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* privado **ESTeCard** de gestão de cartões pré pagos.
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 App (Model-View-Controller)**. Dê-lhe o nome "**ESTeCard**".
- Na vista **Index** do controlador **Home**, mostre a imagem fornecida com este enunciado.
- Apague a ação **`Privacy`** do controlador **Home** e remova do menu do *site* esta opção. Comente também a opção **Home.** O acesso à página de entrada vai-se fazer pela opção de menu que tem o nome da aplicação.

Nível 2

- Para o modelo, crie uma classe **`PrePaidCard`** que será usada para os cartões pré pagos geridos pelo *site*. Defina para a classe criada as seguintes propriedades implícitas: <**`Id`**, **`Name`**, **`Entity`**, **`Code`**, **`Credit`**>. **`Id`** é um *long*, **`Code`** é um inteiro, **`Name`** e **`Entity`** são *string* e **`Credit`** é um valor decimal.
- Para efeitos de teste, crie a classe **`EstCards`** numa nova pasta **Data** dentro do projeto. Dentro da classe crie um campo estático **cards** com uma lista de cartões e preencha-o com 3 cartões: <**1234567887654321**, "**José Martins**", "**ESTeBank**", **5123**, **320.0**>, <**3536776512871456**, "**Carlos Santos**", "**ESTeBank**", **4141**, **30.0**> e <**7786209856342765**, "**Rui Rosa**", "**ESTeBank**", **2723**, **5230.0**>. Crie na mesma classe a propriedade estática **`ESTeCards`** de leitura que retorna a lista criada.
- Crie um controlador **Cards** (*template* **MVC Controller - Empty**) e nessa classe defina um campo com uma lista de cartões que contem os cartões criados antes:

```
private List<PrePaidCard> cards = EstCards.ESTeCards;
```

- Associe à ação **`Index`** do controlador que criou uma nova **Razor View** com base no *template* **List** e no modelo **PrePaidCard**. Na página principal do site, inclua uma opção de menu **EstCards** que dá acesso a esta vista.
- Comente na vista criada os links de **edit**, **details** e **delete** e elimine a coluna que visualiza o código do cartão.
- Para que possa fornecer um *footer* diferente nas páginas do *site*, substitua o elemento **<footer>** da página de **Layout** por uma **secção** com o nome **Footer** utilizando o *html helper* **`RenderSection`**. Parametrize-a como não obrigatória (**`required: false`**).
- Coloque no fim da **View Index** do controlador **Home** uma "*@section Footer*" contendo o texto: "**Address: EST Setúbal - Estefanilha - Phone: 265 790 000**".

Programação Visual
Trabalho de Laboratório nº 4

Nível 3

- Acrescente agora a ação **`Create`** ao controlador **Cards**. Associe a esta ação uma nova **Razor View** com base no *template* **Create** e no modelo **PrePaidCard**. Na vista criada, comente os elementos ligados à atribuição do código.
- A ação anterior leva a que seja mostrada uma vista para receber a informação de um novo cartão. Quando se preenchem os dados do novo cartão e se submetem ao servidor é feito um pedido com o verbo **HttpPost** para uma ação **`Create`**. Uma vez que a ação **`Create`** criada responde apenas a pedidos **HttpGet**, deve acrescentar uma nova ação **Create** com o atributo **HttpPost** que tem como parâmetro um objeto **`card`** do tipo **`PrePaidCard`**. Dentro desta ação, se o parâmetro **`card`** vier com o valor **null** deve voltar à vista associada, caso contrário, deve atribuir um código aleatório entre 0 e 9999 à propriedade **`Code`** do parâmetro **`card`** e adicionar este objeto à coleção de cartões definida. No final deve voltar à ação **`Index`**.

Nível 4

- Para facilitar ao utilizador encontrar os cartões que procura quando seleciona a opção **Cards** do menu que está no *site*, inclua na vista **Index** associada um *form* que recebe o texto da procura do cartão pelo seu **Id**. Este texto deve ser enviado de volta ao servidor, para uma nova ação **`Index`** do mesmo controlador. Esta ação contém o atributo **HttpPost** e recebe o texto que vai ser usado para filtrar o Id dos cartões. Neste caso, a lista dos cartões que contiverem o texto recebido deve ser enviada de volta para a mesma vista **Index** para ser mostrada.
- Na pasta **Cards** que está dentro da pasta **Views** acrescente a uma vista parcial com o nome **_CreditCard**. Dentro desta vista parcial mostre a imagem **CreditCardIcon** fornecida e ao lado a data atual.
- Inclua a vista parcial criada por cima do texto **`<h1>Create</h1>`** que está na vista **Create.cshtml** da mesma pasta.

Nível 5

- Pretende-se agora poder editar um cartão de crédito De facto, a vista **index** gerada já inclui um *link* para uma ação **`Edit`** que foi comentada porque não estava criada. Sendo assim, comece por descomentar o link substituindo-o depois por:

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

- A alteração efetuada implica que se irá chamar a ação **`Edit`** e que esta irá receber o valor do **Id** do cartão como parâmetro. Sendo assim crie a ação **`Edit(long? id)`** dentro do controlador **`Cards`**. Dentro da ação deve procurar, na lista de cartões, o cartão com o **id** recebido e passar esse cartão para a vista. Crie depois a vista com o *template* **Edit** e o modelo **`PrePaidCard`**. Teste e verifique que a página é mostrada com os dados do cartão quando se seleciona o *link* do mesmo na página **Index**.

Programação Visual
Trabalho de Laboratório nº 4

- Tal como fez na vista da ação **`Create`**, inclua a vista parcial **_CreditCard** por cima do texto <h1>Edit</h1> que está na vista **Edit.cshtml**.
- Para completar a edição do cartão, deve guardar as alterações efetuadas diretamente no objeto respetivo da coleção de cartões. Neste caso, o botão **Save** da vista de edição vai chamar novamente a ação **`Edit`** 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 cartões, a ação **`Edit(long id, PrePaidCard card)`** e anote a ação com o atributo **`[HttpPost].`** Dentro da ação criada, procure o cartão com o **`id`** recebido e altere toda a informação desse cartão com os dados recebidos do utilizador que vêm dentro do parâmetro **`card.`** Atenção, não mude o valor do **`Id`** no cartão da coleção. Para segurança, pode mesmo tornar apenas de leitura o elemento html **input** usado na vista para editar esta propriedade. Depois, em vez de voltar a chamar a vista, redirecione para a ação **Index**.
  Nota: Nesta aplicação não se preocupe em verificar os dados recebidos dado que apenas se pretende demonstrar o funcionamento da ação de editar.
- Inclua na

Desafio

- Implemente as restantes ações CRUD para apagar um cartão e para visualizar os detalhes de um cartão. 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