Programação Visual
Trabalho de Laboratório nº 4

Objetivo	MVC – Introdução. Aplicações básicas com utilização de *Layouts* responsivos.
Programa	Pretende-se criar o *site* da papelaria da ESR 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 **Visual C#-> Web -> ASP.NET Core Web Application**. Selecione o *template* **Empty** e dê-lhe o nome "**ESTaPapelaria**".
- Crie uma pasta **Controllers** e adicione-lhe um controlador **Home** usando o *template* **MVC Controller - Empty**. Crie depois uma ação (eventualmente já estará criada aquando da criação do controlador).
- O controlador que criou tem apenas uma ação **`Index`** que ainda não possui *uma* **View** associada. Sendo assim, adicione-lhe uma **View** (não selecione *partial view* e não escolha *com layout*).
- Altere a **View** criada para que mostre a imagem disponibilizada com este enunciado.

Nota: para configurar a aplicação como uma aplicação **Asp.net MVC** deve ter o seguinte código na classe **`StartUp`**:

```
public class Startup
{
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddMvc();
    }

    public void Configure(IApplicationBuilder app, IHostingEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }

        app.UseStaticFiles();

        app.UseMvc(routes =>
        {
            routes.MapRoute(
                 name: "Default",
                 template: "{controller}/{action}/{id?}",
                 defaults: new {controller = "Home", action = "Index"}
            );
        });
    }
}
```

Programação Visual
Trabalho de Laboratório nº 4

Nível 2

- Vai agora criar um *layout* para utilizar nas páginas do site. Sendo assim, adicione à aplicação uma pasta **Shared** dentro da pasta das vistas e a seguir adicione-lhe uma nova **View** com o nome **_ESTLayout**. Depois, substitua o **html** da página por:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!--aqui vão colocar-se os links para os módulos JavaScript e CSS necessários-->
  </head>
  <body>
    <!-- aqui vai colocar-se o conteúdo da pagina/layout… -->
  </body>
</html>
```

- Adicione as seguintes linhas dentro da ***tag*** **<head>**:

```
<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />
<link rel="stylesheet" href="~/css/site.css" />
```

- Dentro da ***tag*** **<body>** coloque uma *tag* **<nav>** (ou seja, *navigator*):

```
<nav id="myNavbar" class="navbar navbar-default navbar-inverse navbar-fixed-left"
role="navigation">
    <!–irá colocar aqui as tags <div class="navbar-header"> e
    <div class="navbar-collapse collapse"> conforme se descreve abaixo -->
</nav>
```

- Dentro da ***tag*** **<nav>**, coloque uma *tag* ** **, de forma a ter-se um *layout* que se adapta a dispositivos móveis:

```
<div class="container-fluid">
</div>
```

- Dentro da ***tag*** ** **, ponha a ** ** que vai conter o botão que permite colapsar as opções do navegador, quando acede à página num dispositivo móvel. Faça assim:

```
<div class="navbar-header ">
   <button type="button" class="navbar-toggle" data-toggle="collapse"
    data-target=".navbar-collapse"> <!-- num smartphone, esconde opções do menu-->
      <span class="sr-only">Toggle navigation</span> <!--for screen readers-->
      <span class="icon-bar"></span> <!--para criar o symbolo “≡” dentro do botão-->
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
   </button>
   <a class="navbar-brand collapse navbar-collapse" href="#">EST-a-Papelaria</a>
</div>
```

- Dentro do ** ** e a seguir a ** **, coloque as opções de menu disponíveis para o *site*. Mais tarde, irá modificar estas opções através de *html helpers*. Faça assim:

```
<div class="navbar-collapse collapse" >
   <ul>
      <li class="active"><a href="#" target="_blank">Home</a></li>
      <li><a href="#" target="_blank">Listar produtos</a></li>
      <li><a href="#" target="_blank">Contactos</a></li>
   </ul>
</div>
```

- Dentro da *tag* **<body>** e a seguir à *tag* **<nav id="myNavbar" …>**, coloque a chamada à instrução *Razor* que vai permitir inserir as vistas parciais e a *tag footer*. Faça assim:

```
<div class="container body-content container-fluid">
   <!--as vistas parciais vão ser geradas (renderized) aqui-->
   @RenderBody()
   <hr />
   <footer>
       <!-- &copy é substituído pelo símbolo copyright-->
       <p>&copy; @DateTime.Now.Year – EST-a-Papelaria Online</p>
   </footer>
</div>
```

Programação Visual
Trabalho de Laboratório nº 4

- Coloque agora uma *tag* **<style>** na **<head>** do documento, para definir o tamanho e posição do menu *navigator*, que vai ficar à esquerda da página. Faça assim:

```
<style>
   .navbar-fixed-left {
       position: fixed;  /* fixar o navegador */
       border-radius: 0;
       height: 100%;     /* ocupar todo o tamanho na vertical*/
   }

   .container {
       padding-left: 220px; /*faz o render a 220px da esquerda*/
   }
</style>
```

- Por fim, antes da *tag* final **</body>** coloque as seguintes:

```
<script src="~/lib/jquery/dist/jquery.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.js"></script>
<script src="~/js/site.js" asp-append-version="true"></script>

@RenderSection("Scripts", required: false)
```

- Copie agora o conteúdo do ficheiro **wwwroot.zip** para dentro da pasta respetiva.
- Crie um novo controlador **PapelariaController** e defina uma vista para a ação **index**. Neste caso, use como *layout* a página `_ESTLayout.cshtml` que criou.
- Para ter acesso à página que criou, na *view* **index** do controlador *home* coloque a imagem dentro do seguinte componente html:

```
<a href="/Papelaria/Index">
    <!-- Coloque a imagem aqui  -->
</a>
```

- Compile o projeto e visualize o resultado obtido. Redimensione o tamanho da página de forma a que fique com as dimensões de um smartphone e visualize o resultado.

Programação Visual
Trabalho de Laboratório nº 4

Nível 3

- Para o modelo, crie a pasta "**Models"** e acrescente-lhe 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`**>.
- Crie um controlador **Artigos** (*template* **empty**) e nessa classe defina como atributo uma lista de artigos e preencha-a com 3 items, por exemplo: <**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 agora uma nova ação **`Listar`** dentro do controlador **Artigos**. Associe a esta ação uma nova **view** que irá mostrar a lista de artigos. Neste caso, passe para a vista associada a lista de artigos e crie a vista com base no *template* **List** e no modelo **Artigo**, usando como página de *layout*, *a página* **_ESTLayout.cshtml** criada antes.
- Altere agora a **View** da ação **`Index`** do controlador **Papelaria** que deverá mostrar o texto "**EST – a – Papelaria**".
- Para fazer a ligação entre as páginas existentes adicione *links* ao *layout* que definiu para as páginas **Index** e **Listar** dos controladores, respetivamente, **Papelaria** e **Artigos**. Faça-o dentro do layout **_ESTLayout.cshtml**. Nota: para definir os *links* use *Html helpers* ou *tag helpers*.
- Teste.
- Para que possa fornecer um *footer* diferente nas páginas que criou, substitua o *footer* estático que está na página de *layout* por um dinâmico. Para isso, crie uma secção com o nome **Footer** na parte inferior da página, utilizando o *html helper* **`RenderSection`**, com o nome **"footer"** e parametrizada 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** o texto "**Veja as nossas promoções**".
- Para facilitar a criação de páginas no futuro, e caso ainda não exista, pode criar uma página **_ViewStart** na pasta das **Views** e colocar lá a seleção da página de *layout* que é feita nas várias *views*. (`Layout = "~/Views/Shared/_ESTLayout.cshtml";`).

Nível 4

- 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. Inclua uma opção de menu na página de layout para as promoções.
- 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 5

- 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**.

Desafio

- Crie uma vista parcial que mostra a informação de um único artigo e utilize-a na vista "**listar.cshtml**" para visualizar os vários artigo. A visualização de cada artigo deve estar devidamente formatada para que seja agradável para o utilizador.

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