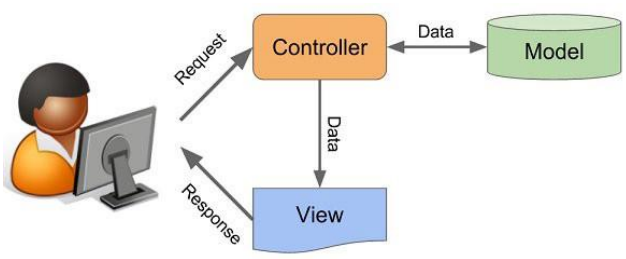
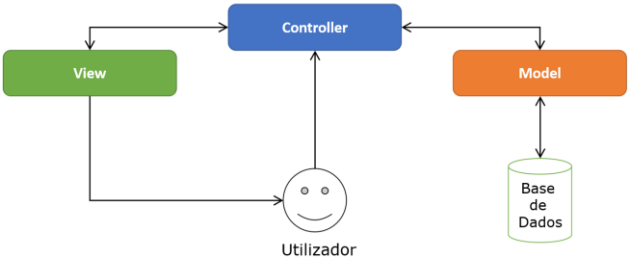
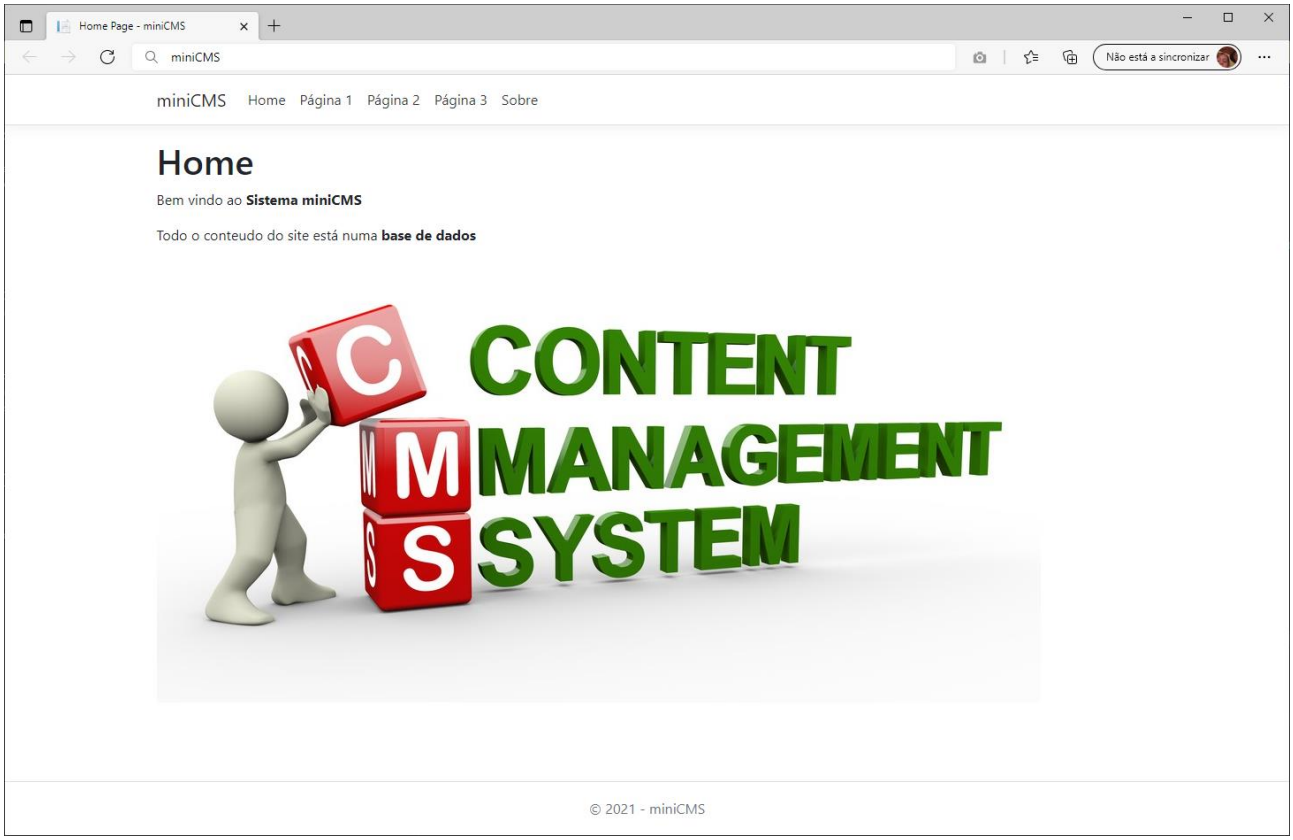
Objetivo:

- Criar um minCMS (Content Management Sytem)
- Usar a Entity Framework
- Abordagem Code First
- Frontoffice e backoffice
- Editor HTML (richtexteditor)



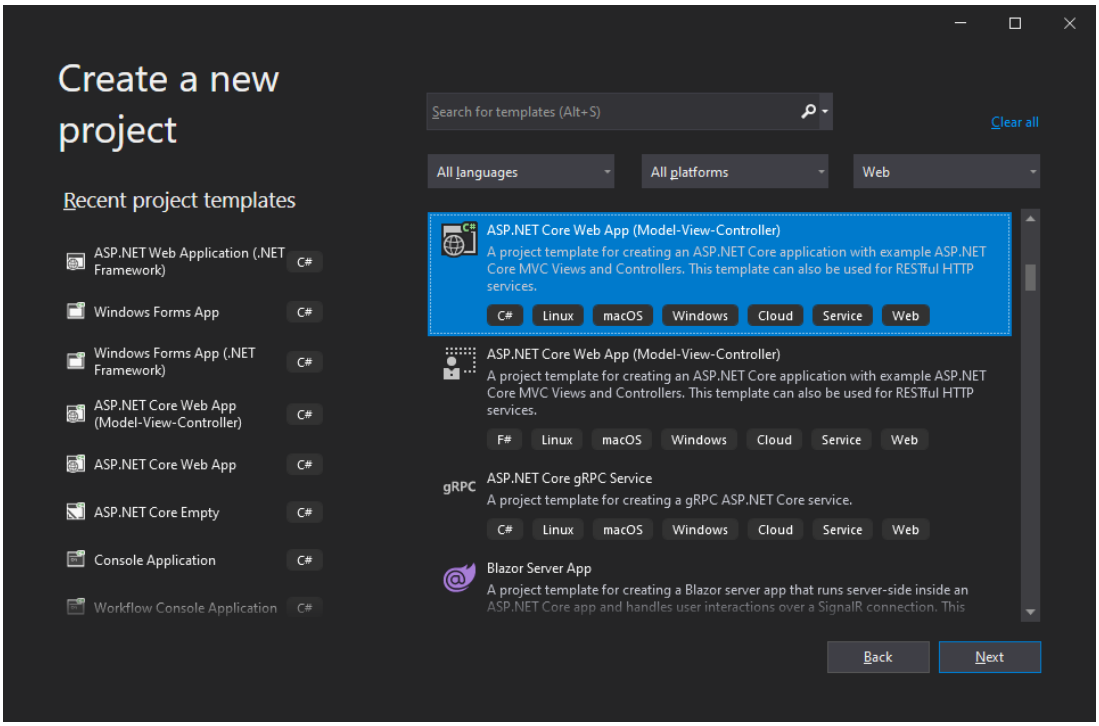


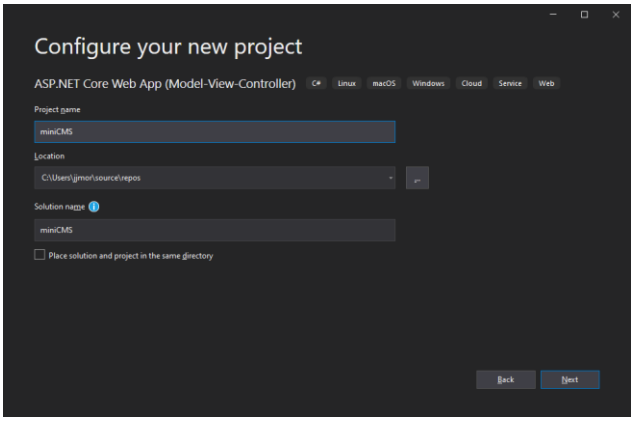


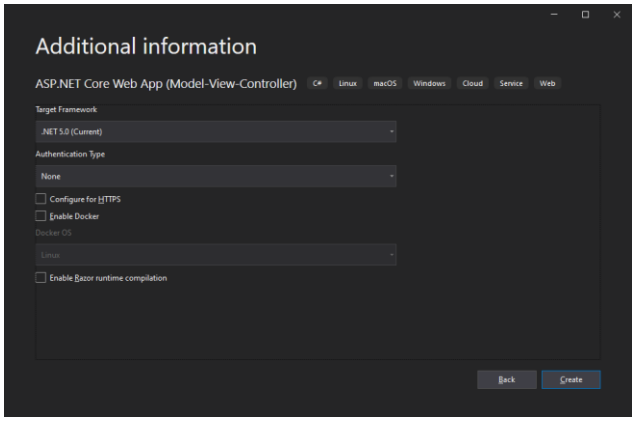
Criar aplicação ASP.NET Core

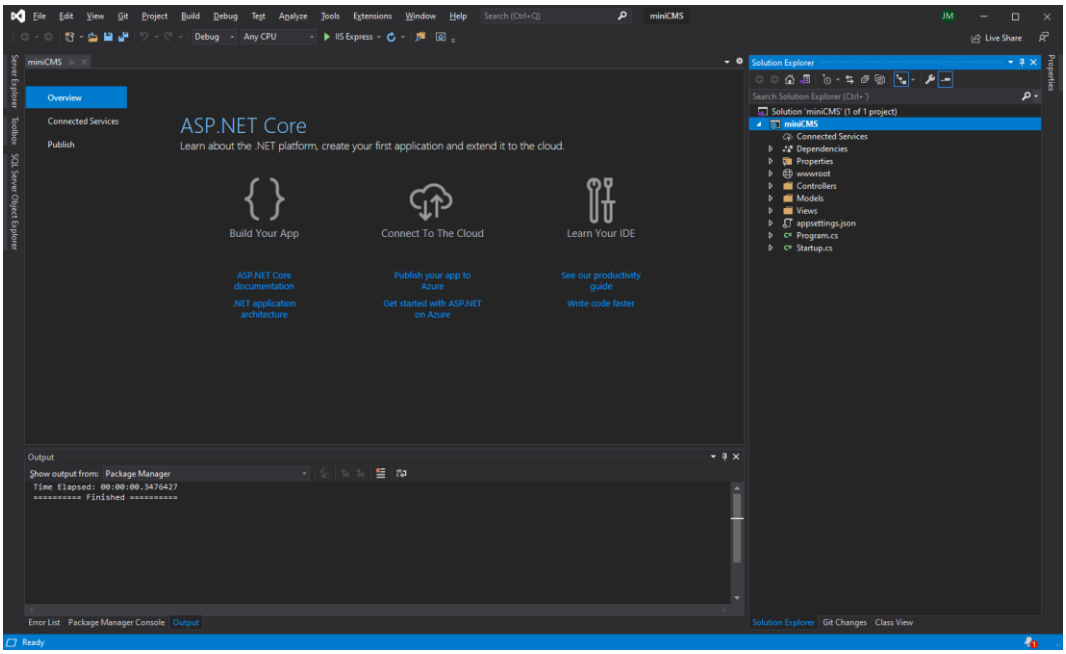
1. Podemos criar o projeto ASP.NET Core baseado no Template MVC (01-ASP.NET Core-Conceitos fundamentais) ou Empty (02-ASP.NET Core-Projeto de raiz)
   Obs: podemos usar o projeto modelo disponível no Moodle (00_AspNetCore_MVC_base_bootstrap.zip).

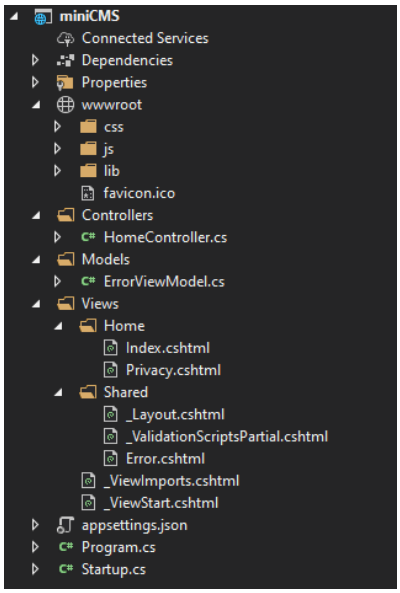
   Uma vez que nos exercícios anteriores criamos sempre um projeto de raiz (ASP.NET core Empty) para conhecer todos os componentes e passos de configuração de uma aplicação ASP.NET core MVC, para este exercício vamos criar o projeto a partir do modelo **ASP.NET Core Web App (Model-View-Controller)**

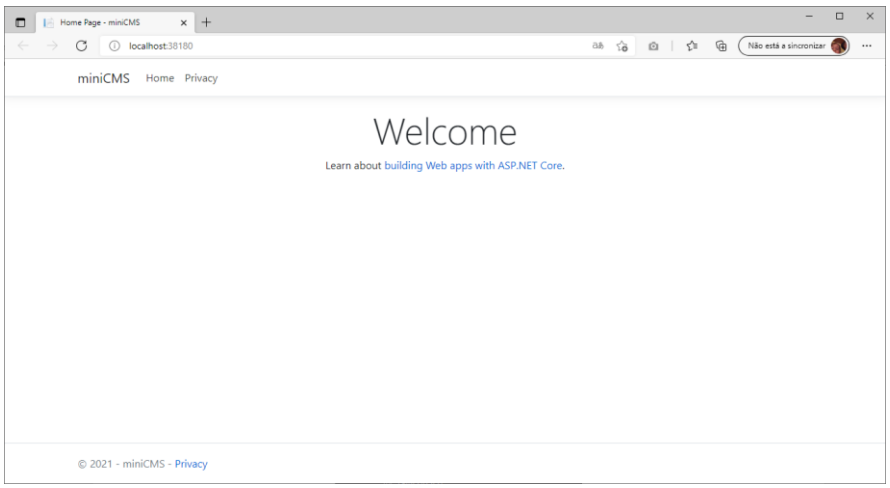












Template base

Code First

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).

1. Instalar o EF (Entity Framework)
2. Criar o modelo de dados
3. Criar e registar o contexto de base de dados (DbContext)
4. Classe de inicialização

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
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)

Conteudo	Tipo	DataAnnotations
Id	int	
Pagina	string	[StringLength(50)]
Titulo	string	[StringLength(50)]
Texto	string	
Autor	string	[StringLength(50)]
Data	DateTime	

Criar classe Conteudo.cs , apenas com as propriedades

Models\Conteudo.cs

```
public class Conteudo
    {
        // prop tab tab -> cria a prop
        // public int MyProperty { get; set; }

        public int Id { get; set; }

        [StringLength(50)]
        public string Pagina { get; set; }

        [StringLength(50)]
        public string Titulo { get; set; }

        public string Texto { get; set; }

        [StringLength(50)]
        public string Autor { get; set; }
        public DateTime Data { get; set; }    }
```

3. Criar e registar o contexto de base de dados (DbContext) na pasta Data

Data\DbMiniCMSContext.cs

```
  public class DbMiniCMSContext: DbContext
    {
        public DbMiniCMSContext(DbContextOptions<DbMiniCMSContext> options) : base(options)
        {

        }

        public DbSet<Conteudo> Conteudos { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Conteudo>().ToTable("Conteudo");
        }
    }
```

Registar o Contexto de base de dados

ConnectionString

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

Por questões de segurança, e também para mais fácil gestão, vamos colocar a Connection String (“miniCMSConnection”) num ficheiro externo de configurações:

appsettings.json

A base dados será criada com o nome de “miniCMS”

appsettings.json

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

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

}
```

Startup.cs

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

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.
public void ConfigureServices(IServiceCollection services)
{
  services.AddDbContext<DbMiniCMSContext>(options => options.UseSqlServer(Configuration.GetConnectionString("miniCMSConnection")));
 services.AddControllersWithViews();
}
```

4. Classe de inicialização

Neste exemplo, **em vez de usar migrações de dados** (para criar a BD e respetivas tabelas), vamos usar uma classe para inicializar a Base dados.

Classe DbInitializer.cs na pasta Data com o método (função) `Initialize` para inicializar a Base dados

Data\DbInitializer.cs

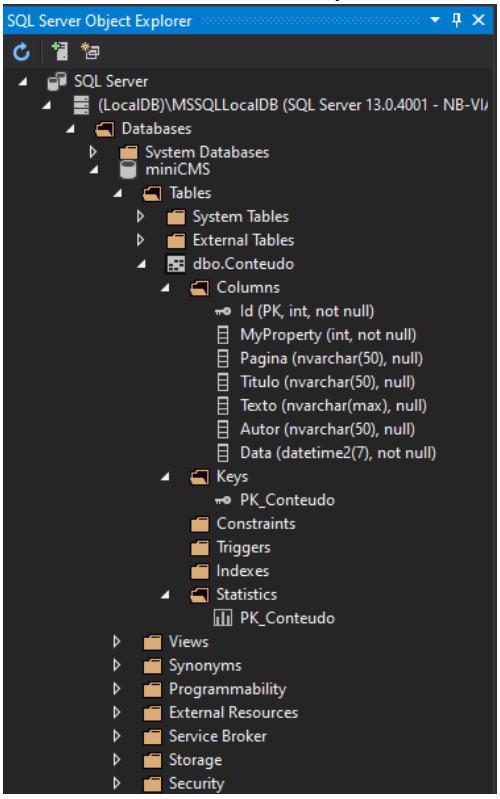
```
public class DbInitializer
    {

        public static void Initialize(DbMiniCMSContext context)
        {
            context.Database.EnsureCreated();

        }
    }
```

Invocar o método `Initialize` da classe DbInitializer no **Programa.cs**

Program.cs

```
public static void Main(string[] args)
        {
            //CreateHostBuilder(args).Build().Run();
            var host = CreateHostBuilder(args).Build();
            CreateDbIfNotExists(host);
            host.Run();
        }

private static void CreateDbIfNotExists(IHost host)
        {
            using (var scope = host.Services.CreateScope())
            {
                var services = scope.ServiceProvider;
                try
                {
                    var context = services.GetRequiredService<DbMiniCMSContext>();
                    DbInitializer.Initialize(context);
                }
                catch (Exception ex)
                {
                    var logger = services.GetRequiredService<ILogger<Program>>();
                    logger.LogError(ex, "An error occurred creating the DB.");
                }
            }
        }
```

Ao correr a aplicação será criada a BD com a respetiva tabela.



Vamos povoar a tabela Conteudo (ao criar a BD) com um registo, com a informação que deve ser apresentada na página Home

HTML:

```
<h1>Home</h1>

<p>Bem vindo ao <b>Sistema miniCMS</b></p>
<p>Todo o conteudo do site está numa <b>base de dados</b></p>
<img src='cms/media/minicms.jpg' />
```

Obs: criar a pasta `wwwroot\cms\media`

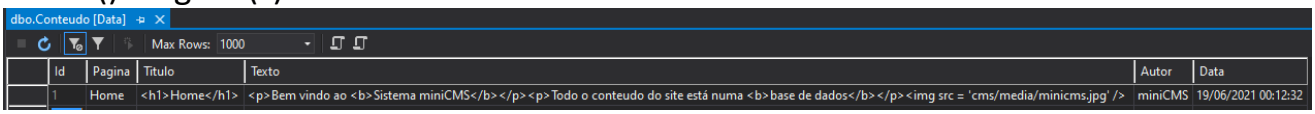
Colocar a imagem `minicms.jpg` na pasta `media`

Na **Classe** DbInitializer vamos adicionar ao método (função) `Initialize` o código para criar um registo na tabela Conteudo

Data\DbInitializer.cs

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

            // Look for any conteudo.
            if (context.Conteudos.Any())
            {
                return;   // DB has been seeded
            }

            Conteudo novoConteudo = new Conteudo
            {
                Pagina = "Home",
                Titulo = "<h1>Home</h1>",
                Texto = "<p>Bem vindo ao <b>Sistema miniCMS</b></p>" +
                "<p>Todo o conteudo do site está numa <b>base de dados</b></p>" +
                "<img src = '/cms/media/minicms.jpg' /> ",
                Autor = "miniCMS",
                Data = DateTime.Now
            };

            context.Conteudos.Add(novoConteudo);
            context.SaveChanges();

        }
    }
```

A primeira vez que a aplicação é executada a BD é criada com a respetiva tabela e com o()s registo(s).


dbo.Conteudo [Data]

Max Rows: 1000

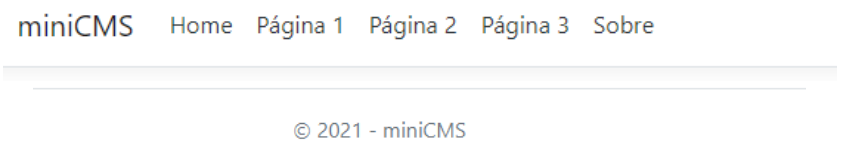
Id	Pagina	Titulo	Texto	Autor	Data
1	Home	Home	Bem vindo ao Sistema miniCMS Todo o conteudo do site está numa base de dados 	miniCMS	19/06/2021 00:12:32

Se mudar o modelo:

-Eliminar a base de dados

-E correr novamente a aplicação.

Algumas modificações ao layout:



Views\Shared_Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - miniCMS</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
            <div class="container">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">miniCMS</a>
                <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-controls=
                        "navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation"><span class="navbar-toggler-icon"></span>
                </button>
  <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
    <ul class="navbar-nav flex-grow-1">
     <li class="nav-item">
       <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
     </li>
     <li class="nav-item">
       <a class="nav-link text-dark" asp-area="" asp-controller="CMS" asp-action="Page" asp-route-id="pagina1">Página 1</a>
     </li>
     <li class="nav-item">
       <a class="nav-link text-dark" asp-area="" asp-controller="CMS" asp-action="Page" asp-route-id="pagina2">Página 2</a>
     </li>
     <li class="nav-item">
       <a class="nav-link text-dark" asp-area="" asp-controller="CMS" asp-action="Page" asp-route-id="pagina3">Página 3</a>
     </li>
     <li class="nav-item">
       <a class="nav-link text-dark" asp-area="" asp-controller="CMS" asp-action="Page" asp-route-id="sobre">Sobre</a>
     </li>
    </ul>
  </div>
 </div>
</nav>
</header>
     <div class="container">
         <main role="main" class="pb-3">
             @RenderBody()
         </main>
     </div>

     <footer class="border-top footer text-muted">
        <div class="container" style="text-align:center;">
            &copy; 2021 - miniCMS
        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

Eliminar:

Ação `Privacy` do controller `HomeController.cs`

A View: `Views\Home\Privacy.cshtml`

Rotas:

Página Home

`.../Home/index`

```
<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
```

Outras páginas

Para todas as outras páginas, vamos criar um **controller** (**CMS**) com a ação **Page** para receber a requisição de cada página

`.../CMS/Page/pagina1`
`.../CMS/Page/pagina2`
`.../CMS/Page/pagina3`
`.../CMS/Page/sobre`

```
<a class="nav-link text-dark" asp-area="" asp-controller="CMS" asp-action="Page" asp-route-id="pagina1">Página 1</a>
```

Tabela Conteudo

```
Titulo = "<h1>Página 1</h1>",
Texto = "<p>Conteudo da página 1</p>"
Autor = "miniCMS",
Data = DateTime.Now
```

Backoffice (Admin)

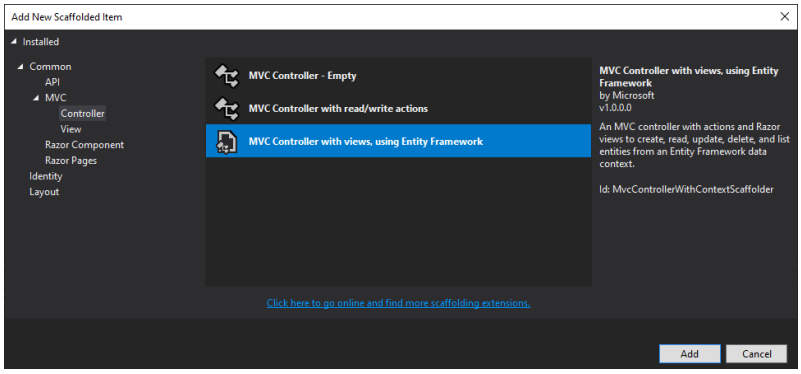
Para o backoffice vamos usar o Scaffolding que é uma framework de geração de código que adiciona códigos automaticamente e cria páginas de visualização (Views) e controladores (Controllers).

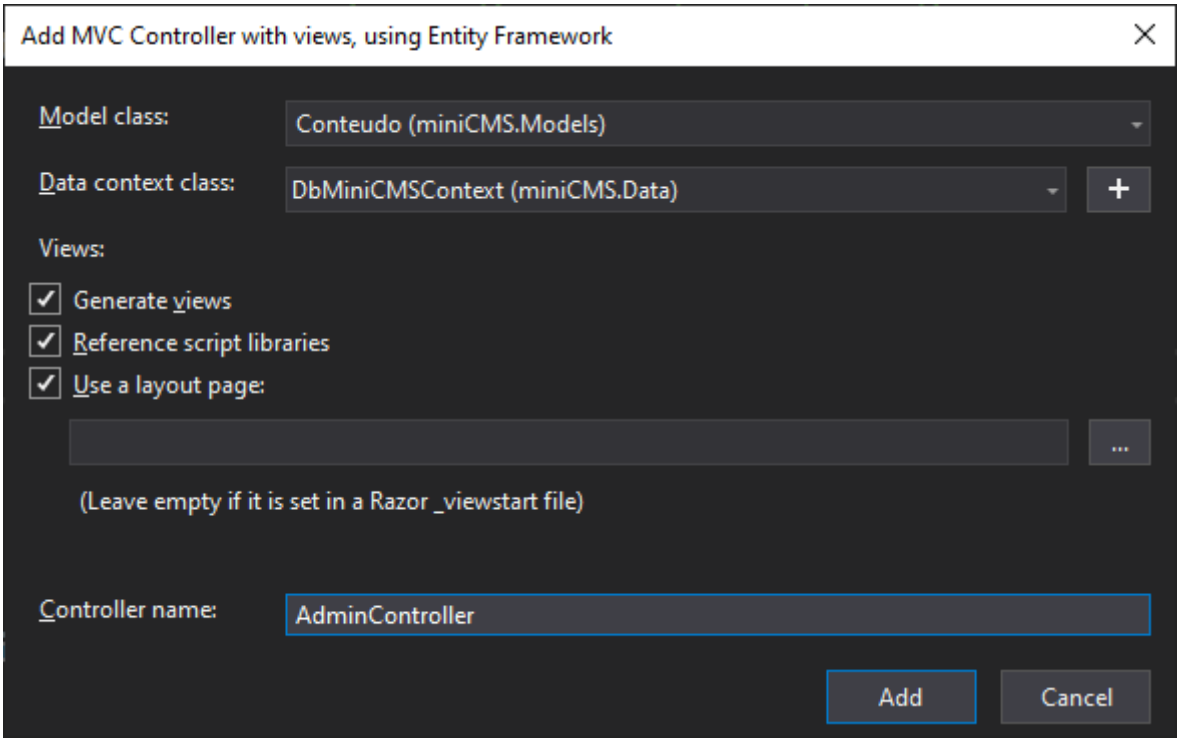
O Scaffolding torna o trabalho do programador mais fácil criando controladores (Controllers) e páginas de visualização (Views) para o modelo de dados (Models).

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

Na pasta Controller, tecla lado direito do rato e selecionar Add | Controller

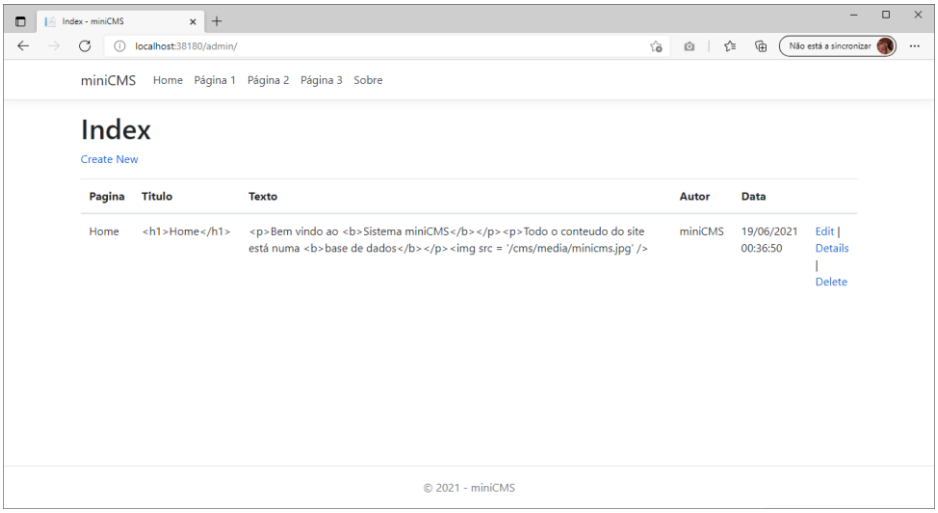
-MVC Controller with Views, using Entity Framework





Código gerado automaticamente pode ser visualizado no **anexo** neste documento.

Para entrar no backoffice: .../Admin/

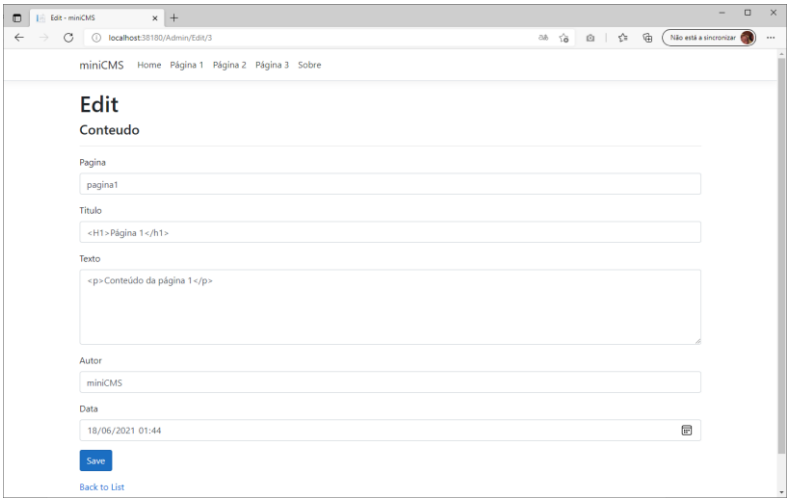


Mais tarde, colocamos um sistema de login, para que apenas quem tenha autorização possa aceder.

Admin: ajustes

Views | Create.cshtml | Edit.cshtml

```
<div class="col-md-12">

<div class="form-group">
        <label asp-for="Texto" class="control-label"></label>
        @*<input asp-for="Texto" class="form-control" />*@
        <textarea asp-for="Texto" class="form-control" rows="5"></textarea>
        <span asp-validation-for="Texto" class="text-danger"></span>
</div>
```



Mais tarde, colocamos um editor richtext para edição de HTML (CKEditor, Tiny,etc...)

Programar a ação `Index` do controller `Home`

Quando a ação index é invocada deve efetuar as seguintes tarefas:
-Recuperar da BD miniCMS da tabela Conteudo, o registo cujo campo Pagina= "Home"

(Código construído a partir do código AdminController\Details)

Controllers\HomeController.cs

```
public IActionResult Index()
 {
    // recuperar da BD miniCMS da tabela Conteudo, o registo cujo campo Pagina = "Home"
    var paginaHome =  _context.Conteudos.FirstOrDefault(m => m.Pagina == "Home");

    if (paginaHome == null)
       {
          return NotFound();
       }

       return View(paginaHome);
 }
```

Temos que invocar o contexto de base dados, pois é este que faz toda a interação com a Base de Dados.

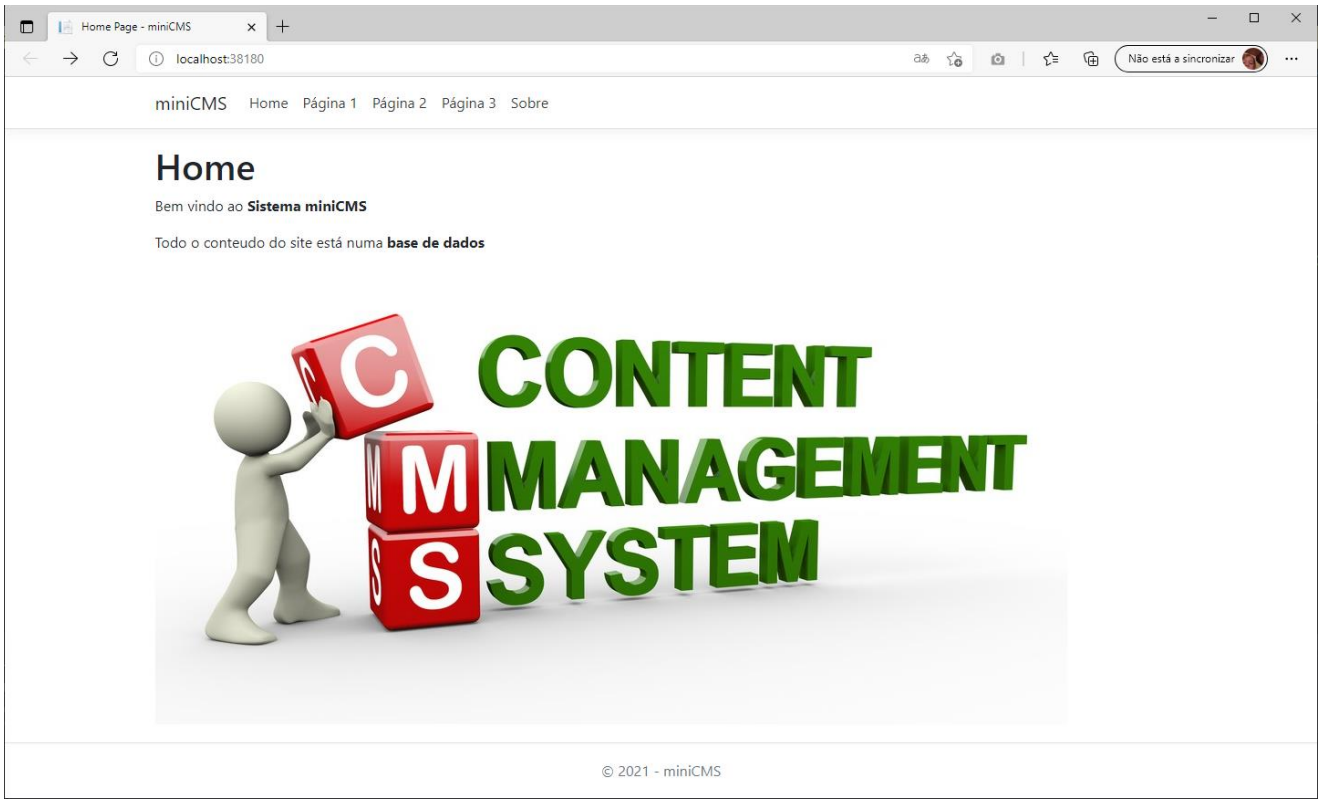
Controllers\HomeController.cs

```
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        private readonly DbMiniCMSContext _context;


        public HomeController(ILogger<HomeController> logger, DbMiniCMSContext context)
        {
            _logger = logger;
            _context = context;
        }

        //private readonly DbMiniCMSContext _context;
        //public HomeController(DbMiniCMSContext context)
        //{
        //    _context = context;
        //}

        public IActionResult Index()
        {
            // recuperar da BD miniCMS da tabela Conteudo, o registo cujo campo Pagina = "Home"
            var paginaHome =  _context.Conteudos.FirstOrDefault(m => m.Pagina == "Home");

            if (paginaHome == null)
            {
                return NotFound();
            }

            return View(paginaHome);
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
        public IActionResult Error()
        {
           return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
        }
    }
```

Criar a View Home\index.cshtml

Views\Home\index.cshtml

```
@model miniCMS.Models.Conteudo
@{
    ViewData["Title"] = "Home Page";
}

@Html.Raw(@Model.Titulo)

@Html.Raw(@Model.Texto)
```



Programar a ação Page do controller CMS

Criar o Controller (Empty) `CMS` e criar ação `Page` preparada para receber um `id`

```
.../CMS/Page/pagina1
.../CMS/Page/{id}
```

```
public IActionResult Page(string id)
 {
     return View();
 }
```

Quando a ação `Page` é invocada deve efetuar as seguintes tarefas:
-Recuperar da BD miniCMS da tabela Conteudo, o registo cujo campo Pagina= "Home"

(Codigo construído a partir do código AdminController\Details ou HomeController\Index)

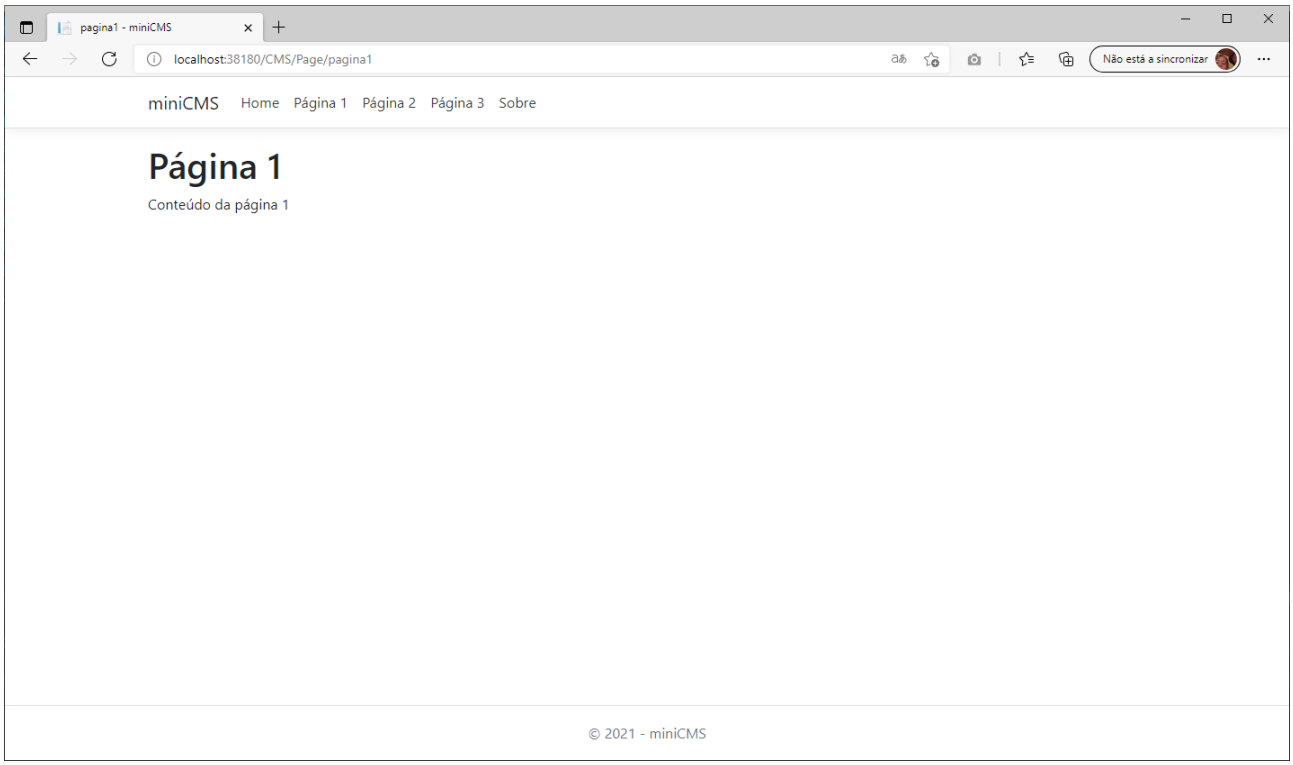
Controllers\CMSController.cs

```
using miniCMS.Data;

namespace miniCMS.Controllers
{
    public class CMSController : Controller
    {
        private readonly DbMiniCMSContext _context;

        public CMSController(DbMiniCMSContext context)
        {
            _context = context;
        }

        public IActionResult Index()
        {
            return View();
        }
        public IActionResult Page(string id)
        {
            // recuperar da BD miniCMS da tabela Conteudo, o registo cujo campo Pagina = id
            var pagina = _context.Conteudos.FirstOrDefault(m => m.Pagina == id);

            if (pagina == null)
            {
                return NotFound();
            }

            return View(pagina);
        }

    }
}
```

Criar a View CMS\Page.cshtml

Views\CMS\Page.cshtml

```
@model miniCMS.Models.Conteudo
@{
    ViewData["Title"] = Model.Pagina;
}


@Html.Raw(@Model.Titulo)

@Html.Raw(@Model.Texto)
```



Implementar o sistema de login para controlar o acesso ao backoffice

Alterar o nome da View Admin\Index.cshtml para Admin.cshtml

Alterar o nome da Acão Index do controller Admin para Admin

```
public async Task<IActionResult> Admin()
 {
     return View(await _context.Conteudos.ToListAsync());
 }
```

Criar a Acão Index do controller Admin

```
public IActionResult Index()
 {
     return View();
 }
```

Criar a View Index

Views\Admin\Index.cshtml

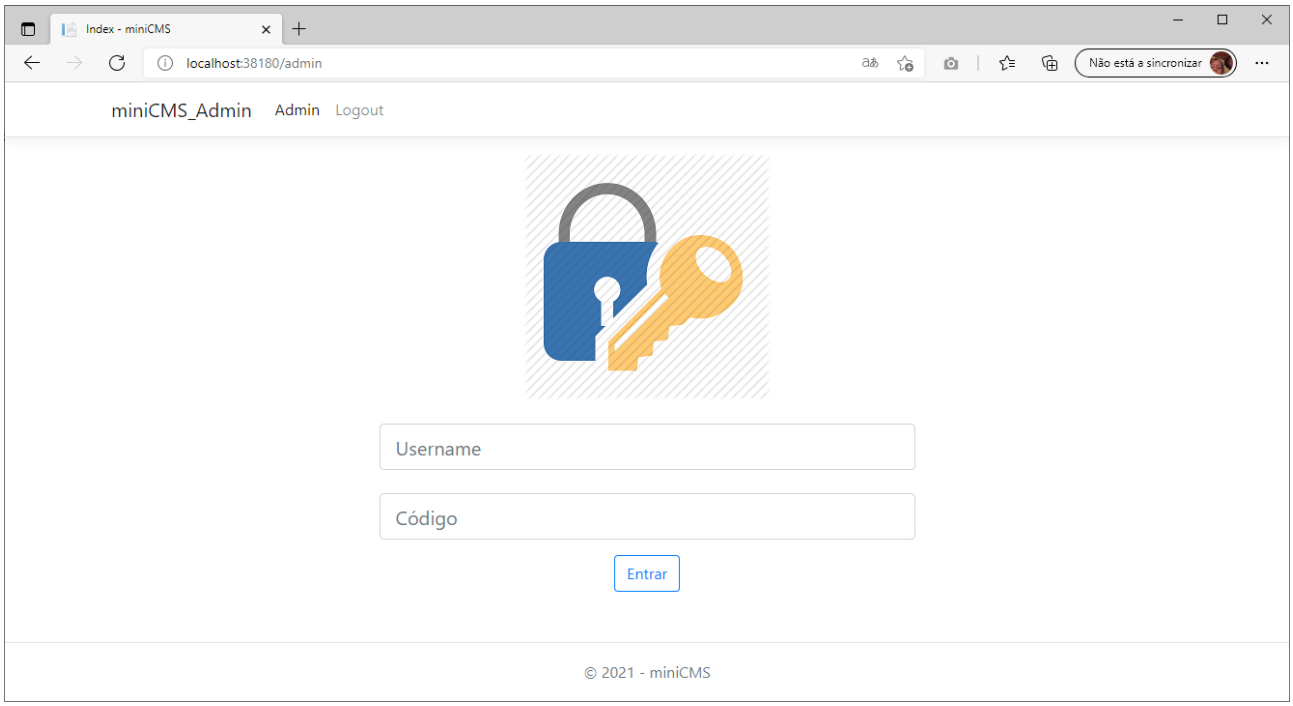
```
@{
    ViewData["Title"] = "Index";
    Layout = "_LayoutAdmin";
}


<form asp-action="Index" method="post">

    <div style="width:50%; margin:auto; text-align:center;">

        <div class="mb-4">
            <img src="~/cms/media/login.png" />
        </div>

        <div class="mb-4">
           <input type="text" name="UserName" class="form-control form-control-lg" placeholder="Username" required>
        </div>

        <div class="mb-3">
           <input type="password" name="Password" class="form-control form-control-lg" placeholder="Código" required>
        </div>

        <div class="mb-3">
            <button class="btn btn-outline-primary" type="submit">Entrar</button>
        </div>
    </div>

</form>
```

O sistema de backoffice terá um layout diferente do frontoffice, pelo menos a nível de menus

-Na pasta `Views\Shared\` duplicar o ficheiro `_Layout.cshtml` e alterar o nome para `_LayoutAdmin.cshtml`

Views\Shared_LayoutAdmin.cshtml

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - miniCMS</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-
shadow mb-3">
            <div class="container">
              <a class="navbar-brand" asp-area="" asp-controller="Admin" asp-action="Index">miniCMS_Admin</a>
                <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-
collapse" aria-controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
                    <ul class="navbar-nav">
                        <li class="nav-item">
                            <a class="nav-link active" aria-current="page" asp-area="" asp-
controller="Admin" asp-action="Index">Admin</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" asp-area="" asp-controller="Admin" asp-
action="Logout">Logout</a>
                        </li>
                    </ul>
                </div>
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted">
        <div class="container" style="text-align:center;">
            &copy; 2021 - miniCMS
        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```



No controller Admin, vamos colocar a receção do formulário da View Admin\Index e de acordo com username e pasword entra na página de Admin ou fica no Login (Index)
Vamos considerar a conta: username: `admin` ; password: `admin`
Podemos criar uma tabela de Utilizadores

Controllers\AdminController.cs

```
public IActionResult Index()
{
    return View();
}

[HttpPost]
public IActionResult Index(IFormCollection formCollection)
{
    string user = formCollection["UserName"];
    string pass = formCollection["UserName"];


    if (user == "admin" && pass == "admin")
    {
        return RedirectToAction("Admin");
    }
    else
    {
        return RedirectToAction("Index");
    }

}
```

Já controla o acesso à página de Admin, sempre que aceder via Admin/index, mas se colocar na URL Admin/admin diretamente, entra.

.../Admin/Admin
.../Admin/Create

Para impedir temos que guardar (variável) se o utilizador fez login com sucesso ou não, e, em cada página que queremos controlar o acesso, temos que verificar o estado dessa variável.
Vamos usar um sistema de armazenamento de variáveis de sessões no servidor.

Temos que ativar as sessões em Startup.cs

Startup.cs

```
public void ConfigureServices(IServiceCollection services)
        {
            services.AddSession();
            services.AddDbContext<DbMiniCMSContext>(options => options.UseSqlServer(Configuration.GetConnectionString("miniCMSConnection")));
            services.AddControllersWithViews();
        }

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            ...
            app.UseStaticFiles();

            app.UseRouting();

            app.UseAuthorization();

              app.UseSession();
            ...
     }
```

Assim, ao fazer login com sucesso vamos armazenar numa variável de sessão (“Login”).

Controllers\AdminController.cs

```
        public IActionResult Index()
        {
            return View();
        }

        [HttpPost]
        public IActionResult Index(IFormCollection formCollection)
        {
            string user = formCollection["UserName"];
            string pass = formCollection["UserName"];

            HttpContext.Session.SetString("Login", "false");

            if (user == "admin" && pass == "admin")
            {
                HttpContext.Session.SetString("Login", "true");
                return RedirectToAction("Admin");
            }
            else
            {
                return RedirectToAction("Index");
            }
        }
```

Agora, na acção Admin vamos controlar se fez login, verificando a variável de sessão "Login"

Controllers\AdminController.cs

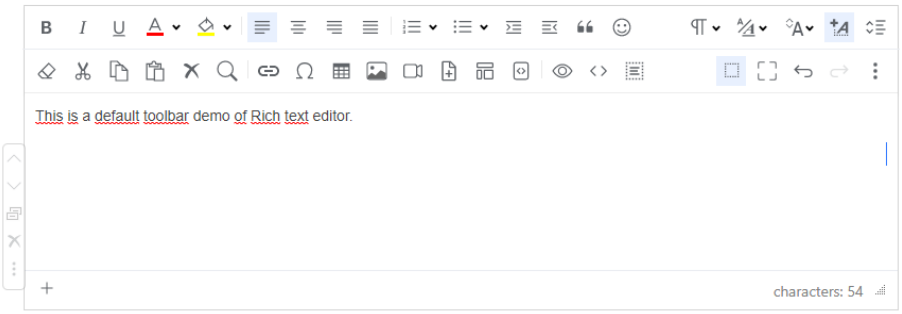
```
public async Task<IActionResult> Admin()
 {

     var login = HttpContext.Session.GetString("Login");
     if (login == null)
     {
         return RedirectToAction("Index");
     }

     if (bool.Parse(login) == false)
     {
         return RedirectToAction("Index");
     }

     return View(await _context.Conteudos.ToListAsync());
 }
```

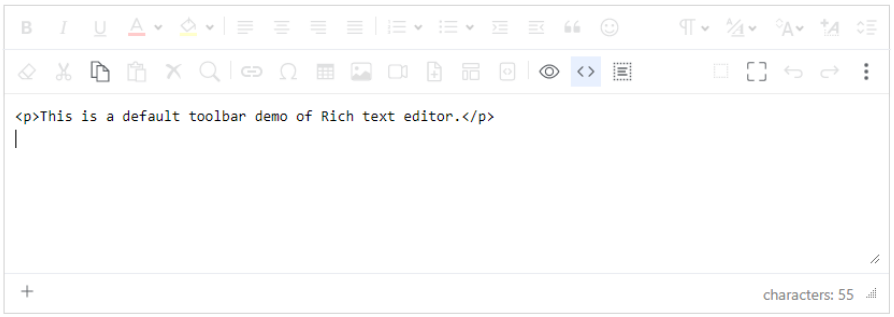
Podemos colocar este código em todas as páginas que queremos controlar a entrada, sem fazer Login.

Adicionar ao CMS um editor HTML para auxiliar na escrita de artigos.

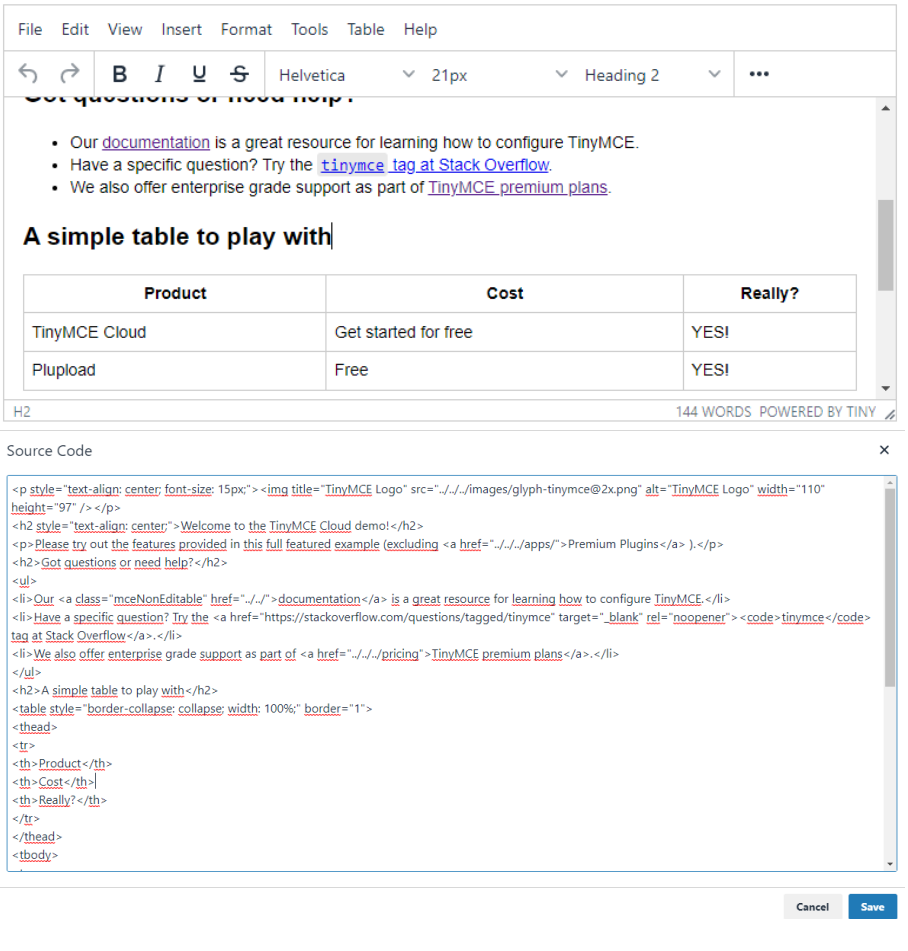
Exemplos:

https://richtexteditor.com/



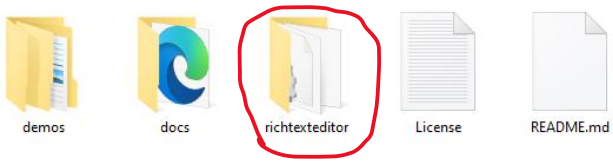


https://www.tiny.cloud/

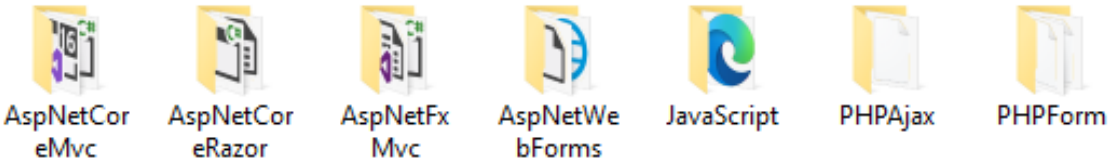


Usar o Rich Text Editor

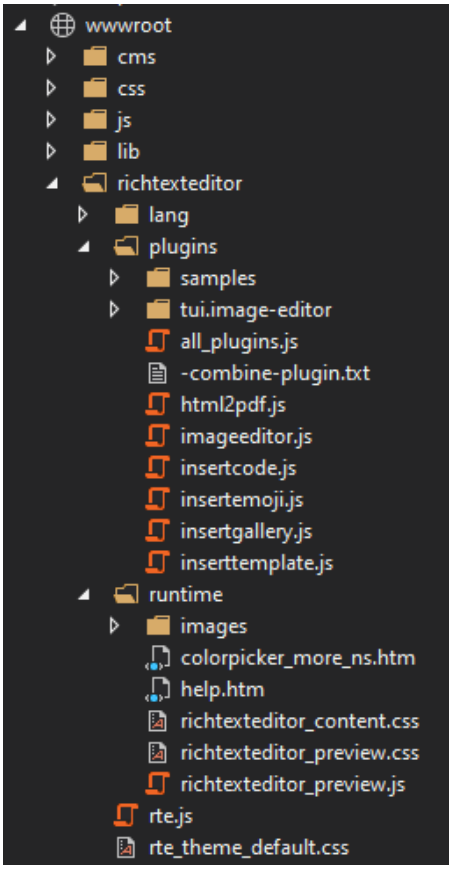
1. Fazer download a partir de https://richtexteditor.com/



A pasta `Demos` tem exemplos de uso para as várias linguagens/tecnologias



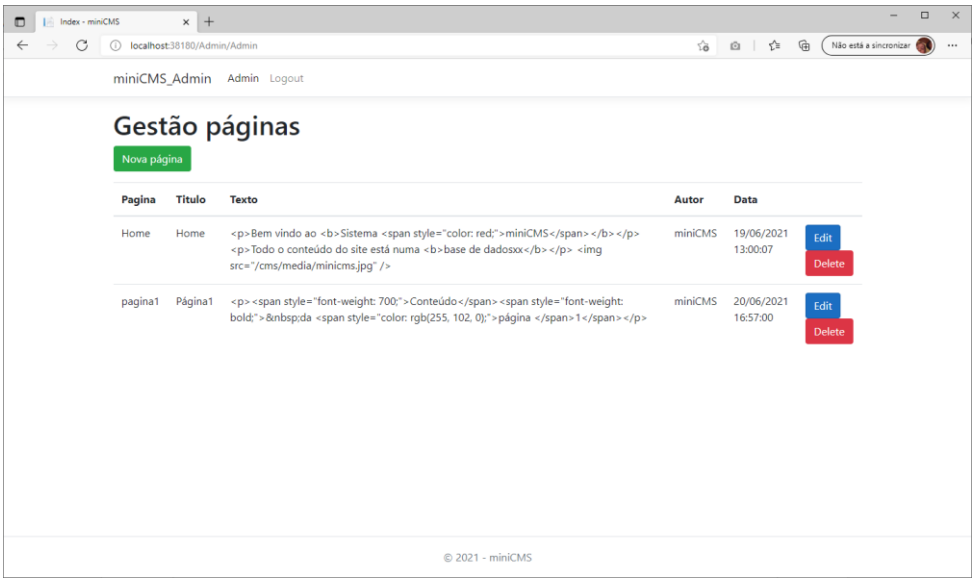
2. Copiar a pasta `richtexteditor` para a pasta `wwwroot` da aplicação

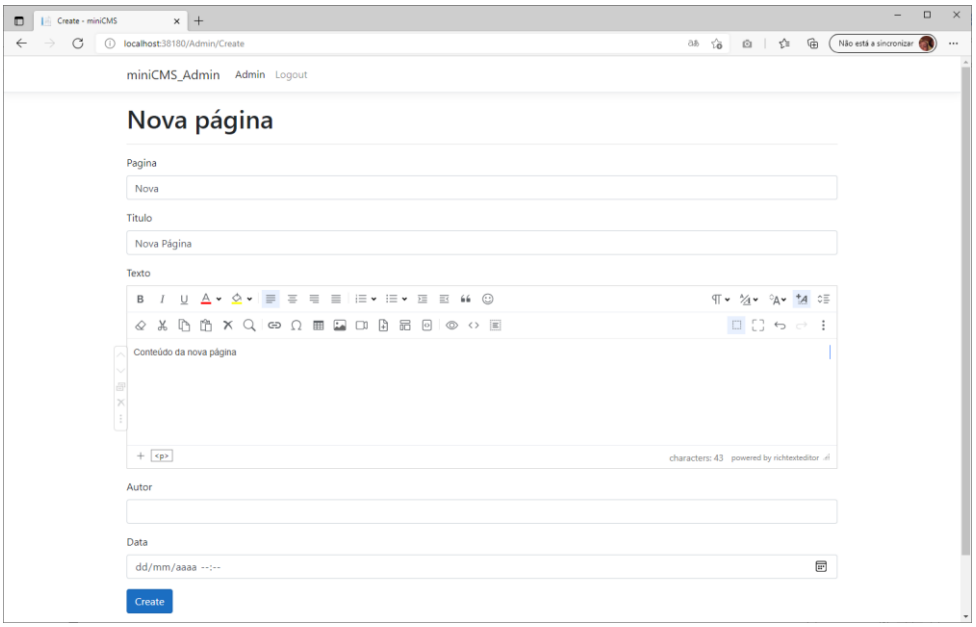


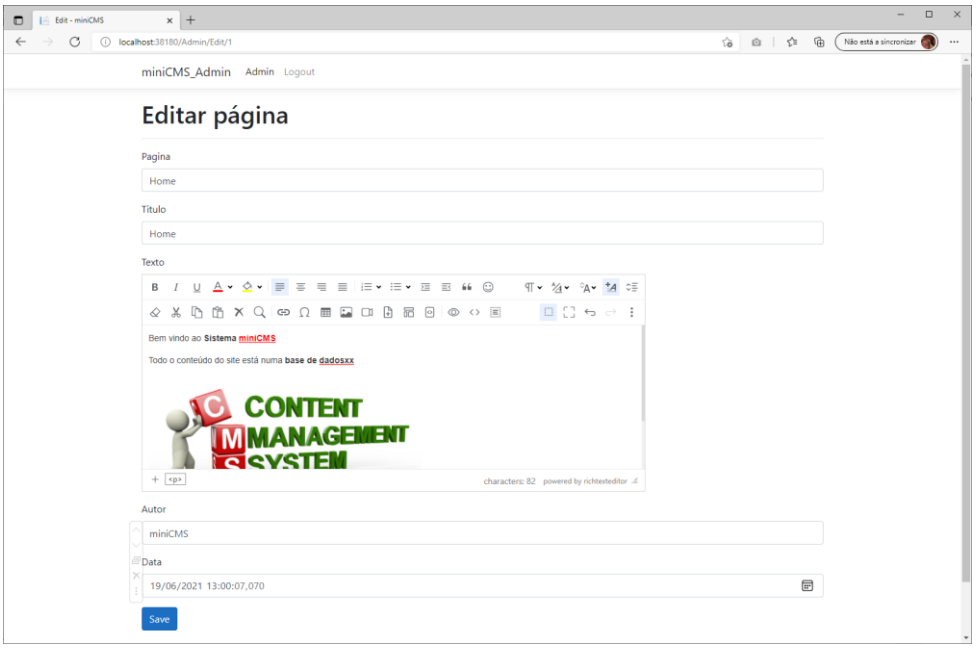
3. Na View onde pretendemos usar o editor colocamos o seguinte código
   -Referenciar richtexteditor (js e css) e o script o código js para criar o editor

Views\Admin\Create.cshtml

```
@model miniCMS.Models.Conteudo

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

<link href="~/richtexteditor/rte_theme_default.css" rel="stylesheet" />
<script src="~/richtexteditor/rte.js"></script>
<script src="~/richtexteditor/plugins/all_plugins.js"></script>
<script src="~/richtexteditor/rte.js"></script>

<h1>Nova página</h1>
<hr />
<div class="row">
    <div class="col-md-12">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Pagina" class="control-label"></label>
                <input asp-for="Pagina" class="form-control" />
                <span asp-validation-for="Pagina" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Titulo" class="control-label"></label>
                <input asp-for="Titulo" class="form-control" />
                <span asp-validation-for="Titulo" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Texto" class="control-label"></label>

                <textarea id="div_editor" asp-for="Texto" class="form-control" rows="5"></textarea>

                <span asp-validation-for="Texto" class="text-danger"></span>
            </div>

            <script>
                var editor1 = new RichTextEditor(document.getElementById("div_editor"));

            </script>

            <div class="form-group">
                <label asp-for="Autor" class="control-label"></label>
                <input asp-for="Autor" class="form-control" />
                <span asp-validation-for="Autor" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Data" class="control-label"></label>
                <input asp-for="Data" class="form-control" />
                <span asp-validation-for="Data" 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="Admin">Back to List</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```


Index - miniCMS
localhost:38180/Admin/Admin
miniCMS_Admin
Admin
Logout
Gestão páginas
Nova página
Pagina
Titulo
Texto
Autor
Data
Home
Home
Bem vindo ao Sistema miniCMS Todo o conteúdo do site está numa base de dadosxx
miniCMS
19/06/2021 13:00:07
Edit
Delete
pagina1
Página1
Conteúdo da página 1
miniCMS
20/06/2021 16:57:00
Edit
Delete
© 2021 - miniCMS


Create - miniCMS
localhost:38180/Admin/Create
miniCMS_Admin
Admin
Logout
Nova página
Pagina
Nova
Titulo
Nova Página
Texto
Conteúdo da nova página
characters: 43
powered by richtexteditor
Autor
Data
dd/mm/aaaa --:--
Create


Edit - miniCMS
localhost:38180/Admin/Edit/1
miniCMS_Admin
Admin
Logout
Editar página
Pagina
Home
Titulo
Home
Texto
Bem vindo ao Sistema miniCMS
Todo o conteúdo do site está numa base de dadosxx
CONTENT
MANAGEMENT
SYSTEM
characters: 82
powered by richtexteditor
Autor
miniCMS
Data
19/06/2021 13:00:07,070
Save

Anexo: Código gerado automaticamente Backoffice (Admin)

Código gerado automaticamente

Controllers\AdminController.cs

```
namespace miniCMS.Controllers
{
    public class AdminController : Controller
    {
        private readonly DbMiniCMSContext _context;

        public AdminController(DbMiniCMSContext context)
        {
            _context = context;
        }

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

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

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

            return View(conteudo);
        }

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

        // POST: Admin/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,Pagina,Titulo,Texto,Autor,Data")] Conteudo conteudo)
        {
            if (ModelState.IsValid)
            {
                _context.Add(conteudo);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(conteudo);
        }
```

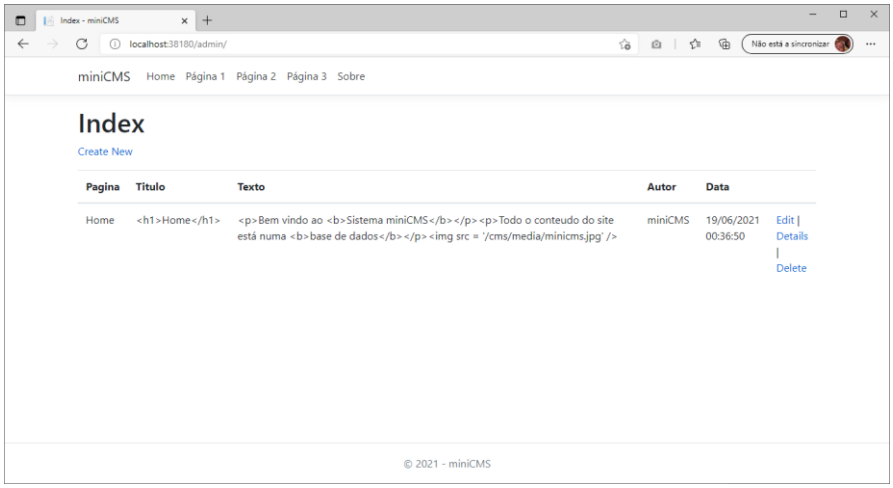
```
        // GET: Admin/Edit/5
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

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

        // POST: Admin/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,Pagina,Titulo,Texto,Autor,Data")] Conteudo conteudo)
        {
            if (id != conteudo.Id)
            {
                return NotFound();
            }

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

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

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

            return View(conteudo);
        }

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

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

Código gerado automaticamente (Views)

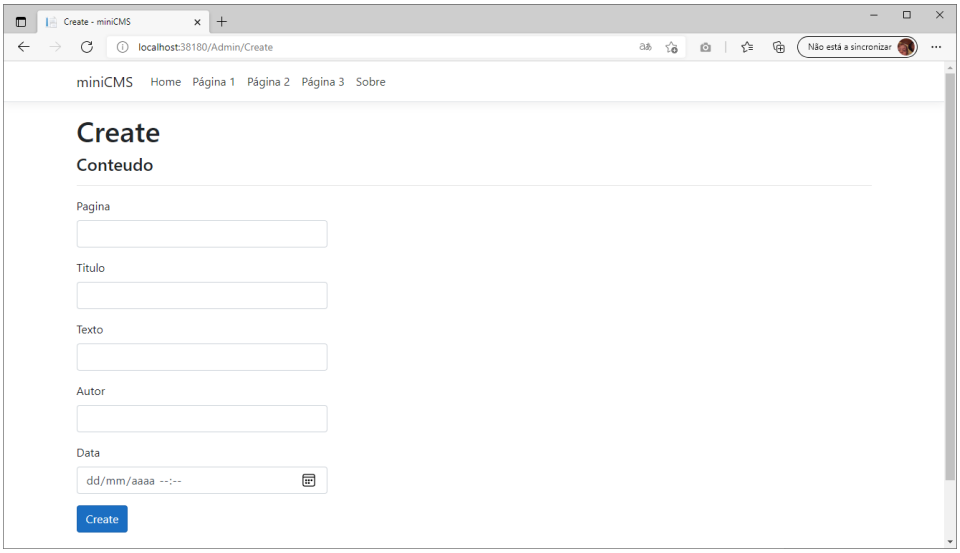
Views\Admin\Index.cshtml

```
@model IEnumerable<miniCMS.Models.Conteudo>
@{
    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.Pagina)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Titulo)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Texto)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Autor)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Data)
            </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
@foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Pagina)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Titulo)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Texto)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Autor)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Data)
            </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>
```



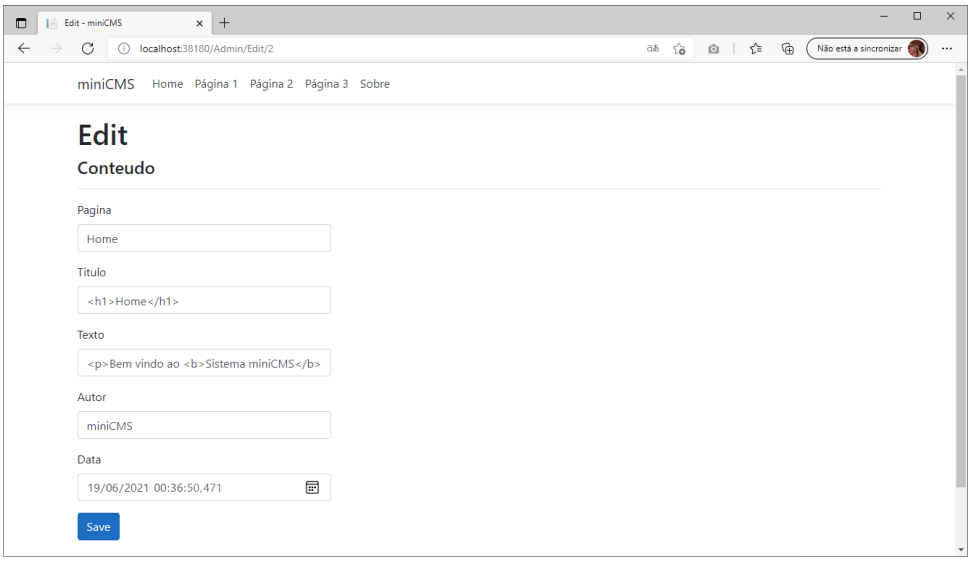
Views\Admin\Create.cshtml

```
@model miniCMS.Models.Conteudo

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

<h1>Create</h1>

<h4>Conteudo</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="Pagina" class="control-label"></label>
                <input asp-for="Pagina" class="form-control" />
                <span asp-validation-for="Pagina" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Titulo" class="control-label"></label>
                <input asp-for="Titulo" class="form-control" />
                <span asp-validation-for="Titulo" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Texto" class="control-label"></label>
                <input asp-for="Texto" class="form-control" />
                <span asp-validation-for="Texto" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Autor" class="control-label"></label>
                <input asp-for="Autor" class="form-control" />
                <span asp-validation-for="Autor" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Data" class="control-label"></label>
                <input asp-for="Data" class="form-control" />
                <span asp-validation-for="Data" 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>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```



Views\Admin\Edit.cshtml

```
@model miniCMS.Models.Conteudo

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

<h1>Edit</h1>

<h4>Conteudo</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="Pagina" class="control-label"></label>
                <input asp-for="Pagina" class="form-control" />
                <span asp-validation-for="Pagina" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Titulo" class="control-label"></label>
                <input asp-for="Titulo" class="form-control" />
                <span asp-validation-for="Titulo" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Texto" class="control-label"></label>
                <input asp-for="Texto" class="form-control" />
                <span asp-validation-for="Texto" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Autor" class="control-label"></label>
                <input asp-for="Autor" class="form-control" />
                <span asp-validation-for="Autor" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Data" class="control-label"></label>
                <input asp-for="Data" class="form-control" />
                <span asp-validation-for="Data" 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>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```



Views\Admin\delete.cshtml

```
@model miniCMS.Models.Conteudo

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

<h1>Delete</h1>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Conteudo</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Pagina)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Pagina)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Titulo)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Titulo)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Texto)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Texto)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Autor)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Autor)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Data)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Data)
        </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>
```