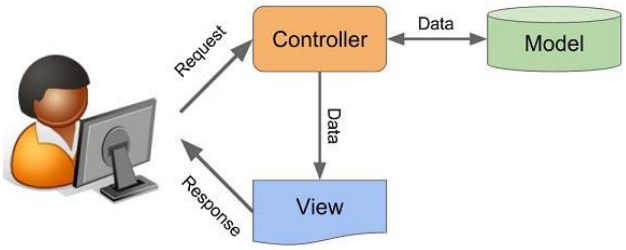
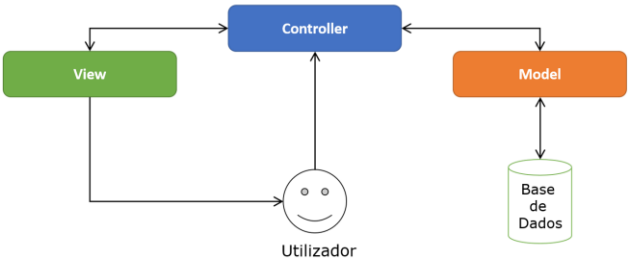
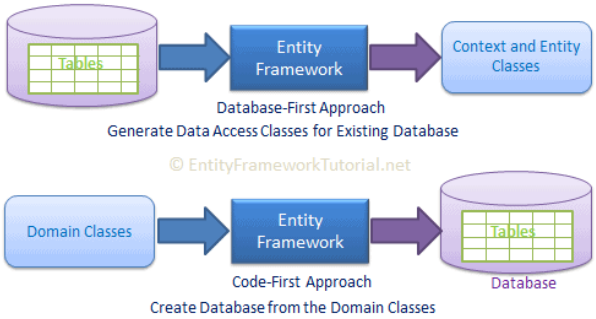
:: ASP.NET Core MVC :: Base dados – Database First ::

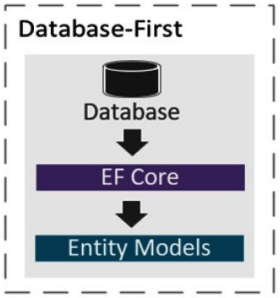
Objetivo:

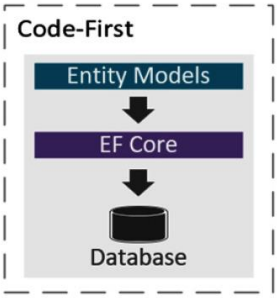
- Conhecer a Entity Framework
- Criar um projeto Web ASP.NET Core MVC
- Base de dados - Abordagem Database First
- CRUD (Create, Read, Update, Delete)
- Controllers e Views automáticos (Scaffold)

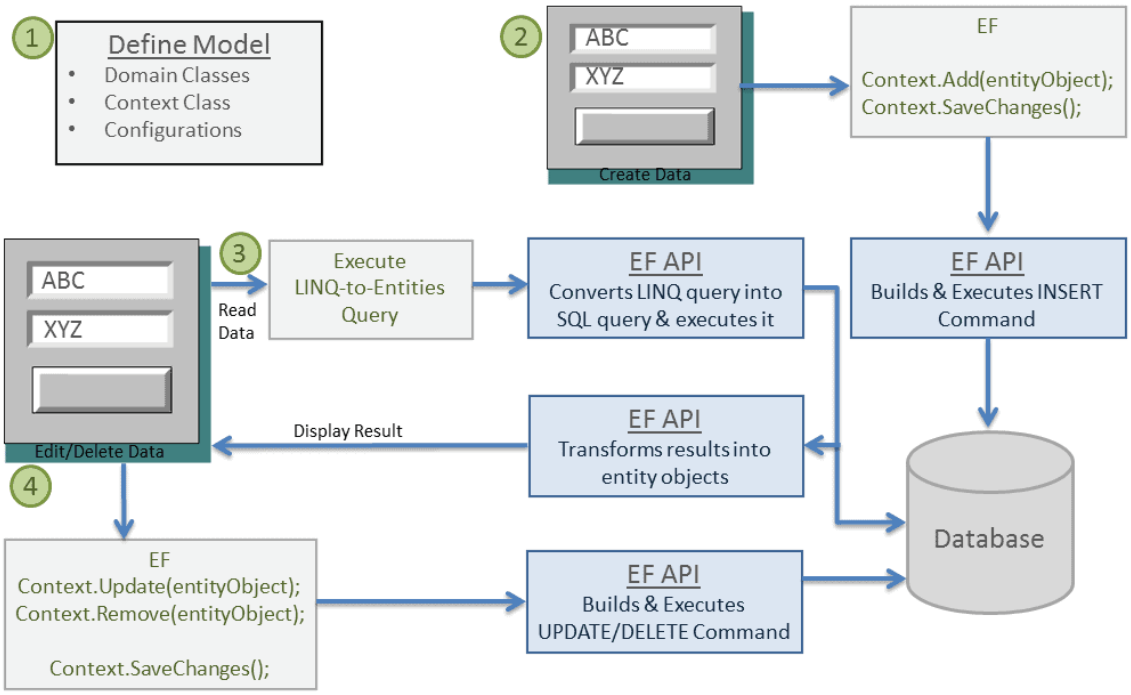










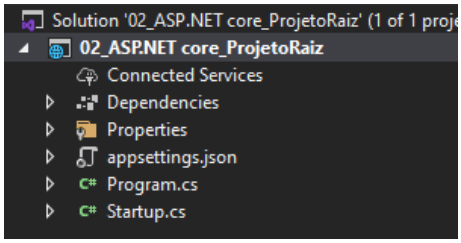


Entity Framework

Para mais detalhes:
ASP.NET Core-Base dados-Entity Framework.pdf

Criar aplicação ASP.NET Core

1. Projeto ASP.NET Core baseado no Template MVC (ASP.NET Core-Conceitos fundamentais) ou Empty (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";
}
```

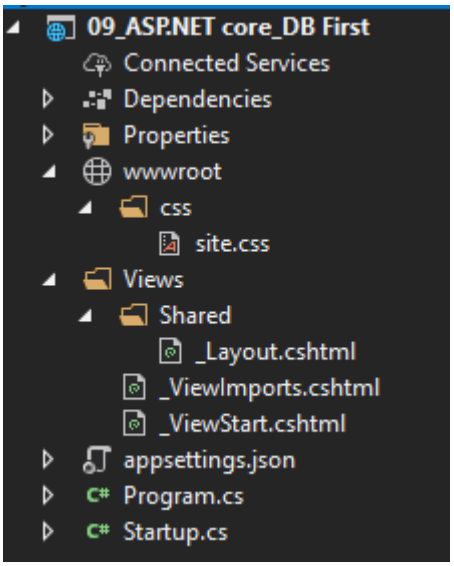
Criar o Layout

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-+0n0xVW2eSR5OomGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOOl7+AMvyTG2x" crossorigin="anonymous">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-
gtEjrD/SeCtmISkJkNUaaKMoLD0//ElJ19smozuHV6z3Iehds+3Ulb9Bn9Plx0x4" 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="Clientes" asp-action="Index">Clientes</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

```
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();
        }

    }
}
```

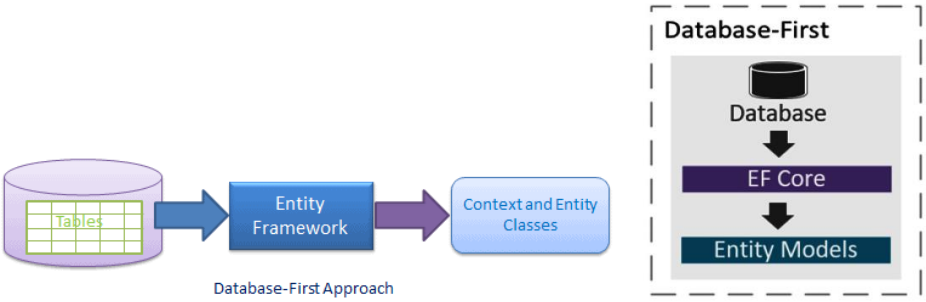
3. Adicionar uma View para a ação Index (04-ASP.NET Core-Views)

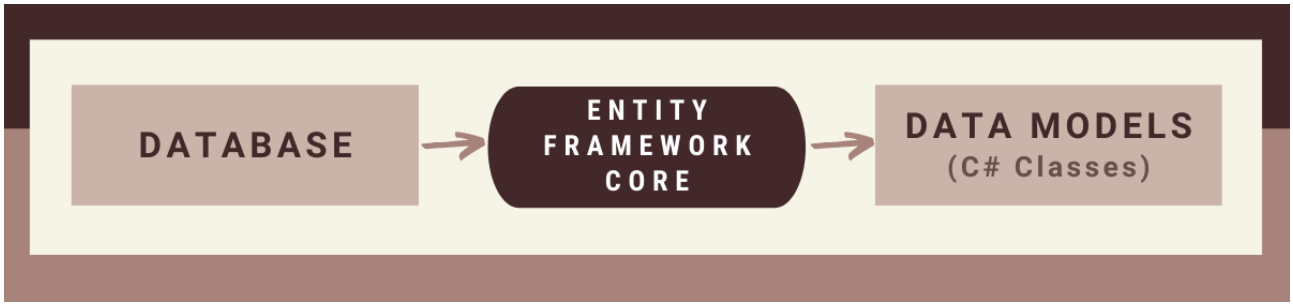
Views\Home\index.cshtml

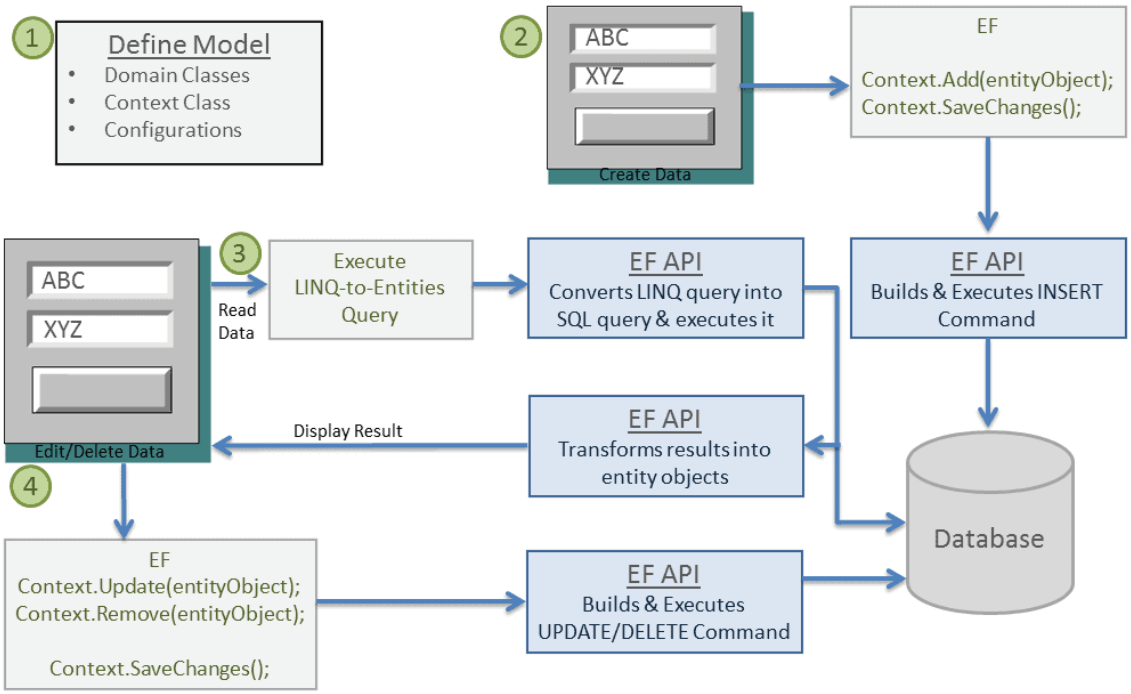
```
@{
    ViewData["Title"] = "Index";
}

<h1>Index</h1>
```

Database First

A abordagem **Database First** permite fazer engenharia reversa (reverse engineer) a partir de uma base de dados existente para um modelo de dados.

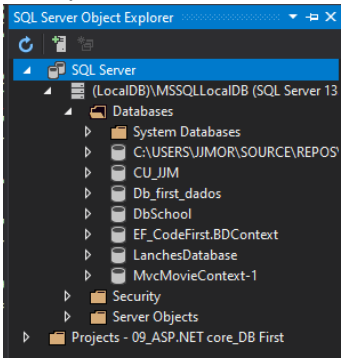




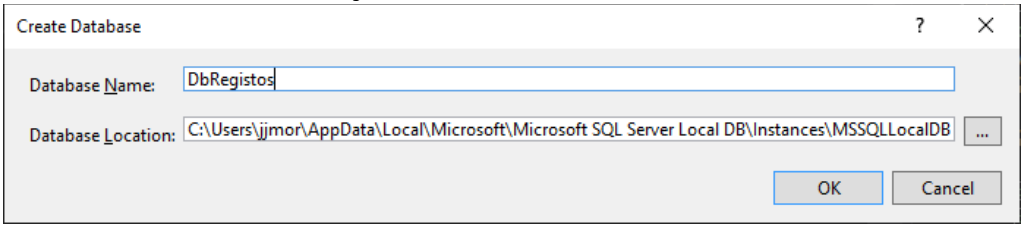


1. Criar a base de dados, ou usar uma que já exista
   Como, no exemplo, vamos usar SQL Server, podemos criar no SQL Server, ou usar o SQL Server Object Explorer do VS Studio

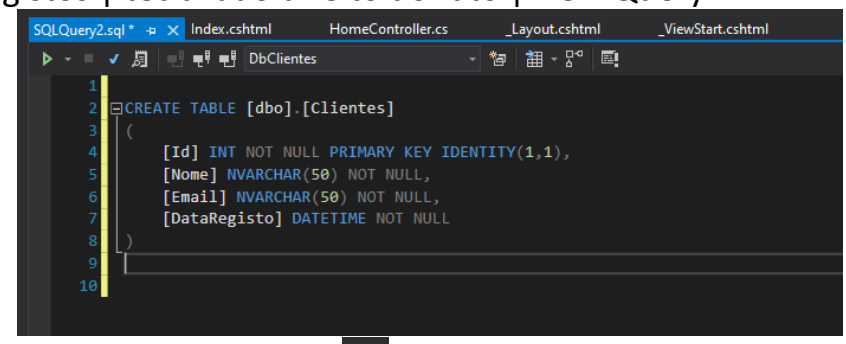
   No VS Studio:
   1. View > SQL Server Object Explorer
   2. Abrir SQL Server > (localdb)\MS S QLLocalDB



2. Na pasta *Databases*, fazer tecla direita do rato | Add New Database
   Definir nome e localização

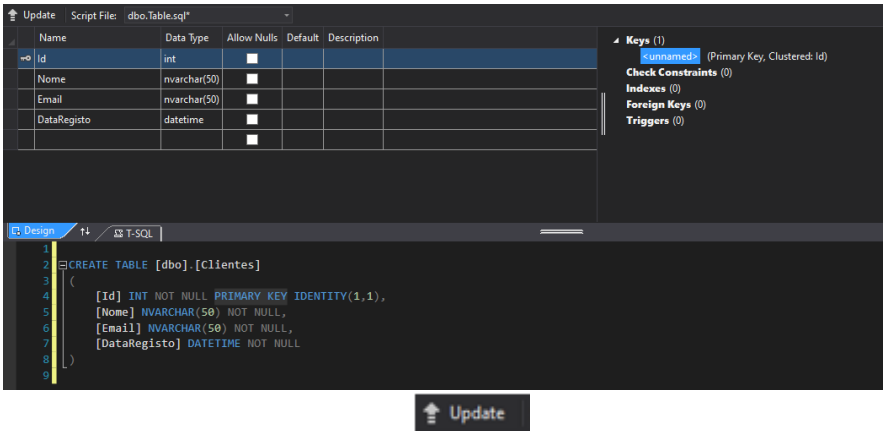


3. Criar as tabelas
   DbRegistos | tecla lado direito do rato | New Query




Execute

ou
DbRegistos | Tables | Tecla lado direto do rato | Add New Table



Script SQL

`CREATE TABLE [dbo].[Clientes]`
`(`
`  [Id] INT NOT NULL PRIMARY KEY IDENTITY(1,1),`
`  [Nome] NVARCHAR(30) NOT NULL,`
`  [Email] NVARCHAR(50) NOT NULL,`
`  [DataRegisto] DATETIME NOT NULL`
`  )`

Neste exemplo vamos usar a abordagem **Database First** que permite fazer engenharia reversa (reverse engineer) a partir de uma base de dados existente para um modelo de dados.

4. 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.SqlServer
Install-Package Microsoft.EntityFrameworkCore.SqlServer.Design
```

Ou Assistente
Tools | NuGet Package Manager | Manager Nugets Packages for Solution

5. Gerar Modelo a partir da base de dados

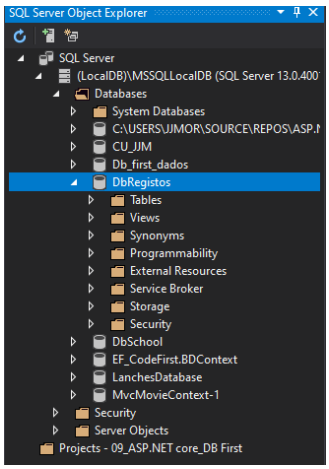
Comando **Scaffold-DbContext**

Cria os modelos a partir da base de dados e o contexto de base de dados para poder manipular a mesma.

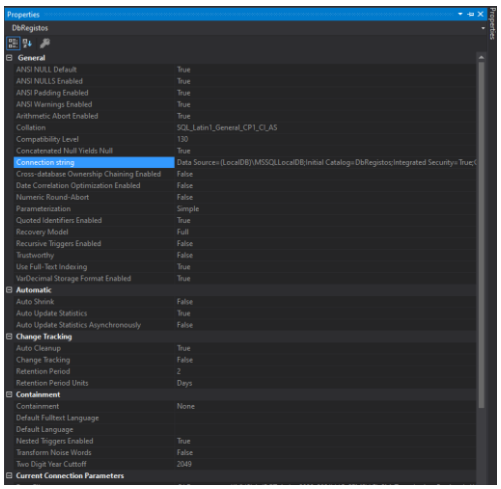
```
Scaffold-DbContext [-Connection] [-Provider] [-OutputDir] [-ContextDir] [-Context]
[-Schemas>] [-Tables>] [-DataAnnotations] [-Force] [-Project] [-StartupProject]
[<CommonParameters>]
```

https://docs.microsoft.com/en-us/ef/core/cli/powershell

Obter os parâmetros de ligação com a base de dados (Connection String, https://www.connectionstrings.com/)



Selecionar Base dados | Propriedades (**F4**)



Connection string	Data Source=(LocalDB)\MSSQLLocalDB;Initial Catalog=DbRegistos;Integrated Security=True;

`ConnectionString`

```
Data Source=(LocalDB)\MSSQLLocalDB;Initial Catalog=DbRegistos;Integrated
Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;
ApplicationIntent=ReadWrite; MultiSubnetFailover=False
```

Já temos a *Connection String*, queremos que os modelos de dados das tabelas (classes) sejam criados na pasta Models e o contexto de base de dados (Classe) seja criado na pasta Data

Parâmetros a mudar, os outros ficam por omissão:

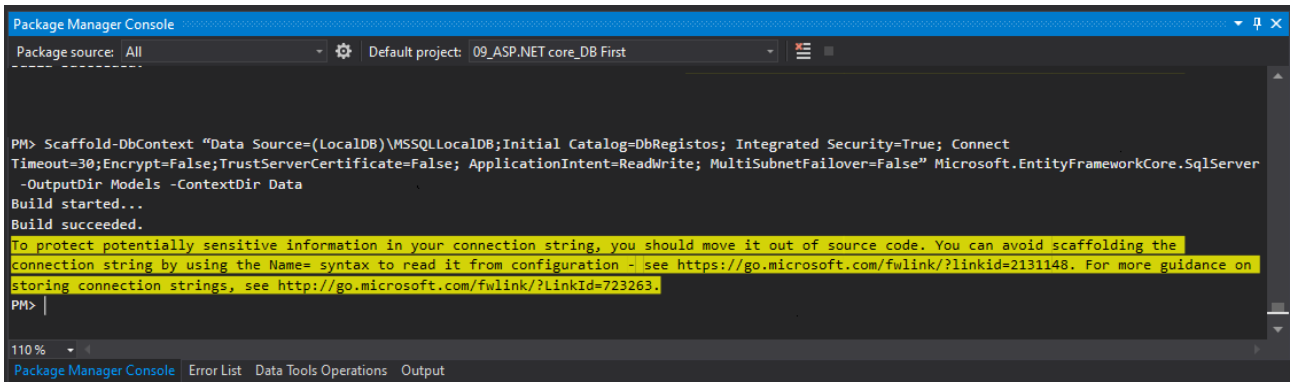
```
-Connection String "Connection String"
-OutputDir Models
-ContextDir Data
```

Linha de comando (Visual Studio)

Tools | NuGet Package Manager | Package Manager Console

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

Comando Scaffold-DbContext
Scaffold-DbContext **"Data Source=(LocalDB)\MSSQLLocalDB;Initial Catalog=DbRegistos; Integrated Security=True; Connect Timeout=30;Encrypt=False;TrustServerCertificate=False; ApplicationIntent=ReadWrite; MultiSubnetFailover=False"** Microsoft.EntityFrameworkCore.SqlServer **-OutputDir** Models **-ContextDir** Data



Depois de executado o comando anterior com sucesso, é gerado o seguinte código (classes) nas pastas: Models e Data

Models\Estudante.cs

```
using System;
using System.Collections.Generic;

#nullable disable

namespace _10_ASP.NET_core_BD_First.Models
{
    public partial class Cliente
    {
        public int Id { get; set; }
        public string Nome { get; set; }
        public string Email { get; set; }
        public DateTime DataRegisto { get; set; }
    }
}
```

Data\DbRegistosContext.cs

```
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata;
using _10_ASP.NET_core_BD_First.Models;

#nullable disable

namespace _10_ASP.NET_core_BD_First.Data
{
    public partial class DbRegistosContext : DbContext
    {
        public DbRegistosContext()
        {
        }

        public DbRegistosContext(DbContextOptions<DbRegistosContext> options)
            : base(options)
        {
        }

        public virtual DbSet<Cliente> Clientes { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            if (!optionsBuilder.IsConfigured)
            {
#warning To protect potentially sensitive information in your connection string, you should move it out of source code.
You can avoid scaffolding the connection string by using the Name= syntax to read it from configuration - see
https://go.microsoft.com/fwlink/?linkid=2131148. For more guidance on storing connection strings, see
http://go.microsoft.com/fwlink/?LinkId=723263.
                optionsBuilder.UseSqlServer("Data Source=(LocalDB)\\MSSQLLocalDB;Initial Catalog=DbRegistos; Integrated
Security=True; Connect Timeout=30;Encrypt=False;TrustServerCertificate=False; ApplicationIntent=ReadWrite;
MultiSubnetFailover=False");
            }
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.HasAnnotation("Relational:Collation", "SQL_Latin1_General_CP1_CI_AS");

            modelBuilder.Entity<Cliente>(entity =>
            {
                entity.Property(e => e.DataRegisto).HasColumnType("datetime");

                entity.Property(e => e.Email)
                    .IsRequired()
                    .HasMaxLength(50);

                entity.Property(e => e.Nome)
                    .IsRequired()
                    .HasMaxLength(30);
            });

            OnModelCreatingPartial(modelBuilder);
        }

        partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
    }
}
```

Como alertado por questões de segurança, e também para mais fácil gestão, vamos colocar a Connection String num ficheiro externo de configurações: appsettings.json

appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",

  "ConnectionStrings": {
    "RegistosConnection": "Data Source=(LocalDB)\\MSSQLLocalDB;Initial Catalog=DbRegistos; Integrated
Security=True; Connect Timeout=30; Encrypt=False; TrustServerCertificate=False;ApplicationIntent=ReadWrite;
MultiSubnetFailover=False"
  }

}
```

6. Registar o Contexto de base de dados

Startup.cs

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

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

   services.AddControllersWithViews();

}
```

Controllers e Views automáticos (Scaffold)

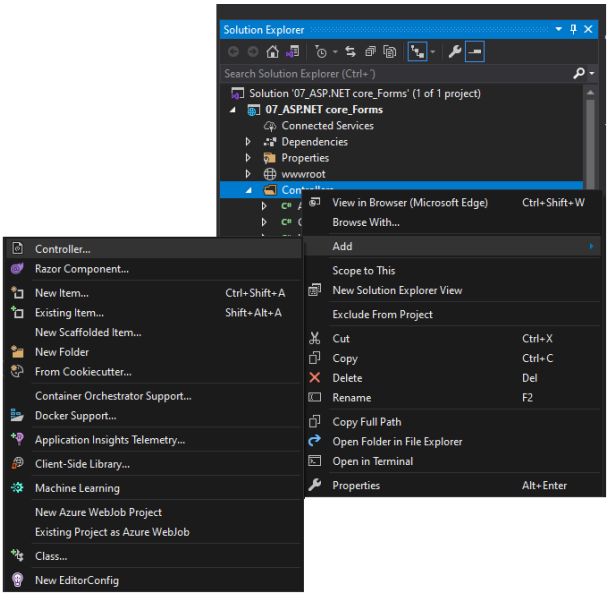
Scaffolding é uma framework de geração de código que adiciona códigos automaticamente e cria páginas de visualização (Views) e controladores (Controllers).

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

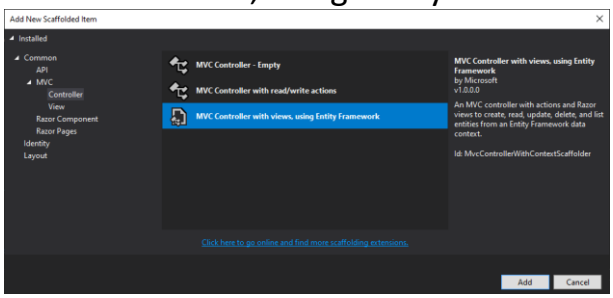
Gera códigos e páginas para a operação CRUD (**C**reate (Criar), **R**ead (Ler), **U**pdate (Atualizar) e **D**elete (Excluir)).

Controllers e Views automáticos (Entity Framework)

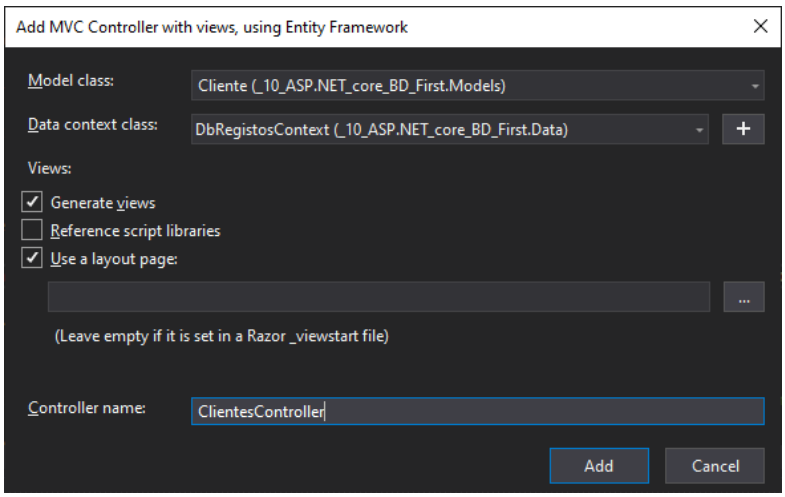
Em Solution Explorer, sobre o nome da pasta "*Controllers*", fazer Tecla direita do rato e seleconar **Add** -> **New Scaffolded** ou **Controller**, como na figura a seguir:

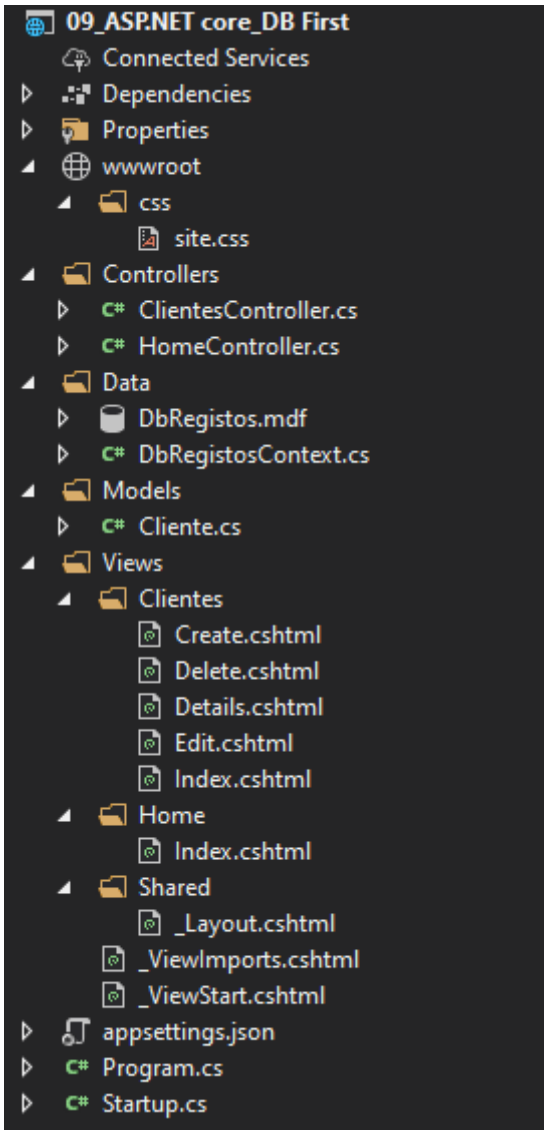


Selecionar "MVC Controller with views, using Entity Framework"



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





Código gerado automaticamente (Controllers)

Controllers\ClientesController.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 ClientesController : Controller
    {
        private readonly DbRegistosContext _context;

        public ClientesController(DbRegistosContext context)
        {
            _context = context;
        }

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

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

            var cliente = await _context.Clientes
                .FirstOrDefaultAsync(m => m.Id == id);
            if (cliente == null)
            {
                return NotFound();
            }

            return View(cliente);
        }

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

        // POST: Clientes/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("Id,Nome,Email,DataRegisto")] Cliente cliente)
        {
            if (ModelState.IsValid)
            {
                _context.Add(cliente);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(cliente);
        }
```

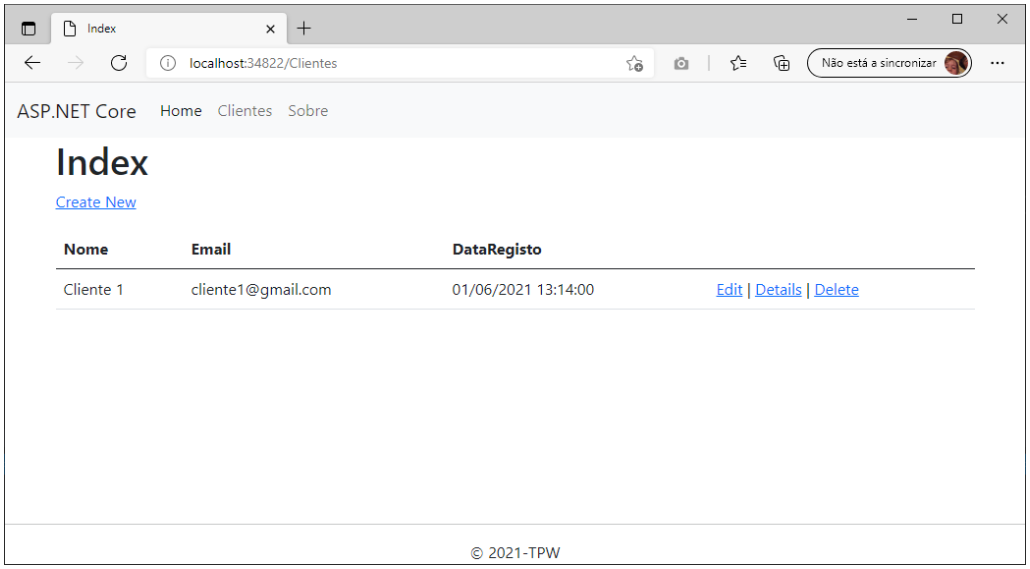
```
        // GET: Clientes/Edit/5
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

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

        // POST: Clientes/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("Id,Nome,Email,DataRegisto")] Cliente cliente)
        {
            if (id != cliente.Id)
            {
                return NotFound();
            }

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

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

            var cliente = await _context.Clientes
                .FirstOrDefaultAsync(m => m.Id == id);
            if (cliente == null)
            {
                return NotFound();
            }

            return View(cliente);
        }

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

        private bool ClienteExists(int id)
        {
            return _context.Clientes.Any(e => e.Id == id);
        }
    }
}
```

Código gerado automaticamente (Views)

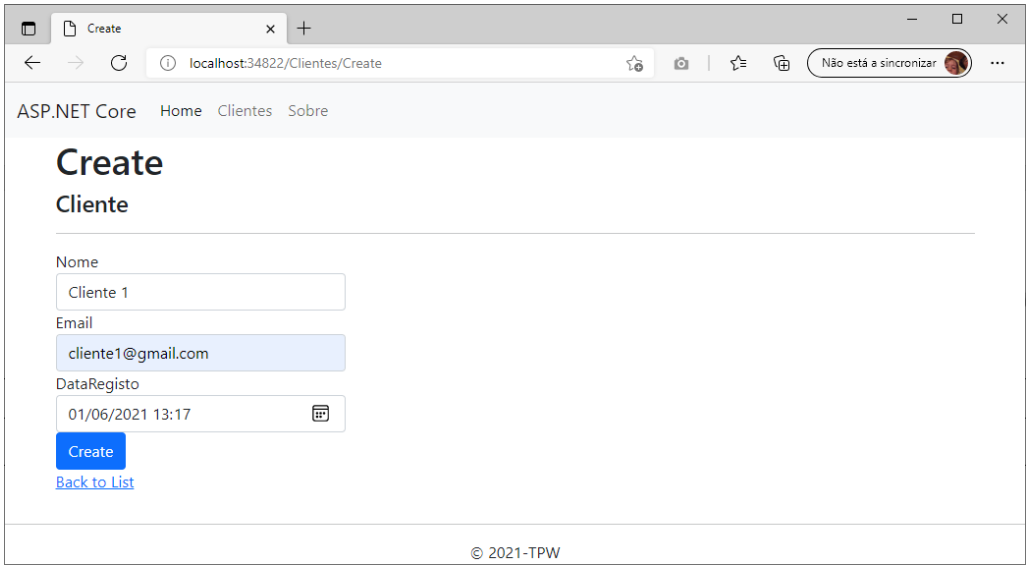
Views\Clientes\Index.cshtml

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

@{
    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.Nome)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Email)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.DataRegisto)
            </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
@foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Nome)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Email)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.DataRegisto)
            </td>
            <td>
                <a asp-action="Edit" asp-route-id="@item.Id">Edit</a> |
                <a asp-action="Details" asp-route-id="@item.Id">Details</a> |
                <a asp-action="Delete" asp-route-id="@item.Id">Delete</a>
            </td>
        </tr>
}
    </tbody>
</table>
```



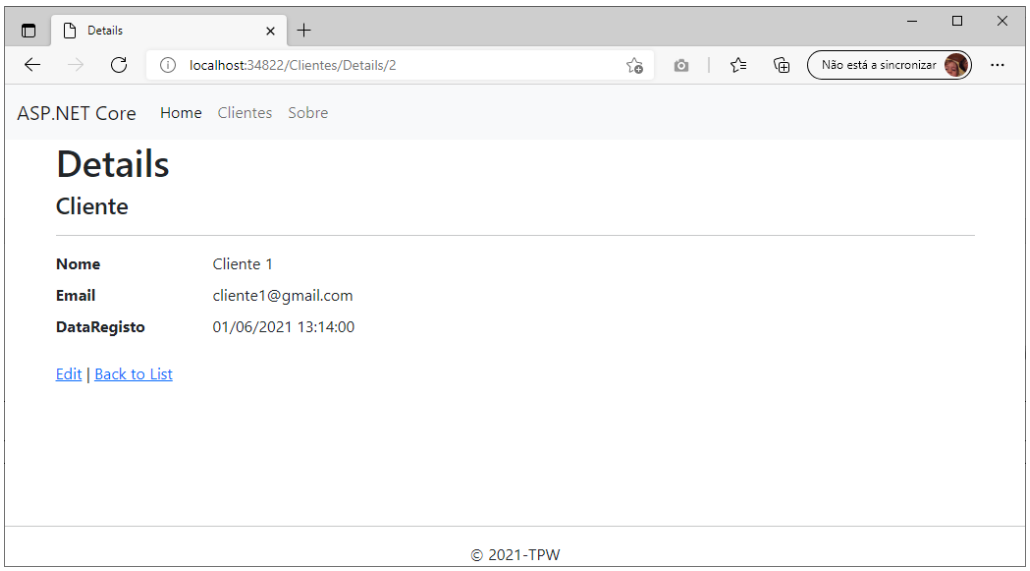
Views\ Clientes\Create.cshtml

```
@model _10_ASP.NET_core_BD_First.Models.Cliente

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

<h1>Create</h1>

<h4>Cliente</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="Nome" class="control-label"></label>
                <input asp-for="Nome" class="form-control" />
                <span asp-validation-for="Nome" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Email" class="control-label"></label>
                <input asp-for="Email" class="form-control" />
                <span asp-validation-for="Email" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="DataRegisto" class="control-label"></label>
                <input asp-for="DataRegisto" class="form-control" />
                <span asp-validation-for="DataRegisto" class="text-danger"></span>
            </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>
```



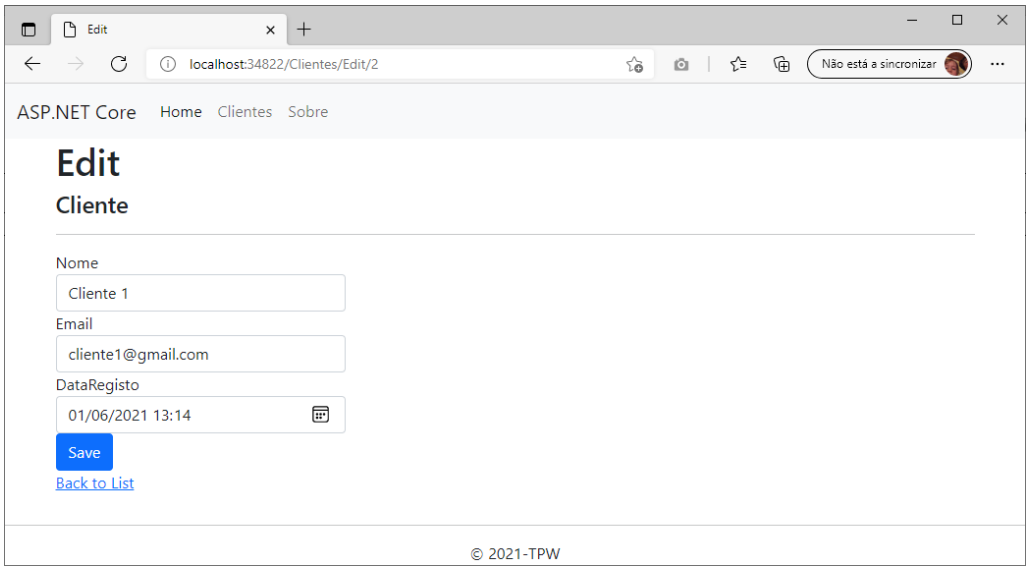
Views\ Clientes\Details.cshtml

```
@model _10_ASP.NET_core_BD_First.Models.Cliente

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

<h1>Details</h1>

<div>
    <h4>Cliente</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Nome)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Nome)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Email)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Email)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.DataRegisto)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.DataRegisto)
        </dd>
    </dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model.Id">Edit</a> |
    <a asp-action="Index">Back to List</a>
</div>
```



Views\ Clientes\Edit.cshtml

```
@model _10_ASP.NET_core_BD_First.Models.Cliente

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

<h1>Edit</h1>

<h4>Cliente</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="Id" />
            <div class="form-group">
                <label asp-for="Nome" class="control-label"></label>
                <input asp-for="Nome" class="form-control" />
                <span asp-validation-for="Nome" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Email" class="control-label"></label>
                <input asp-for="Email" class="form-control" />
                <span asp-validation-for="Email" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="DataRegisto" class="control-label"></label>
                <input asp-for="DataRegisto" class="form-control" />
                <span asp-validation-for="DataRegisto" 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\ Clientes\Delete.cshtml

```
@model _10_ASP.NET_core_BD_First.Models.Cliente

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

<h1>Delete</h1>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Cliente</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Nome)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Nome)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Email)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Email)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.DataRegisto)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.DataRegisto)
        </dd>
    </dl>

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