
Programação Web

Aulas Teóricas – Capítulo 2 – 2.4

1º Semestre - 2023/2024

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra



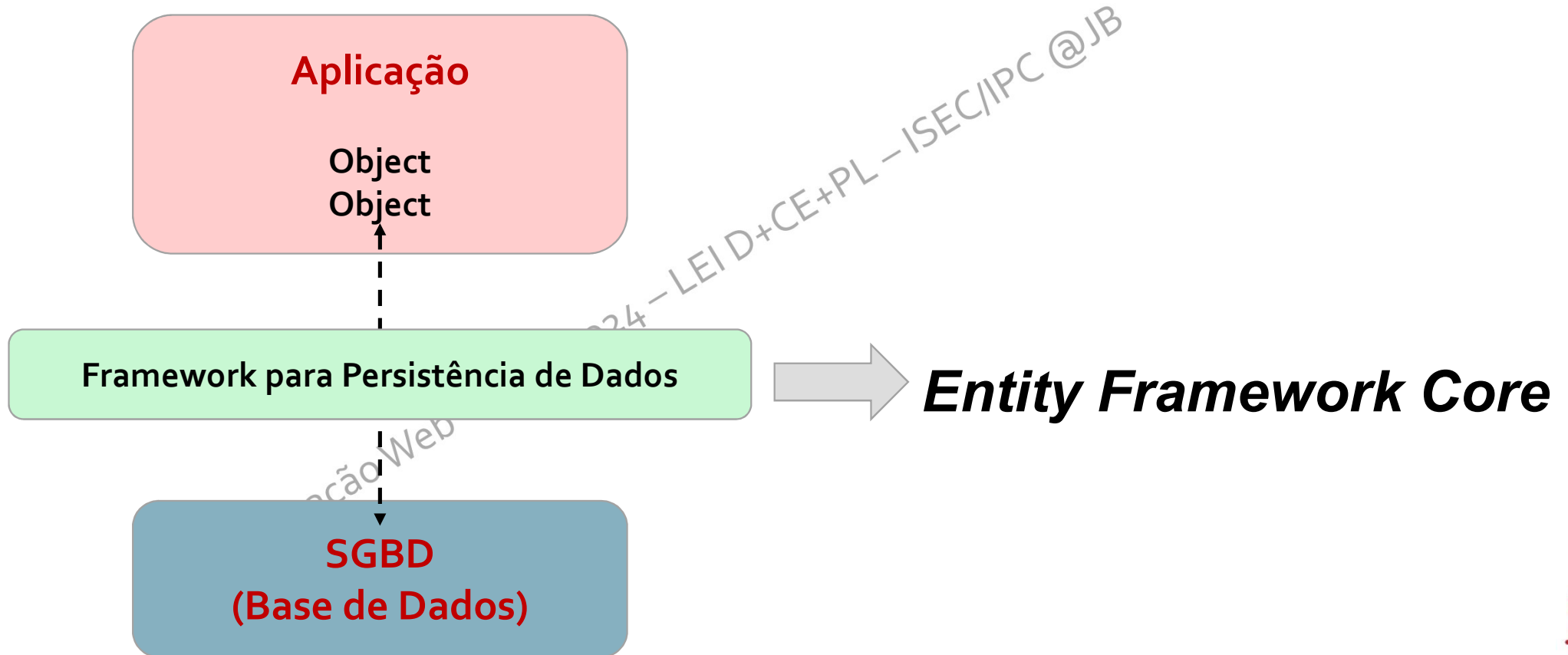
Programação Web

Manipulação de Dados em ASP.NET Core

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

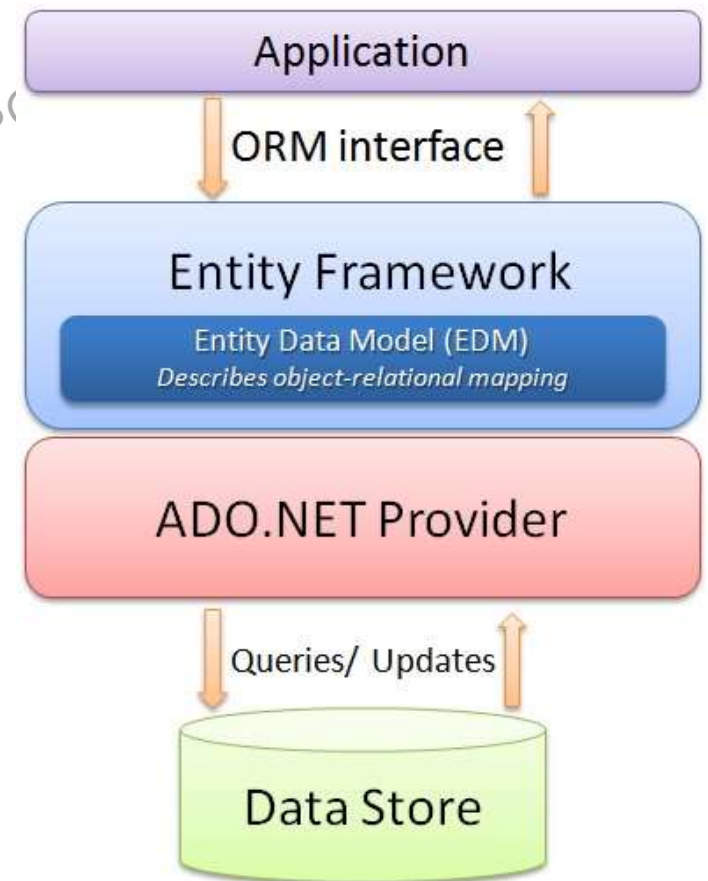


Entity Framework Core (EF)



Entity Framework Core (EF)

- *Object Relational Mapper (ORM)*
- *Open Source*
- Aumentar a produtividade do programador, reduzindo tarefas redundantes para persistência de dados



Entity Framework Core (EF)

- **Entity Framework Core (EF Core)** é uma versão leve, extensível e multiplataforma do Entity Framework
- O EF Core introduz muitas melhorias e novos recursos quando comparado com as versões anteriores
- O EF Core mantém a experiência do programadores habituados a usar versões antigas do EF e a maioria das APIs de alto nível permanecem as mesmas

Entity Framework Core (EF)

- Ao mesmo tempo, o **EF Core** é construído sobre um conjunto completamente novo de componentes principais. Isso significa que o EF Core não herda automaticamente todos os recursos de anteriores versões do EF
 - Alguns desses recursos serão exibidos em lançamentos futuros, outros recursos menos usados não serão implementados no **EF Core**
- O novo núcleo permitiu adicionar alguns recursos ao **EF Core** que não serão implementados nas versões anteriores

Entity Framework Core

- Formas para especificar o modelo de dados

1. *Code First*

- Parte-se das classes do domínio e a **EF** gera a Base de Dados
- Iremos dar particular atenção à utilização de *Code First*

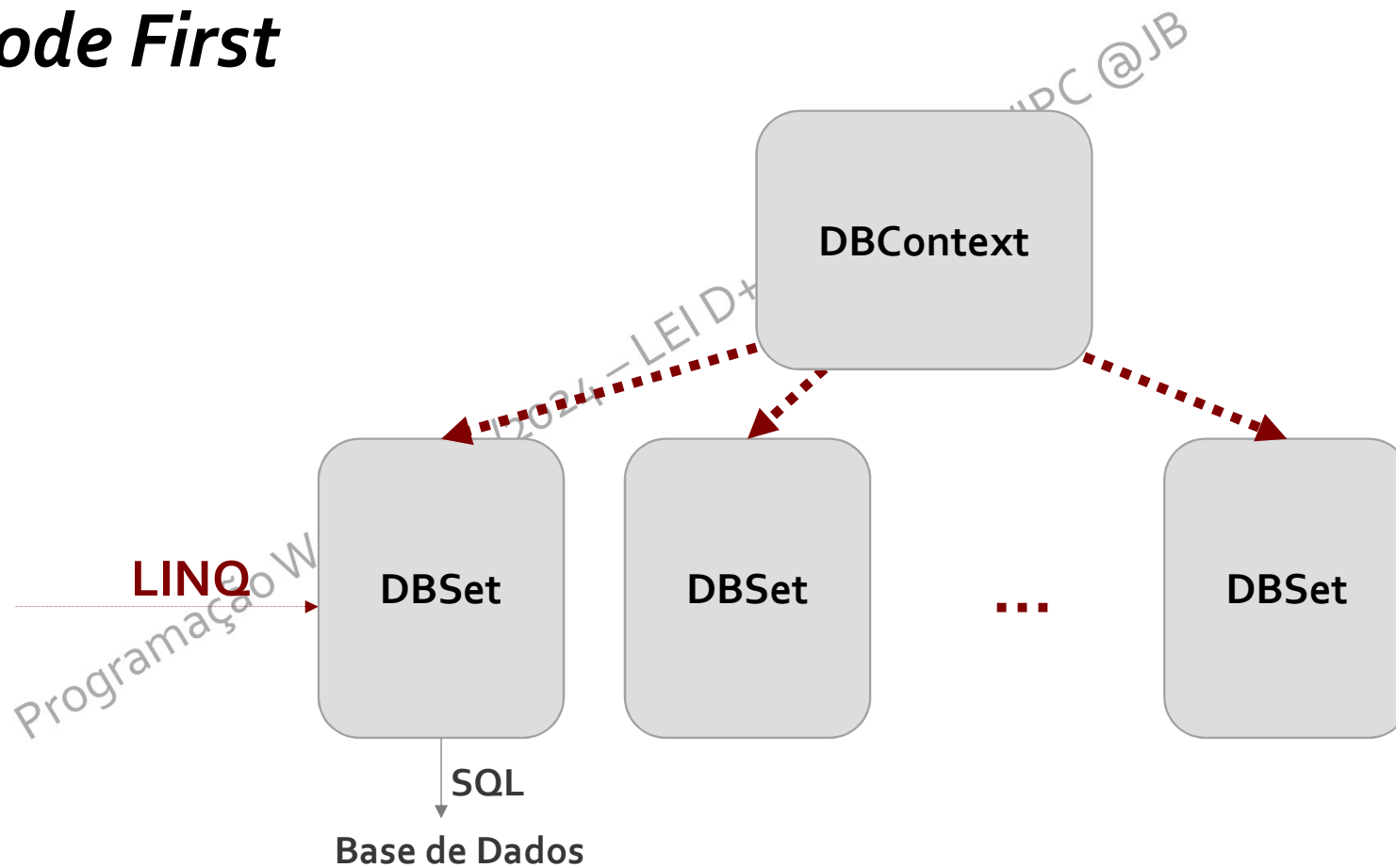
2. *Database First*

- Parte-se da Base de Dados para obter as classes de domínio

- Model First - Esta abordagem era possível com versões anteriores do ASP.Net, se bem que já na altura pouco suportada pela MS mas actualmente está descontinuada no Core

Entity Framework

1. Code First



Entity Framework – Code First

Etapas a seguir para a utilização da **Entity Framework** em **CodeFirst**

1. Instalação da **Entity Framework**
2. Definição do **Modelo de Domínio**
3. Criação do **Contexto** (classe que herda da **DbContext** e que é uma abstração da base de dados usada no acesso aos dados)
4. Registrar no método **Services** de **Program.cs** o **DbContext** como um serviço
5. Especificação da **Connection String** no ficheiro **appsettings.json**

Especificação das Entidades a Utilizar

2. Definição do **Modelo de Domínio**

Na pasta *Models* crie as classes que representem as entidade de domínio
Por exemplo:



Criação do Modelo Curso

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
namespace AutoWEB.Models
{
    26 referências
    public class Curso
    {
        15 referências
        public int Id { get; set; }
        [Display(Name = "Nome", Prompt = "Introduza o Nome do curso!")]
        [Required(ErrorMessage = "Indique o nome do curso!")]
        19 referências
        public string Nome { get; set; }
        [Display(Name = "Descrição", Prompt = "Introduza a Descrição do curso!")]
        [Required(ErrorMessage = "Descreva o curso!")]
        11 referências
        public string Descricao { get; set; }
        [Display(Name = "Descrição Resumida", Prompt = "Introduza a Descrição Resumida do curso!")]
        [Required(ErrorMessage = "Descreva resumidamente o curso!")]
        17 referências
        public string DescricaoResumida { get; set; }
        [Display(Name = "Requisitos", Prompt = "Introduza os Requisitos do curso!")]
        [Required(ErrorMessage = "Indique os requisitos do curso!")]
        10 referências
        public string Requisitos { get; set; }
        [Display(Name = "Idade mínima", Prompt = "Idade Mínima",
            Description = "Idade mínima para poder frequentar esta formação")]
        [Required(ErrorMessage = "Indique a Idade Mínima para este curso!")]
        [Range(14, 100, ErrorMessage = "Mínimo: 14 anos, Máximo: 100 anos")]
        10 referências
        public int IdadeMinima { get; set; }
        [Display(Name = "Curso activo?")]
        12 referências
        public bool Disponivel { get; set; }
        [Display(Name = "Curso em Destaque?")]
        11 referências
        public bool EmDestaque { get; set; }
        [Required(ErrorMessage = "Indique o Preço do curso!")]
        [Display(Name = "Preço", Prompt = "Introduza o Preço", Description = "Preço do curso")]
        [DisplayFormat(DataFormatString = "{0:C2}")]
        12 referências
        public decimal? Preco { get; set; }
        [Display(Name = "Categoria")]
        11 referências
        public int? CategoriaId { get; set; }
        10 referências
        public Categoria categoria { get; set; }
    }
}
```

Criação do Contexto

3. Definição do **Contexto** da Base de Dados

Na pasta **Data** crie o arquivo **ApplicationDbContext**

```
using AutoWEB.Models;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace AutoWEB.Data
{
    16 referências
    public class ApplicationDbContext : IdentityDbContext <ApplicationUser>
    {
        19 referências
        public DbSet<Curso> Cursos { get; set; }
        17 referências
        public DbSet<Categoria> Categorias { get; set; }
        12 referências
        public DbSet<Agendamento> Agendamentos { get; set; }
        18 referências
        public DbSet<TipoDeAula> TipoDeAula { get; set; }

        0 referências
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
            : base(options)
        {
        }
    }
}
```

DbContext como Serviço – Program.cs

4. Registrar em *Program.cs* o *DbContext* como um serviço

No ficheiro *program.cs*, tem de se especificar o *DbContext* como um serviço

```
builder.Services.AddDbContext<ApplicationDbContext>(options =>  
    options.UseSqlServer(connectionString));  
  
...
```

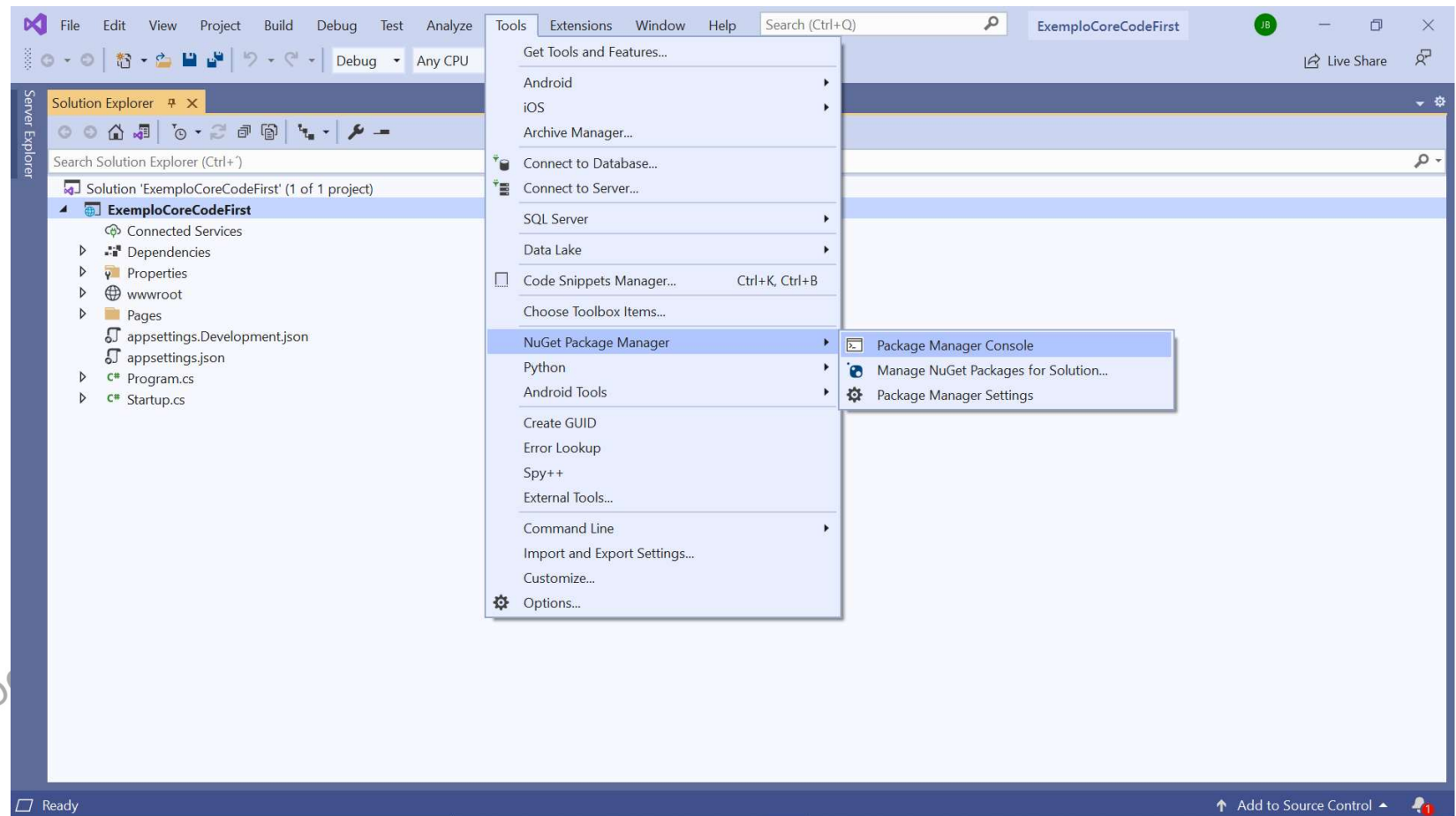
Especificação da *Connection String*

5. Especificação da *Connection String* no ficheiro *appsettings.json*

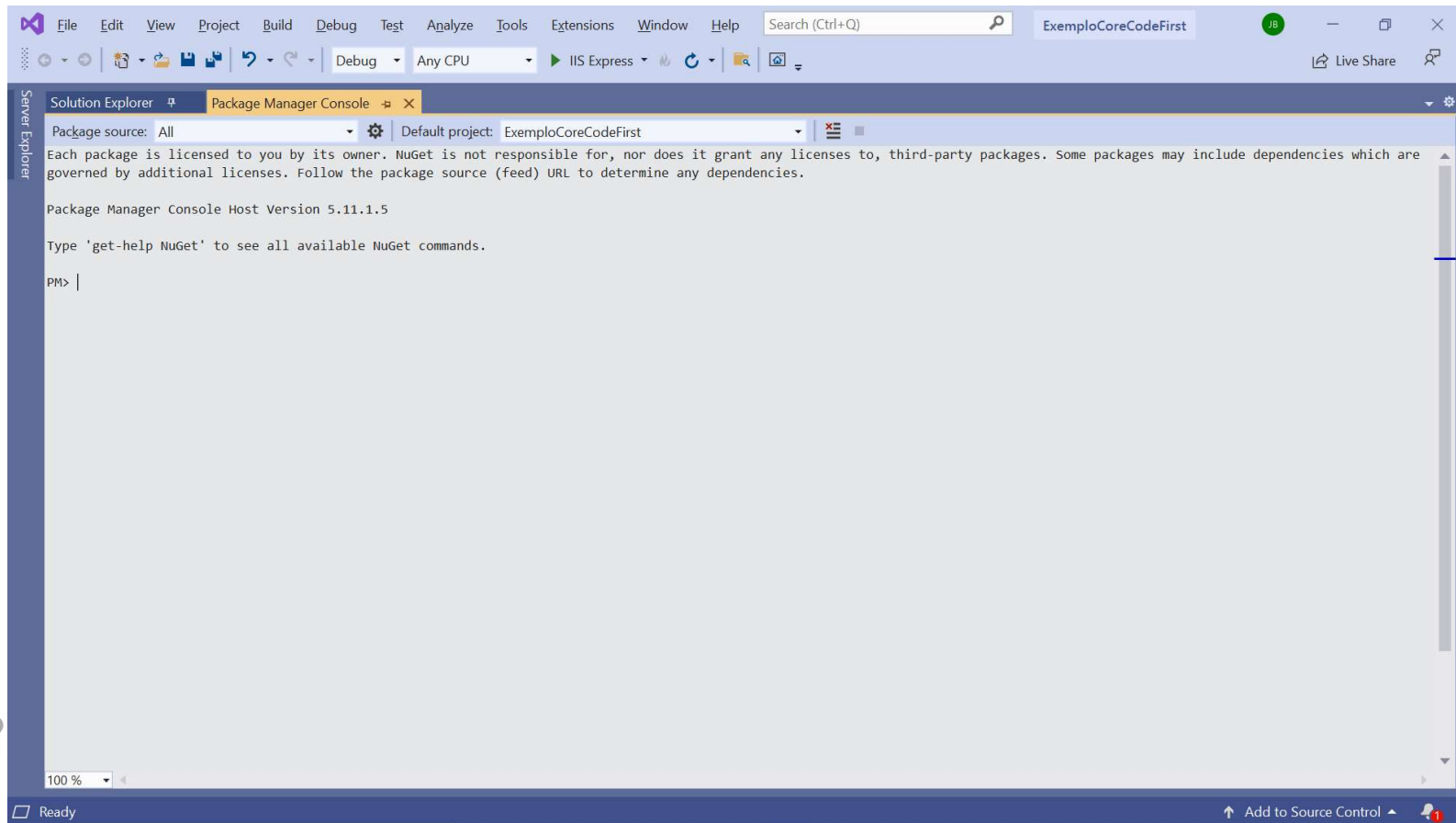
No ficheiro *appsettings.json*, define-se uma *string* de conexão com a base de dados usando uma instância local da BD

```
"ConnectionStrings": {  
  "DefaultConnection":  
    "Server=(localdb)\\mssqllocaldb;Database=AutoWeb;Trusted_Connection=True;MultipleActiveResultSets=true"  
},  
...
```

Ativação das *Migrations*

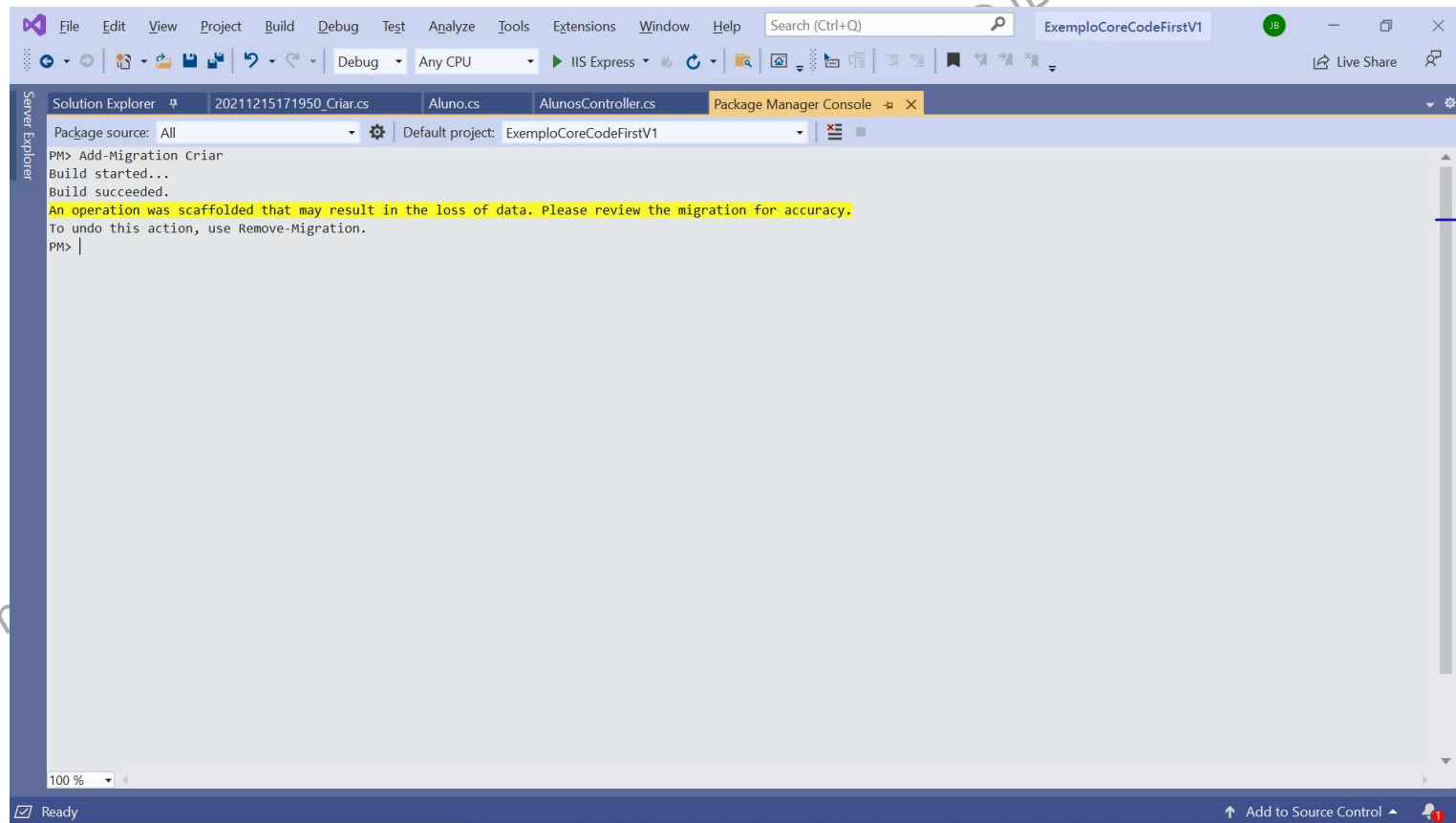


Package Manager Console



Migrations

PM> Add-Migration Criar ou >dotnet ef migrations add Criar



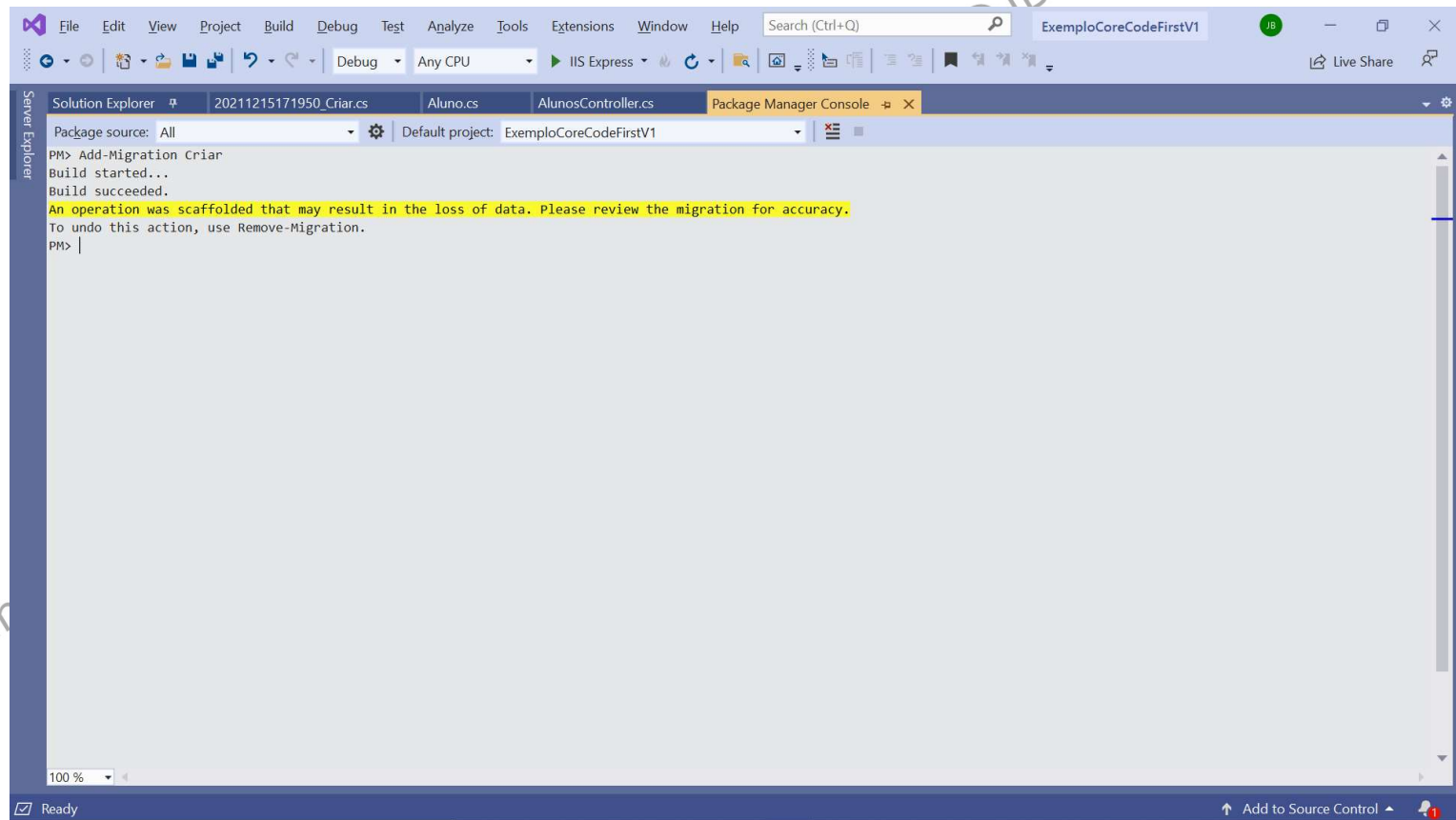
The screenshot shows the Visual Studio interface with the Package Manager Console open. The console displays the following output:

```
PM> Add-Migration Criar
Build started...
Build succeeded.
An operation was scaffolded that may result in the loss of data. Please review the migration for accuracy.
To undo this action, use Remove-Migration.
PM> |
```

The console window is titled 'ExemploCoreCodeFirstV1' and shows the 'Package Manager Console' tab. The 'Solution Explorer' on the left shows the project structure with files like '20211215171950_Criar.cs', 'Aluno.cs', and 'AlunosController.cs'.

Migrations

PM> update-database

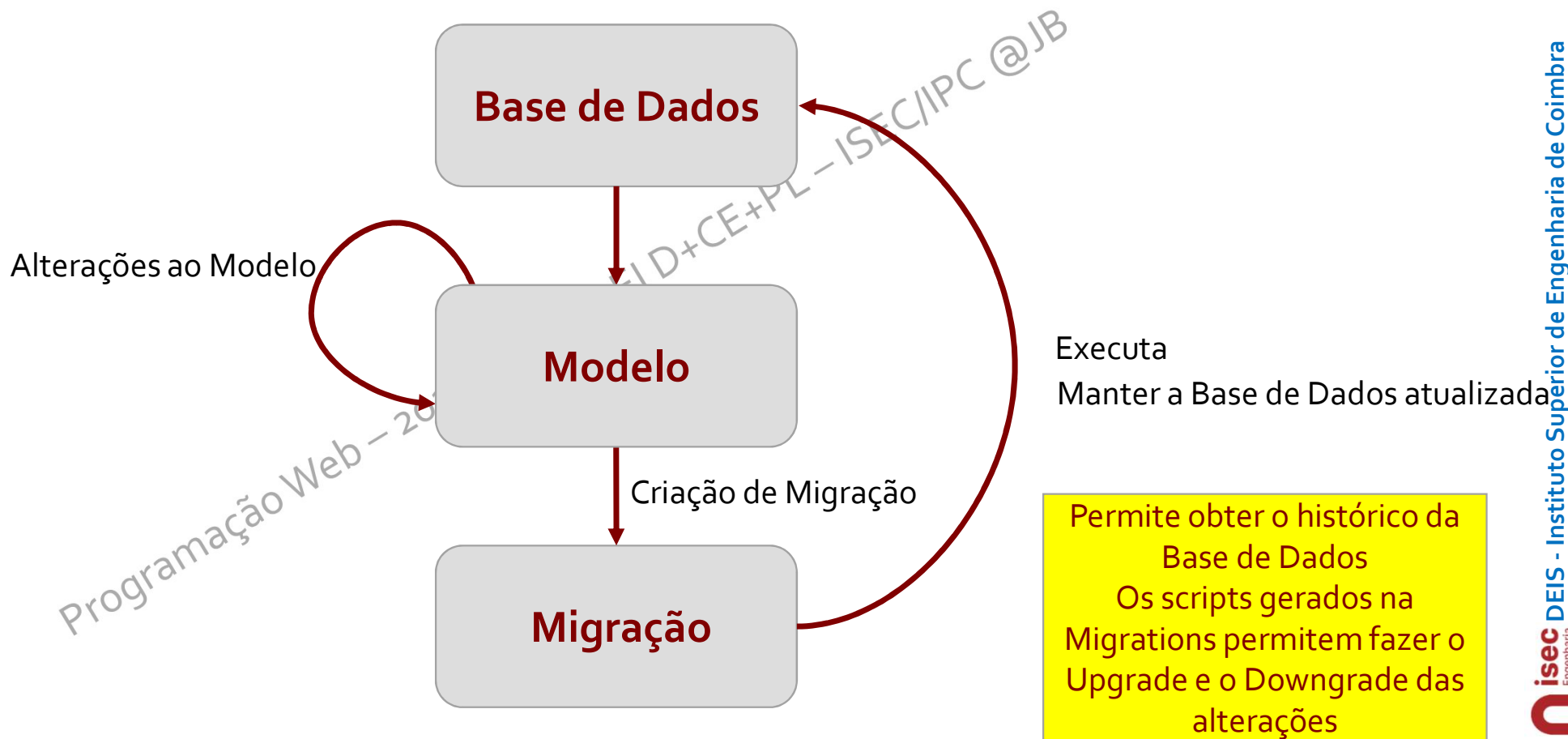


The screenshot shows the Visual Studio Package Manager Console for a project named 'ExemploCoreCodeFirstV1'. The console output is as follows:

```
Package source: All
Default project: ExemploCoreCodeFirstV1
PM> Add-Migration Criar
Build started...
Build succeeded.
An operation was scaffolded that may result in the loss of data. Please review the migration for accuracy.
To undo this action, use Remove-Migration.
PM>
```

The message 'An operation was scaffolded that may result in the loss of data. Please review the migration for accuracy.' is highlighted in yellow in the original image. The console also shows the status 'Ready' at the bottom left and 'Add to Source Control' at the bottom right.

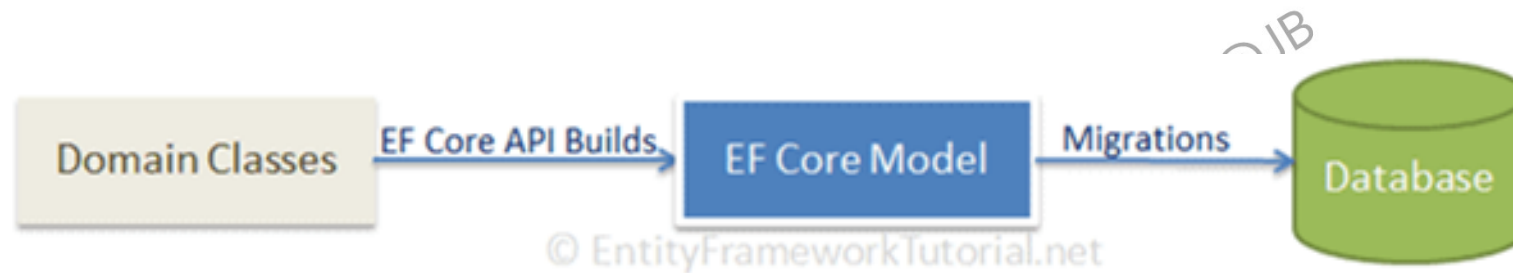
Processo Iterativo



*Mas qual o objectivo das
Migrações (Migrations) ?*

Programação Web – 2022/2024 – FEP+PL – JEP@JB

EF Core Migrations



A *EF Core API* constrói o modelo *EF Core* a partir das classes de domínio (entidades) e as *migrations EF Core* criarão ou atualizarão o esquema da base de dados com base no modelo *EF Core*

Sempre que as classes de domínio são alteradas, é necessário executar novamente a *migration* para manter o esquema da base de dados atualizado

EF Core Migrations

As *migrations EF Core* são um conjunto de comandos que se executam na consola do gestor de pacotes *NuGet* ou na interface de linha de comando *dotnet*

A tabela seguinte mostram-se os comandos de *migration* mais importantes no *EF Core*

EF Core Migrations - Comandos

Comando PMC	Comando dotnet CLI	Utilização
add-migration <migration name>	Add <migration name>	Cria a migration adicionando uma migration snapshot.
Remove-migration	Remove	Remove a última migration snapshot.
Update-database	Update	Actualiza o esquema da base de dados baseada na última migration snapshot.
Script-migration	Script	Gera um script SQL usando todas as migration snapshots.

EF Core Migrations - Comandos

Adicionando uma Migration

Inicialmente, definiram-se as classes de domínio

Após esta definição, ainda não se tem a base de dados para armazenar os dados das classes de domínio.

O passo seguinte é então criar essa BD e para isso utiliza-se uma *migration*

Selecciona-se a opção do menu Ferramentas -> Gerenciador de pacotes NuGet ->

Console do gerenciador de pacotes no Visual Studio

Com isto é aberto a Consola do NuGet e nesta consola executa-se o comando para adicionar uma migration

PM> add-migration MyFirstMigration

Caso se use a interface de linha de comando dotnet, executa-se o comando **CLI**

> dotnet ef migrations add MyFirstMigration

EF Core Migrations - Comandos

Criar ou Actualizar a Base de Dados

Use um dos seguintes comandos para criar ou atualizar o esquema da base de dados

Package Manager Console

PM> Update-Database

CLI

> dotnet ef database update

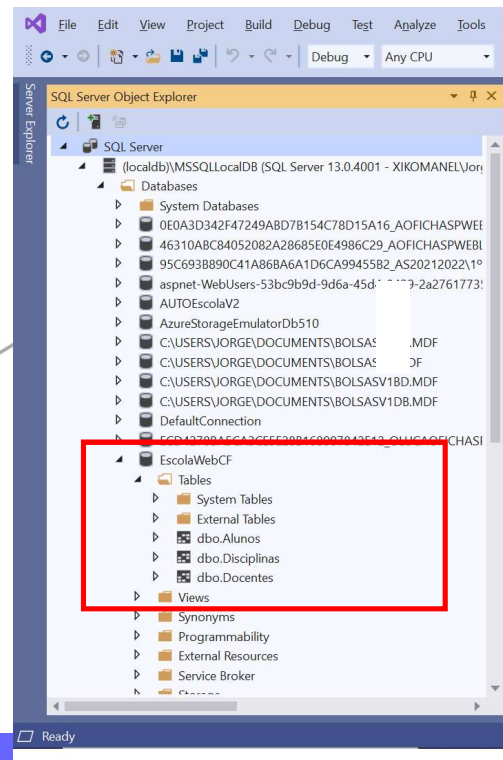
O comando *Update* criará a base de dados com base no contexto e nas classes de domínio e no *snapshot* da *migration*, criado usando o comando *add-migration* ou *add*

- Na primeira migração, também será criada uma tabela chamada *__EFMigrationsHistory*, que armazenará o nome de todas as *migrations*, como e quando serão aplicadas à base de dados

EF Core Migrations - Comandos

Criar ou Actualizar a Base de Dados

Após a execução deste comando em que **MyFirstMigration** é o nome de uma **migration** será criada no SQL Server a Base de Dados e as respectivas tabelas



Migrattions – Passos Principais

Usar Migrações Code First

2 Passos Principais:

1. Adicionar as alterações

PM> Add-Migration *darumnomequalqueraadicao*

Se falhar, fazer o Clean da Solution e a seguir o Rebuild

2. Actualizar a Base de Dados

PM> Update-Database

Mapeamento entre tipo C# e tipo SQL

Tipo de Dados em C#	Tipo de Dados na coluna da DB	Tipo de dados da Coluna da chave Primária e Comprimento
int	int	int, Coluna Identity
string	nvarchar(Max)	nvarchar(128)
decimal	decimal(18,2)	decimal(18,2)
float	real	real
byte[]	varbinary(Max)	varbinary(128)
datetime	datetime	datetime
bool	bit	bit
byte	tinyint	tinyint
short	smallint	smallint
long	bigint	bigint
double	float	float
char	No mapping	No mapping
sbyte	No mapping (throws exception)	No mapping
object	No mapping	No mapping

Nota: Na **class C#**, a propriedade referente à **Key** deve ser definida como **long**

DataAnnotations

- **DataAnnotations**

- Simples de especificar
- Menos flexível
- Útil para anotações simples como obrigatórios, tamanhos máximos...
- *DataAnnotations* tem limitações de configuração

Programação Web – 2022/2024 – LEI@CE+PL – ISEC/IPC @JB

DataAnnotations: Key

■ Chaves Primárias

```
using System.ComponentModel.DataAnnotations;
```

```
[Key]
```

```
private int NA1uno { get; set; }
```

```
[Key]
```

```
[DatabaseGenerated(DatabaseGeneratedOption.None)]  
private string SIGLA { get; set; }
```

DatabaseGeneratedOption

- None
- Identity
- Computed

DataAnnotations: Keys Compostas

- Chaves Primárias Compostas

```
public class Inscricoes
{
    [Key]
    [Column(Order=1)]
    public int DisciplinaId { get; set; }

    [Key, Column(Order=2)]
    public int AlunoId { get; set; }

    public DateTime DataInscricao { get; set; }
}
```

DataAnnotations: NotMapped

[NotMapped]

```
public int Idade { get; set; }
```

```
public string PrimeiroNome  
{  
    get { return AlunoNome; }  
}
```

```
public int IdadeGet  
{  
    get { return 0; }  
}
```


DataAnnotations: Índices

```
[Index]
```

```
public int DisciplinaId { get; set; }
```

```
[Index(IsUnique = true)]
```

```
public int DisciplinaId { get; set; }
```

ASP.Net Core Entity Framework Data Base First

Programação Web – 2022/2024 – LEC+CE+PL – ISEC/IPC @JB

Entity Framework – Data Base First

O ***Entity Framework Database First*** é uma abordagem alternativa no desenvolvimento de sites para a *Web* utilizando-se o ***ASP.Net Core Entity Framework***.

- Pode ser utilizada quer se disponha já de uma base de dados, por exemplo a utilizada num site que se quer actualizar ou quer se tenha desenvolvido especifica e previamente uma Base de Dados para um novo site
 - No entanto é comum que no desenvolvimento de raiz de um site, se utilize a abordagem *Code First*

Entity Framework – Data Base First

No caso de se dispor já de uma Base de Dados de um *site* em produção, a Base de Dados respectiva já deverá ter um número muito grande de tabelas as quais também já deverão conter um número muito significativo de dados inseridos

- Neste caso não faria sentido não se utilizar essa Base de Dados, por exemplo numa actualização desse *site*

Entity Framework – Data Base First

Nesta situação de se dispor já de uma Base de Dados, se se adoptasse a abordagem *Code First*, poderiam ocorrer alguns problemas:

- Ter-se-ia de se definir manualmente cada uma das classes C# correspondentes a cada uma das tabelas e as respectivas relações
- Não se poderia usar migrações *Code First* porque a base de dados existente provavelmente estará a ser gerida de uma maneira diferente.
- Embora isto não fosse um problema, perder-se-ia uma das grandes vantagens de se usar o *Code First*

Entity Framework – Data Base First

Deste modo, nestas situações é preferível usar a abordagem **Data Base First**:

- O que se faria criando manualmente o código das classes **C#**, será feita pela **EF** através da inspecção da base de dados existente.
- O resultado dessa acção de análise da **BD** pela **EF** será então a criação das classes de dados **C#**

Entity Framework – Data Base First

Para que o **EF** realize essa inspecção da **BD** e crie as respectivas classes **C#**, basta executar o seguinte comando:

Scaffold-DbContext "MyConnectionStringHere"

E o Entity Framework fará o resto por nós