Implemente as restantes ações CRUD de criação de um novo artigo, apagar um artigo e visualizar os detalhes de um artigo. Pode usar os *templates* respetivos na criação das vistas associadas.

Notas

Para os identificadores siga as convenções adotadas pelo C#, nomeadamente:

- A notação camelCase para o nome das variáveis locais e identificadores privados.
- A notação PascalCase para os nomes públicos dos métodos e classes
- Não utilize o símbolo '_' nos identificadores
- Não use abreviaturas