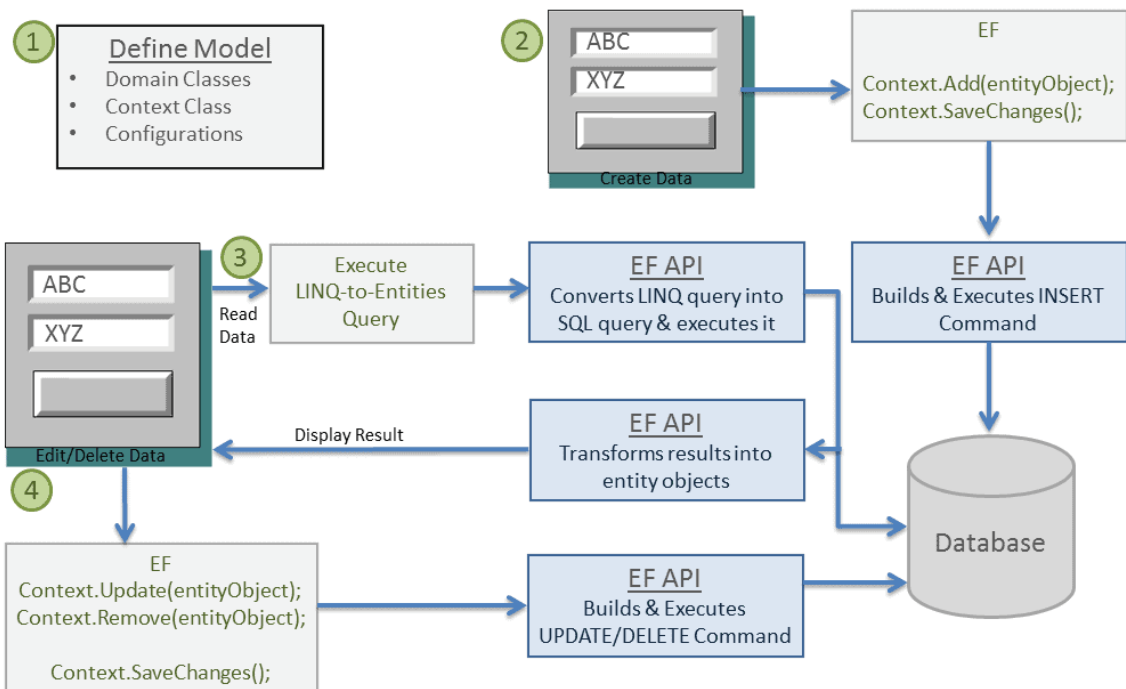
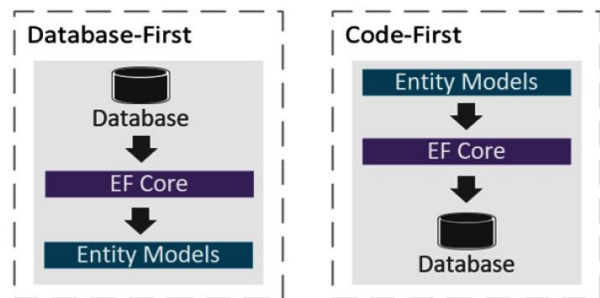
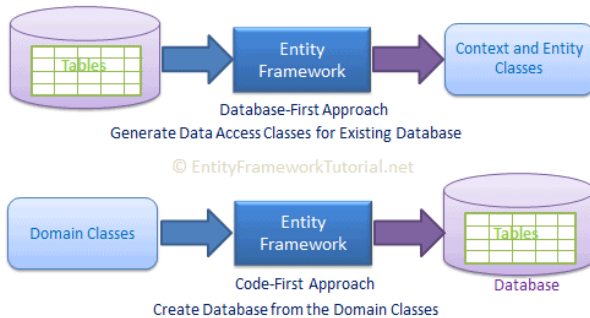
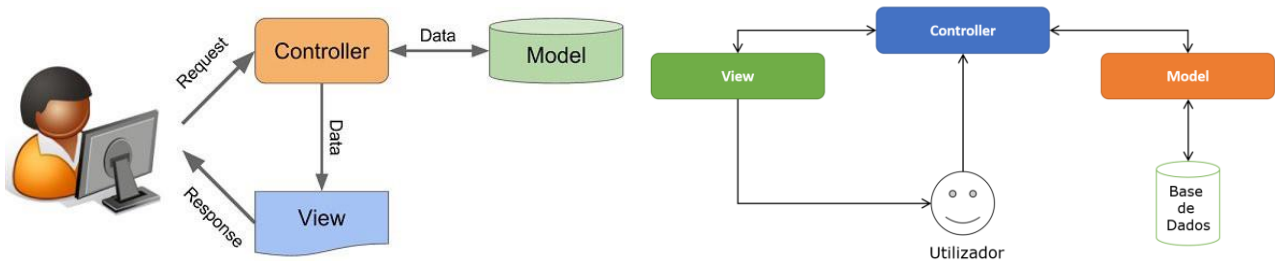


Objetivo:

- Conhecer a Entity Framework
- Criar um projeto Web ASP.NET Core MVC
- Base de dados - Abordagem Code First
- CRUD (Create, Read, Update, Delete)



Entity Framework	3
Criar aplicação ASP.NET Core	3
Code First	6
Controllers e Views automáticos (Scaffold)	14
Controllers e Views automáticos (Entity Framework)	14

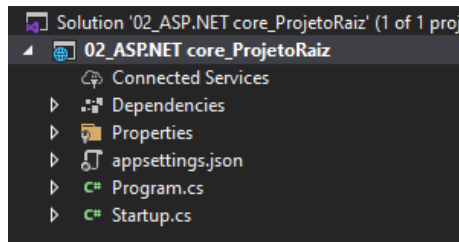
Entity Framework

Para mais detalhes:

ASP.NET Core-Base dados-Entity Framework.pdf

Criar aplicação ASP.NET Core

1. o projeto ASP.NET Core baseado no Template MVC (01-ASP.NET Core-Conceitos fundamentais) ou Empty (02-ASP.NET Core-Projeto de raiz)



Configurar aplicação para: ASP.NET Core MVC

Startup.cs

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();
}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseRouting();
    app.UseStaticFiles();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");
    });
}
```

ViewImports

Views_ViewImports.cshtml

```
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

ViewStart

Views_ViewStart.cshtml

```
@{
    Layout = "_Layout";
}
```

Criar o Layout

wwwroot\css\site.css

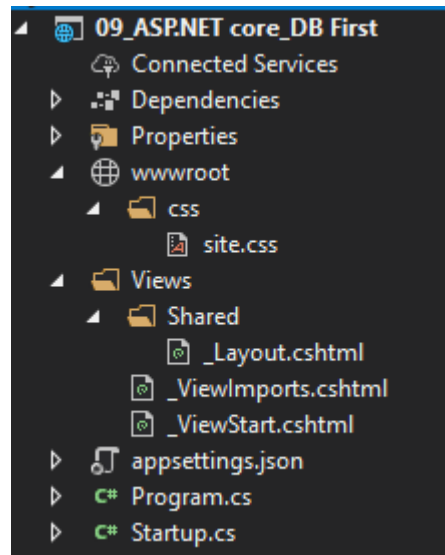
```
.rodape {
    text-align: center;
    /*Colocar o rodapé na base (fundo do browser)*/
    position: absolute;
    bottom: 0;
    width: 100%;
}
```

Views\Shared_Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>@ViewBag.Title</title>
    <link href="/css/site.css" rel="stylesheet" />
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css" rel="stylesheet"
    integrity="sha384-+0n0xVW2eSR5OomGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOO17+AMvyTG2x" crossorigin="anonymous">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-
    gtEjrD/SeCtmISKkKNUaaKMoLD0//E1J19smozuhV6z3Iehds+3U1b9Bn9P1x0x4" crossorigin="anonymous"></script>
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-lg navbar-light bg-light">
            <div class="container-fluid">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">ASP.NET Core</a>
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"
                    aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="collapse navbar-collapse" id="navbarNav">
                    <ul class="navbar-nav">
                        <li class="nav-item">
                            <a class="nav-link active" aria-current="page" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" asp-area="" asp-controller="Blogs" asp-action="Index">Blogs</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" asp-area="" asp-controller="Posts" asp-action="Index">Posts</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" asp-area="" asp-controller="Home" asp-action="Sobre">Sobre</a>
                        </li>
                    </ul>
                </div>
            </div>
        </nav>
    </header>

    <div class="container">
        @RenderBody()
    </div>

    <footer class="rodape">
        <hr />
        &copy; 2021-TPW
    </footer>
</body>
</html>
```



2. Adicionar um Controller (03-ASP.NET Core-Controllers)

Controllers\HomeController.cs
<pre>using Microsoft.AspNetCore.Mvc; using System; using System.Collections.Generic; using System.Linq; using System.Threading.Tasks; namespace _07_ASP.NET_core_Forms.Controllers { public class HomeController : Controller { public IActionResult Index() { return View(); } } }</pre>

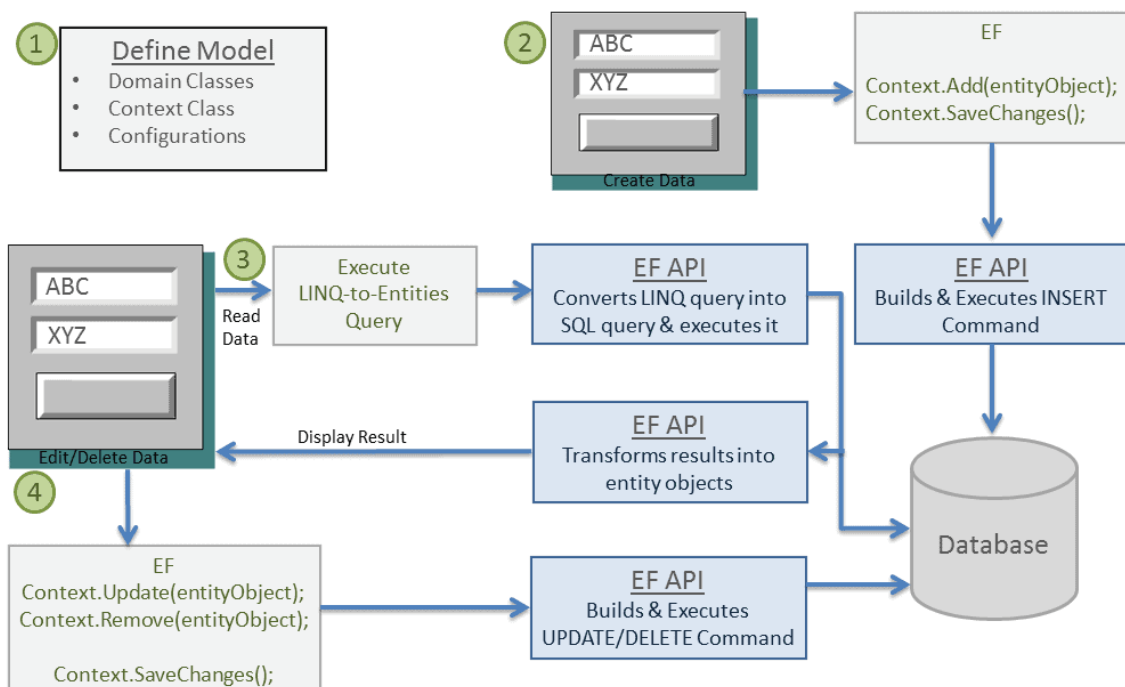
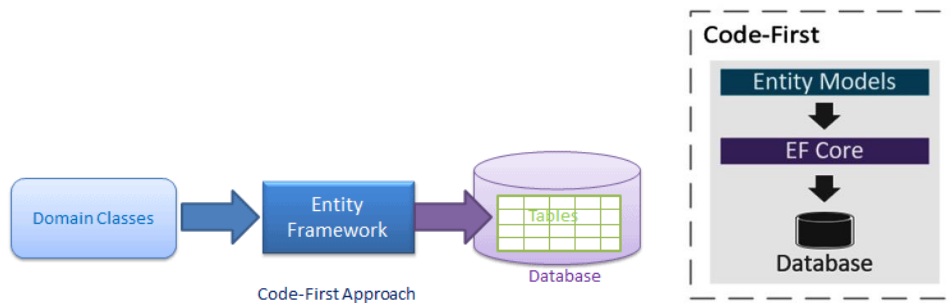
3. Adicionar uma View para a ação Index (04-ASP.NET Core-Views)

Views\Home\index.cshtml
<pre>@{ ViewData["Title"] = "Index"; } <h1>Index</h1></pre>

Code First

A abordagem **Code First** tem por objetivo a criação de uma nova base de dados e posterior atualização a partir do modelo de dados (classes). Assim, o Code First cria uma base de dados vazia ao qual adicionará novas tabelas.

O Code First permite a definição do modelo de dados usando classes C# ou VB.Net e através dos comandos da Entity Framework gerar a respectiva base de dados e tabelas com a possibilidade de atualizar a base de dados posteriormente.



Neste exemplo vamos usar a abordagem **Code First** que permite a criação de uma nova base de dados e posterior atualização a partir do modelo de dados (classes).

Tarefas:

1. Devemos instalar o EF (Entity Framework)
2. Criar o modelo de dados
3. Criar e registar o contexto de base de dados (DbContext)
4. Criar as Migrações

1. Instalar (adicionar ao projeto) o EF (Entity Framework)

Linha de comando (Visual Studio)

Tools | NuGet Package Manager | Package Manager Console

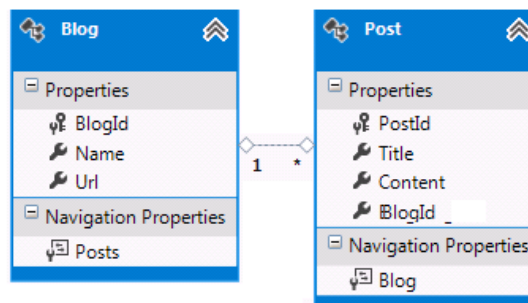
Podemos usar **Tab** para pedir ajuda de contexto

```
Install-Package Microsoft.EntityFrameworkCore -Version 5.0.6
Install-Package Microsoft.EntityFrameworkCore.Tools
Install-Package Microsoft.EntityFrameworkCore.Design
Install-Package Microsoft.EntityFrameworkCore.SqlServer
```

Ou Assistente

Tools | NuGet Package Manager | Manager Nugets Packages for Solution

2. Criar o Modelo de dados (Entidades)



Models\Blog.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace _10_ASP.NET_core_BD_First.Models
{
    public class Blog
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public ICollection<Post> Posts { get; set; }
    }
}
```

A propriedade Id torna-se a coluna chave primária da tabela, desta classe. Por padrão, o EF Core interpreta uma propriedade que é nomeada ID ou ClassNameID como a chave primária. Assim, podia ser BlogId

A propriedade Posts é uma propriedade de navegação. As propriedades de navegação são ligações a outras entidades com as quais esta (tabela) se relaciona. Neste caso, a propriedade Posts da entidade Blog contém todas as entidades Post a ela associadas.

Models\Post.cs
<pre>using System; using System.Collections.Generic; using System.Linq; using System.Threading.Tasks; namespace _10_ASP.NET_core_BD_First.Models { public class Post { public int PostId { get; set; } public string Title { get; set; } public string Content { get; set; } public int BlogId { get; set; } public Blog Blog { get; set; } } }</pre>

A propriedade BlogId é uma chave estrangeira e a propriedade de navegação correspondente é Blog. Uma entidade Post está associada somente a uma entidade Blog, portanto, a propriedade é simples.

3. Criar o contexto de base de dados (DbContext)

Data\DbBlogContext.cs
<pre>using _10_ASP.NET_core_BD_First.Models; using Microsoft.EntityFrameworkCore; using System; using System.Collections.Generic; using System.Linq; using System.Threading.Tasks; namespace _10_ASP.NET_core_BD_First.Data { public class DbBlogContext: DbContext { public DbSet<Blog> Blogs { get; set; } public DbSet<Post> Posts { get; set; } } }</pre>

4. Registrar o Contexto de base de dados

ConnectionString

<pre>Data Source=(LocalDB)\MSSQLLocalDB;Initial Catalog= DbLogs;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False; ApplicationIntent=ReadWrite; MultiSubnetFailover=False</pre>
--

Por questões de segurança, e também para mais fácil gestão, vamos colocar a Connection String num ficheiro externo de configurações: appsettings.json
A base dados será criada com o nome de “DbBlogs”

appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "BlogsConnection": "Data Source=(LocalDB)\\MSSQLLocalDB;Initial Catalog=DbBlogs; Integrated Security=True; Connect Timeout=30; Encrypt=False; TrustServerCertificate=False;ApplicationIntent=ReadWrite; MultiSubnetFailover=False"
  }
}
```

Startup.cs

```
public IConfiguration Configuration { get; }
public Startup(IConfiguration configuration)
{
    Configuration = configuration;
}

public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<DbBlogContext>(options => options.UseSqlServer(Configuration.GetConnectionString("BlogsConnection")));

    services.AddControllersWithViews();
}
```

5. Definir o data provider da base de dados (DbContext)

Data\ DbBlogContext.cs

```
using _10_ASP.NET_core_BD_First.Models;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace _10_ASP.NET_core_BD_First.Data
{
    public class DbBlogContext: DbContext
    {
        public IConfiguration Configuration { get; }
        public DbBlogContext(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public DbSet<Blog> Blogs { get; set; }
        public DbSet<Post> Posts { get; set; }
    }
}
```

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    if (!optionsBuilder.IsConfigured)
    {
        optionsBuilder.UseSqlServer(Configuration.GetConnectionString("BlogsConnection"));
    }
}
}
```

6. Criar as migrações

Permitem manter o modelo de dados em sincronia com a base de dados. Assim, servem para criar pela primeira vez, a base de dados e respetivas tabelas. E depois sempre que alteramos o modelo de dados (classes), podemos novamente atualizar a base de dados.

Os ficheiros de migração (atualização) serão criadas na pasta Data\Migrations

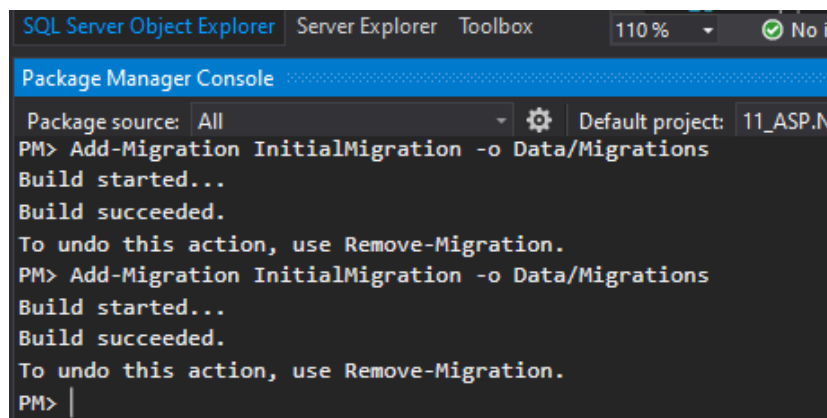
Comando Add-Migration

```
Add-Migration InitialMigration -o Data/Migrations
```

Linha de comando (Visual Studio)

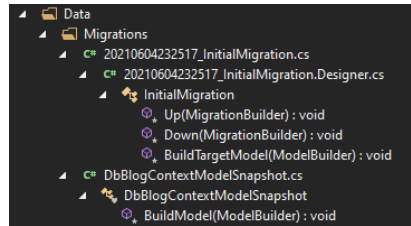
Tools | NuGet Package Manager | Package Manager Console

Podemos usar **Tab** para pedir ajuda de contexto



```
SQL Server Object Explorer | Server Explorer | Toolbox | 110 % | No i
Package Manager Console
Package source: All | Settings | Default project: 11_ASP.N
PM> Add-Migration InitialMigration -o Data/Migrations
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
PM> Add-Migration InitialMigration -o Data/Migrations
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
PM> |
```

Ficheiros criados:



Data\Migrations\20210604232517_InitialMigration.cs

```
using Microsoft.EntityFrameworkCore.Migrations;

namespace _10_ASP.NET_core_BD_First.Data.Migrations
{
    public partial class InitialMigration : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Blogs",
                columns: table => new
                {
                    BlogId = table.Column<int>(type: "int", nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                    Name = table.Column<string>(type: "nvarchar(max)", nullable: true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Blogs", x => x.BlogId);
                });

            migrationBuilder.CreateTable(
                name: "Posts",
                columns: table => new
                {
                    PostId = table.Column<int>(type: "int", nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                    Title = table.Column<string>(type: "nvarchar(max)", nullable: true),
                    Content = table.Column<string>(type: "nvarchar(max)", nullable: true),
                    BlogId = table.Column<int>(type: "int", nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Posts", x => x.PostId);
                    table.ForeignKey(
                        name: "FK_Posts_Blogs_BlogId",
                        column: x => x.BlogId,
                        principalTable: "Blogs",
                        principalColumn: "BlogId",
                        onDelete: ReferentialAction.Cascade);
                });

            migrationBuilder.CreateIndex(
                name: "IX_Posts_BlogId",
                table: "Posts",
                column: "BlogId");
        }

        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "Posts");

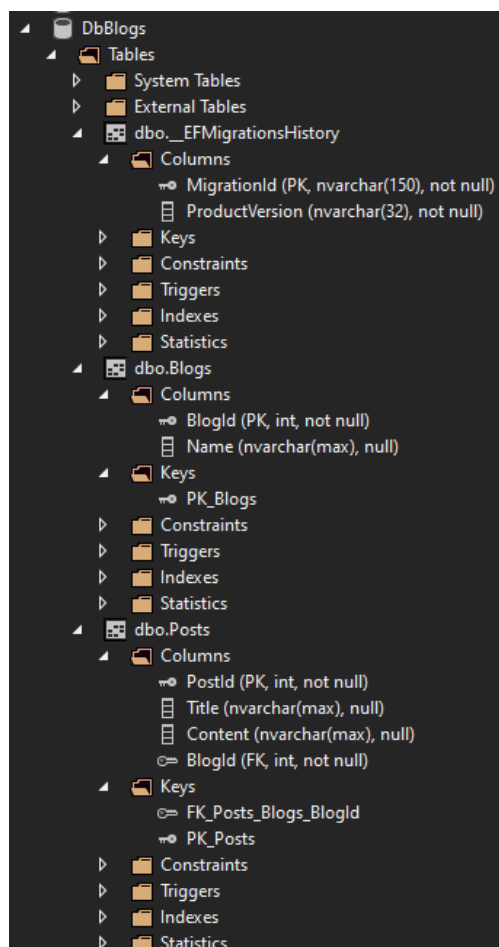
            migrationBuilder.DropTable(
                name: "Blogs");
        }
    }
}
```

Uma vez criados os ficheiros de migração, podem os aplicar. Como é a primeira vez, será criada a Base de dados e respetivas tabelas.

Comando Update-Database
Update-Database

```
Package Manager Console
Package source: All
PM> Update-Database
Build started...
Build succeeded.
Applying migration '20210604232517_InitialMigration'.
Done.
PM>
```

Depois de executado o comando anterior com sucesso, teremos a base de dados (DbBlogs) e respetivas tabelas criadas (Blogs, Posts) com as respetivas chaves.



Em vez de usar migrações de dados (para criar a BD e respectivas tabelas), podemos usar uma classe para inicializar a Base dados.

Classe para inicializar a Base dados

Data\DbInitializer.cs

```
public class DbInitializer
{
    public static void Initialize(DbSchoolContext context)
    {
        context.Database.EnsureCreated();
    }
}
```

Program.cs

```
public static void Main(string[] args)
{
    //CreateHostBuilder(args).Build().Run();

    var host = CreateHostBuilder(args).Build();

    CreateDbIfNotExists(host);

    host.Run();
}
```

A primeira vez que a aplicação é executada a BD é criada com as respectivas tabelas e com os registos.

Se mudar o modelo:

- Eliminar a base de dados
- E correr novamente a aplicação.

Controllers e Views automáticos (Scaffold)

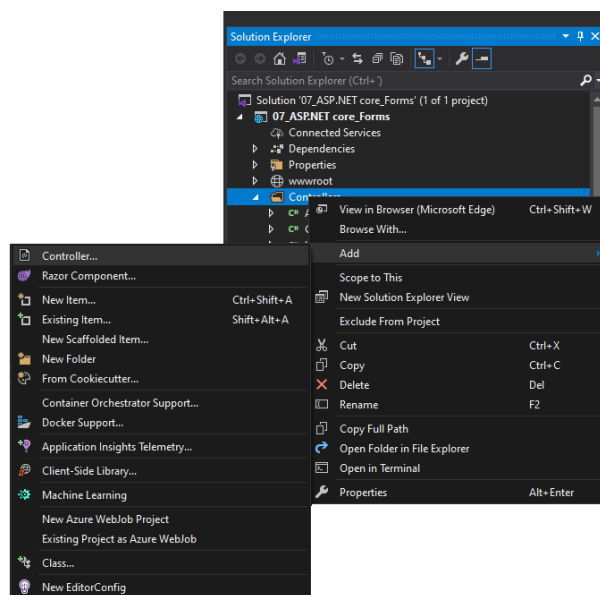
Scaffolding é uma framework de geração de código que adiciona códigos automaticamente e cria páginas de visualização (Views) e controladores (Controllers).

O Scaffolding torna o trabalho do programador mais fácil criando controladores (Controllers) e páginas de visualização (Views) para o modelo de dados (Models).

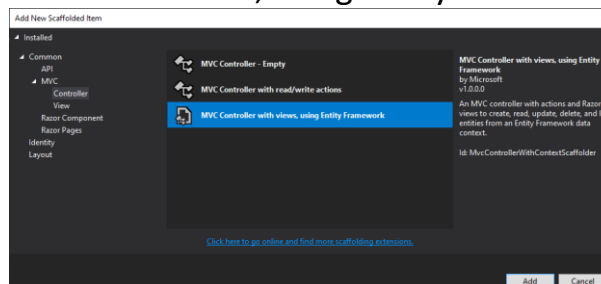
Gera códigos e páginas para a operação CRUD (**C**reate (Criar), **R**ead (Ler), **U**ppdate (Atualizar) e **D**elete (Excluir)).

Controllers e Views automáticos (Entity Framework)

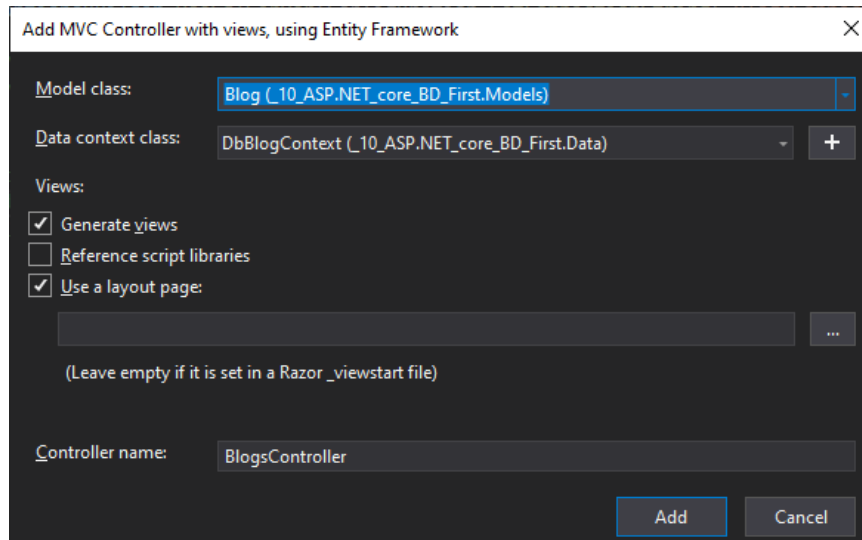
Em Solution Explorer, sobre o nome da pasta “*Controllers*”, fazer Tecla direita do rato e seleconar **Add -> New Scaffolded** ou **Controller**, como na figura a seguir:



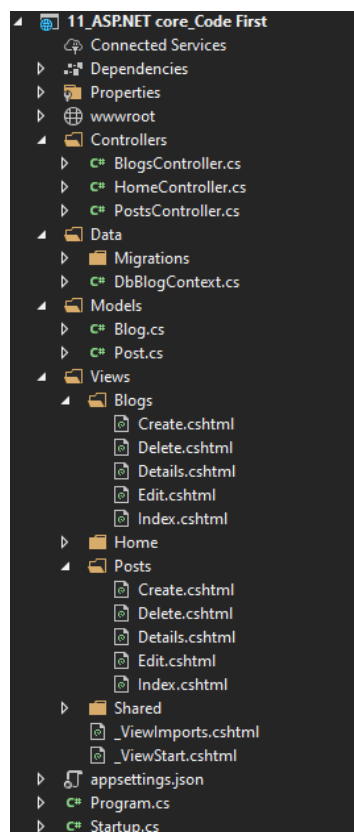
Selecionar “MVC Controller with views, using Entity Framework”



Selecionar a Classe do modelo de dados, criar o contexto da base de dados (+) e definir Nome para o controller. Assinalar “Generate views” para que as Views sejam também criadas. Adicionar



BlogsController, PostsController e Views Blogs, Posts criadas automaticamente



URL: .../Blogs e .../Posts

Código gerado automaticamente (Controllers)

Controllers\BlogsController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using _10_ASP.NET_core_BD_First.Data;
using _10_ASP.NET_core_BD_First.Models;

namespace _10_ASP.NET_core_BD_First.Controllers
{
    public class BlogsController : Controller
    {
        private readonly DbBlogContext _context;

        public BlogsController(DbBlogContext context)
        {
            _context = context;
        }

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

        // GET: Blogs/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var blog = await _context.Blogs
                .FirstOrDefaultAsync(m => m.BlogId == id);
            if (blog == null)
            {
                return NotFound();
            }

            return View(blog);
        }

        // GET: Blogs/Create
        public IActionResult Create()
        {
            return View();
        }

        // POST: Blogs/Create
        // To protect from overposting attacks, enable the specific properties you want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("BlogId,Name")] Blog blog)
        {
            if (ModelState.IsValid)
            {
                _context.Add(blog);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(blog);
        }
    }
}
```



```
// GET: Blogs/Edit/5
public async Task<IActionResult> Edit(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var blog = await _context.Blogs.FindAsync(id);
    if (blog == null)
    {
        return NotFound();
    }
    return View(blog);
}

// POST: Blogs/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id, [Bind("BlogId,Name")] Blog blog)
{
    if (id != blog.BlogId)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(blog);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!BlogExists(blog.BlogId))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    return View(blog);
}
```

```
// GET: Blogs/Delete/5
public async Task<IActionResult> Delete(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var blog = await _context.Blogs
        .FirstOrDefaultAsync(m => m.BlogId == id);
    if (blog == null)
    {
        return NotFound();
    }

    return View(blog);
}

// POST: Blogs/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var blog = await _context.Blogs.FindAsync(id);
    _context.Blogs.Remove(blog);
    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

private bool BlogExists(int id)
{
    return _context.Blogs.Any(e => e.BlogId == id);
}
}
```

Código gerado automaticamente (Views)

Views\Blogs\Index.cshtml

```
@model IEnumerable<_10_ASP.NET_core_BD_First.Models.Blog>

@{
    ViewData["Title"] = "Index";
}

<h1>Index</h1>

<p>
    <a asp-action="Create">Create New</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Name)
            </th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model) {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Name)
                </td>
                <td>
                    <a asp-action="Edit" asp-route-id="@item.BlogId">Edit</a> |
                    <a asp-action="Details" asp-route-id="@item.BlogId">Details</a> |
                    <a asp-action="Delete" asp-route-id="@item.BlogId">Delete</a>
                </td>
            </tr>
        }
    </tbody>
</table>
```

Views\Blogs\Create.cshtml

```
@model _10_ASP.NET_core_BD_First.Models.Post

@{
    ViewData["Title"] = "Create";
}

<h1>Create</h1>

<h4>Post</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Title" class="control-label"></label>
                <input asp-for="Title" class="form-control" />
                <span asp-validation-for="Title" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Content" class="control-label"></label>
                <input asp-for="Content" class="form-control" />
                <span asp-validation-for="Content" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="BlogId" class="control-label"></label>
                <select asp-for="BlogId" class="form-control" asp-items="ViewBag.BlogId"></select>
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>
```

Views\Blogs\Details.cshtml

```
@model _10_ASP.NET_core_BD_First.Models.Post

@{
    ViewData["Title"] = "Details";
}

<h1>Details</h1>

<div>
    <h4>Post</h4>
    <hr />
    <dl class="row">
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Title)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Title)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Content)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Content)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Blog)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Blog.BlogId)
        </dd>
    </dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model.PostId">Edit</a> |
    <a asp-action="Index">Back to List</a>
</div>
```

Views\Blogs\Edit.cshtml

```
@model _10_ASP.NET_core_BD_First.Models.Post

@{
    ViewData["Title"] = "Edit";
}

<h1>Edit</h1>

<h4>Post</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="PostId" />
            <div class="form-group">
                <label asp-for="Title" class="control-label"></label>
                <input asp-for="Title" class="form-control" />
                <span asp-validation-for="Title" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Content" class="control-label"></label>
                <input asp-for="Content" class="form-control" />
                <span asp-validation-for="Content" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="BlogId" class="control-label"></label>
                <select asp-for="BlogId" class="form-control" asp-items="ViewBag.BlogId"></select>
                <span asp-validation-for="BlogId" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Save" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>
```

Views\Blogs\Delete.cshtml

```
@model _10_ASP.NET_core_BD_First.Models.Post

@{
    ViewData["Title"] = "Delete";
}

<h1>Delete</h1>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Post</h4>
    <hr />
    <dl class="row">
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Title)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Title)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Content)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Content)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Blog)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Blog.BlogId)
        </dd>
    </dl>

    <form asp-action="Delete">
        <input type="hidden" asp-for="PostId" />
        <input type="submit" value="Delete" class="btn btn-danger" /> |
        <a asp-action="Index">Back to List</a>
    </form>
</div>
```