

> Ficha Prática Nº2 - Entity Framework Core

Objetivos:

- Introdução à manipulação de dados com o 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 que _Layout utilizar

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

>> Entity Framework (EF)

- Conjunto de tecnologias ADO.NET que dão suporte ao desenvolvimento de aplicações orientadas a dados.
- 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 relativo ao acesso e à manipulação dos dados (T-SQL, PL/SQL, etc.)

>> Entity Framework Core (EF Core)

- Desenvolvida de raiz
- Open source - <https://github.com/dotnet/efcore>
- Multiplatform
- Suporta 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 Model.
- Um Model é composto por classes (entity classes) e um objeto de contexto que representa a ligação à base de dados.
- Este objeto de contexto permite ler e atualizar dados (inserir, atualizar, apagar) da base de dados.
- A EF Core permite duas abordagens na criação de modelos:
 - Code First – Primeiro criamos os modelos (classes POCO) e depois geramos a base de dados com base nestas classes.
 - Database First – Os modelos (classes) são gerados com base numa base de dados existente.

- Exemplo de um Model:

```
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 a propriedades das classes.
- Fornecem meta-dados adicionais sobre as classes ou as suas propriedades
- Não são específicos do Entity Framework Core – Fazem parte do .NET Core Framework.
- No ASP.NET MVC Core é possível utilizar estes atributos para validação dos Models
- Dois tipos de atributos:
 - Data Modeling Attributes
 - Validation Related Attributes

>> Chaves Primárias

- O EF depende de cada entidade ter “uma chave” que é usada para identificar a entidade.
- No modelo **Code First** está definida uma convenção - *chaves implícitas*, em que o **Code First** irá procurar uma propriedade chamada “Id”, ou uma combinação do **nome de classe** e “Id”, como “**Cursold**”. Essa propriedade será mapeada para uma coluna do “tipo” chave primária na base de dados.
- Se optar por não seguir a convenção das chaves implícitas então é necessário definir qual a propriedade que corresponde à chave primária através da utilização do atributo [Key].

>> Chaves Compostas

- O EF suporta chaves compostas.
- Neste caso é necessário indicar quais as propriedades que compõem a chave [Key] e qual 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 CONTROLLERS E VIEWS QUE TRABALHEM COM OS MODELOS CRIADOS

>> Modelos, Migrações e Base de Dados

1. Crie o Modelo POCO que representa 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

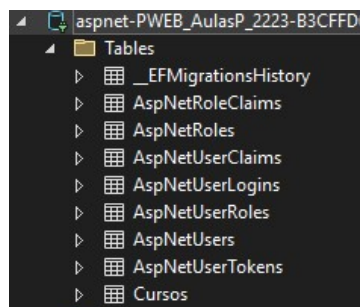
```
PM> Add-Migration MigracaoInicial
```

3.2. Analise o ficheiro (`*_MigracaoInicial.cs`) de migração gerado pelo comando anterior na diretoria `\data\Migrations\`

3.3. Execute o seguinte comando na Consola do Package Manager

```
PM> Update-Database
```

Foi gerada uma base de dados SQL Server com as seguintes tabelas:



__EFMigrationsHistory

histórico de migrações – nome das migrações aplicadas

AspNet*

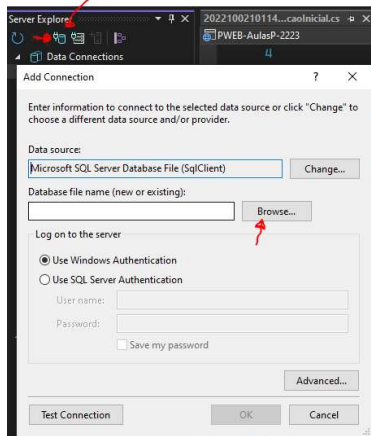
Tabelas necessárias para o Identity (tema abordado mais tarde)

Cursos

Modelo mapeado na classe de contexto

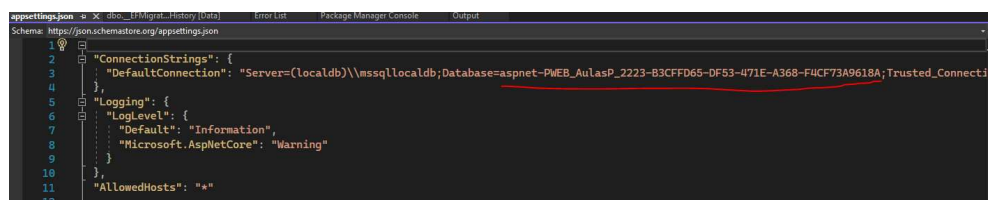
- Abra a base de dados criada e verifique se a estrutura gerada é igual à apresentada no ponto anterior.

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 definida no ficheiro *appsettings.json*



- A base de dados é criada na home dir do utilizador, por exemplo em sistemas Windows:

c:\users\nome_do_utilizador\

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

c:\users\nome_do_utilizador\aspnet-PWEB_AulasP_2223-B3CFD65-DF53-471E-A368-F4CF73A9618A.mdf

- Adicione os seguintes registos na tabela Cursos:

| Id | Nome | Disponivel |
|----|--|------------|
| 1 | Categoria AM - Ciclomotor (idade mínima 16 anos) | True |
| 2 | Categoria A1 - Motociclo - 11kw/125cc (idade mínima 16 anos) | True |
| 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** – adicione as seguintes propriedades:

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

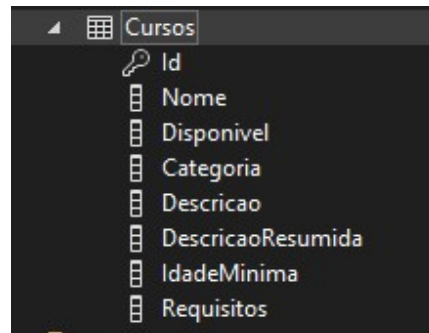
7. Crie uma migração com o comando **Add-Migration**.

8. Analise o ficheiro criado com a migração.

9. Atualize a base de dados com o comando **Update-Database**.

10. Verifique as alterações efetuadas na base de dados.

A tabela de Cursos ficou com as colunas apresentadas na imagem à direita.



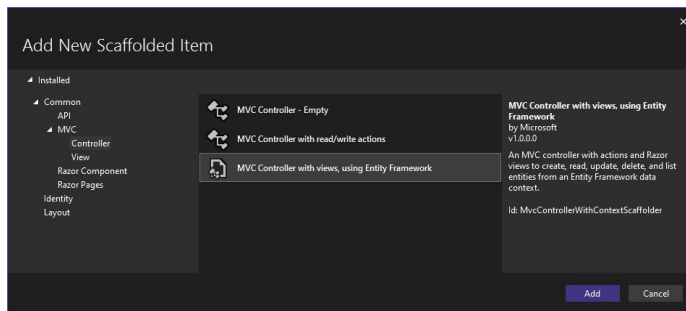
| Id | Nome | Disponivel | Categoria | Descricao | DescricaoResumida | IdadeMinima | Requisitos |
|----|------|------------|-----------|-----------|-------------------|-------------|------------|
|----|------|------------|-----------|-----------|-------------------|-------------|------------|

>> Entity Framework & Controllers & Views

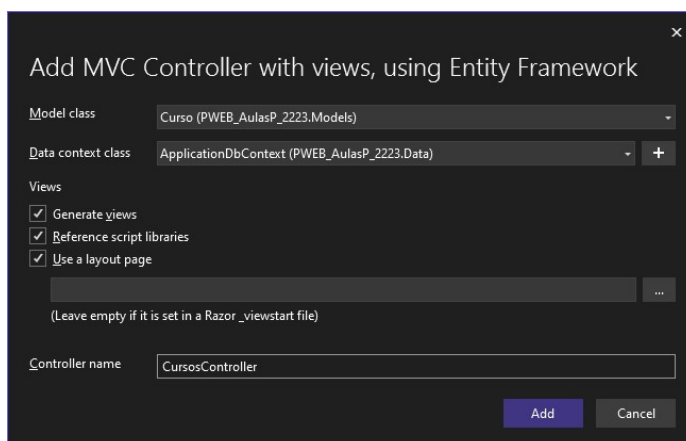
11. Crie um Controller e respectivas vistas que permitam realizar as seguintes operações:

- Listar Cursos
- Ver detalhes de um Curso
- Editar um Curso
- Inserir um Curso
- Apagar um Curso

Para isso, no explorador da solução, clique com o botão direito em cima do projeto ou da pasta Controllers e escolha a opção “Add new Scaffolded Item” e no ecrã seguinte escolha a opção **MVC » MVC Controller with views, using Entity Framework**, conforme a imagem seguinte:

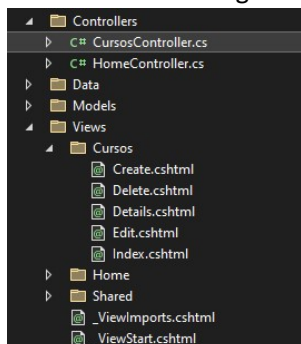


No ecrã seguinte escolhe o Modelo a partir do qual pretende gerar o controller e as views - no nosso caso é o Modelo Curso – bem como a classe de contexto a ser utilizada.

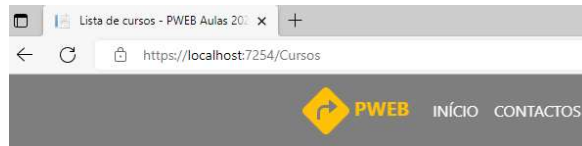


Altere o nome do controller para “**CursosController**” em vez de “CursosController”.

12. Analise os ficheiros gerados – Views e controller



13. Execute a aplicação e chame a View Index do controller Cursos.



14. Adicione uma entrada no menu superior da aplicação que chame a View do ponto anterior.

Teste o menu e garanta que o link está a funcionar.

15. O template gráfico criado na ficha laboratorial número 1 tem um ligeiro “problema” de layout fazendo com que o conteúdo das views geradas fiquem por baixo do menu superior. Para resolver esse problema:

- Faça 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 de _layout existente.
- Edite todas as vistas “do” Controller Cursos e adicione a seguinte linha de código:

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

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

16. Analise o controller 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());
        }
    }
}
```

Analise com o docente:

- Variável de contexto
- Construtor c/ Injeção de dependência
- Métodos assíncronos
- Views e Models
- `_context.Update();`
- `SaveChangesAsync();`
- `ModelState.IsValid`

17. Altere a View **Index** por forma a mostrar uma tabela com a seguinte estrutura

PWEB INÍCIO CONTACTOS CURSOS

REGISTO ENTRAR

Lista de Cursos

[Create New](#)

| Nome | Disponível | |
|---|-------------------------------------|---|
| 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 - Quadriciclo (idade mínima 16 anos) | <input checked="" type="checkbox"/> | Edit Details Delete |
| Categoria A1B1 - Motociclo + Quadriciclo (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`

18. Altere o texto dos links “Edite, Details, Delete, Create new” para “Editar, Detalhes, Apagar e Adicionar curso”.

19. Adicione as opções de consulta “Todos”, “Ativos”, “Inativos”, conforme a seguinte imagem:

PWEB INÍCIO CONTACTOS CURSOS

REGISTO ENTRAR

Lista de cursos Inativos [\(adicionar\)](#)

[Todos](#) | [Ativos](#) | [Inativos](#) |

| Nome | Disponível | |
|--|--------------------------|---|
| 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`

20. Faça as alterações necessárias no controller **Cursos** por forma a que quando o utilizador clica nestes links, só sejam mostrados os cursos que correspondam à escolha selecionada.
21. Faça as alterações necessárias no **controller Home** e na **View Index** deste por forma a substituir os três cursos existentes pela lista de cursos que existem na base de dados e que estão ativos (disponível=true).

Notas:

- Deve manter o mesmo formato de apresentação usado na ficha laboratorial número 1 (formato card).
 - A classe Curso ainda não tem preço pelo que nesta fase vamos construir o card sem o preço.
22. Altere o botão “**saber mais**” do card para redirecionar o utilizador para a View Detalhes do Controller Cursos, passando como parâmetro o Id do curso selecionado.
 23. Altere a classe **Curso** – adicione a seguinte propriedade:
 - Preço – **Decimal?**
 24. Crie uma nova migração e atualize a base de dados.
 25. Faça as alterações necessárias nas views Details, Insert, Edit por forma a contemplar esta nova propriedade.
 26. Faça as alterações necessárias no controller Cursos por forma a que seja possível modificar esta propriedade (inserir, editar).

No final deve ser capaz de inserir um novo curso com preço, editar o preço de um curso, bem como ver todos os detalhes do curso (preço incluído).

>> Exercícios extra

27. Altere a classe Curso – adicione a seguinte propriedade:
 - EmDestaque – `bool`
28. Crie uma Migração e atualize a base de dados
29. Modifique o controller Cursos e as respectivas vistas por forma a contemplarem esta nova propriedade.
30. Modifique o comportamento da Página Inicial da aplicação para mostrar apenas os cursos que estejam activos (`disponível=true`) e em destaque (`EmDestaque=true`).
31. Criar um Model que represente uma Categoria de Carta (A, B, C, D, etc.)
 - Id – `int`
 - Nome – `string`
 - Descricao – `string`
 - Disponivel – `bool`
32. Altere o ficheiro de contexto e adicione este modelo
33. Crie uma Migração e atualize a base de dados
34. Crie uma propriedade de navegação nas classes Categoria e Curso que permita relacionar ambas (relação 1-N).
 - Uma Categoria tem uma lista de cursos
 - Um Curso pertence a uma Categoria