Programação Visual
Trabalho de Laboratório nº 6

Objectivo	MVC – Introdução. Aplicações básicas com acesso a base de dados através do *Entity Framework* e utilização do pacote *Microsoft Identity*.
Programa	Pretende-se continuar o desenvolvimento da aplicação **EsteCar**, em que desta vez, a aplicação deverá permitir aos utilizadores visualizar os carros e proceder ao aluguer de um veículo por um período de tempo. Os utilizadores deverão poder autenticar-se e alterar o seu perfil.
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 aplicação **ASP.Net Core Web Application**, **MVC**, dê-lhe o nome de **EstCarII** e escolha como método de autenticação **Individual User Accounts** .
- Na diretoria **Models**, crie os seguintes modelos (pode obtê-los da resolução do laboratório anterior – **lab5**):
  a. **`Carro`** – **`CarroId, Modelo, NumeroDePortas, EmissoesCO2, TipoDeCaixa, MarcaId.`**
  b. **`Marca – MarcaId, Designacao.`**
- Acrescente uma propriedade de texto **`FicheiroFoto`** à classe **`Carro`**, com uma dimensão máxima de **`255`**. Crie uma pasta chamada **Fotos** dentro da diretoria **wwwroot**.
- Gere, por *scaffolding*, os controladores MVC para as Marcas e para os Carros. Selecione a opção que inclui as vistas usando a ***Entity Framework***. Em ambos os casos, use o contexto **ApplicationDbContext** criado antes pelo **Microsoft Identity**.
- Adicione à barra de menus os *links* para a *view Index* dos controladores que criou e corrija as várias vistas do controlador **Home** de forma a que estejam personalizadas para esta aplicação.
- Compile a aplicação e aplique as modificações na base de dados, aplicando o habitual **add-migration** seguido do **update-database** no **package Manager Console** (**PMC**).
- Verifique apenas que está a funcionar com as alterações efetuadas. Não adicione nenhuma marca ou carro.

Nível 2

- Para definir um conjunto de dados iniciais, na pasta dos dados (**Data**) crie uma classe **`DbInitializer`** e dentro desta classe defina o seguinte método:

```
public static async Task Initialize(ApplicationDbContext context)       {
    context.Database.EnsureCreated();
    if (!context.Marca.Any())         {
        // Adicionar Marcas para testes
        context.SaveChanges(); // é necessário gravar de imediato
                               // para se poder usar as FK nos carros
    }
    if (!context.Carro.Any())     {
        // Adicionar Carros para testes
    }
}
```

- Acrescente algumas **Marcas**: Ferrary, Porche, BMW, …
- Acrescente alguns **Carros**: (1, TestaRossa, …), (2, 911 Carrera, …), …
  (o primeiro campo de cada carro é a FK **`MarcaId`**)

Programação Visual
Trabalho de Laboratório nº 6

- Na classe **`Program`**, comente o código do método **`Main`** e acrescente o seguinte código:

```
public static void Main(string[] args)
{
    var host = CreateHostBuilder(args).Build();
    using (var scope = host.Services.CreateScope())
    {
        var services = scope.ServiceProvider;
        try
        {
            var context = services.GetRequiredService<ApplicationDbContext>();
            DbInitializer.Initialize(context).Wait();
        }
        catch (Exception ex)
        {
            var logger = services.GetRequiredService<ILogger<Program>>();
            logger.LogError(ex, "An error occurred while seeding the database.");
        }
    }
    host.Run();
}
```

- Compile a aplicação e verifique que aparecem os dados iniciais que forneceu.

Nível 3

- Pretende-se agora criar uma classe **`Cliente`** estendendo o *user* criado pelo pacote *Microsoft Identity* que foi instalado inicialmente na aplicação. Uma vez que não tem acesso a este código, adicione à pasta do projeto um novo *scaffold item* selcionando a opção *identity*. Na janela de diálogo que abriu escolha o *override* dos ficheiros *Account\Register* e *Account\Manage\index* e o contexto da aplicação criado antes.
- No modelo, crie a classe **`Cliente`** que deriva de **`IdentityUser`** e acrescente-lhe a propriedade **`Nome`**.
- Assegure que a classe do contexto deriva de **`IdentityDbContext<Cliente>`**
- Será necessário agora corrigir as vistas *Register* e *Index* relativas aos ficheiros que adicionou antes. Para isso siga este tutorial (apenas as alterações realçadas referentes ao nome): https://docs.microsoft.com/en-us/aspnet/core/security/authentication/add-user-data?view=aspnetcore-3.0&tabs=visual-studio#add-custom-user-data-to-the-identity-db
  Atenção que não está a usar agora a classe **`IdentityUser`** dos utilizadores, mas sim a classe **`Cliente`**.
- Na classe **`StartUp`**, método **`ConfigureServices`** também é necessário substituir o **`IdentityUser`** por **`Cliente`**.
- Também a vista **`_LoginPartial`** necessita de ser atualizada com a nova classe **`Cliente`** em vez de **`IdentityUser`**
- Teste criando registando um novo cliente.

Programação Visual
Trabalho de Laboratório nº 6

Nível 4

- Para a gestão do *site* **EsteCar** será ainda necessário fazer a gestão dos alugueres de carros. Defina no modelo uma classe **`Aluguer`** para os alugueres com as propriedades **`AluguerId, CarroId, UserId`** (*String*), **`LocalDeEntrega, LocalDeRecolha`**, **`DataInicio, DataFim.`**
- Anote as propriedades que representem datas na classe **`Aluguer`** com o seguinte atributo: `[DataType(DataType.Date)]`.
- Anote a propriedade **`UserId`** com o atributo: `[ForeignKey("Cliente")]`
- Coloque as propriedades de navegação **`Carro`** e **`Cliente`** na classe **`Aluguer`**.
- De seguida crie o controlador para os alugueres usando o *template* apropriado. No layout, inclua uma entrada de menu para este controlador
- Atualize a base de dados.
- Teste a aplicação.

Nível 5

- Os excêntricos proprietários do *site* não admitem alugueres de carros aos fins-de-semana. Crie um **atributo** de validação (**`ValidationAttribute`**) que faça esta verificação. Compile e teste.
- Altere o nível de autorização para o controlador alugueres, de forma que somente os utilizadores registados conseguem ter acesso aos alugueres. Adicionalmente, somente os administradores podem editar ou apagar alugueres. Faça apenas a anotação das ações, sem criar o papel de administrador e sem corrigir a vista para que os links respetivos não sejam mostrados.
- Altere as regras de criação de passwords para os utilizadores de forma a que a única restrição seja um texto com no mínimo 8 carateres.

Desafio

- Altere o controlador do **Carro** e respetivas vistas para que se consiga efetuar o *upload* e visualização das fotografias dos veículos existentes no *site*

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