

> Ficha Prática Nº2 - Entity Framework Core

Objetivos:

- Introdução à manipulação de dados com a Entity Framework Core.
- Rever e consolidar os conceitos de Controller, Views e Models.
- Introdução aos ViewModels, ViewBag, ViewData.
- O que é o _ViewStart e como se define qual o _Layout a utilizar.

Parte I – Conceitos

(<https://learn.microsoft.com/en-us/ef/core/>)

>> Entity Framework (EF)

- Conjunto de tecnologias ADO.NET que dão apoio ao desenvolvimento de aplicações orientadas para os dados.
- A EF é um ORM (object-relational mapper) – Mapeamento objeto-relacional – que permite trabalhar com dados relacionais na forma de objetos .NET, específicos do domínio.
- Foco na lógica do negócio e não no acesso aos dados, ou seja, elimina a maioria do código necessário relacionado com o acesso e a manipulação dos dados (T-SQL, PL/SQL, etc.)

>> Entity Framework Core (EF Core)

- Multiplatform
- Open source - <https://github.com/dotnet/efcore>
- Pode ser utilizada com diferentes tipos de bases de dados - <https://learn.microsoft.com/en-us/ef/core/providers/?tabs=dotnet-core-cli>

>> Model

- Na EF o acesso aos dados é realizado através de um modelo.
- Um modelo é composto por classes (entity classes) e um objeto de contexto, que representa a ligação à base de dados.
- O objeto de contexto permite ler e atualizar os dados (criar, alterar, remover) da base de dados.
- A EF Core permite duas abordagens na criação de modelos:
 - Code First – Primeiro criam-se os modelos (classes POCO) e depois gera-se a base de dados, com base nestas classes.

- Database First – Os modelos (classes) são gerados a partir de uma base de dados existente.
- Exemplo de um modelo:

```
public class Aluno
{
    public int Id { get; set; }
    public string Nome { get; set; }
    public string Morada { get; set; }
    public DateTime DataNascimento { get; set; }
}
```

>> Data Annotations

- Atributos aplicados às classes ou às propriedades das classes.
- Fornecem meta-dados adicionais sobre as classes ou das suas propriedades.
- Não são específicos da Entity Framework Core – fazem parte da .NET Core Framework.
- Na ASP.NET MVC Core é possível utilizar estes atributos para a validação dos modelos
- Existem dois tipos de atributos:
 - Data Modeling Attributes
 - Validation Related Attributes

>> Chaves Primárias

- A EF depende de cada entidade ter «uma chave» que é usada para identificar a entidade.
- No processo **Code First** está definida uma convenção – **chaves implícitas** – em que o **Code First** irá procurar uma propriedade com a designação **Id**, ou uma combinação do **nome de classe** e **Id**, como **Cursoid**. Essa propriedade será mapeada numa (irá corresponder a uma) coluna do tipo chave primária numa tabela da base de dados.
- Se optar por não seguir a convenção das chaves implícitas então é necessário definir qual é a propriedade que corresponde à chave primária, através da utilização do atributo `[Key]` numa das propriedades do modelo.

>> Chaves Compostas

- A EF permite a definição de chaves compostas.
- Neste caso, é necessário indicar quais são as propriedades que compõem a chave `[Key]` e qual é a ordem `[Column(Order=1)]` pela qual a chave composta é formada.

```
public class Passport
{
    [Key]
    [Column(Order=1)]
    public int PassportNumber { get; set; }
    [Key]
    [Column(Order = 2)]
    public string IssuingCountry { get; set; }
    public DateTime Issued { get; set; }
    public DateTime Expires { get; set; }
}
```

Parte II – Exercícios

- CRIAR MODELOS
- GERAR, ANALISAR E APLICAR MIGRAÇÕES
- ANALISAR A BASE DE DADOS
- GERAR, ANALISAR E MODIFICAR CONTROLADORES E VISTAS QUE TRABALHEM COM OS MODELOS CRIADOS

Modelos, Migrações e Base de Dados

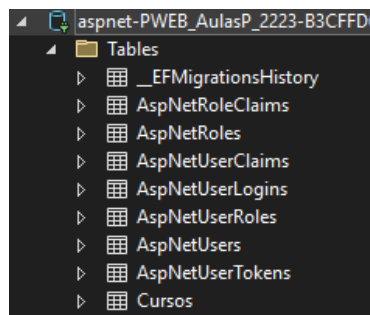
1. Crie o modelo POCO que represente um Curso, com as seguintes propriedades:
 - Id - `int`
 - Nome - `string`
 - Disponível - `bool`
2. Altere o ficheiro de contexto e mapeie este modelo.
 - 2.1. Edite o ficheiro `Data\ApplicationDbContext.cs` e adicione o seguinte código:


```
public DbSet<Curso> Cursos{ get; set; }
```
 3. Crie a migração Inicial.
 - 3.1. Abra a consola do Package Manager (Package Manager Console) e execute o seguinte comando


```
PM> Add-Migration MigracaoInicial
```
 - 3.2. Analise o ficheiro (`*_MigracaoInicial.cs`) da migração gerada na diretoria `\data\Migrations\`
 - 3.3. Execute o seguinte comando na Package Manager Console


```
PM> Update-Database
```

Com isto, foi gerada uma base de dados SQL Server com as seguintes tabelas:



__EFMigrationsHistory

Histórico das migrações – nome das migrações aplicadas.

AspNet*

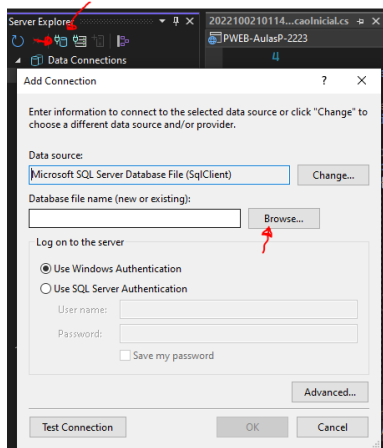
Tabelas necessárias para a Identity (um tema que será abordado mais tarde).

Cursos

Modelo mapeado na classe de contexto.

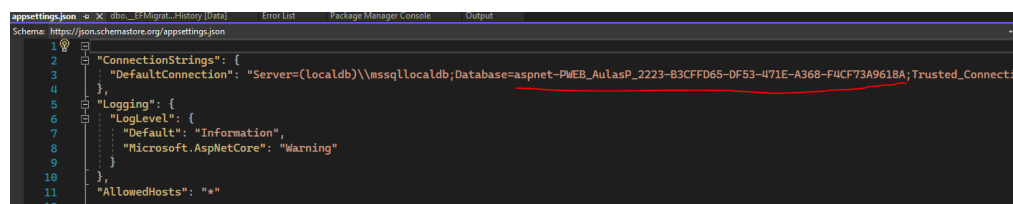
4. Abra a base de dados criada e verifique se a estrutura gerada é igual à apresentada na figura (do ponto 3).

Se for necessário, abra o **Server Explorer** e clique no botão – **nova ligação** e, de seguida, escolha o ficheiro com a base de dados.



Notas

- O nome da base de dados gerada está definido no ficheiro *appsettings.json*



- A base de dados é criada na **home dir** do utilizador:

c:\users\nome_do_utilizador\

- No exemplo, o caminho para o ficheiro com a base de dados é:

c:\users\nome_do_utilizador\aspnet-PWEB_AulasP_2223-B3CFFD65-DF53-471E-A368-F4CF73A9618A.mdf

5. Adicione os seguintes registos na tabela Cursos:

Id	Nome	Disponível
1	Categoria AM - Ciclomotor (idade mínima 16 anos)	True
2	Categoria A1 - Motociclo - 11kw/125cc (idade mínima 16 anos)	True

Id	Nome	Disponível
3	Categoria A2 - Motociclo - 35kw (idade mínima 18 anos)	True
4	Categoria A - Motociclo (idade mínima 24 anos)	True
5	Categoria B1 - Quadriciclo (idade mínima 16 anos)	True
6	Categoria A1B1 - Motociclo + Quadriciclo (idade mínima 16 anos)	True
7	Categoria A2B - Ligeiro + Motociclo - 35kw (idade mínima 18 anos)	True
8	Categoria AB - Ligeiro + Motociclo (idade mínima 24 anos)	True
9	Categoria B - Ligeiro Caixa Automática (idade mínima 18 anos)	True
10	Categoria A1B - Motociclo 125cc + Ligeiro (idade mínima 18 anos)	True

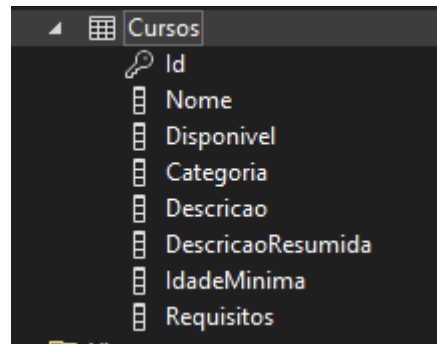
6. Altere a classe **Curso**, adicionando as seguintes propriedades:

- Categoria - `string`
- Descricao - `string`
- DescricaoResumida - `string`
- Requisitos - `string`
- IdadeMinima - `int`

7. Atualize a base de dados.

8. Verifique as alterações que foram efetuadas na base de dados.

A tabela Cursos ficou com as colunas apresentadas na imagem.

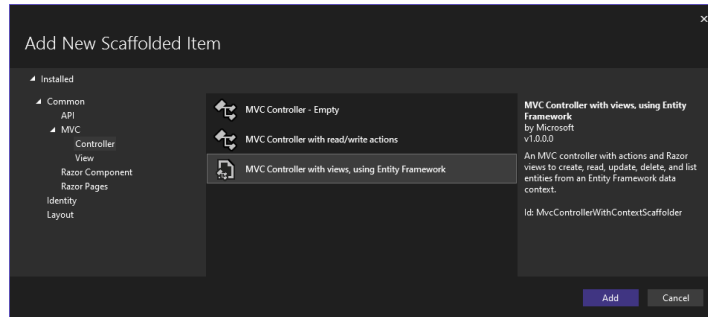


>> Entity Framework & Controllers & Views

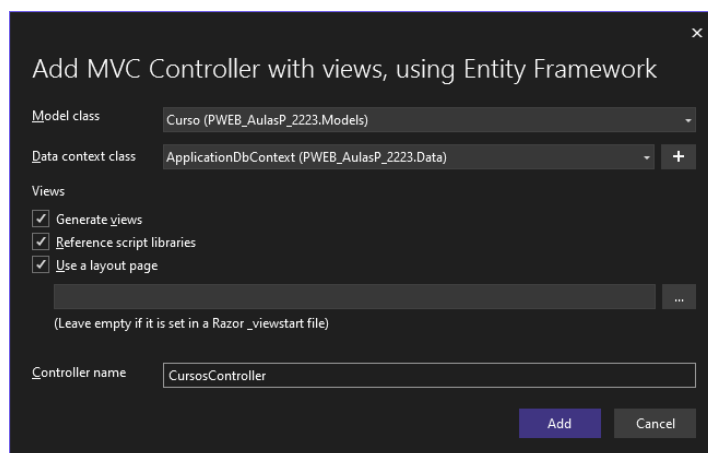
9. Crie um controlador e as respetivas vistas para que se possam efetuar as seguintes operações:

- Listar Cursos.
- Ver os detalhes de um Curso.
- Editar um Curso.
- Criar um Curso.
- Apagar um Curso.

Para isso, no «explorador» da solução, clique com o botão direito em cima do nome do projeto, ou da pasta Controllers, e escolha a opção **Add new Scaffolded Item**. Na «janela» seguinte escolha a opção **MVC » MVC Controller with views, using Entity Framework**:

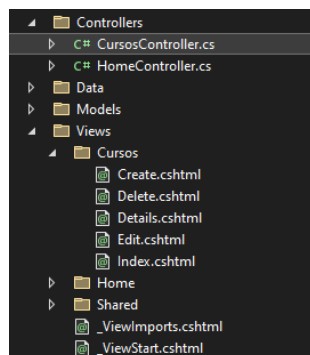


Na «janela» seguinte escolha o **modelo** e a **classe de contexto**, a partir dos quais pretende gerar o controlador e as vistas relacionadas (o modelo **Curso** e a classe de contexto **ApplicationDbContext**).



Altere o nome do controlador de **CursoesController** para **CursosController**.

10. Analise os ficheiros gerados – as vistas e o controlador.



11. Execute a aplicação e passe a execução da aplicação para a vista Index do controlador Cursos.
12. Adicione uma entrada no menu superior da aplicação que chame a vista Index do controlador Cursos. Teste o menu e garanta que o *link* está a funcionar.
13. O *template* gráfico, criado na Ficha 1, tem um ligeiro problema de *layout* fazendo com que o conteúdo das vistas geradas fiquem por baixo do menu superior. Para resolver esse problema:
 - Faça o download dos ficheiros **_Layout.cshtml** e **_Layout2.cshtml**, que estão disponíveis no Moodle.
 - Copie estes ficheiros para a pasta **Views\Shared** e substitua o ficheiro **_layout** existente.
 - Edite todas as vistas do controlador Cursos e adicione a seguinte linha de código:

Layout = "~/Views/Shared/_Layout2.cshtml";

```
@{
    ViewData["Title"] = "Cursos";
    Layout = "~/Views/Shared/_Layout2.cshtml";
}
```

14. Analise o controlador Cursos.

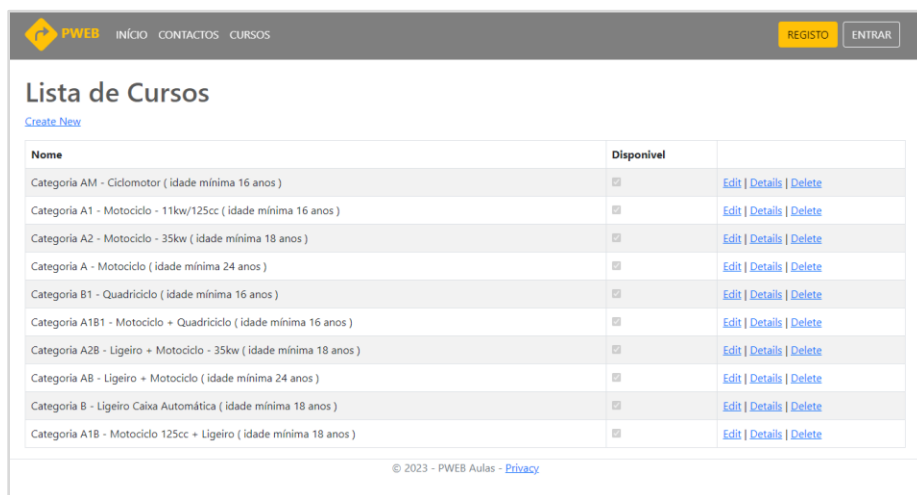
```
namespace PWEB_AulasP_2223.Controllers
{
    1 reference
    public class CursosController : Controller
    {
        private readonly ApplicationDbContext _context;

        0 references
        public CursosController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: Cursos
        3 references
        public async Task<IActionResult> Index()
        {
            return View(await _context.Cursos.ToListAsync());
        }
    }
}
```

- Variável de contexto.
- Construtor com Injeção das dependências (DI).
- Métodos assíncronos.
- Views e Models.
- `_context.Update();`
- `SaveChangesAsync();`
- `ModelState.IsValid`

15. Altere a vista **Index** para mostrar uma tabela com a seguinte estrutura:



Nome	Disponivel	
Categoria AM - Ciclomotor (idade mínima 16 anos)	<input checked="" type="checkbox"/>	Edit Details Delete
Categoria A1 - Motociclo - 11kw/125cc (idade mínima 16 anos)	<input checked="" type="checkbox"/>	Edit Details Delete
Categoria A2 - Motociclo - 35kw (idade mínima 18 anos)	<input checked="" type="checkbox"/>	Edit Details Delete
Categoria A - Motociclo (idade mínima 24 anos)	<input checked="" type="checkbox"/>	Edit Details Delete
Categoria B1 - Quadríciclo (idade mínima 16 anos)	<input checked="" type="checkbox"/>	Edit Details Delete
Categoria A1B1 - Motociclo + Quadríciclo (idade mínima 16 anos)	<input checked="" type="checkbox"/>	Edit Details Delete
Categoria A2B - Ligeiro + Motociclo - 35kw (idade mínima 18 anos)	<input checked="" type="checkbox"/>	Edit Details Delete
Categoria AB - Ligeiro + Motociclo (idade mínima 24 anos)	<input checked="" type="checkbox"/>	Edit Details Delete
Categoria B - Ligeiro Caixa Automática (idade mínima 18 anos)	<input checked="" type="checkbox"/>	Edit Details Delete
Categoria A1B - Motociclo 125cc + Ligeiro (idade mínima 18 anos)	<input checked="" type="checkbox"/>	Edit Details Delete

© 2023 - PWEB Aulas - [Privacy](#)

Para isso, na tag table, necessita de adicionar as seguintes classes de estilo `table table-bordered table-striped table-hover`

16. Altere o texto dos links **Edit**, **Details**, **Delete** e **Create new** para **Editar**, **Detalhes**, **Apagar** e **Adicionar curso**.

17. Adicione as opções de consulta **Todos**, **Ativos** e **Inativos**



Nome	Disponivel	
Categoria A2 - Motociclo - 35kw (idade mínima 18 anos)	<input type="checkbox"/>	Edit Details Delete

© 2023 - PWEB Aulas - [Privacy](#)

Estas opções (*links*) devem ser criadas com a TAG `<a>` e com recurso aos tag-helpers `ASP-ACTION` e `ASP-ROUTE`.

TAG-HELPERS

[HTTPS://LEARN.MICROSOFT.COM/EN-US/ASPNET/CORE/MVC/VIEWS/TAG-HELPERS/INTRO?VIEW=ASPNETCORE-6.0](https://learn.microsoft.com/en-us/aspnet/core/mvc/views/tag-helpers/intro?view=aspnetcore-6.0)

Exemplo:

```
<a asp-action="Index" asp-route-disponivel="true">Activos</a>
```


18. Faça as alterações necessárias no controlador **Cursos** para que, quando o utilizador clica nestes *links*, só sejam mostrados os cursos que correspondam à escolha selecionada.
19. Faça as alterações necessárias no controlador **Home** e na vista **Index** deste controlador para substituir os três cursos existentes pela lista dos cursos que existem na base de dados e que estão ativos (disponíveis).

Tenha em consideração que

- Deve manter o formato de apresentação usado na Ficha 1 (formato *card*).
 - Nesta fase, a classe **Curso** ainda não tem uma propriedade que represente o valor do preço do curso. Assim, nesta fase do desenvolvimento da aplicação, defina os *card* sem o valor do preço.
20. Altere o botão «**saber mais**» dos *card* para redirecionar o utilizador para a vista **Details** do controlador **Cursos**, passando como parâmetro o **Id** do curso selecionado.
 21. Altere a classe **Curso**, adicionando a seguinte propriedade:
 - Preço – *Decimal?*
 22. Atualize a base de dados.
 23. Faça as alterações necessárias nas vistas **Index**, **Details**, **Create** e **Edit** para usarem esta propriedade.
 24. Faça as alterações necessárias no controlador **Cursos** para que seja possível modificar esta propriedade (criar, editar).

Nesta fase a aplicação deve ser capaz de criar um curso, com o preço, editar o preço de um curso e ver os detalhes do curso (com o preço incluído).

25. Altere a classe **Curso**, adicionando a seguinte propriedade:
 - EmDestaque – *bool*
26. Atualize a base de dados.
27. Modifique o controlador **Cursos** e as respetivas vistas para usarem esta propriedade.
28. Modifique o comportamento da Página Inicial da aplicação para mostrar apenas os cursos que estejam ativos e em destaque.