Programação Visual

Trabalho de Laboratório nº 8

Objetivo	MVC – Introdução. Utilização de *ViewComponents* e utilização/definição de serviços
Programa	Pretende-se continuar o desenvolvimento da aplicação **MiniESTS** .
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

- Descarregue e descompacte o ficheiro "**Lab 8 – Materiais**" fornecido com este enunciado. Depois de descompactar, compile, crie a migração inicial e a base de dados.
- Execute a aplicação e confirme que está a funcionar. Deve encontrar já alguns produtos quando abre os detalhes de qualquer uma das lojas.
- Dentro do projeto **MiniESTS**, pretende-se agora criar um **ViewComponent** que mostre o total de produtos em promoção disponíveis numa determinada loja. Para este efeito, comece por criar este componente e na vista associada escreva uma tabela **com o texto mostrado**:

> **Produtos**
> Vários produtos em promoção disponíveis

- Teste este componente, colocando-o dentro na página **Index** do Controlador **Home** em substituição do texto que lá está.

Programação Visual
Trabalho de Laboratório nº 8

Nível 2

- O componente anterior está incompleto porque ainda não é mostrado o número de produtos de uma determinada loja. Para que isso aconteça, ir-se-á criar um serviço a partir do qual se pode obter a informação dos produtos. Crie então, dentro da pasta **Services**, a seguinte interface:

```
public interface IProdutosEmPromocaoNaLoja
{
    IEnumerable<Produto> ProdutosEmPromocao(string localidade);
}
```

A seguir, crie uma classe `ProdutosComDesconto` que implementa a interface anterior. Dentro desta classe, obtenha o contexto por *dependency injection* (como é feito dentro do controlador **Produtos**) e, a partir desse contexto, retorne a lista de produtos com desconto da loja da localidade recebida como parâmetro do método.

- Para usar o serviço criado, registe-o na classe **`Program.cs`** como *Scoped*:

```
builder.Services.AddScoped<IProdutosEmPromocaoNaLoja,ProdutosComDesconto>();.
```

Para testar o serviço, obtenha a lista de produtos da loja de "Setúbal" **diretamente na págin**a **Index** do controlador **Home** usando a injeção de dependência do serviço nesta vista (consulte https://docs.microsoft.com/en-us/aspnet/core/mvc/views/dependency-injection?view=aspnetcore-6.0). Deverá aparecer o texto "**Existem X produtos em promoção em Setúbal!**". O número de produtos é obtido a partir do número de elementos da lista de produtos "***Stock***".

Nível 3

- Altere o *ViewComponent* criado no nível 1 de forma que este receba a lista de Produtos disponíveis numa determinada loja, usando o serviço `IProdutosEmPromocaoNaLoja`, e passe o total de produtos para a vista associada. A localidade da loja deve ser recebida como parâmetro do método **`InvokeAsync`** do **ViewComponent**. Com esta alteração, a vista do **ViewComponent** deve passar a ser tipificada com o modelo **`int.`** Corrija a vista para que passe a obter o número de produtos da Loja de "Setúbal" passando a mostrar:

Produtos
2 produtos em promoção disponíveis

Corrija a utilização deste componente na vista **Index** do controlador **Home** colocando como argumento, na invocação do componente, a localidade "Lisboa".

Programação Visual
Trabalho de Laboratório nº 8

Nível 4

- Infelizmente o nosso **ViewComponent** não mostra a loja onde existem produtos disponíveis, nem o tipo de produtos. Para que possam ser visualizados o número de produtos, a localidade da loja e os tipos de produtos disponíveis crie um **ViewModel** com propriedades para estes valores e use este **ViewModel** na vista do **ViewComponent.** Neste caso o texto a mostrar deve incluir a informação recebida. Por exemplo: "5 produtos em promoção disponíveis em Lisboa. Os produtos em promoção são dos tipos: Bebidas, Carnes, Congelados". Implemente ainda um *dropdown* que possibilite ao utilizador escolher a loja que é passada ao **ViewComponent**.

Nível 5

- Para completar a aplicação, pretende-se criar e usar um serviço de *email* que permita a confirmação do *email* fornecido quando um utilizador se regista. Neste caso, vamos utilizar a biblioteca **MailKit**. Sendo assim execute o seguinte comando NuGet:

```
Install-Package MailKit
```

- Para usar este serviço de email, altere a classe ***EmailSender*** que está dentro da pasta ***Services***. O código desta classe está no ficheiro **EmailSender.txt**, fornecido com os materiais que acompanham este laboratório**.**
- Coloque a entrada ***EmailSender*** (fornecida no ficheiro **EmailSenderJson.txt**) no ficheiro de configurações **appsettings.json**. No texto, estão as definições de uma conta de email que foi criada para este laboratório e que será removida posteriormente.
- Substitua o registo do serviço por:

```
builder.Services.AddTransient<IEmailSender, EmailSender>(i =>
   new EmailSender(
        builder.Configuration["EmailSender:Host"],
        builder.Configuration.GetValue<int>("EmailSender:Port"),
        builder.Configuration.GetValue<bool>("EmailSender:EnableSSL"),
        builder.Configuration["EmailSender:UserName"],
        builder.Configuration["EmailSender:Password"]
   )
);
```

- Altere a configuração do serviço de **Identity** para que seja necessário a confirmação do email do registo.
- Teste registando-se com um email pessoal válido. Verifique que recebeu o pedido de confirmação do registo.

Desafio

- Crie um controlador para os utilizadores que receba a lista dos utilizadores a partir de um serviço criado para o efeito.

Programação Visual

Trabalho de Laboratório nº 8

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 de qualquer forma '_' nos identificadores
- Não use abreviaturas