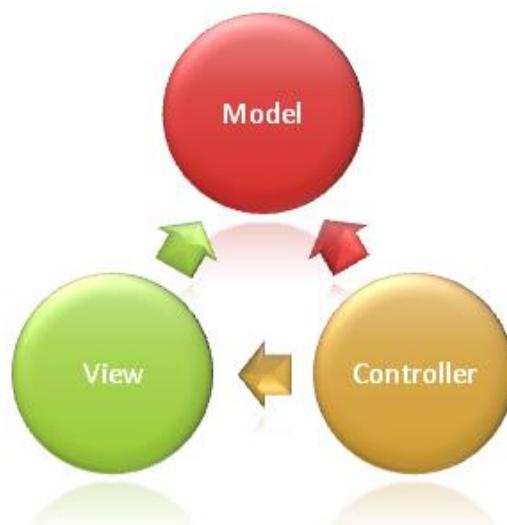


## O que é o padrão MVC?

O padrão de arquitetura **MVC (Model-View-Controller)** separa um aplicativo em três grupos de componentes principais: **Modelos**, **Exibições** e **Controladores**. Esse padrão ajuda a obter a separação de interesses. Usando esse padrão, as solicitações de usuário são encaminhadas para um **Controlador**, que é responsável por trabalhar com o **Modelo** para executar as ações do usuário e/ou recuperar os resultados de consultas. O **Controlador** escolhe a **Exibição** a ser exibida para o usuário e fornece-a com os dados do **Modelo** solicitados.

O seguinte diagrama mostra os três componentes principais e quais deles referenciam os outros:



Essa descrição das responsabilidades ajuda você a dimensionar o aplicativo em termos de complexidade, porque é mais fácil de codificar, depurar e testar algo (modelo, exibição ou controlador) que tem um único trabalho. É mais difícil atualizar, testar e depurar um código que tem dependências distribuídas em duas ou mais dessas três áreas. Por exemplo, a lógica da interface do usuário tende a ser alterada com mais frequência do que a lógica de negócios. Se o código de apresentação e a lógica de negócios forem combinados em um único objeto, um objeto que contém a lógica de negócios precisa ser modificado sempre que a interface do usuário é alterada. Isso costuma introduzir erros e exige um novo teste da lógica de negócios após cada alteração mínima da interface do usuário.

**Observação:** A exibição e o controlador dependem do modelo. No entanto, o modelo não depende da exibição nem do controlador. Esse é um dos principais benefícios da separação. Essa separação permite que o modelo seja criado e testado de forma independente da apresentação visual.

## Responsabilidades do Modelo

O Modelo em um aplicativo MVC representa o estado do aplicativo e qualquer lógica de negócios ou operação que deve ser executada por ele. A lógica de negócios deve ser encapsulada no modelo, juntamente com qualquer lógica de implementação, para persistir o estado do aplicativo. As exibições fortemente tipadas

normalmente usam tipos **ViewModel** criados para conter os dados a serem exibidos nessa exibição. O controlador cria e popula essas instâncias de **ViewModel** com base no modelo.

## Responsabilidades da Exibição

As exibições são responsáveis por apresentar o conteúdo por meio da interface do usuário. Eles usam o **Razor** mecanismo de exibição para inserir o código .net na marcação **HTML**. Deve haver uma lógica mínima nas exibições e qualquer lógica contida nelas deve se relacionar à apresentação do conteúdo. Se você precisar executar uma grande quantidade de lógica em arquivos de exibição para exibir dados de um modelo complexo, considere o uso de um Componente de Exibição, **ViewModel** ou um modelo de exibição para simplificar a exibição.

## Responsabilidades do Controlador

Os controladores são os componentes que cuidam da interação do usuário, trabalham com o modelo e, em última análise, selecionam uma exibição a ser renderizada. Em um aplicativo MVC, a exibição mostra apenas informações; o controlador manipula e responde à entrada e à interação do usuário. No padrão MVC, o controlador é o ponto de entrada inicial e é responsável por selecionar quais tipos de modelo serão usados para o trabalho e qual exibição será renderizada (daí seu nome – ele controla como o aplicativo responde a determinada solicitação).

**Observação:** Os controladores não devem ser excessivamente complicados por muitas responsabilidades. Para evitar que a lógica do controlador se torne excessivamente complexa, efetue *push* da lógica de negócios para fora do controlador e insira-a no modelo de domínio.

**Dica:** Se você achar que as ações do controlador executam com frequência os mesmos tipos de ações, move essas ações comuns para filtros.

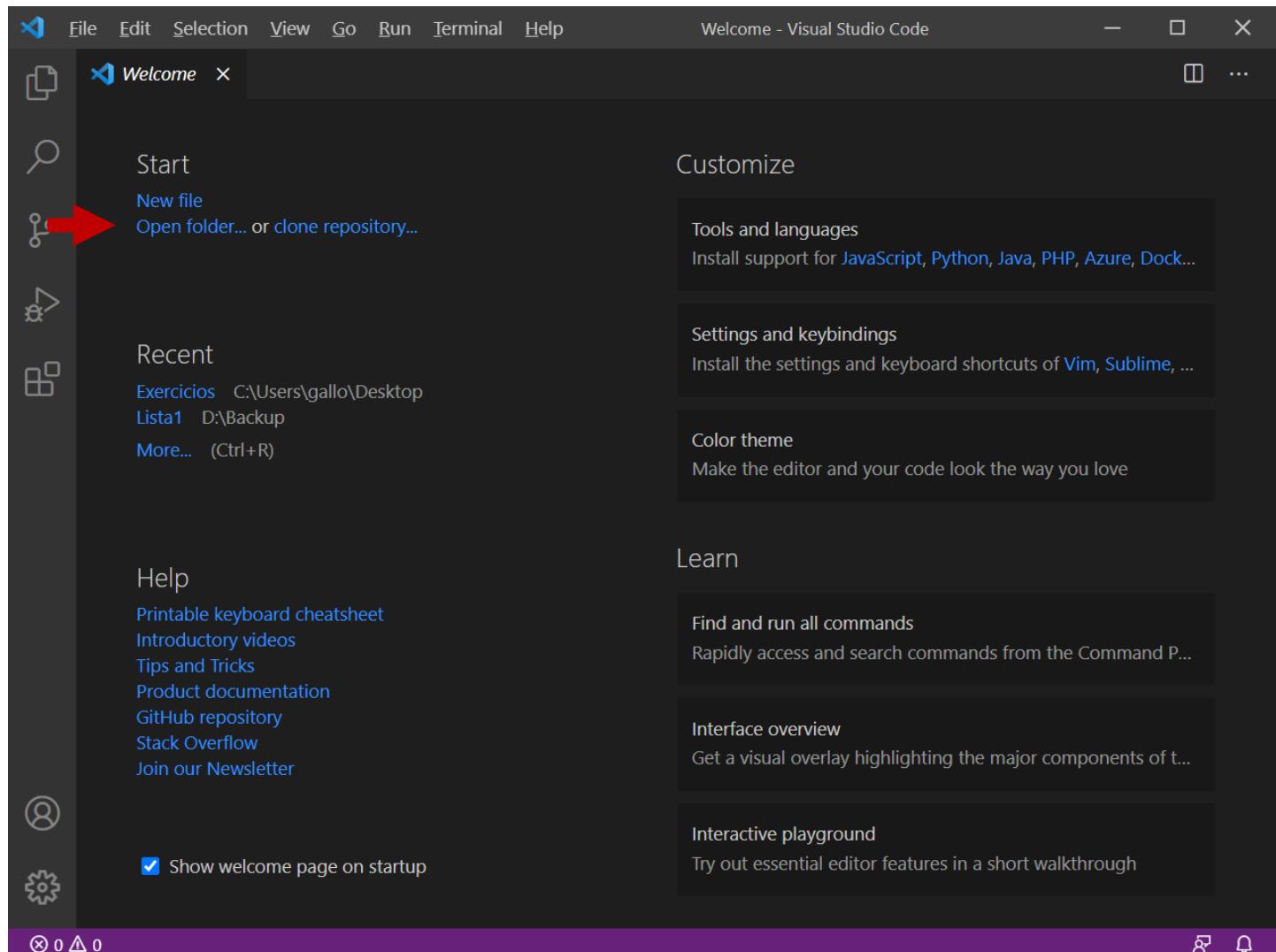
## O que é ASP.NET Core MVC

A estrutura do ASP.NET Core MVC é uma estrutura de apresentação leve, de software livre e altamente testável, otimizada para uso com o ASP.NET Core.

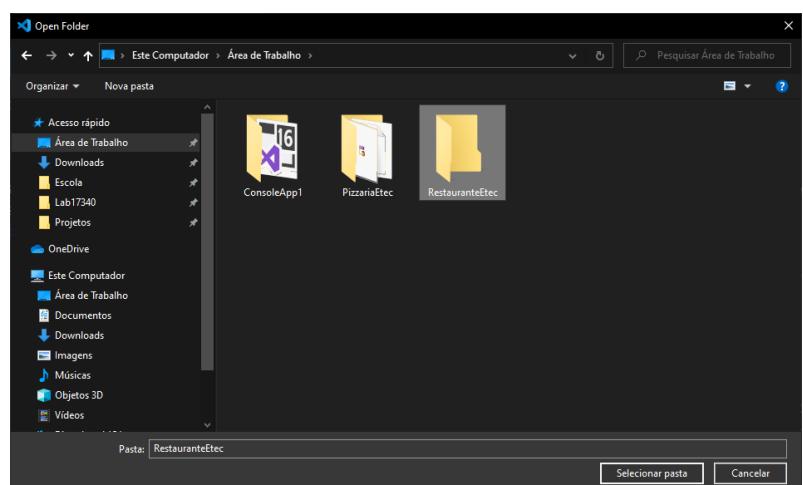
ASP.NET Core MVC fornece uma maneira com base em padrões para criar sites dinâmicos que habilitam uma separação limpa de preocupações. Ele lhe dá controle total sobre a marcação, dá suporte ao desenvolvimento amigável a TDD e usa os padrões da web mais recentes.

# RESTAURANTEETEC

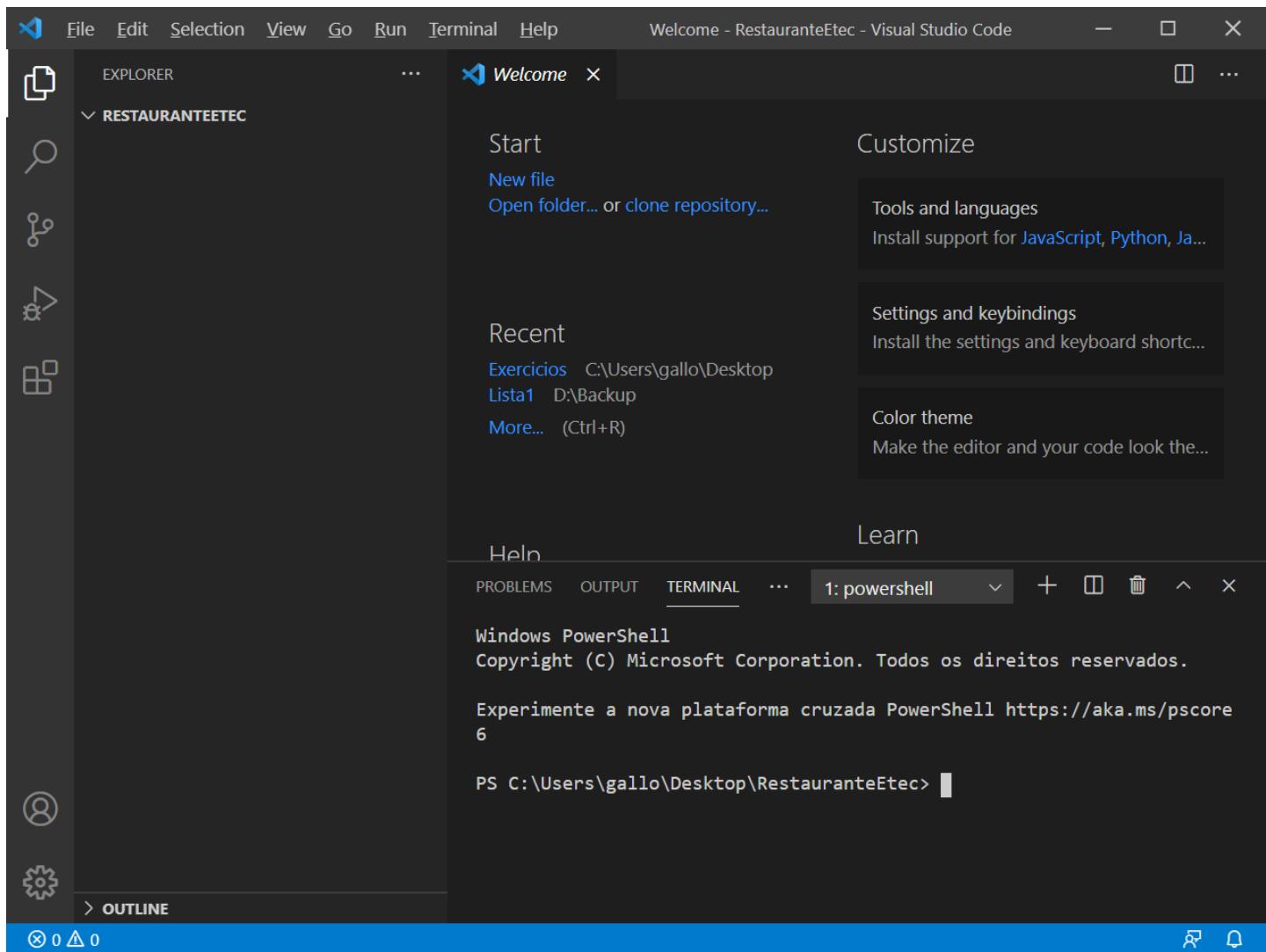
Iniciaremos o desenvolvimento do projeto **RestauranteEtec**, pela criação do projeto. Abra o **Visual Studio Code** e selecione no menu **File** a opção “**Open Folder**”, ou conforme a imagem abaixo, use a opção da tela de bem-vindo (**Welcome**).



Na janela navega do **Open Folder**, navegue até o local onde deseja salvar seu projeto, e crie uma pasta com o nome **RestauranteEtec**, conforme a imagem ao lado. Com esta pasta selecionada, clique no botão **[Selecionar pasta]**, que ela estará aberta e disponível para uso no **Visual Studio Code**.



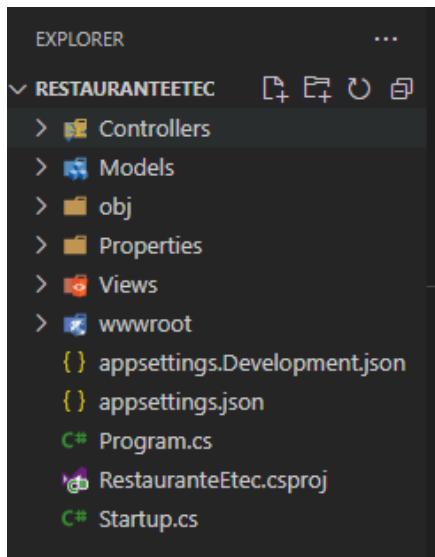
Agora no **Visual Studio Code**, vamos usar o terminal para criar nosso projeto. Para abrir o terminal clique: **[Ctrl] + ' (Control + Aspas)**.



Com o terminal aberto o que precisamos fazer agora é digitar o comando:

```
dotnet new mvc -lang "C#" -f net5.0
```

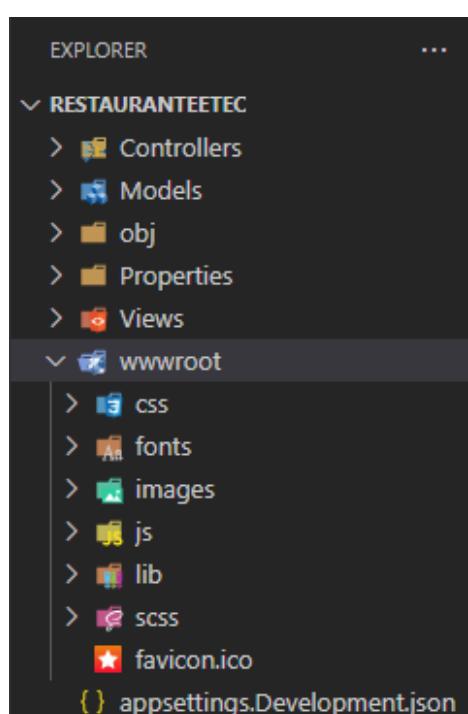
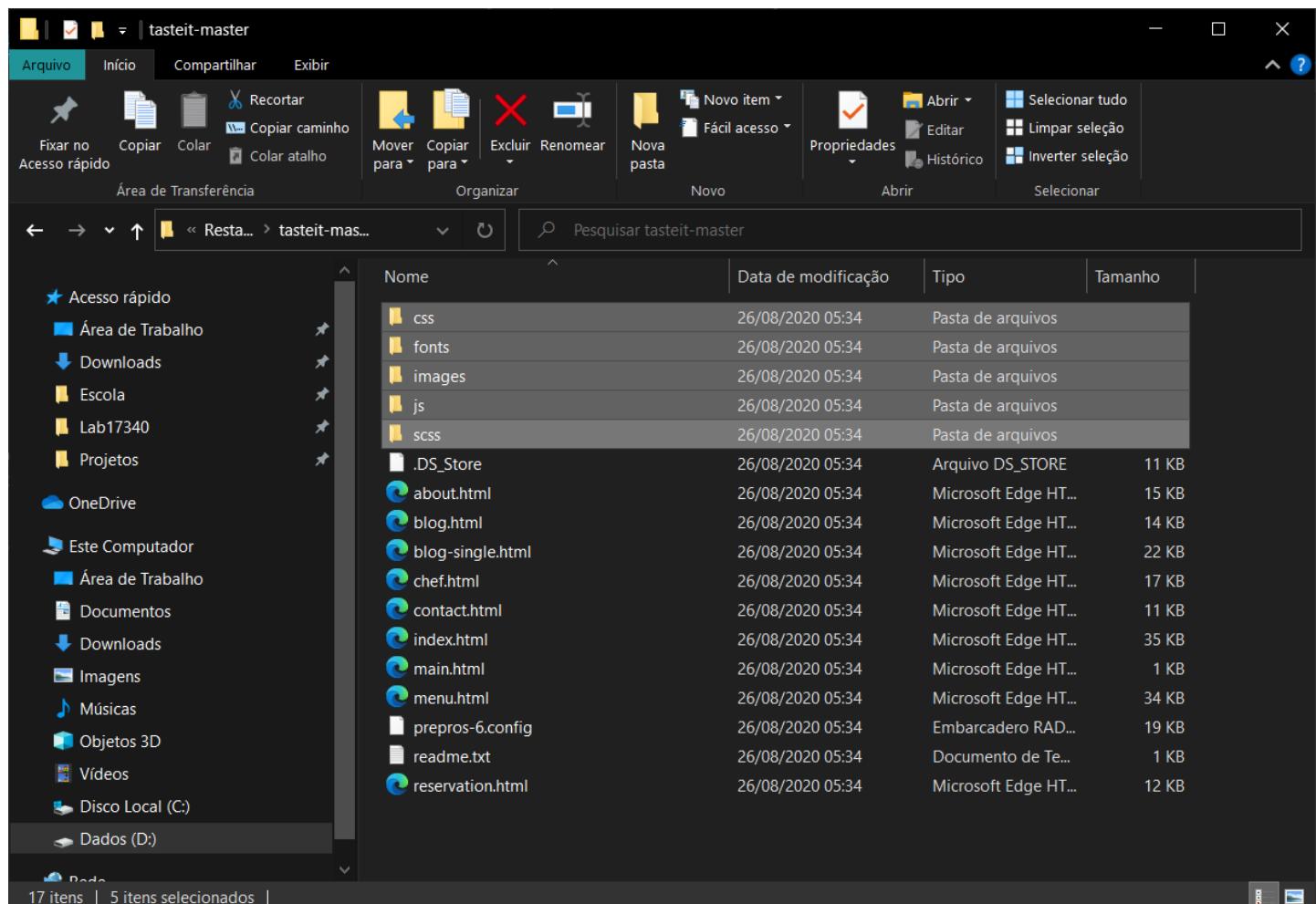
No caso aqui, especificamos que o projeto possui o padrão “**MVC**” a linguagem de programação é **C#** e o **framework** a ser usado o **.NET 5**. É importante lembrar desses passos, pois sua máquina pode possuir mais de uma versão do **SDK** instalado.



E com isso o projeto está pronto para começar o desenvolvimento, conforme pode ser observado na imagem ao lado que mostra a estrutura de pastas do projeto.

## Criando o Layout das Páginas

Agora vamos copiar os arquivos e códigos de nossa página (**template**) **HTML** para dentro de nosso projeto. Descompacte o arquivo [tasteit-master.zip](#) e em seguida copie todas as suas pastas, e cole na **wwwroot** do nosso projeto, conforme mostra a figura abaixo:



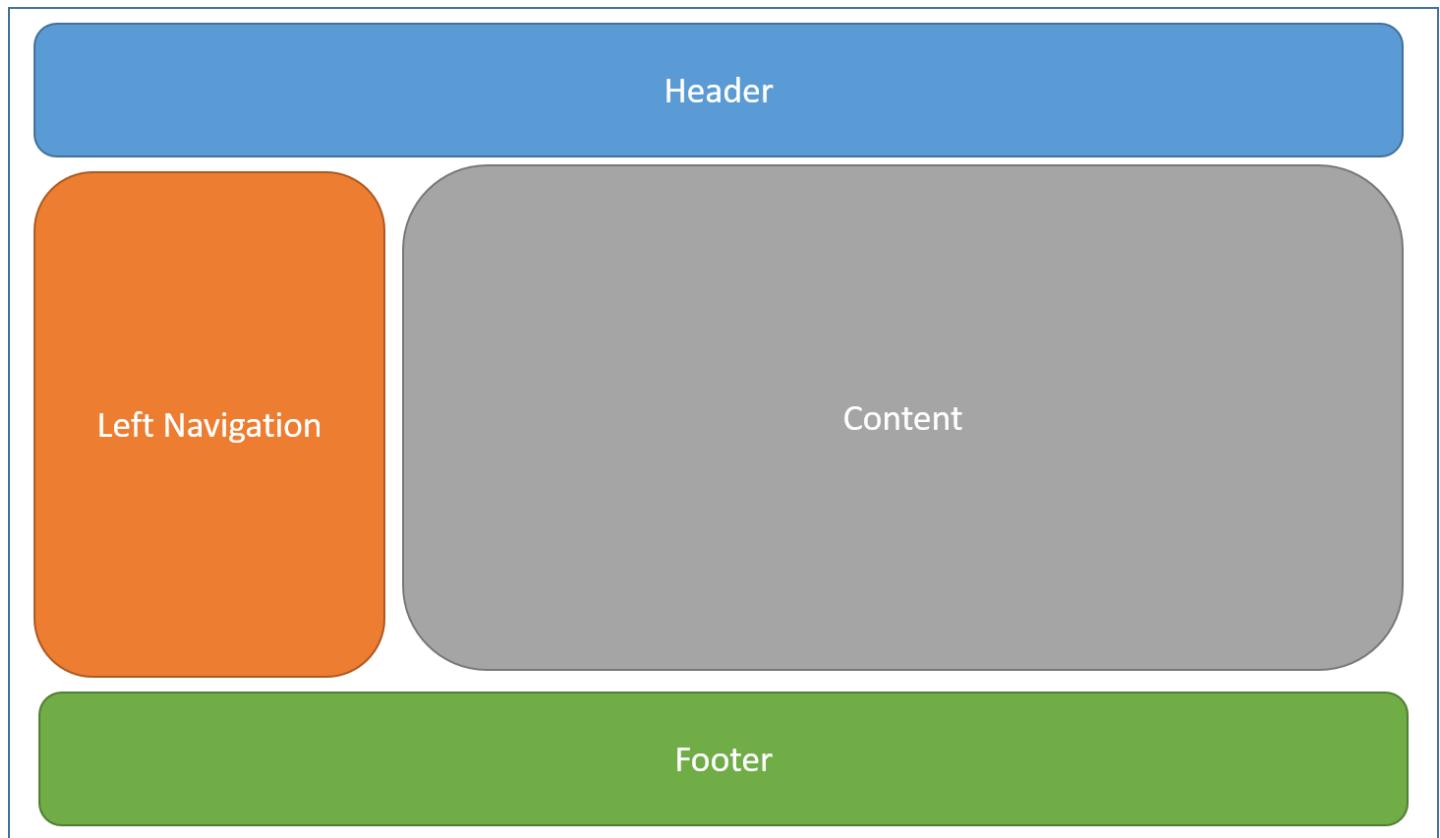
Caso o **Windows** avise que já existem arquivos com o mesmo nome na pasta **wwwroot**, clique no botão para substituir os arquivos existentes.

Com isso os arquivos estáticos necessários a correta exibição das páginas estarão disponíveis em seu projeto. E a pasta **wwwroot** deverá apresentar a mesma estrutura da imagem ao lado.

Nosso próximo passo aora é editar o arquivo **\_Layout.cshtml** que está localizado na pasta **Views\Shared** para criar o mesmo padrão das páginas do template.

## O que é um layout

A maioria dos aplicativos **Web** tem um **layout** comum que fornece aos usuários uma experiência consistente durante sua navegação de uma página a outra. O **layout** normalmente inclui elementos comuns de interface do usuário, como o cabeçalho do aplicativo, elementos de menu ou de navegação e rodapé.



Estruturas **HTML** comuns, como **scripts** e **folhas de estilo**, também são usadas frequentemente por muitas páginas em um aplicativo. Todos esses elementos compartilhados podem ser definidos em um arquivo de **layout**, que pode ser referenciado por qualquer exibição usada no aplicativo. Os **layouts** reduzem o código duplicado nas exibições.

Por convenção, o layout padrão de um aplicativo **ASP.NET Core** é chamado **\_Layout.cshtml**.

O layout define um modelo de nível superior para exibições no aplicativo. Aplicativos não exigem um layout. Os aplicativos podem definir mais de um layout, com diferentes exibições que especificam layouts diferentes.

Vamos alterar o arquivo existente incluindo parte do código da página **index.html** do **template tasteit-master**, abra este arquivo no bloco de notas ou outro editor de sua preferência copie todo o seu conteúdo e substitua o código do arquivo **\_Layout.cshtml**.

Agora recorte no **\_Layout.cshtml** todo o código entre as linhas 70 a 794 e cole esse código no lugar da **DIV** que existe no arquivo **Index.cshtml** na pasta **Views\Home** (voltaremos nele logo mais).

Voltando ao **\_Layout.cshtml**, no lugar do código recortado, escreva:

```
@RenderBody()
```

No final do `_Layout.cshtml`, antes da tag `</body>` inclua a linha de código abaixo:

```
@await RenderSectionAsync("Scripts", required: false)
```

Para finalizarmos nosso layout, precisamos fazer algumas alterações nos caminhos dos arquivos estáticos, e nos links para que sejam direcionadas as ações do `HomeController`. Também vamos incluir um efeito de classe dinâmica para atribuir a classe “`active`” apenas ao link da página que estiver aberta, vamos fazer isso utilizando o `ViewData` e a `ViewBag`.

Abaixo, segue o código completo do arquivo `Views\Shared\_Layout.cshtml`

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <title>RestauranteEtec - @ViewData["Title"]</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <link
        href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500;700;900&display=swap"
        rel="stylesheet">
    <link
        href="https://fonts.googleapis.com/css2?family=Dancing+Script:wght@400;500;600;700&display=sw
        ap" rel="stylesheet">

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
    awesome/4.7.0/css/font-awesome.min.css">

    <link rel="stylesheet" href="~/css/animate.css">

    <link rel="stylesheet" href="~/css/owl.carousel.min.css">
    <link rel="stylesheet" href="~/css/owl.theme.default.min.css">
    <link rel="stylesheet" href="~/css/magnific-popup.css">

    <link rel="stylesheet" href="~/css/bootstrap-datepicker.css">
    <link rel="stylesheet" href="~/css/jquery.timepicker.css">

    <link rel="stylesheet" href="~/css/flaticon.css">
    <link rel="stylesheet" href="~/css/style.css">
</head>
<body>

    <div class="wrap">
        <div class="container">
            <div class="row justify-content-between">
                <div class="col-12 col-md d-flex align-items-center">
                    <p class="mb-0 phone"><span class="mailus">Phone no:</span> <a
                        href="#">+00 1234 567</a> or <span class="mailus">email us:</span> <a
                        href="#">emailsample@email.com</a></p>
                </div>
                <div class="col-12 col-md d-flex justify-content-md-end">
                    <p class="mb-0">Mon - Fri / 9:00-21:00, Sat - Sun / 10:00-20:00</p>
                    <div class="social-media">
```

```

        <p class="mb-0 d-flex">
            <a href="#" class="d-flex align-items-center justify-content-center"><span class="fa fa-facebook"><i class="sr-only">Facebook</i></span></a>
            <a href="#" class="d-flex align-items-center justify-content-center"><span class="fa fa-twitter"><i class="sr-only">Twitter</i></span></a>
            <a href="#" class="d-flex align-items-center justify-content-center"><span class="fa fa-instagram"><i class="sr-only">Instagram</i></span></a>
            <a href="#" class="d-flex align-items-center justify-content-center"><span class="fa fa-dribbble"><i class="sr-only">Dribbble</i></span></a>
        </p>
    </div>
</div>
</div>

<nav class="navbar navbar-expand-lg navbar-dark ftco-navbar bg-dark ftco-navbar-light" id="ftco-navbar">
    <div class="container">
        <a class="navbar-brand" asp-controller="Home" asp-action="Index">
            RestauranteEtec
        </a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#ftco-nav" aria-controls="ftco-nav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="oi oi-menu"></span> Menu
        </button>

        <div class="collapse navbar-collapse" id="ftco-nav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item @(ViewBag.Title == "Home" ? "active" : "")><a asp-controller="Home" asp-action="Index" class="nav-link">Home</a></li>
                <li class="nav-item @(ViewBag.Title == "Quem Somos" ? "active" : "")><a asp-controller="Home" asp-action="QuemSomos" class="nav-link">Quem Somos</a></li>
                <li class="nav-item @(ViewBag.Title == "Chefes" ? "active" : "")><a asp-controller="Home" asp-action="Chefes" class="nav-link">Chefes</a></li>
                <li class="nav-item @(ViewBag.Title == "Menu" ? "active" : "")><a asp-controller="Home" asp-action="Menu" class="nav-link">Menu</a></li>
                <li class="nav-item @(ViewBag.Title == "Reservas" ? "active" : "")><a asp-controller="Home" asp-action="Reservas" class="nav-link">Reservas</a></li>
                <li class="nav-item @(ViewBag.Title == "Blog" ? "active" : "")><a asp-controller="Home" asp-action="Blog" class="nav-link">Blog</a></li>
                <li class="nav-item @(ViewBag.Title == "Contatos" ? "active" : "")><a asp-controller="Home" asp-action="Contatos" class="nav-link">Contatos</a></li>
            </ul>
        </div>
    </div>
</nav>
<!-- END nav -->

@RenderBody()

<footer class="ftco-footer ftco-no-pb ftco-section">
    <div class="container">

```

```

<div class="row mb-5">
    <div class="col-md-6 col-lg-3">
        <div class="ftco-footer-widget mb-4">
            <h2 class="ftco-heading-2">Taste.it</h2>
            <p>Far far away, behind the word mountains, far from the countries  

Vokalia and Consonantia, there live the blind texts. Separated they live in  

Bookmarksgrove</p>
            <ul class="ftco-footer-social list-unstyled float-md-left float-lft  

mt-3">
                <li class="ftco-animate"><a href="#"><span class="fa fa-  

twitter"></span></a></li>
                <li class="ftco-animate"><a href="#"><span class="fa fa-  

facebook"></span></a></li>
                <li class="ftco-animate"><a href="#"><span class="fa fa-  

instagram"></span></a></li>
            </ul>
        </div>
    </div>
    <div class="col-md-6 col-lg-3">
        <div class="ftco-footer-widget mb-4">
            <h2 class="ftco-heading-2">Open Hours</h2>
            <ul class="list-unstyled open-hours">
                <li class="d-flex"><span>Monday</span><span>9:00 -  

24:00</span></li>
                <li class="d-flex"><span>Tuesday</span><span>9:00 -  

24:00</span></li>
                <li class="d-flex"><span>Wednesday</span><span>9:00 -  

24:00</span></li>
                <li class="d-flex"><span>Thursday</span><span>9:00 -  

24:00</span></li>
                <li class="d-flex"><span>Friday</span><span>9:00 -  

02:00</span></li>
                <li class="d-flex"><span>Saturday</span><span>9:00 -  

02:00</span></li>
                <li class="d-flex"><span>Sunday</span><span> Closed</span></li>
            </ul>
        </div>
    </div>
    <div class="col-md-6 col-lg-3">
        <div class="ftco-footer-widget mb-4">
            <h2 class="ftco-heading-2">Instagram</h2>
            <div class="thumb d-sm-flex">
                <a href="#" class="thumb-menu img" style="background-image:  

url(@Url.Content("~/images/insta-1.jpg"));">  

                </a>
                <a href="#" class="thumb-menu img" style="background-image:  

url(@Url.Content("~/images/insta-2.jpg"));">  

                </a>
                <a href="#" class="thumb-menu img" style="background-image:  

url(@Url.Content("~/images/insta-3.jpg"));">  

                </a>
            </div>
            <div class="thumb d-flex">

```

```

                <a href="#" class="thumb-menu img" style="background-image:
url(@Url.Content("~/images/insta-4.jpg"));">
                </a>
                <a href="#" class="thumb-menu img" style="background-image:
url(@Url.Content("~/images/insta-5.jpg"));">
                </a>
                <a href="#" class="thumb-menu img" style="background-image:
url(@Url.Content("~/images/insta-6.jpg"));">
                </a>
            </div>
        </div>
    </div>
<div class="col-md-6 col-lg-3">
    <div class="ftco-footer-widget mb-4">
        <h2 class="ftco-heading-2">Newsletter</h2>
        <p>Far far away, behind the word mountains, far from the
countries.</p>
        <form action="#" class="subscribe-form">
            <div class="form-group">
                <input type="text" class="form-control mb-2 text-center"
placeholder="Enter email address">
                <input type="submit" value="Subscribe" class="form-control
submit px-3">
            </div>
        </form>
    </div>
</div>
<div class="container-fluid px-0 bg-primary py-3">
    <div class="row no-gutters">
        <div class="col-md-12 text-center">

            <p class="mb-0">
                <!-- Link back to Colorlib can't be removed. Template is licensed
under CC BY 3.0. -->
                Copyright &copy;
                <script>document.write(new Date().getFullYear());</script> All rights
reserved | This template is made with <i class="fa fa-heart" aria-hidden="true"></i> by <a
href="https://colorlib.com" target="_blank">Colorlib</a>
                <!-- Link back to Colorlib can't be removed. Template is licensed
under CC BY 3.0. -->
            </p>
        </div>
    </div>
</div>
</footer>

<!-- loader -->
<div id="ftco-loader" class="show fullscreen"><svg class="circular" width="48px"
height="48px"><circle class="path-bg" cx="24" cy="24" r="22" fill="none" stroke-width="4"
stroke="#eeeeee" /><circle class="path" cx="24" cy="24" r="22" fill="none" stroke-width="4"
stroke-miterlimit="10" stroke="#F96D00" /></svg></div>

```

```

<script src="~/js/jquery.min.js"></script>
<script src="~/js/jquery-migrate-3.0.1.min.js"></script>
<script src="~/js/popper.min.js"></script>
<script src="~/js/bootstrap.min.js"></script>
<script src="~/js/jquery.easing.1.3.js"></script>
<script src="~/js/jquery.waypoints.min.js"></script>
<script src="~/js/jquery.stellar.min.js"></script>
<script src="~/js/owl.carousel.min.js"></script>
<script src="~/js/jquery.magnific-popup.min.js"></script>
<script src="~/js/jquery.animateNumber.min.js"></script>
<script src="~/js/bootstrap-datepicker.js"></script>
<script src="~/js/jquery.timepicker.min.js"></script>
<script src="~/js/scrollax.min.js"></script>
<script src="~/js/main.js"></script>

    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>

```

Abaixo, segue o código completo do arquivo **Views\Home\Index.cshtml**

```

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

<section class="hero-wrap">
    <div class="home-slider owl-carousel js-fullheight">
        <div class="slider-item js-fullheight" style="background-
image:url(@Url.Content("~/images/bg_1.jpg"));">
            <div class="overlay"></div>
            <div class="container">
                <div class="row no-gutters slider-text js-fullheight align-items-center
justify-content-center">
                    <div class="col-md-12 ftco-animate">
                        <div class="text w-100 mt-5 text-center">
                            <span class="subheading">Restaurante Etec</span>
                            <h1>Cozinhando Desde</h1>
                            <span class="subheading-2">1993</span>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <div class="slider-item js-fullheight" style="background-
image:url(@Url.Content("~/images/bg_2.jpg"));">
            <div class="overlay"></div>
            <div class="container">
                <div class="row no-gutters slider-text js-fullheight align-items-center
justify-content-center">

```

```
<div class="col-md-12 ftco-animate">
    <div class="text w-100 mt-5 text-center">
        <span class="subheading">Restaurante Etec</span>
        <h1>A Melhor Qualidade</h1>
        <span class="subheading-2 sub">Pratos Diversos</span>
    </div>
</div>
</div>
</div>
</div>
</section>
<section class="ftco-section ftco-wrap-about ftco-no-pb ftco-no-pt">
    <div class="container">
        <div class="row no-gutters">
            <div class="col-sm-4 p-4 p-md-5 d-flex align-items-center justify-content-center bg-primary">
                <form action="#" class="appointment-form">
                    <h3 class="mb-3">Reserve sua Mesa</h3>
                    <div class="row">
                        <div class="col-md-12">
                            <div class="form-group">
                                <input type="text" class="form-control" placeholder="Nome">
                            </div>
                        </div>
                        <div class="col-md-12">
                            <div class="form-group">
                                <input type="email" class="form-control" placeholder="E-mail">
                            </div>
                        </div>
                        <div class="col-md-12">
                            <div class="form-group">
                                <input type="text" class="form-control" placeholder="Fone">
                            </div>
                        </div>
                        <div class="col-md-12">
                            <div class="form-group">
                                <div class="input-wrap">
                                    <div class="icon"><span class="fa fa-calendar"></span></div>
                                    <input type="text" class="form-control book_date" placeholder="Data">
                                </div>
                            </div>
                        </div>
                        <div class="col-md-12">
                            <div class="form-group">
                                <div class="input-wrap">
                                    <div class="icon"><span class="fa fa-clock-o"></span></div>
                                    <input type="text" class="form-control book_time" placeholder="Hora">
                                </div>
                            </div>
                        </div>
                    </div>
                </form>
            </div>
        </div>
    </div>

```

```

                </div>
            </div>
            <div class="col-md-12">
                <div class="form-group">
                    <div class="form-field">
                        <div class="select-wrap">
                            <div class="icon"><span class="fa fa-chevron-down"></span></div>
                            <select name="" id="" class="form-control">
                                <option value="">Convidados</option>
                                <option value="">1</option>
                                <option value="">2</option>
                                <option value="">3</option>
                                <option value="">4</option>
                                <option value="">5</option>
                            </select>
                        </div>
                    </div>
                </div>
                <div class="col-md-12">
                    <div class="form-group">
                        <input type="submit" value="Reserve agora" class="btn btn-white py-3 px-4">
                    </div>
                </div>
            </div>
        </form>
    </div>
    <div class="col-sm-8 wrap-about py-5 ftco-animate img" style="background-image: url(@Url.Content("~/images/about.jpg"));">
        <div class="row pb-5 pb-md-0">
            <div class="col-md-12 col-lg-7">
                <div class="heading-section mt-5 mb-4">
                    <div class="pl-lg-3 ml-md-5">
                        <span class="subheading">Sobre nós</span>
                        <h2 class="mb-4">Bem vindo ao RestauranteEtec</h2>
                    </div>
                </div>
                <div class="pl-lg-3 ml-md-5">
                    <p>On her way she met a copy. The copy warned the Little Blind Text, that where it came from it would have been rewritten a thousand times and everything that was left from its origin would be the word "and" and the Little Blind Text should turn around and return to its own, safe country. A small river named Duden flows by their place and supplies it with the necessary regelialia. It is a paradigmatic country, in which roasted parts of sentences fly into your mouth.</p>
                </div>
            </div>
        </div>
    </div>
</div>
</div>
</section>
```

```

<section class="ftco-section ftco-intro" style="background-image:
url(@Url.Content("~/images/bg_3.jpg"));">
    <div class="overlay">
    </div>
    <div class="container">
        <div class="row">
            <div class="col-md-12 text-center">
                <span>Agendamentos para</span>
                <h2>Jantares Particulares & Happy Hours</h2>
            </div>
        </div>
    </div>
</section>
<section class="ftco-section">
    <div class="container">
        <div class="row justify-content-center mb-5 pb-2">
            <div class="col-md-7 text-center heading-section ftco-animate">
                <span class="subheading">Especialidades</span>
                <h2 class="mb-4">Nosso Menu</h2>
            </div>
        </div>
        <div class="row">
            <div class="col-md-6 col-lg-4">
                <div class="menu-wrap">
                    <div class="heading-menu text-center ftco-animate">
                        <h3>Café da Manhã</h3>
                    </div>
                    <div class="menus d-flex ftco-animate">
                        <div class="menu-img img" style="background-image:
url(@Url.Content("~/images/breakfast-1.jpg"));"></div>
                        <div class="text">
                            <div class="d-flex">
                                <div class="one-half">
                                    <h3>Beef Roast Source</h3>
                                </div>
                                <div class="one-fourth">
                                    <span class="price">$29</span>
                                </div>
                            </div>
                            <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,
<span>Tomatoe</span></p>
                        </div>
                    </div>
                    <div class="menus d-flex ftco-animate">
                        <div class="menu-img img" style="background-image:
url(@Url.Content("~/images/breakfast-2.jpg"));"></div>
                        <div class="text">
                            <div class="d-flex">
                                <div class="one-half">
                                    <h3>Beef Roast Source</h3>
                                </div>
                                <div class="one-fourth">
                                    <span class="price">$29</span>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>

```

```
        </div>
        <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,
<span>Tomatoe</span></p>
    </div>
</div>
<div class="menus border-bottom-0 d-flex ftco-animate">
    <div class="menu-img img" style="background-image:
url(@Url.Content("~/images/breakfast-3.jpg"));"></div>
    <div class="text">
        <div class="d-flex">
            <div class="one-half">
                <h3>Beef Roast Source</h3>
            </div>
            <div class="one-fourth">
                <span class="price">$29</span>
            </div>
        </div>
        <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,
<span>Tomatoe</span></p>
    </div>
</div>
<span class="flat flaticon-bread" style="left: 0;"></span>
<span class="flat flaticon-breakfast" style="right: 0;"></span>
</div>
</div>

<div class="col-md-6 col-lg-4">
    <div class="menu-wrap">
        <div class="heading-menu text-center ftco-animate">
            <h3>Almoço</h3>
        </div>
        <div class="menus d-flex ftco-animate">
            <div class="menu-img img" style="background-image:
url(@Url.Content("~/images/lunch-1.jpg"));"></div>
            <div class="text">
                <div class="d-flex">
                    <div class="one-half">
                        <h3>Beef Roast Source</h3>
                    </div>
                    <div class="one-fourth">
                        <span class="price">$29</span>
                    </div>
                </div>
                <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,
<span>Tomatoe</span></p>
            </div>
</div>
<div class="menus d-flex ftco-animate">
            <div class="menu-img img" style="background-image:
url(@Url.Content("~/images/lunch-2.jpg"));"></div>
            <div class="text">
                <div class="d-flex">
                    <div class="one-half">
                        <h3>Beef Roast Source</h3>
                    </div>
```

```
        </div>
        <div class="one-fourth">
            <span class="price">$29</span>
        </div>
    </div>
    <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,<br/><span>Tomatoe</span></p>
        </div>
    </div>
    <div class="menus border-bottom-0 d-flex ftco-animate">
        <div class="menu-img img" style="background-image:<br/>url(@Url.Content("~/images/lunch-3.jpg"));"></div>
        <div class="text">
            <div class="d-flex">
                <div class="one-half">
                    <h3>Beef Roast Source</h3>
                </div>
                <div class="one-fourth">
                    <span class="price">$29</span>
                </div>
            </div>
            <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,<br/><span>Tomatoe</span></p>
            </div>
        </div>
        <span class="flat flaticon-pizza" style="left: 0;"></span>
        <span class="flat flaticon-chicken" style="right: 0;"></span>
    </div>
</div>

<div class="col-md-6 col-lg-4">
    <div class="menu-wrap">
        <div class="heading-menu text-center ftco-animate">
            <h3>Jantar</h3>
        </div>
        <div class="menus d-flex ftco-animate">
            <div class="menu-img img" style="background-image: url(images/dinner-1.jpg);"></div>
            <div class="text">
                <div class="d-flex">
                    <div class="one-half">
                        <h3>Beef Roast Source</h3>
                    </div>
                    <div class="one-fourth">
                        <span class="price">$29</span>
                    </div>
                </div>
                <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,<br/><span>Tomatoe</span></p>
                </div>
            </div>
            <div class="menu-img img" style="background-image: url(images/dinner-2.jpg);"></div>
        </div>
    </div>

```

```
<div class="text">
    <div class="d-flex">
        <div class="one-half">
            <h3>Beef Roast Source</h3>
        </div>
        <div class="one-fourth">
            <span class="price">$29</span>
        </div>
    </div>
    <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,
<span>Tomatoe</span></p>
</div>
</div>
<div class="menus border-bottom-0 d-flex ftco-animate">
    <div class="menu-img img" style="background-image: url(images/dinner-3.jpg);"></div>
    <div class="text">
        <div class="d-flex">
            <div class="one-half">
                <h3>Beef Roast Source</h3>
            </div>
            <div class="one-fourth">
                <span class="price">$29</span>
            </div>
        </div>
        <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,
<span>Tomatoe</span></p>
        </div>
    </div>
    <span class="flat flaticon-omelette" style="left: 0;"></span>
    <span class="flat flaticon-burger" style="right: 0;"></span>
</div>
</div>

<!-- -->
<div class="col-md-6 col-lg-4">
    <div class="menu-wrap">
        <div class="heading-menu text-center ftco-animate">
            <h3>Sobremessas</h3>
        </div>
        <div class="menus d-flex ftco-animate">
            <div class="menu-img img" style="background-image:
url(images/dessert-1.jpg);"></div>
            <div class="text">
                <div class="d-flex">
                    <div class="one-half">
                        <h3>Beef Roast Source</h3>
                    </div>
                    <div class="one-fourth">
                        <span class="price">$29</span>
                    </div>
                </div>
                <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,
<span>Tomatoe</span></p>
            </div>
        </div>
    </div>
```

```
</div>
</div>
<div class="menus d-flex ftco-animate">
    <div class="menu-img img" style="background-image:
url(images/dessert-2.jpg);"></div>
    <div class="text">
        <div class="d-flex">
            <div class="one-half">
                <h3>Beef Roast Source</h3>
            </div>
            <div class="one-fourth">
                <span class="price">$29</span>
            </div>
        </div>
        <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,
<span>Tomatoe</span></p>
    </div>
</div>
<div class="menus border-bottom-0 d-flex ftco-animate">
    <div class="menu-img img" style="background-image:
url(images/dessert-3.jpg);"></div>
    <div class="text">
        <div class="d-flex">
            <div class="one-half">
                <h3>Beef Roast Source</h3>
            </div>
            <div class="one-fourth">
                <span class="price">$29</span>
            </div>
        </div>
        <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,
<span>Tomatoe</span></p>
    </div>
</div>
<span class="flat flaticon-cupcake" style="left: 0;"></span>
<span class="flat flaticon-ice-cream" style="right: 0;"></span>
</div>
</div>

<div class="col-md-6 col-lg-4">
    <div class="menu-wrap">
        <div class="heading-menu text-center ftco-animate">
            <h3>Vinhos</h3>
        </div>
        <div class="menus d-flex ftco-animate">
            <div class="menu-img img" style="background-image: url(images/wine-
1.jpg);"></div>
            <div class="text">
                <div class="d-flex">
                    <div class="one-half">
                        <h3>Beef Roast Source</h3>
                    </div>
                    <div class="one-fourth">
                        <span class="price">$29</span>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

```
                </div>
            </div>
            <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,
<span>Tomatoe</span></p>
                </div>
            </div>
            <div class="menus d-flex ftco-animate">
                <div class="menu-img img" style="background-image: url(images/wine-2.jpg);"></div>
                <div class="text">
                    <div class="d-flex">
                        <div class="one-half">
                            <h3>Beef Roast Source</h3>
                        </div>
                        <div class="one-fourth">
                            <span class="price">$29</span>
                        </div>
                    </div>
                    <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,
<span>Tomatoe</span></p>
                </div>
            </div>
            <div class="menus border-bottom-0 d-flex ftco-animate">
                <div class="menu-img img" style="background-image: url(images/wine-3.jpg);"></div>
                <div class="text">
                    <div class="d-flex">
                        <div class="one-half">
                            <h3>Beef Roast Source</h3>
                        </div>
                        <div class="one-fourth">
                            <span class="price">$29</span>
                        </div>
                    </div>
                    <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,
<span>Tomatoe</span></p>
                </div>
            </div>
            <span class="flat flaticon-wine" style="left: 0;"></span>
            <span class="flat flaticon-wine-1" style="right: 0;"></span>
        </div>
    </div>

    <div class="col-md-6 col-lg-4">
        <div class="menu-wrap">
            <div class="heading-menu text-center ftco-animate">
                <h3>Bebidas & Chá</h3>
            </div>
            <div class="menus d-flex ftco-animate">
                <div class="menu-img img" style="background-image: url(images/drink-1.jpg);"></div>
                <div class="text">
                    <div class="d-flex">
                        <div class="one-half">
```

```
                <h3>Beef Roast Source</h3>
            </div>
            <div class="one-fourth">
                <span class="price">$29</span>
            </div>
        </div>
        <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,
<span>Tomatoe</span></p>
    </div>
</div>
<div class="menus d-flex ftco-animate">
    <div class="menu-img img" style="background-image: url(images/drink-2.jpg);"></div>
    <div class="text">
        <div class="d-flex">
            <div class="one-half">
                <h3>Beef Roast Source</h3>
            </div>
            <div class="one-fourth">
                <span class="price">$29</span>
            </div>
        </div>
        <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,
<span>Tomatoe</span></p>
    </div>
</div>
<div class="menus border-bottom-0 d-flex ftco-animate">
    <div class="menu-img img" style="background-image: url(images/drink-3.jpg);"></div>
    <div class="text">
        <div class="d-flex">
            <div class="one-half">
                <h3>Beef Roast Source</h3>
            </div>
            <div class="one-fourth">
                <span class="price">$29</span>
            </div>
        </div>
        <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,
<span>Tomatoe</span></p>
    </div>
</div>
<span class="flat flaticon-wine" style="left: 0;"></span>
<span class="flat flaticon-wine-1" style="right: 0;"></span>
</div>
</div>
</div>
</section>

<section class="ftco-section testimony-section" style="background-image: url(@Url.Content("~/images/bg_5.jpg"));">
    <div class="overlay">
        </div>
```

```
<div class="container">
    <div class="row justify-content-center mb-3 pb-2">
        <div class="col-md-7 text-center heading-section heading-section-white ftco-animate">
            <span class="subheading">Relatos</span>
            <h2 class="mb-4">Clientes Satisfeitos</h2>
        </div>
    </div>
    <div class="row ftco-animate justify-content-center">
        <div class="col-md-7">
            <div class="carousel-testimony owl-carousel ftco-owl">
                <div class="item">
                    <div class="testimony-wrap text-center">
                        <div class="text p-3">
                            <p class="mb-4">Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts.</p>
                            <div class="user-img mb-4" style="background-image: url(@Url.Content("~/images/person_1.jpg"))">
                                <span class="quote d-flex align-items-center justify-content-center">
                                    <i class="fa fa-quote-left"></i>
                                    </span>
                                </div>
                                <p class="name">John Gustavo</p>
                                <span class="position">Cliente</span>
                            </div>
                        </div>
                    </div>
                </div>
                <div class="item">
                    <div class="testimony-wrap text-center">
                        <div class="text p-3">
                            <p class="mb-4">Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts.</p>
                            <div class="user-img mb-4" style="background-image: url(@Url.Content("~/images/person_1.jpg"))">
                                <span class="quote d-flex align-items-center justify-content-center">
                                    <i class="fa fa-quote-left"></i>
                                    </span>
                                </div>
                                <p class="name">John Gustavo</p>
                                <span class="position">Cliente</span>
                            </div>
                        </div>
                    </div>
                </div>
                <div class="item">
                    <div class="testimony-wrap text-center">
                        <div class="text p-3">
                            <p class="mb-4">Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts.</p>
                            <div class="user-img mb-4" style="background-image: url(@Url.Content("~/images/person_1.jpg"))">
                                <span class="quote d-flex align-items-center justify-content-center">
                                    <i class="fa fa-quote-left"></i>
                                    </span>
                                </div>
                                <p class="name">John Gustavo</p>
                                <span class="position">Cliente</span>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
```



```
</div>
<div class="row">
    <div class="col-md-6 col-lg-3 ftco-animate">
        <div class="staff">
            <div class="img" style="background-image: url(images/chef-4.jpg);"></div>
            <div class="text px-4 pt-2">
                <h3>John Gustavo</h3>
                <span class="position mb-2">CEO, Co Founder</span>
                <div class="faded">
                    <p>I am an ambitious workaholic, but apart from that, pretty simple person.</p>
                    <ul class="ftco-social d-flex">
                        <li class="ftco-animate"><a href="#"><span class="icon-twitter"></span></a></li>
                        <li class="ftco-animate"><a href="#"><span class="icon-facebook"></span></a></li>
                        <li class="ftco-animate"><a href="#"><span class="icon-google-plus"></span></a></li>
                        <li class="ftco-animate"><a href="#"><span class="icon-instagram"></span></a></li>
                    </ul>
                </div>
            </div>
        </div>
    </div>
    <div class="col-md-6 col-lg-3 ftco-animate">
        <div class="staff">
            <div class="img" style="background-image: url(images/chef-2.jpg);"></div>
            <div class="text px-4 pt-2">
                <h3>Michelle Fraulen</h3>
                <span class="position mb-2">Head Chef</span>
                <div class="faded">
                    <p>I am an ambitious workaholic, but apart from that, pretty simple person.</p>
                    <ul class="ftco-social d-flex">
                        <li class="ftco-animate"><a href="#"><span class="icon-twitter"></span></a></li>
                        <li class="ftco-animate"><a href="#"><span class="icon-facebook"></span></a></li>
                        <li class="ftco-animate"><a href="#"><span class="icon-google-plus"></span></a></li>
                        <li class="ftco-animate"><a href="#"><span class="icon-instagram"></span></a></li>
                    </ul>
                </div>
            </div>
        </div>
    </div>
    <div class="col-md-6 col-lg-3 ftco-animate">
        <div class="staff">
            <div class="img" style="background-image: url(images/chef-3.jpg);"></div>
            <div class="text px-4 pt-2">
                <h3>Alfred Smith</h3>
                <span class="position mb-2">Chef Cook</span>
            </div>
        </div>
    </div>

```

```

        <div class="faded">
            <p>I am an ambitious workaholic, but apart from that, pretty
simple person.</p>
            <ul class="ftco-social d-flex">
                <li class="ftco-animate"><a href="#"><span class="icon-
twitter"></span></a></li>
                <li class="ftco-animate"><a href="#"><span class="icon-
facebook"></span></a></li>
                <li class="ftco-animate"><a href="#"><span class="icon-
google-plus"></span></a></li>
                <li class="ftco-animate"><a href="#"><span class="icon-
instagram"></span></a></li>
            </ul>
        </div>
    </div>
<div class="col-md-6 col-lg-3 ftco-animate">
    <div class="staff">
        <div class="img" style="background-image: url(images/chef-1.jpg);"></div>
        <div class="text px-4 pt-2">
            <h3>Antonio Santibanez</h3>
            <span class="position mb-2">Chef Cook</span>
            <div class="faded">
                <p>I am an ambitious workaholic, but apart from that, pretty
simple person.</p>
                <ul class="ftco-social d-flex">
                    <li class="ftco-animate"><a href="#"><span class="icon-
twitter"></span></a></li>
                    <li class="ftco-animate"><a href="#"><span class="icon-
facebook"></span></a></li>
                    <li class="ftco-animate"><a href="#"><span class="icon-
google-plus"></span></a></li>
                    <li class="ftco-animate"><a href="#"><span class="icon-
instagram"></span></a></li>
                </ul>
            </div>
        </div>
    </div>
</div>
</div>
</section>

<section class="ftco-section ftco-no-pt ftco-no-pb">
    <div class="container">
        <div class="row d-flex">
            <div class="col-md-6 d-flex">
                <div class="img img-2 w-100 mr-md-2" style="background-image:
url(@Url.Content("~/images/bg_6.jpg"));"></div>
                <div class="img img-2 w-100 ml-md-2" style="background-image:
url(@Url.Content("~/images/bg_4.jpg"));"></div>
            </div>
            <div class="col-md-6 ftco-animate makereservation p-4 p-md-5">

```

```

<div class="heading-section ftco-animate mb-5">
    <span class="subheading">Este é o Nosso Segredo</span>
    <h2 class="mb-4">Ingredientes Perfeitos</h2>
    <p>
        Far far away, behind the word mountains, far from the countries
        Vokalia and Consonantia, there live the blind texts. Separated they live in Bookmarksgrove
        right at the coast of the Semantics, a large language ocean.
    </p>
    <p><a href="#" class="btn btn-primary">Saiba Mais</a></p>
</div>
</div>
</div>
</div>
</section>

<section class="ftco-section bg-light">
<div class="container">
    <div class="row justify-content-center mb-5 pb-2">
        <div class="col-md-7 text-center heading-section ftco-animate">
            <span class="subheading">Blog</span>
            <h2 class="mb-4">Recentes</h2>
        </div>
    </div>
    <div class="row">
        <div class="col-md-4 ftco-animate">
            <div class="blog-entry">
                <a href="blog-single.html" class="block-20" style="background-image:
url('images/image_1.jpg');">
                </a>
                <div class="text px-4 pt-3 pb-4">
                    <div class="meta">
                        <div><a href="#">August 3, 2020</a></div>
                        <div><a href="#">Admin</a></div>
                    </div>
                    <h3 class="heading"><a href="#">Even the all-powerful Pointing has no
control about the blind texts</a></h3>
                    <p class="clearfix">
                        <a href="#" class="float-left read btn btn-primary">Read more</a>
                        <a href="#" class="float-right meta-chat"><span class="fa fa-
comment"></span> 3</a>
                    </p>
                </div>
            </div>
        </div>
        <div class="col-md-4 ftco-animate">
            <div class="blog-entry">
                <a href="blog-single.html" class="block-20" style="background-image:
url('images/image_2.jpg');">
                </a>
                <div class="text px-4 pt-3 pb-4">
                    <div class="meta">
                        <div><a href="#">August 3, 2020</a></div>
                        <div><a href="#">Admin</a></div>
                    </div>

```

```
<h3 class="heading"><a href="#">Even the all-powerful Pointing has no control about the blind texts</a></h3>
<p class="clearfix">
    <a href="#" class="float-left read btn btn-primary">Read more</a>
    <a href="#" class="float-right meta-chat"><span class="fa fa-comment"></span> 3</a>
</p>
</div>
</div>
</div>
<div class="col-md-4 ftco-animate">
    <div class="blog-entry">
        <a href="blog-single.html" class="block-20" style="background-image: url('images/image_3.jpg');">
            </a>
        <div class="text px-4 pt-3 pb-4">
            <div class="meta">
                <div><a href="#">August 3, 2020</a></div>
                <div><a href="#">Admin</a></div>
            </div>
            <h3 class="heading"><a href="#">Even the all-powerful Pointing has no control about the blind texts</a></h3>
            <p class="clearfix">
                <a href="#" class="float-left read btn btn-primary">Read more</a>
                <a href="#" class="float-right meta-chat"><span class="fa fa-comment"></span> 3</a>
            </p>
            </div>
        </div>
    </div>
</div>
</div>
</section>
<section class="ftco-section ftco-no-pt ftco-no-pb ftco-intro bg-primary">
    <div class="container py-5">
        <div class="row py-2">
            <div class="col-md-12 text-center">
                <h2>Fazemos Deliciosas &amp; Nutritivas Refeições</h2>
                <a href="#" class="btn btn-white btn-outline-white">Reserve sua mesa agora</a>
            </div>
        </div>
    </div>
</section>
```

## Criando as Demais Páginas do Projeto

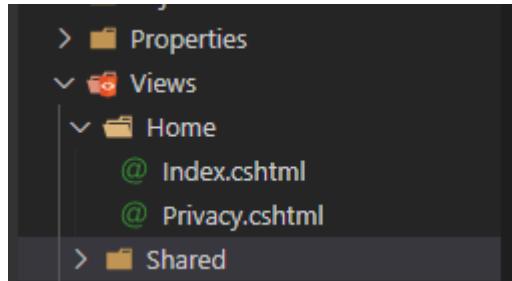
No **MVC** para criar páginas, precisamos criar uma **Action** (Ação) no **Controller** responsável pela página. Além das ações criamos as **Views (Exibições)** na pasta com o mesmo nome do **Controller** na **View**.

**1º - No HomeController, vamos adicionar os códigos abaixo:**

```
0 references
21     public IActionResult Index()
22     {
23         return View();
24     }
25
26     0 references
27     public IActionResult QuemSomos()
28     {
29         return View();
30     }
31
32     0 references
33     public IActionResult Chefes()
34     {
35         return View();
36     }
37
38     0 references
39     public IActionResult Menu()
40     {
41         return View();
42     }
43
44     0 references
45     public IActionResult Reservas()
46     {
47         return View();
48     }
49
50     0 references
51     public IActionResult Blog()
52     {
53         return View();
54     }
55
56     0 references
57     public IActionResult Contatos()
58     {
59         return View();
60     }
```

Vale lembrar que no nosso **Layout**, já deixamos configurados os links da **NavBar** para corresponder as ações criadas acima.

Agora para cada ação (**com exceção da Index**) que já foi desenvolvida, vamos adicionar na pasta **Views\Home** uma **Razor Page** com o mesmo nome, conforme a figura abaixo:



Para criar **Controllers** e **Views** através de linha de comando é necessário possuir a ferramenta **dotnet-aspnet-codegenerator** instalada. Para realizar sua instalação pasta executar o seguinte comando no terminal.

```
dotnet tool install -g dotnet-aspnet-codegenerator --version  
5.0.2
```

Essa ferramenta ficará disponível para todos os seus projetos, e mais informações sobre ela pode ser consultada diretamente no site da **Microsoft**: <https://docs.microsoft.com/pt-br/aspnet/core/fundamentals/tools/dotnet-aspnet-codegenerator?view=aspnetcore-6.0>.

Agora precisamos instalar e referenciar dois pacotes necessários a geração dos **Controladores** e **Views**.

```
dotnet add package Microsoft.VisualStudio.Web.CodeGenerators.Mvc --  
version 5.0.2
```

e

```
dotnet add package Microsoft.VisualStudio.Web.CodeGeneration.Design --  
version 5.0.2
```

Com isso temos a nossa disposição uma lista de comandos para criação de **Controladores** e **Views**. Você pode conferir a lista de opções para criação de **View**, por exemplo, através do comando abaixo:

```
dotnet-aspnet-codegenerator view -h
```

Vamos criar nossa primeira **View** – “**Quem Somos**”, abra o terminal e execute o seguinte comando:

```
dotnet-aspnet-codegenerator view QuemSomos Empty -udl -outDir  
"Views/Home"
```

A primeira parte do comando é a chamada a ferramenta de geração de código especificando que queremos criar uma **View**, em seguida temos o nome da **View “QuemSomos”**, a palavra **Empty** especifica o tipo de **template (Empty – Vazio**, para **Cruds** temos os: **Create – Criar, Edit – Editar, Delete – Excluir, Details – Detalhes e List – Página Index** com uma **lista** para exibir os dados cadastrados), o argumento “**-udl**” especifica que queremos usar o layout padrão (**\_Layout.cshtml**) e o ”**-outDir**” é necessário para especificar o local onde o **View** será criada.

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows the project structure for 'RESTAURANTEETEC' with files like .vscode, bin, Controllers (HomeController.cs), Models, obj, Properties, Views (Home, Index.cshtml, Privacy.cshtml, QuemSomos.cshtml), and Shared. The Editor tab bar at the top has tabs for \_Layout.cshtml, Index.cshtml, HomeController.cs, and QuemSomos.cshtml (which is currently active). The status bar at the bottom shows the file path: Views > Home > QuemSomos.cshtml.

```

1  @{
2      ViewData["Title"] = "QuemSomos";
3  }
4
5
6  <h1>QuemSomos</h1>
7
8

```

Vamos começar pela edição da página, e logo no começo da **View** altere o “**Title**” para “**Quem Somos**”, incluindo um espaço entre as palavras.

The screenshot shows the QuemSomos.cshtml file in the Editor. The code has been modified to include a space between 'Quem' and 'Somos' in the ViewData['Title'] assignment.

```

1  @{
2      ViewData["Title"] = "Quem Somos";
3  }
4

```

Agora copie o código do arquivo **about.html** do **template tasteit.master**, das linhas 70 a 244 e cole abaixo no arquivo **QuemSomos**, em seguida faça a alteração no código copiado alterando os caminhos das imagens, conforme feito no **Index.cshtml**.

Também faça a correção dos links **href** para **asp-action**.

Para facilitar o entendimento segue abaixo o código final da página QuemSomos

```

@{
    ViewData["Title"] = "Quem Somos";
}

<section class="hero-wrap hero-wrap-2" style="background-image:
url(@Url.Content("~/images/bg_5.jpg"));" data-stellar-background-ratio="0.5">
    <div class="overlay">
    </div>
    <div class="container">

```

```

<div class="row no-gutters slider-text align-items-end justify-content-center">
    <div class="col-md-9 ftco-animate text-center mb-5">
        <h1 class="mb-2 bread">Quem Somos</h1>
        <p class="breadcrumbs"><span class="mr-2"><a asp-action="Index">Home <i class="fa fa-chevron-right"></i></a></span> <span>Quem Somos <i class="fa fa-chevron-right"></i></span></p>
    </div>
</div>
</section>

<section class="ftco-section ftco-no-pt ftco-no-pb">
    <div class="container">
        <div class="row d-flex">
            <div class="col-md-6 d-flex">
                <div class="img img-2 w-100 mr-md-2" style="background-image: url(@Url.Content("~/images/bg_6.jpg"));"></div>
                <div class="img img-2 w-100 ml-md-2" style="background-image: url(@Url.Content("~/images/bg_4.jpg"));"></div>
            </div>
            <div class="col-md-6 ftco-animate makereservation p-4 p-md-5">
                <div class="heading-section ftco-animate mb-5">
                    <span class="subheading">Este é o Nosso Segredo</span>
                    <h2 class="mb-4">Ingredientes Perfeitos</h2>
                    <p>
                        Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts. Separated they live in Bookmarksgrove right at the coast of the Semantics, a large language ocean.
                    </p>
                    <p><a href="#" class="btn btn-primary">Saiba Mais</a></p>
                </div>
            </div>
        </div>
    </div>
</section>

<section class="ftco-section ftco-counter img" id="section-counter" style="background-image: url(@Url.Content("~/images/bg_4.jpg")); data-stellar-background-ratio="0.5">
    <div class="container">
        <div class="row d-md-flex align-items-center justify-content-center">
            <div class="col-lg-10">
                <div class="row d-md-flex align-items-center">
                    <div class="col-md d-flex justify-content-center counter-wrap ftco-animate">
                        <div class="block-18">
                            <div class="text">
                                <strong class="number" data-number="100">0</strong>
                                <span>Tasty Dishes</span>
                            </div>
                        </div>
                    </div>
                    <div class="col-md d-flex justify-content-center counter-wrap ftco-animate">
                        <div class="block-18">
                            <div class="text">
                                <strong class="number" data-number="4000">0</strong>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>

```

```

                <span>Dishes Served</span>
            </div>
        </div>
    </div>
    <div class="col-md d-flex justify-content-center counter-wrap ftco-animate">
        <div class="block-18">
            <div class="text">
                <strong class="number" data-number="10">0</strong>
                <span>Restaurants</span>
            </div>
        </div>
    </div>
    <div class="col-md d-flex justify-content-center counter-wrap ftco-animate">
        <div class="block-18">
            <div class="text">
                <strong class="number" data-number="10000">0</strong>
                <span>Happy Customers</span>
            </div>
        </div>
    </div>
</div>
</div>
</section>

<section class="ftco-section ftco-no-pt ftco-no-pb ftco-intro bg-primary">
    <div class="container py-5">
        <div class="row py-2">
            <div class="col-md-12 text-center">
                <h2>Fazemos Deliciosas & Nutritivas Refeições</h2>
                <a href="#" class="btn btn-white btn-outline-white">Reserve sua mesa agora</a>
            </div>
        </div>
    </div>
</section>

<section class="ftco-section testimony-section" style="background-image: url(@Url.Content("~/images/bg_5.jpg"));">
    <div class="overlay">
    </div>
    <div class="container">
        <div class="row justify-content-center mb-3 pb-2">
            <div class="col-md-7 text-center heading-section heading-section-white ftco-animate">
                <span class="subheading">Relatos</span>
                <h2 class="mb-4">Clientes Satisfeitos</h2>
            </div>
        </div>
        <div class="row ftco-animate justify-content-center">
            <div class="col-md-7">
                <div class="carousel-testimony owl-carousel ftco-owl">
                    <div class="item">

```

```
<div class="testimony-wrap text-center">
    <div class="text p-3">
        <p class="mb-4">Far far away, behind the word mountains, far
from the countries Vokalia and Consonantia, there live the blind texts.</p>
        <div class="user-img mb-4" style="background-image:
url(@Url.Content("~/images/person_1.jpg"))">
            <span class="quote d-flex align-items-center justify-
content-center">
                <i class="fa fa-quote-left"></i>
                </span>
            </div>
            <p class="name">John Gustavo</p>
            <span class="position">Cliente</span>
        </div>
    </div>
<div class="item">
    <div class="testimony-wrap text-center">
        <div class="text p-3">
            <p class="mb-4">Far far away, behind the word mountains, far
from the countries Vokalia and Consonantia, there live the blind texts.</p>
            <div class="user-img mb-4" style="background-image:
url(@Url.Content("~/images/person_1.jpg"))">
                <span class="quote d-flex align-items-center justify-
content-center">
                    <i class="fa fa-quote-left"></i>
                    </span>
                </div>
                <p class="name">John Gustavo</p>
                <span class="position">Cliente</span>
            </div>
        </div>
    </div>
<div class="item">
    <div class="testimony-wrap text-center">
        <div class="text p-3">
            <p class="mb-4">Far far away, behind the word mountains, far
from the countries Vokalia and Consonantia, there live the blind texts.</p>
            <div class="user-img mb-4" style="background-image:
url(@Url.Content("~/images/person_1.jpg"))">
                <span class="quote d-flex align-items-center justify-
content-center">
                    <i class="fa fa-quote-left"></i>
                    </span>
                </div>
                <p class="name">John Gustavo</p>
                <span class="position">Cliente</span>
            </div>
        </div>
    </div>
<div class="item">
    <div class="testimony-wrap text-center">
        <div class="text p-3">
            <p class="mb-4">Far far away, behind the word mountains, far
from the countries Vokalia and Consonantia, there live the blind texts.</p>
            <div class="user-img mb-4" style="background-image:
url(@Url.Content("~/images/person_1.jpg"))">
                <span class="quote d-flex align-items-center justify-
content-center">
                    <i class="fa fa-quote-left"></i>
                    </span>
                </div>
                <p class="name">John Gustavo</p>
                <span class="position">Cliente</span>
            </div>
        </div>
    </div>
<div class="item">
    <div class="testimony-wrap text-center">
        <div class="text p-3">
```

```

        <p class="mb-4">Far far away, behind the word mountains, far
from the countries Vokalia and Consonantia, there live the blind texts.</p>
        <div class="user-img mb-4" style="background-image:
url(@Url.Content("~/images/person_1.jpg"))">
            <span class="quote d-flex align-items-center justify-
content-center">
                <i class="fa fa-quote-left"></i>
            </span>
        </div>
        <p class="name">John Gustavo</p>
        <span class="position">Cliente</span>
    </div>
</div>
<div class="item">
    <div class="testimony-wrap text-center">
        <div class="text p-3">
            <p class="mb-4">Far far away, behind the word mountains, far
from the countries Vokalia and Consonantia, there live the blind texts.</p>
            <div class="user-img mb-4" style="background-image:
url(@Url.Content("~/images/person_1.jpg"))">
                <span class="quote d-flex align-items-center justify-
content-center">
                    <i class="fa fa-quote-left"></i>
                </span>
            </div>
            <p class="name">John Gustavo</p>
            <span class="position">Cliente</span>
        </div>
    </div>
    </div>
</div>
</div>
</div>
</section>
```

Agora é com você, repita os passos da “**Quem Somos**”, para realizar a criação das **Views: Chefes, Menu, Reservas, Blog e Contatos**, usando a tabela abaixo como referência.

View	Title	Arquivo HTML	Linhas
<b>QuemSomos</b>	Quem Somos	about.html	70 – 244
<b>Chefes</b>	Chefes	chef.html	70 – 266
<b>Menu</b>	Menu	menu.html	70 – 810
<b>Reservas</b>	Reservas	reservation.html	70 – 168
<b>Blog</b>	Blog	blog.html	70 – 205
<b>Contatos</b>	Contatos	contact.html	70 – 141

Não esqueça de incluir o **ViewData[“Title”]** e fazer as alterações nos caminhos das imagens e links.

## Criando os Models do Projeto

Os **modelos (Model)** são utilizados para manipular informações de forma mais detalhada, sendo recomendado que, sempre que possível, se utilize dos modelos para realizar consultas, cálculos e todas as regras de negócio do nosso site ou sistema.

Fazendo uma análise das páginas, durante as aulas, verificamos alguns objetos que vamos modelar como nossas classes, abaixo, segue a lista analisada:

### Reserva

Id	int
NomePessoa	string 60 not null
EmailPessoa	string 100 not null
FonePessoa	string 20 not null
DataCadastro	DateTime
DataReserva	DateTime not null
Convidados	byte not null
Status	byte

### Cargo

Id	int
Nome	string 30 not null

### Categoria

Id	int
Nome	string 30 not null

### Funcionario

Id	int
Nome	string 60 not null
Descricao	string 500 not null
Foto	string 200
Cargold	int not null
ExibirHome	bool
OrdemExibicao	byte
Ativo	bool

### Produto

Id	int
Nome	string 60 not null
Descricao	string 500 not null
Preco	decimal not null
Categoriald	int not null
Foto	string 200
ExibirHome	bool
Ativo	bool

### Blog

Id	int
DataCadastro	DateTime
Titulo	string 100 not null
Texto	string 8000 not null
Imagen	string 200

### Relato

Id	int
Texto	string 1000 not null
NomePessoa	string 60 not null
FotoPessoa	string 200
TipoPessoa	string 30 not null
DataCadastro	datetime
ExibirHome	bool
OrdemExibicao	byte
Ativo	bool

### Contato

Id	int
DataContato	DateTime
NomePessoa	string 60 not null
EmailPessoa	string 100 not null
Assunto	string 100 not null
Mensagem	string 500 not null
Status	byte
Retorno	string 500

Usando essa análise vamos agora criar os nossos modelos. Caso você tenha instalado a extensão “**C# Extensions**”, pode simplesmente clicar com o botão direito na pasta **Models** e escolher a opção **New C# -> Class**, e na caixa que é apresentada digite **Categoria** e pressione **Enter**, agora é só alterar o código da classe conforme a imagem abaixo:

```
C# Categoria.cs X
Models > C# Categoria.cs > ...
1  using System.ComponentModel.DataAnnotations;
2  using System.ComponentModel.DataAnnotations.Schema;
3
4  namespace RestauranteEtec.Models
5  {
6      [Table("Categoria")]
7      12 references
8      public class Categoria
9      {
10         [Key]
11         3 references
12         public int Id { get; set; }
13
14         [Required(ErrorMessage = "Por favor, informe o Nome da Categoria")]
15         [StringLength(30, ErrorMessage = "O Nome deve possuir no máximo 30 caracteres")]
16         4 references
17         public string Nome { get; set; }
18     }
19 }
```

No código acima, as linhas 7, 10, 13 e 14, são anotações de dados (**Data Annotation**). Para usar essas anotações são necessários os **usings** das linhas 2 e 3.

A validação dos dados é a primeira e mais importante etapa na proteção de um aplicativo. Ela impede que o aplicativo processe entradas indesejadas que podem produzir resultados imprevisíveis. A plataforma .NET oferece o recurso conhecido como **DataAnnotation** presente no **namespace System.ComponentModel.DataAnnotations** que possui várias classes e atributos para nos ajudar a validar dados. Mais informações podem ser obtidas em: <https://www.learnentityframeworkcore.com/configuration/data-annotation-attributes>.

Utilizando a lista que fizemos e as duas classes de exemplo, crie as demais classes de modelo, lembre-se de adicionar as anotações de dados.

- **[Key]** nas chaves primárias, ou seja os **Id**.
- **[StringLength(tamanho)]** para as propriedades texto (**string**).
- **[Required]** para as propriedades obrigatórios, aquelas na lista marcadas com **NotNull**

Agora vamos a criação da classe, **Produto**, seguindo o mesmo processo da **Categoria**:

## C# Produto.cs X

```
Models > C# Produto.cs > ...
1  using System.ComponentModel.DataAnnotations;
2  using System.ComponentModel.DataAnnotations.Schema;
3
4  namespace RestauranteEtec.Models
5  {
6      [Table("Produto")]
7      11 references
8      public class Produto
9      {
10         [Key]
11         3 references
12         public int Id { get; set; }
13
14         [Required(ErrorMessage = "Por favor, informe o Nome do Produto")]
15         [StringLength(50, ErrorMessage = "O Nome deve possuir no máximo 50 caracteres")]
16         4 references
17         public string Nome { get; set; }
18
19         [Display(Name = "Descrição")]
20         [Required(ErrorMessage = "Por favor, informe a Descrição do Produto")]
21         [StringLength(200, ErrorMessage = "A Descrição deve possuir no máximo 50 caracteres")]
22         4 references
23         public string Descricao { get; set; }
24
25         [Display(Name = "Preço")]
26         [Required(ErrorMessage = "Por favor, informe o Preço do Produto")]
27         4 references
28         public decimal Preco { get; set; }
29
30         [StringLength(200)]
31         4 references
32         public string Foto { get; set; }
33
34         [Display(Name = "Categoria")]
35         [Required(ErrorMessage = "Por favor, informe a Categoria do Produto")]
36         6 references
37         public int CategoriaId { get; set; }
38         2 references
39         public Categoria Categoria { get; set; }
40
41         [Display(Name = "Destaque?")]
42         4 references
43         public bool ExibirHome { get; set; }
44
45         4 references
46         public bool Ativo { get; set; } = true;
47     }
48 }
```

Acrescentando mais um exemplo, abaixo é apresentado a **Model** de Relatos, onde temos uma propriedade **DateTime**, e outra **bool**. Aqui também está colocando na frente dos métodos de “**get e set**” destas propriedades valores padrões, que serão assumidos quando um objeto dessa classe for criado. Além de uma nova **Anotação de Dados**, para fixar a forma padrão de exibição da **Data** da propriedade **DataCadastro**.

## C# Relato.cs X

Models &gt; C# Relato.cs &gt; ...

```
1  using System;
2  using System.ComponentModel.DataAnnotations;
3  using System.ComponentModel.DataAnnotations.Schema;
4
5  namespace RestauranteEtec.Models
6  {
7      [Table("Relato")]
8      11 references
9      public class Relato
10     {
11         [Key]
12         3 references
13         public int Id { get; set; }
14
15         [Required(ErrorMessage = "Por favor, informe o Texto do Relato")]
16         [StringLength(500)]
17         4 references
18         public string Texto { get; set; }
19
20         [Required(ErrorMessage = "Por favor, informe o Nome da Pessoa")]
21         [StringLength(60, ErrorMessage = "O Nome da Pessoa deve possuir no máximo 60 caracteres")]
22         4 references
23         public string NomePessoa { get; set; }
24
25         [Required(ErrorMessage = "Por favor, informe o Tipo da Pessoa")]
26         [StringLength(30, ErrorMessage = "O Tipo da Pessoa deve possuir no máximo 30 caracteres")]
27         4 references
28         public string TipoPessoa { get; set; }
29
30         [StringLength(200)]
31         4 references
32         public string FotoPessoa { get; set; }
33
34         [DisplayFormat(DataFormatString = "{0:dd/MM/yyyy hh:mm}")]
35         2 references
36         public DateTime DataCadastro { get; set; } = DateTime.Now;
37
38         4 references
39         public bool ExibirHome { get; set; }
40
41         4 references
42         public byte OrdemExibicao { get; set; }
43
44         4 references
45         public bool Ativo { get; set; } = true;
46     }
47 }
```

## Criando uma Interface

Uma interface contém definições para um grupo de funcionalidades relacionadas que uma classe não abstrata ou uma **struct** deve implementar. Uma interface pode definir **static** métodos, que devem ter uma implementação. A partir do C# 8,0, uma interface pode definir uma implementação padrão para membros. Uma interface não pode declarar dados de instância, como campos, propriedades implementadas automaticamente ou eventos de propriedade.

Usando interfaces, você pode, por exemplo, incluir o comportamento de várias fontes em uma classe. Essa funcionalidade é importante em C# porque a linguagem não dá suporte a várias heranças de classes. Além disso, use uma interface se você deseja simular a herança para **structs**, pois eles não podem herdar de outro **struct** ou classe.

O nome de uma interface deve ser um nome de **identificador C#** válido. Por convenção, os nomes de interface começam com uma letra maiúscula I.

Qualquer classe ou struct que implemente a interface **IEquatable<T>** deve conter uma definição para um método **Equals** que corresponda à assinatura que a interface específica. Como resultado, você pode contar com uma classe que implementa **IEquatable<T>** para conter um método **Equals** com o qual uma instância da classe pode determinar se é igual a outra instância da mesma classe.

A definição de **IEquatable<T>** não fornece uma implementação para o **Equals**. Uma classe ou estrutura pode implementar várias interfaces, mas uma classe só pode herdar de uma única classe.

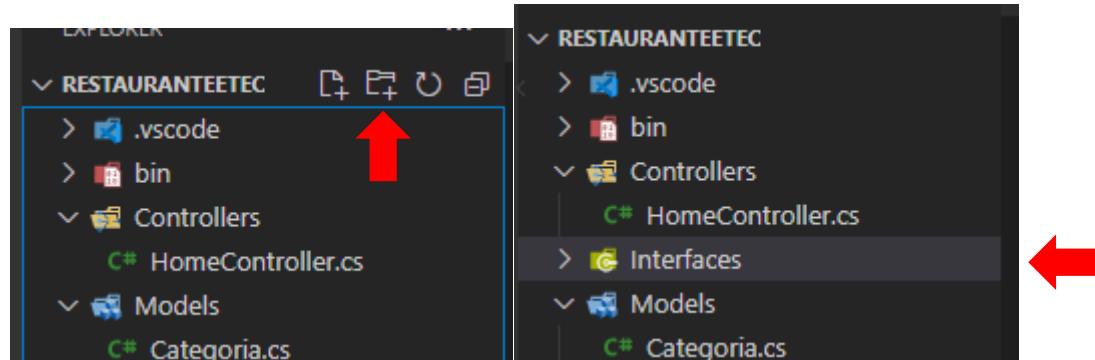
Para obter mais informações sobre classes abstratas, consulte [Classes e membros de classes abstratas e lacrados](#).

As interfaces podem conter métodos de instância, propriedades, eventos, indexadores ou qualquer combinação desses quatro tipos de membro. As interfaces podem conter construtores, campos, constantes ou operadores estáticos. Para obter links para exemplos, consulte as [seções relacionadas](#). Uma interface não pode conter campos de instância, construtores de instância ou finalizadores. Os membros da interface são públicos por padrão e você pode especificar explicitamente modificadores de acessibilidade, como, **public protected internal private protected internal** ou **private protected**. Um **private** membro deve ter uma implementação padrão.

Para implementar um membro de interface, o membro correspondente da classe de implementação deve ser público, não estático e ter o mesmo nome e assinatura do membro de interface.

Quando uma classe ou **struct** implementa uma interface, a classe ou **struct** deve fornecer uma implementação para todos os membros que a interface declara, mas não fornece uma implementação padrão para o. No entanto, se uma classe base implementa uma interface, qualquer classe que é derivada da classe base herda essa implementação.

Em nossa aplicação, vamos criar uma pasta para armazenar nossa interface. Clique no menu lateral, onde os arquivos são exibidos, abaixo do último arquivo, para selecionar a pasta principal do projeto. E então clique no ícone que possui uma pasta com um sinal de adição, para criar uma pasta com o nome **Interfaces**.



Caso você tenha instalado a extensão “**C# Extensions**”, pode simplesmente clicar com o botão direito na pasta **Interfaces** e escolher a opção **New C# -> Interface**, e na caixa que é apresentada digite **IRepository** e pressione **Enter**, agora é só alterar o código a imagem abaixo:

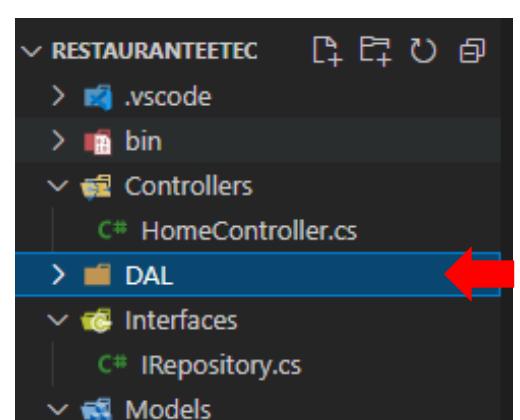
```
C# IRepository.cs •

Interfaces > C# IRepository.cs > ...
1   using System.Collections.Generic;
2
3   namespace RestauranteEtec.Interfaces
4   {
5       0 references
6       public interface IRepository<T>
7       {
8           0 references
9           IEnumarable<T> GetAll();
10          0 references
11          T GetById(int? id);
12          0 references
13          void Add(T model);
14          0 references
15          void Update(T model);
16          0 references
17          void Delete(int? id);
18      }
19  }
```

Agora com essa interface criada, temos a possibilidade de criar as demais classes da nossa **Camada de Acesso da Dados (DAL)**. Essas classes vão implementar os métodos da interface **IRepository** de forma a acessar e possibilitar alteração dos dados de nosso banco de dados.

Em nossa aplicação, vamos criar uma pasta para armazenar as classes **DAL**. Assim como feito anteriormente crie uma pasta com o nome **DAL** na estrutura geral do projeto.

Antes de criarmos nossas classes vamos precisar de um pacote que contém as bibliotecas de acesso ao banco de dados **MySQL**, que estamos usando em nosso projeto.



Para adicionar esse pacote no **Code**, abra seu terminal e execute o comando abaixo:

```
dotnet add package MySql.Data --version 8.0.25
```

Agora sim estamos prontos para criar nossa camada de acesso a dados.

Clique com o botão direito na pasta **DAL** e selecione a opção de adicionar uma nova classe. Coloque o nome da classe de **CargoDAL** e faça a inclusão do código abaixo para informar a implementação da **Interface**:

```
C# CargoDAL.cs 2 ●  
DAL > C# CargoDAL.cs > {} RestauranteEtec.DAL  
1   using System;  
2   using System.Collections.Generic;  
3   using System.Linq;  
4   using System.Threading.Tasks;  
5  
6   namespace RestauranteEtec.DAL  
7   {  
8       0 references  
9       public class CargoDAL : IRepository<Cargo>  
10      {  
11      }  
12  }
```

Para resolver os problemas, precisamos adicionar duas referências a nossa classe (linhas 5 e 6):

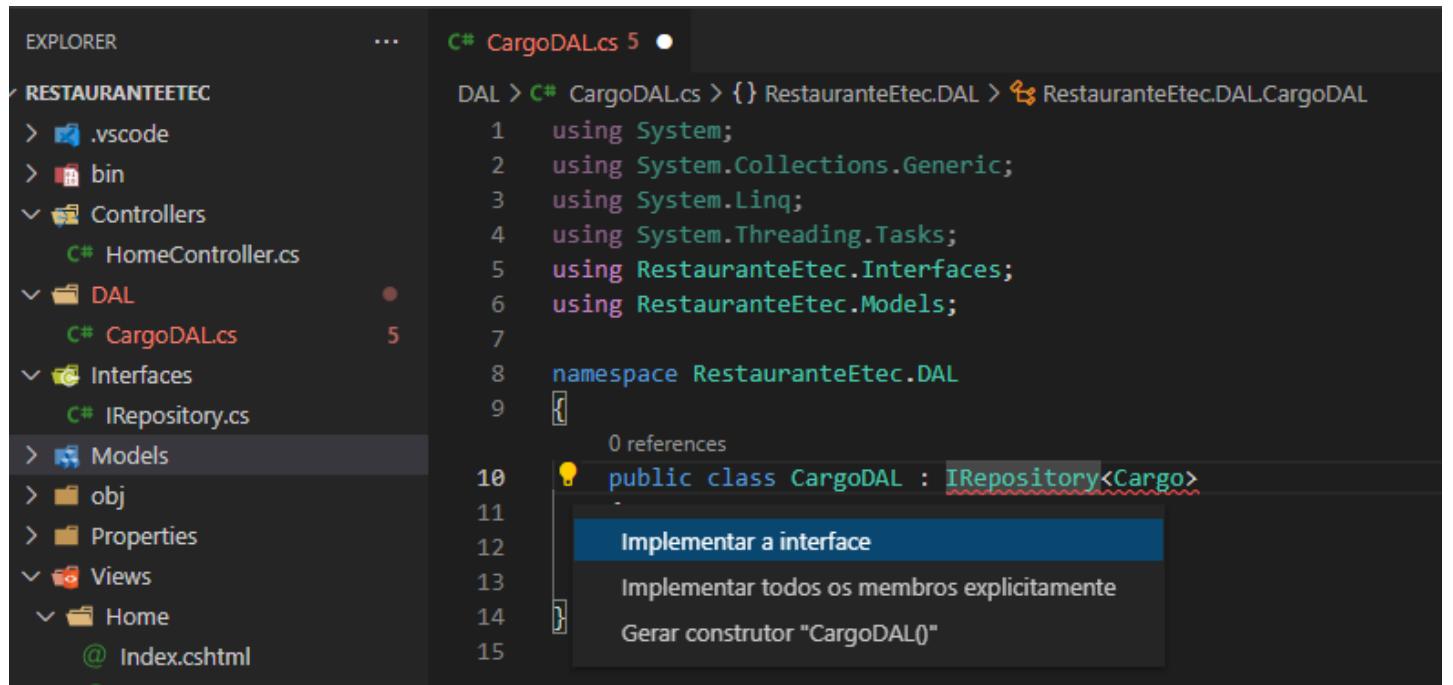


```
C# CargoDAL.cs 5 ●  
DAL > C# CargoDAL.cs > ...  
1   using System;  
2   using System.Collections.Generic;  
3   using System.Linq;  
4   using System.Threading.Tasks;  
5   using RestauranteEtec.Interfaces;  
6   using RestauranteEtec.Models;  
7  
8   namespace RestauranteEtec.DAL  
9   {  
10      0 references  
11      public class CargoDAL : IRepository<Cargo>  
12      {  
13      }  
14  }
```

Agora temos outro problema, no caso, a ferramenta nos avisa que a classe **CargoDAL**, não está implementando os métodos definidos na Interface **IRepository**, por isso fica apontando erro. O que quero dizer

com isso é que não programamos os métodos que foram definidos na interface, não existe ainda na **CargoDAL** os métodos **Add**, **GetAll**, **GetById**, **Update** e **Delete**.

Clicando com o botão direito sobre o erro e abrindo a **Lâmpada** que é exibida você terá uma opção para resolver esse problema, conforme a figura abaixo:



Clicando em **Implementar a interface**, o **Code** gera os métodos da interface que precisamos programar:

```
C# CargoDAL.cs ×
DAL > C# CargoDAL.cs > {} RestauranteEtec.DAL > RestauranteEtec.DAL.CargoDAL
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using RestauranteEtec.Interfaces;
6  using RestauranteEtec.Models;
7
8  namespace RestauranteEtec.DAL
9  {
10     public class CargoDAL : IRepository<Cargo>
11     {
12         public void Add(Cargo model)
13         {
14             throw new NotImplementedException();
15         }
16
17         public void Delete(int? id)
18         {
19             throw new NotImplementedException();
20         }
21
22         public IEnumerable<Cargo> GetAll()
23         {
24             throw new NotImplementedException();
25         }
26
27         public Cargo GetById(int? id)
28         {
29             throw new NotImplementedException();
30         }
31
32         public void Update(Cargo model)
33         {
34             throw new NotImplementedException();
35         }
36     }
37 }
```

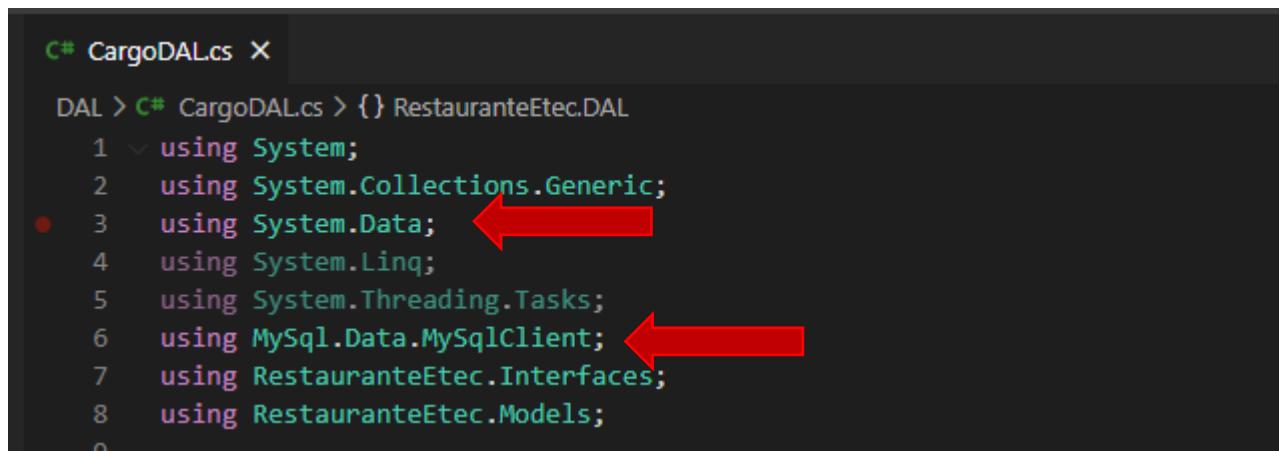
Agora precisamos programar todos os métodos substituindo o código “**throw new ....**” gerado automaticamente, pois ele é apenas para estourar uma exceção pois o método não está programado. Vamos começar pela inclusão de uma **string**, que será utilizada pelos objetos de conexão para identificar nosso banco de dados.

Adicione logo abaixo da abertura da classe a seguinte **string**:

```
0 references
10 public class CargoDAL : IRepository<Cargo>
11 {
12     0 references
13     string connectionString = @"Server=localhost;port=3306;database=RestauranteEtec;uid=root;pwd=root";
```

Aqui temos o nome do servidor “**localhost**”, a porta de conexão do **MySQL “3306”**, o nome do banco de dados “**RestauranteEtec**”, nome do usuário “**root**” e a senha que também é “**root**”.

Antes de programarmos os demais métodos, acrescentes mais dois pacotes ao início do código (linhas 3 e 6):



```
C# CargoDAL.cs X
DAL > C# CargoDAL.cs > {} RestauranteEtec.DAL
1  using System;
2  using System.Collections.Generic;
3  using System.Data; ← Red arrow here
4  using System.Linq;
5  using System.Threading.Tasks;
6  using MySql.Data.MySqlClient; ← Red arrow here
7  using RestauranteEtec.Interfaces;
8  using RestauranteEtec.Models;
9
```

Agora sim, vamos aos códigos, programe conforme as figuras abaixo:

## Função Add

```
0 references
16 public void Add(Cargo model)
17 {
18     using (MySqlConnection conexao = new MySqlConnection(connectionString))
19     {
20         var sql = "insert into Cargo(Nome) values (@Nome)";
21         MySqlCommand comando = new MySqlCommand(sql, conexao);
22         comando.CommandType = CommandType.Text;
23         comando.Parameters.AddWithValue("@Nome", model.Nome);
24
25         conexao.Open();
26         comando.ExecuteNonQuery();
27         conexao.Close();
28     }
29 }
```

## Função Delete

```
0 references
31     public void Delete(int? id)
32     {
33         using (MySqlConnection conexao = new MySqlConnection(connectionString))
34         {
35             var sql = "delete from Cargo where Id = @Id";
36             MySqlCommand comando = new MySqlCommand(sql, conexao);
37             comando.CommandType = CommandType.Text;
38
39             comando.Parameters.AddWithValue("@Id", id);
40
41             conexao.Open();
42             comando.ExecuteNonQuery();
43             conexao.Close();
44         }
45     }
```

## Função GetAll

```
0 references
47     public IEnumerable<Cargo> GetAll()
48     {
49         List<Cargo> cargos = new List<Cargo>();
50         using (MySqlConnection conexao = new MySqlConnection(connectionString))
51         {
52             MySqlCommand comando = new MySqlCommand("select * from Cargo", conexao);
53             comando.CommandType = CommandType.Text;
54             conexao.Open();
55             MySqlDataReader leitor = comando.ExecuteReader();
56             while (leitor.Read())
57             {
58                 Cargo cargo = new Cargo()
59                 {
60                     Id = Convert.ToInt32(leitor["Id"]),
61                     Nome = leitor["Nome"].ToString()
62                 };
63                 cargos.Add(cargo);
64             }
65             conexao.Close();
66         }
67         return cargos;
68     }
```

## Função GetById

```
0 references
70  v    public Cargo GetById(int? id)
71    {
72      Cargo cargo = new Cargo();
73      using (MySqlConnection conexao = new MySqlConnection(connectionString))
74      {
75        MySqlCommand comando = new MySqlCommand("select * from Cargo where Id = @Id", conexao);
76        comando.CommandType = CommandType.Text;
77        comando.Parameters.AddWithValue("@Id", id);
78
79        conexao.Open();
80        MySqlDataReader leitor = comando.ExecuteReader();
81        leitor.Read();
82        if (!leitor.HasRows)
83        {
84          conexao.Close();
85          return null;
86        }
87        cargo.Id = Convert.ToInt32(leitor["Id"]);
88        cargo.Nome = leitor["Nome"].ToString();
89        conexao.Close();
90      }
91      return cargo;
92    }
93  }
```

## Função Update

```
0 references
94    public void Update(Cargo model)
95    {
96      using (MySqlConnection conexao = new MySqlConnection(connectionString))
97      {
98        var sql = "update Cargo set" +
99                    " Nome = @Nome" +
100                   " where Id = @Id";
101        MySqlCommand comando = new MySqlCommand(sql, conexao);
102        comando.CommandType = CommandType.Text;
103
104        comando.Parameters.AddWithValue("@Id", model.Id);
105        comando.Parameters.AddWithValue("@Nome", model.Nome);
106
107        conexao.Open();
108        comando.ExecuteNonQuery();
109        conexao.Close();
110      }
111    }
```

## Funções Add, Delete e Update:

O que nosso código faz, é utilizar um **MySqlConnection**, para criar uma conexão com o banco de dados através da **connectionString**, e posteriormente, através de **MySQLCommand**, especificamos o comando **SQL** que queremos executar. O **Parameters** permite adicionar os parâmetros que estão sendo usados na consulta **SQL**, o comando **Open**, abre a conexão, o comando **ExecuteNonQuery**, executa uma consulta sem resultados e o **Close** fecha a conexão. São Funções que não tem retorno, por isso, são executadas conforme a explicação acima, com o comando **ExecuteNonQuery**.

## Funções GetAll e GetById:

Estas funções são semelhantes na parte da conexão e criação dos comandos **SQL**, porém aqui temos funções com retorno de dados (**select**), dessa forma precisamos utilizar um objeto **MySqlDataReader**, para fazer a leitura dos dados retornados pelo comando. No momento da leitura dos dados, precisamos fazer as conversões de tipos de acordo com as propriedades das classes.

## Tabela de conversões:

<b>int</b>	Convert.ToInt32(leitor[“nomeCampo”])
<b>double</b>	Convert.ToDouble(leitor[“nomeCampo”])
<b>decimal</b>	Convert.ToDecimal(leitor[“nomeCampo”])
<b>byte</b>	Convert.ToByte(leitor[“nomeCampo”])
<b>string</b>	Leitor[“nomeCampo”].ToString()
<b>bool</b>	<b>Get:</b> Convert.ToInt32(leitor[“nomeCampo”]) == 1 <b>Add:</b> true ou false ou valor da propriedade <b>Update:</b> model.Ativo ? 1 : 0
<b>DateTime</b>	<b>Get:</b> Convert.ToDateTime(leitor[“nomeCampo”]) <b>Add:</b> model.Propriedade.ToString(“yyyy-MM-dd H:mm”)

Na Tabela de Conversões, temos uma particularidade, propriedades do tipo **bool**, no banco de dados **MySQL**, essas propriedades são campos **tinyint(1)**, portanto quando fazemos a leitura temos um valor 0 ou 1, que precisa ser convertido em operador lógico (verdadeiro ou falso). Para os casos de comando **select (Get e GetById)** usamos uma verificação simples, é igual a 1, verdadeiro, caso contrário é falso. Quanto for um **insert** (método **Add**), estamos cadastrando essa informação então é verdadeiro.

Agora é com você para criar as classes **DAL** abaixo:

- **BlogDAL** ligado ao **Model Blog**
- **CategoriaDAL** ligado ao **Model Categoria**
- **ContatoDAL** ligado ao **Model Contato**
- **RelatoDAL** ligado ao **Model Relato**
- **ReservaDAL** ligado ao **Model Reserva**

As classes de acesso **DAL**, de **Funcionário** e **Produto**, têm uma particularidade, que é exatamente a mesma. Seus objetos são dependentes de outros objetos, no caso o **Funcionário depende do Cargo**; e o **Produto depende da Categoria**. Essa dependência fica explícita nos métodos **GetAll** e **.GetById**. Nos próximos prints, veremos como trabalhar essa dependência, sendo que o restante do código é quase idêntico as demais classes.

Vamos criar agora o arquivo e classe **FuncionarioDAL**, repetindo os mesmos procedimentos das demais classes DAL com os códigos conforme as figuras a seguir:

```
C# FuncionarioDAL.cs ●  
DAL > C# FuncionarioDAL.cs > ...  
1  using System;  
2  using System.Collections.Generic;  
3  using System.Data;  
4  using System.Linq;  
5  using System.Threading.Tasks;  
6  using MySql.Data.MySqlClient;  
7  using RestauranteEtec.Interfaces;  
8  using RestauranteEtec.Models;  
9  
10 namespace RestauranteEtec.DAL  
11 {  
12     0 references  
13     public class FuncionarioDAL: IRepository<Funcionario>  
14     {  
15         0 references  
16         string connectionString = @"Server=localhost;port=3306;database=RestauranteEtec;uid=root;pwd=root";  
17  
18         0 references  
19         public void Add(Funcionario model)  
20         {  
21             throw new NotImplementedException();  
22         }  
23  
24         0 references  
25         public void Delete(int? id)  
26         {  
27             throw new NotImplementedException();  
28         }  
29  
30         0 references  
31         public IEnumerable<Funcionario> GetAll()  
32         {  
33             throw new NotImplementedException();  
34         }  
35  
36         0 references  
37         public Funcionario GetById(int? id)  
38         {  
39             throw new NotImplementedException();  
40         }  
41     }  
42 }
```

Vamos agora a codificação de cada função.

## Função Add:

```
0 references
16  ~ public void Add(Funcionario model)
17    {
18      using (MySqlConnection conexao = new MySqlConnection(connectionString))
19      {
20          var sql = "insert into Funcionario(Nome, Descricao, Foto, CargoId, ExibirHome, OrdemExibicao, Ativo)" +
21          " values (@Nome, @Descricao, @Foto, @CargoId, @ExibirHome, @OrdemExibicao, @Ativo)";
22          MySqlCommand comando = new MySqlCommand(sql, conexao);
23          comando.CommandType = CommandType.Text;
24          comando.Parameters.AddWithValue("@Nome", model.Nome);
25          comando.Parameters.AddWithValue("@Descricao", model.Descricao);
26          comando.Parameters.AddWithValue("@Foto", model.Foto);
27          comando.Parameters.AddWithValue("@CargoId", model.CargoId);
28          comando.Parameters.AddWithValue("@ExibirHome", model.ExibirHome);
29          comando.Parameters.AddWithValue("@OrdemExibicao", model.OrdemExibicao);
30          comando.Parameters.AddWithValue("@Ativo", true);
31
32          conexao.Open();
33          comando.ExecuteNonQuery();
34          conexao.Close();
35      }
36  }
```

## Funções Delete:

```
0 references
38  ~ public void Delete(int? id)
39  {
40      using (MySqlConnection conexao = new MySqlConnection(connectionString))
41      {
42          var sql = "delete from Funcionario where Id = @Id";
43          MySqlCommand comando = new MySqlCommand(sql, conexao);
44          comando.CommandType = CommandType.Text;
45          comando.Parameters.AddWithValue("@Id", id);
46
47          conexao.Open();
48          comando.ExecuteNonQuery();
49          conexao.Close();
50      }
51  }
```

## Função GetAll:

```
0 references
53    public IEnumerable<Funcionario> GetAll()
54    {
55        List<Funcionario> funcionarios = new List<Funcionario>();
56        CargoDAL cargo = new CargoDAL();
57        using (MySqlConnection conexao = new MySqlConnection(connectionString))
58        {
59            MySqlCommand comando = new MySqlCommand("select * from funcionario", conexao);
60            comando.CommandType = CommandType.Text;
61
62            conexao.Open();
63            MySqlDataReader leitor = comando.ExecuteReader();
64            while (leitor.Read())
65            {
66                Funcionario funcionario = new Funcionario()
67                {
68                    Id = Convert.ToInt32(leitor["Id"]),
69                    Nome = leitor["Nome"].ToString(),
70                    Descricao = leitor["Descricao"].ToString(),
71                    Foto = leitor["Foto"].ToString(),
72                    CargoId = Convert.ToInt32(leitor["CargoId"].ToString()),
73                    ExibirHome = Convert.ToByte(leitor["ExibirHome"]) == 1,
74                    OrdemExibicao = Convert.ToByte(leitor["OrdemExibicao"]),
75                    Ativo = Convert.ToInt32(leitor["Ativo"]) == 1
76                };
77                funcionario.Cargo = cargo.GetById(funcionario.CargoId);
78
79                funcionarios.Add(funcionario);
80            }
81            conexao.Close();
82        }
83        return funcionarios;
84    }
85}
```

Repare que na função **GetAll**, utilizamos um objeto **cargo**, para preencher a propriedade **funcionario.Cargo**, através do método  **GetById** do **Cargo**. Isso irá tornar possível, quando formos exibir os dados dos funcionários, exibir também informações do cargo que este funcionário ocupa.

## Função GetById:

```
2 references
86     public Funcionario GetById(int? id)
87     {
88         CargoDAL cargo = new CargoDAL();
89         Funcionario funcionario = new Funcionario();
90         using (MySqlConnection conexao = new MySqlConnection(connectionString))
91         {
92             MySqlCommand comando = new MySqlCommand("select * from Funcionario where Id = @Id", conexao);
93             comando.CommandType = CommandType.Text;
94             comando.Parameters.AddWithValue("@Id", id);
95
96             conexao.Open();
97             MySqlDataReader leitor = comando.ExecuteReader();
98             leitor.Read();
99             if (!leitor.HasRows)
100             {
101                 conexao.Close();
102                 return null;
103             }
104             funcionario.Id = Convert.ToInt32(leitor["Id"]);
105             funcionario.Nome = leitor["Nome"].ToString();
106             funcionario.Descricao = leitor["Descricao"].ToString();
107             funcionario.Foto = leitor["Foto"].ToString();
108             funcionario.CargoId = Convert.ToInt32(leitor["CargoId"].ToString());
109             funcionario.ExibirHome = Convert.ToInt32(leitor["ExibirHome"]) == 1;
110             funcionario.OrdemExibicao = Convert.ToByte(leitor["OrdemExibicao"]);
111             funcionario.Ativo = Convert.ToInt32(leitor["Ativo"]) == 1;
112             funcionario.Cargo = cargo.GetById(funcionario.CargoId);
113             conexao.Close();
114         }
115         return funcionario;
116     }
117 }
```

Repare que na função **GetById**, utilizamos um objeto **cargo**, para preencher a propriedade **funcionario.Cargo**, através do método **.GetById** do **Cargo**. Isso irá tornar possível, quando formos exibir os dados do funcionário, exibir também informações do cargo que este funcionário ocupa.

## Função Update:

```
0 references
118     public void Update(Funcionario model)
119     {
120         using (MySqlConnection conexao = new MySqlConnection(connectionString))
121         {
122             var sql = "update Funcionario set" +
123                     " Nome = @Nome," +
124                     " Descricao = @Descricao," +
125                     " Foto = @Foto," +
126                     " CargoId = @CargoId," +
127                     " ExibirHome = @ExibirHome," +
128                     " OrdemExibicao = @OrdemExibicao," +
129                     " Ativo = @Ativo" +
130                     " where Id = @Id";
131             MySqlCommand comando = new MySqlCommand(sql, conexao);
132             comando.CommandType = CommandType.Text;
133
134             comando.Parameters.AddWithValue("@Id", model.Id);
135             comando.Parameters.AddWithValue("@Nome", model.Nome);
136             comando.Parameters.AddWithValue("@Descricao", model.Descricao);
137             comando.Parameters.AddWithValue("@Foto", model.Foto);
138             comando.Parameters.AddWithValue("@CargoId", model.CargoId);
139             comando.Parameters.AddWithValue("@ExibirHome", model.ExibirHome);
140             comando.Parameters.AddWithValue("@OrdemExibicao", model.OrdemExibicao);
141             comando.Parameters.AddWithValue("@Ativo", model.Ativo);
142
143             conexao.Open();
144             comando.ExecuteNonQuery();
145             conexao.Close();
146         }
147     }
```

Agora podemos fazer uma alteração no código do **HomeController**, para utilizar nossa classe de acesso aos dados dos funcionários e assim, exibir os dados diretamente do banco de dados.

Vamos começar por adicionar os **namespaces DAL e Models**, no **HomeController**.

```
C# HomeController.cs X
Controllers > C# HomeController.cs > {} RestauranteEtec.Controllers > RestauranteEtec.Controllers.HomeController
1  using System;
2  using System.Collections.Generic;
3  using System.Diagnostics;
4  using System.Linq;
5  using System.Threading.Tasks;
6  using Microsoft.AspNetCore.Mvc;
7  using Microsoft.Extensions.Logging;
8  using RestauranteEtec.Models;
9  using RestauranteEtec.DAL;
```



Modifique o código da ação **Index**, conforme a imagem abaixo, para criar uma lista de funcionários ativos, que estão marcados para exibição na home, em ordem de numeração conforme o campo **OrdemExibicao**.

```
22     0 references
23     public IActionResult Index()
24     {
25         var funcionarios = new FuncionarioDAL();
26         var chefes = funcionarios.GetAll().Where(f => f.Ativo && f.ExibirHome).OrderBy(f => f.OrdemExibicao).ToList();
27         ViewData["Chefes"] = chefes;
28     }

```

O objeto **funcionarios**, nos permite executar os acessos e alterações de dados programados nos métodos (**Add**, **Delete**, **GetAll**, **GetById** e **Update**).

O objeto **chefes** então vai receber o resultado do método **GetAll** do **funcionarios**, com alguns filtros extras, adicionados através da biblioteca **Linq**, um **Where**, que nos permite filtrar os funcionários que estão ativos (**Ativo** é booleano então só precisamos especificar o campo para pegar os verdadeiros) e (**&&**) que **ExibirHome** também é verdadeiro; em seguida o **OrderBy**, permite fazer uma ordenação desse resultado, pelo campo **OrdemExibicao**; por fim o **ToList()**, para transformar o resultado em uma lista de objetos do tipo **funcionario**.

Por fim, vamos alterar a **View Index.cshtml**, localizada em **Views\Home**, substituindo as **4 divs de funcionários** (linhas entre 554 e 625, aproximadamente) por um **foreach**. Para facilitar segue abaixo a seção inteira de Chefes alterada (procure no seu Index por “**Our Chefs**” e altere todo o código da seção:

```
567 <section class="ftco-section bg-light">
568     <div class="container">
569         <div class="row justify-content-center mb-5 pb-2">
570             <div class="col-md-7 text-center heading-section ftco-animate">
571                 <span class="subheading">Chefes</span>
572                 <h2 class="mb-4">Nossos Mestres</h2>
573             </div>
574         </div>
575         <div class="row">
576             @foreach (var chefe in ViewBag.Chefes)
577             {
578                 <div class="col-md-6 col-lg-3 ftco-animate">
579                     <div class="staff">
580                         <div class="img" style="background-image: url(@Url.Content(chefe.Foto));"></div>
581                         <div class="text px-4 pt-2">
582                             <h3>@chefe.Nome</h3>
583                             <span class="position mb-2">@chefe.Cargo.Nome</span>
584                             <div class="faded">
585                                 <p>@chefe.Descricao</p>
586                             </div>
587                         </div>
588                     </div>
589                 </div>
590             }
591         </div>
592     </div>
593 </section>
```

Antes de executarmos nosso projeto, precisamos primeiro criar nosso banco de dados. Baixe o **script** do banco **MySQL** [aqui](#). Abra o **SGBD** de sua preferência, execute o script e agora estamos prontos para o comando de execução do projeto.

```
dotnet run
```

Ctrl + Clique do Mouse – No link indicado pela seta da imagem abaixo

```
PS C:\Users\gallo\Desktop\RestauranteEtec> dotnet run
Compilando...
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000 ←
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\gallo\Desktop\RestauranteEtec
```

Execute e verifique o resultado:

The screenshot shows a web browser window with the URL <https://localhost:44349>. The page is titled "RestauranteEtec - Home". The main content area is titled "Chefs" and "Nossos Mestres". There are four cards, each featuring a chef's photo, name, title, and a short bio. The bio for all four chefs is identical: "Eu sou um workaholic ambicioso, mas fora isso, uma pessoa muito simples." Each card ends with a small red ribbon icon.

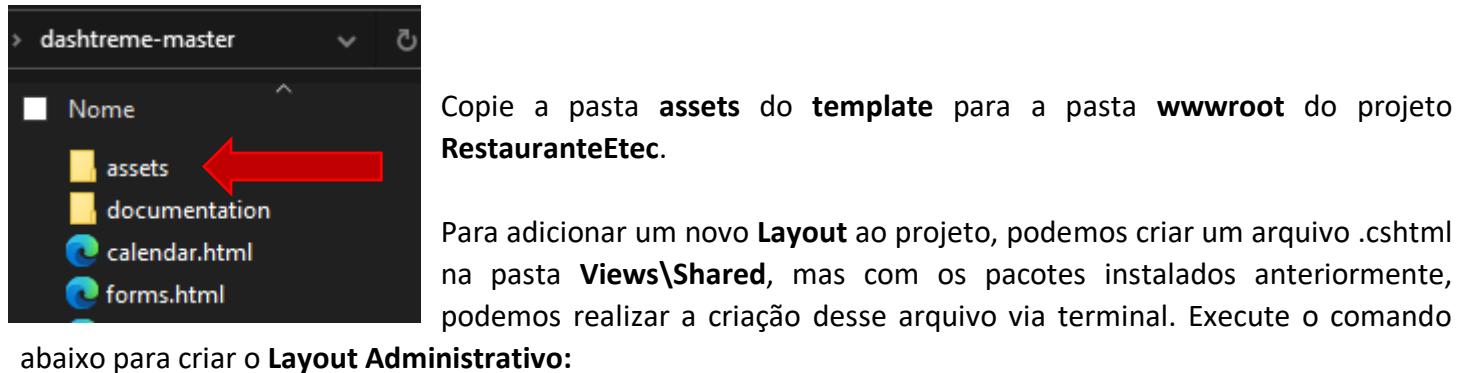
Chef	Title	Bio
João Gustavo	CEO, Co Fundador	Eu sou um workaholic ambicioso, mas fora isso, uma pessoa muito simples.
Michelle Fraulen	Chefe de Cozinha	Eu sou um workaholic ambicioso, mas fora isso, uma pessoa muito simples.
Alfred Smith	Cozinheiro Chefe	Eu sou um workaholic ambicioso, mas fora isso, uma pessoa muito simples.
Antonio Santibanez	Cozinheiro Chefe	Eu sou um workaholic ambicioso, mas fora isso, uma pessoa muito simples.

O processo de deixar as outras páginas dinâmicas é o mesmo que o feito na parte de funcionários, e pode ser repetido para o blog e relatos.

## CRIANDO UM LAYOUT PARA A ÁREA ADMINISTRATIVA

Agora vamos repetir o processo de criação de um layout para criarmos o design que será usado na área administrativa e posteriormente usado para as páginas de **CRUD** (Acrônimo usado para Create, Read Update e Delete, as ações básicas em tabelas de banco de dados: criar, ler, atualizar e excluir).

Faça o download o arquivo **dashtreme-master.zip** disponibilizado na aba **Arquivos** do canal **Geral**, com link [online aqui](#). Descompacte o arquivo e siga os passos abaixo:



```
dotnet-aspnet-codegenerator view _LayoutAdmin Empty -outDir  
"Views/Shared"
```

Agora abra o arquivo **index.html** do **template** em outro editor e copie as linhas de 1 a 175 (inclusive) e cole o conteúdo **sobrescrevendo o código atual do arquivo \_LayoutAdmin**.

Digite logo abaixo do código copiado: **@RenderBody()**

Agora volte ao arquivo **index.html** do **template** e copie as linhas de 443 a 517 e cole no arquivo **\_LayoutAdmin.cshtml**, abaixo do **@RenderBody()**

Desça até a tag **</body>** a inclua a seguinte linha antes da tag: **@await RenderSectionAsync("Scripts", required: false)**

Altere a linha 2 para: **<html lang="pt-br">**

Altere a linha 9 para: **<title>RestauranteEtec - @ViewData["Title"]</title>**

Agora assim como fizemos no primeiro layout, precisamos encontrar todas as referências a pasta **assets** e acrescente antes um **"~/"** (til e barra).

Além disso já vamos fazer algumas alterações no menu superior e lateral, para isso deixei o seu código igual ao código abaixo:

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />
    <meta name="description" content="" />
    <meta name="author" content="" />
    <title>RestauranteEtec - @ ViewData["Title"]</title>
    <!-- loader-->
    <link href="~/assets/css/pace.min.css" rel="stylesheet" />
    <script src="~/assets/js/pace.min.js"></script>
    <!--favicon-->
    <link rel="icon" href="~/assets/images/favicon.ico" type="image/x-icon">
    <!-- Vector CSS -->
    <link href="~/assets/plugins/vectormap/jquery-jvectormap-2.0.2.css" rel="stylesheet" />
    <!-- simplebar CSS-->
    <link href="~/assets/plugins/simplebar/css/simplebar.css" rel="stylesheet" />
    <!-- Bootstrap core CSS-->
    <link href="~/assets/css/bootstrap.min.css" rel="stylesheet" />
    <!-- animate CSS-->
    <link href="~/assets/css/animate.css" rel="stylesheet" type="text/css" />
    <!-- Icons CSS-->
    <link href="~/assets/css/icons.css" rel="stylesheet" type="text/css" />
    <!-- Sidebar CSS-->
    <link href="~/assets/css/sidebar-menu.css" rel="stylesheet" />
    <!-- Custom Style-->
    <link href="~/assets/css/app-style.css" rel="stylesheet" />

</head>

<body class="bg-theme bg-theme2">

    <!-- Start wrapper-->
    <div id="wrapper">

        <!--Start sidebar-wrapper-->
        <div id="sidebar-wrapper" data-simplebar="" data-simplebar-auto-hide="true">
            <div class="brand-logo">
                <a asp-action="Index" asp-controller="Admin">
                    <h5 class="logo-text">Restaurante Etec</h5>
                </a>
            </div>
            <ul class="sidebar-menu do-nicescrol">
                <li>
                    <a asp-action="Index" asp-controller="Admin">
                        <i class="zmdi zmdi-view-dashboard"></i> <span>Dashboard</span>
                    </a>
                </li>

                <li class="sidebar-header text-info">SOCIAL</li>
                <li>
                    <a asp-action="Index" asp-controller="Blog">
                        <i class="zmdi zmdi-blogger"></i> <span>Blog</span>
                    </a>
                </li>
            
```

```

        </a>
    </li>
    <li>
        <a asp-action="Index" asp-controller="Contatos">
            <i class="zmdi zmdi-email"></i> <span>Contatos</span>
            <small class="badge float-right badge-light">Novo</small>
        </a>
    </li>
    <li>
        <a asp-action="Index" asp-controller="Relatos">
            <i class="zmdi zmdi-assignment"></i> <span>Relatos</span>
        </a>
    </li>
    <li>
        <a asp-action="Index" asp-controller="Reservas">
            <i class="zmdi zmdi-calendar-check"></i> <span>Reservas</span>
            <small class="badge float-right badge-light">Novo</small>
        </a>
    </li>

<li class="sidebar-header text-success">MENU</li>
<li>
    <a asp-action="Index" asp-controller="Categorias">
        <i class="zmdi zmdi-filter-list"></i> <span>Categorias</span>
    </a>
</li>
<li>
    <a asp-action="Index" asp-controller="Produtos">
        <i class="zmdi zmdi-cutlery"></i> <span>Produtos</span>
    </a>
</li>

<li class="sidebar-header text-danger">RECURSOS HUMANOS</li>
<li>
    <a asp-action="Index" asp-controller="Cargos">
        <i class="zmdi zmdi-accounts-list"></i> <span>Cargos</span>
    </a>
</li>
<li>
    <a asp-action="Index" asp-controller="Funcionarios">
        <i class="zmdi zmdi-account-circle"></i> <span>Funcionários</span>
    </a>
</li>
</ul>
</div>
<!--End sidebar-wrapper-->
<!--Start topbar header-->
<header class="topbar-nav">
    <nav class="navbar navbar-expand fixed-top">
        <ul class="navbar-nav mr-auto align-items-center">
            <li class="nav-item">
                <a class="nav-link toggle-menu" href="javascript:void();">
                    <i class="icon-menu menu-icon"></i>
                </a>
            </li>
        <ul>

```

```

        </li>
    </ul>

    <ul class="navbar-nav align-items-center right-nav-link">
        <li class="nav-item dropdown-lg">
            <a class="nav-link dropdown-toggle dropdown-toggle-nocaret waves-effect" data-toggle="dropdown" href="javascript:void();">
                <i class="fa fa-envelope-open-o"></i>
            </a>
        </li>
        <li class="nav-item dropdown-lg">
            <a class="nav-link dropdown-toggle dropdown-toggle-nocaret waves-effect" data-toggle="dropdown" href="javascript:void();">
                <i class="fa fa-bell-o"></i>
            </a>
        </li>
        <li class="nav-item">
            <a class="nav-link dropdown-toggle dropdown-toggle-nocaret" data-toggle="dropdown" href="#">
                <span class="user-profile"></span>
                </a>
            <ul class="dropdown-menu dropdown-menu-right">
                <li class="dropdown-item user-details">
                    <a href="javaScript:void();">
                        <div class="media">
                            <div class="avatar"></div>
                            <div class="media-body">
                                <h6 class="mt-2 user-title">Sarajhon Mccoy</h6>
                                <p class="user-subtitle">mccoy@example.com</p>
                            </div>
                        </div>
                    </a>
                </li>
                <li class="dropdown-divider"></li>
                <li class="dropdown-item"><i class="icon-envelope mr-2"></i> Inbox</li>
                <li class="dropdown-divider"></li>
                <li class="dropdown-item"><i class="icon-wallet mr-2"></i> Account</li>
                <li class="dropdown-divider"></li>
                <li class="dropdown-item"><i class="icon-settings mr-2"></i> Setting</li>
                <li class="dropdown-divider"></li>
                <li class="dropdown-item"><i class="icon-power mr-2"></i> Logout</li>
            </ul>
        </li>
    </ul>
</nav>
</header>
<!--End topbar header-->
```

```
<div class="clearfix"></div>

<div class="content-wrapper">
    <div class="container-fluid">

        @RenderBody()
        <!--End Dashboard Content-->
        <!--start overlay-->
        <div class="overlay toggle-menu"></div>
        <!--end overlay-->

    </div>
    <!-- End container-fluid-->

</div><!--End content-wrapper-->
<!--Start Back To Top Button-->
<a href="javaScript:void();" class="back-to-top"><i class="fa fa-angle-double-up"></i> </a>
<!--End Back To Top Button-->
<!--Start footer-->
<footer class="footer">
    <div class="container">
        <div class="text-center">
            Copyright © @DateTime.Now.Year Restaurante Etec
        </div>
    </div>
</footer>
<!--End footer-->
<!--start color switcher-->
<div class="right-sidebar">
    <div class="switcher-icon">
        <i class="zmdi zmdi-settings zmdi-hc-spin"></i>
    </div>
    <div class="right-sidebar-content">

        <p class="mb-0">Gaussian Texture</p>
        <hr>

        <ul class="switcher">
            <li id="theme1"></li>
            <li id="theme2"></li>
            <li id="theme3"></li>
            <li id="theme4"></li>
            <li id="theme5"></li>
            <li id="theme6"></li>
        </ul>

        <p class="mb-0">Gradient Background</p>
        <hr>

        <ul class="switcher">
            <li id="theme7"></li>
            <li id="theme8"></li>
        </ul>
    </div>
</div>
```

```
<li id="theme9"></li>
<li id="theme10"></li>
<li id="theme11"></li>
<li id="theme12"></li>
<li id="theme13"></li>
<li id="theme14"></li>
<li id="theme15"></li>
</ul>

</div>
</div>
<!--end color switcher-->

</div><!--End wrapper-->
<!-- Bootstrap core JavaScript-->
<script src="~/assets/js/jquery.min.js"></script>
<script src="~/assets/js/popper.min.js"></script>
<script src="~/assets/js/bootstrap.min.js"></script>

<!-- simplebar js -->
<script src="~/assets/plugins/simplebar/js/simplebar.js"></script>
<!-- sidebar-menu js -->
<script src="~/assets/js/sidebar-menu.js"></script>
<!-- loader scripts -->
<script src="~/assets/js/jquery.loading-indicator.js"></script>
<!-- Custom scripts -->
<script src="~/assets/js/app-script.js"></script>
<!-- Chart js -->

<script src="~/assets/plugins/Chart.js/Chart.min.js"></script>

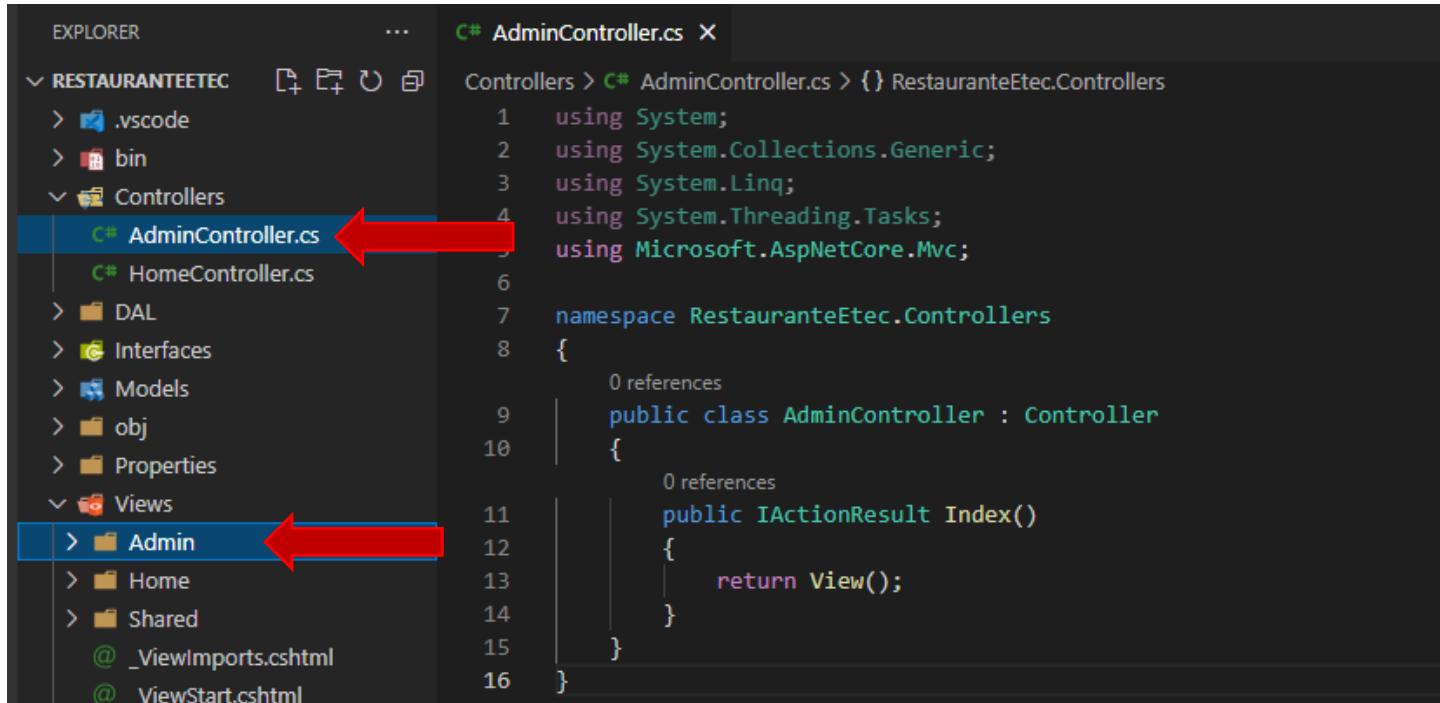
<!-- Index js -->
<script src="~/assets/js/index.js"></script>
@await RenderSectionAsync("Scripts", required: false)

</body>
</html>
```

Vamos agora adicionar um novo **Controlador**, que será responsável por apresentar ao usuário o acesso aos cadastros da área administrativa. Para isso vamos usar o gerador de códigos novamente, abra seu terminal e digite:

```
dotnet-aspnet-codegenerator controller -name AdminController -outDir Controllers
```

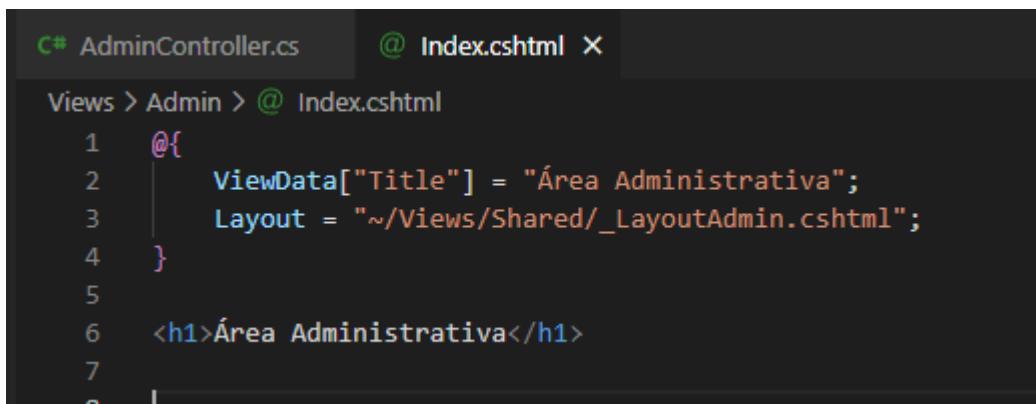
Comando executado, temos um novo **Controller**, conforme mostra na imagem abaixo. Agora precisamos adicionar uma pasta com o nome **Admin**, dentro da pasta **Views**, onde vamos criar a **Index.cshtml** do **AdminController**:



Para criar a **View**, conforme mencionado acima, execute o seguinte comando:

```
dotnet-aspnet-codegenerator view Index Empty -l
"~/Views/Shared/_LayoutAdmin.cshtml" -outDir "Views\Admin"
```

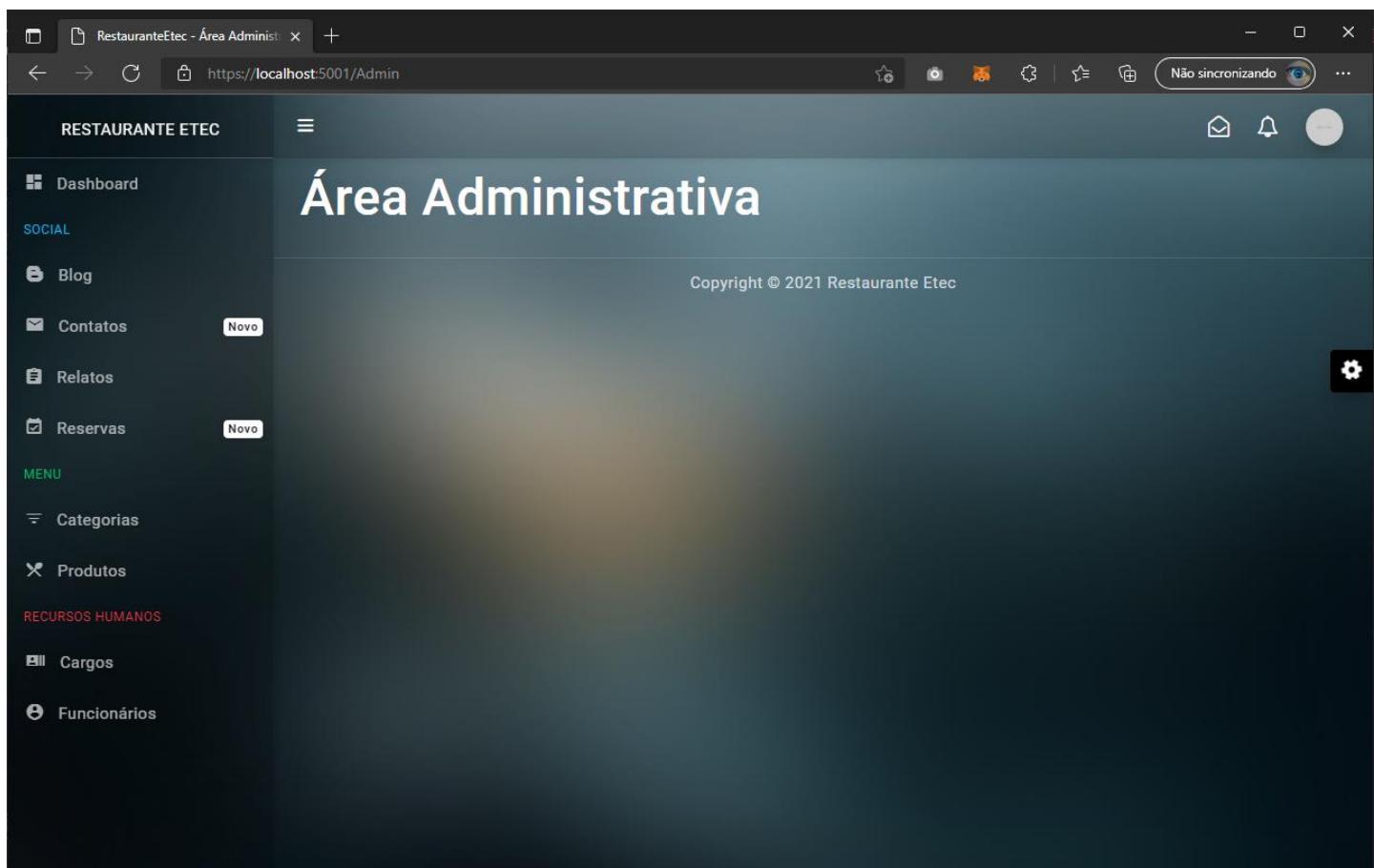
E pronto, nosso layout administrativo está pronto. Caso você queira, pode copiar as linhas que ficaram de fora da cópia anterior do arquivo **index.html** do **template** e colar no arquivo **Index.cshtml** do **Admin**, apenas para ver mais informações ao executar o **Admin**.



Execute o projeto, sem se esquecer de executar o **Xampp** ou do **MySQL**:



Na barra de navegação, acrescente ao final do endereço **/Admin**, e o resultado você confere na próxima imagem:



## CRIANDO O CRUD DE CARGOS

Vamos agora criar o **Controller** CargosController, que ficará responsável por receber e processar as requisições das páginas de exibição e gestão de dados, para realizar as operações de **CRUD** através de nossa camada **DAL**. Para isso abra seu terminal e execute o comando abaixo:

```
dotnet-aspnet-codegenerator controller -name CargosController -outDir Controllers -actions
```

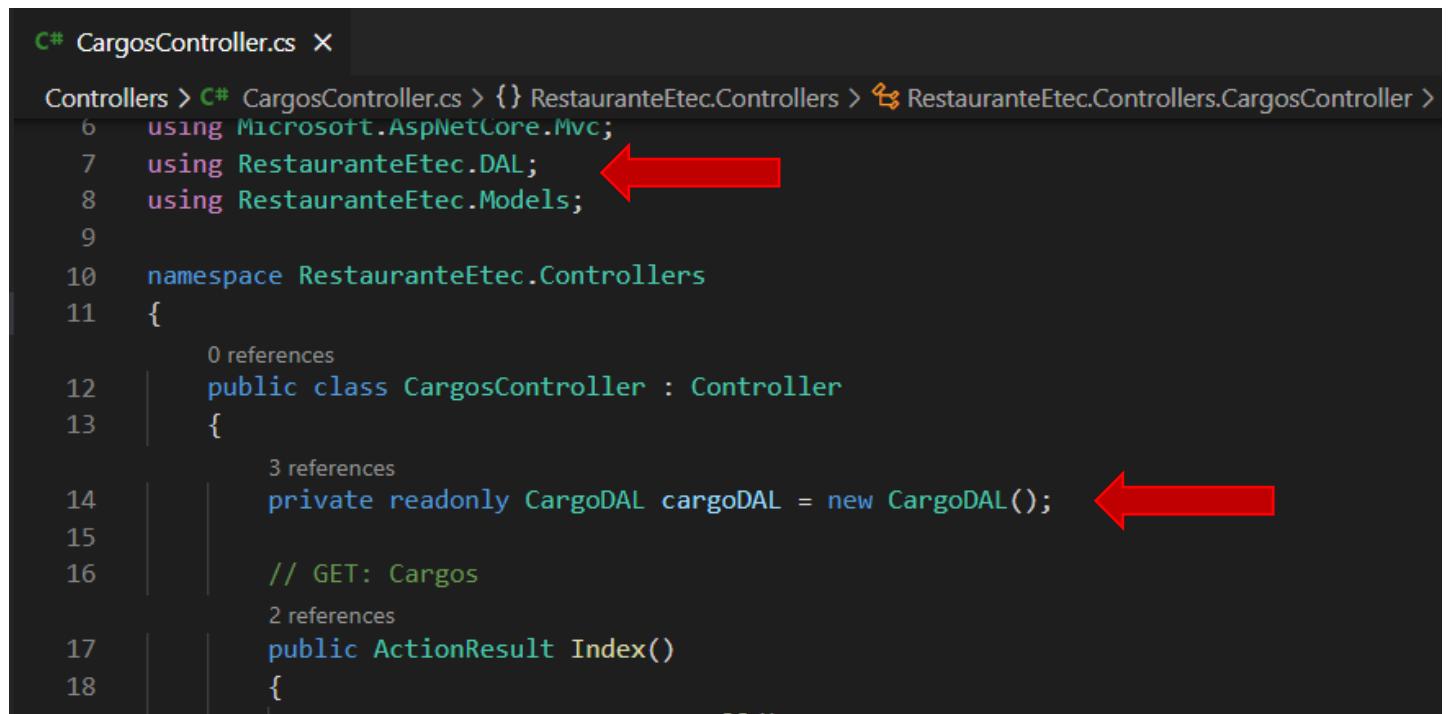
No comando assim, utilizamos a opção “**-name**” para especificar o nome do **Controller**; a opção “**-outDir**” para especificar a pasta onde o controlador será criado e por fim o “**-actions**” para definir que queremos pré-programado no controlador os métodos (**actions**) comuns aos **CRUDs**. Como resultado temos um **Controller** com várias ações para reutilizarmos:

```
C# CargosController.cs ×
Controllers > C# CargosController.cs > {} RestauranteEtec.Controllers > RestauranteEtec.Controllers.CargosController > Create(IFormCollection collection)
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using Microsoft.AspNetCore.Http;
6  using Microsoft.AspNetCore.Mvc;
7
8  namespace RestauranteEtec.Controllers
9  {
10     public class CargosController : Controller
11     {
12         // GET: Cargos
13         public ActionResult Index()
14         {
15             return View();
16         }
17
18         // GET: Cargos/Details/5
19         public ActionResult Details(int id)
20         {
21             return View();
22         }
23
24         // GET: Cargos/Create
25         public ActionResult Create()
26         {
27             return View();
28         }
29
30         // POST: Cargos/Create
31         [HttpPost]
32         [ValidateAntiForgeryToken]
33         public ActionResult Create(IFormCollection collection)
34         {
35             try
36             {

```

Assim como quando adicionamos nossa interface em uma classe, aqui também o **Controller** já é criado com as ações comuns de **CRUD**. Isso nos permite ganhar tempo de digitação, agora vamos editar as **actions** criadas, fazendo uso do **CargosDAL** e posteriormente vamos criar as **Views** de cada **Action**.

Vamos começar incluindo no começo do **Controller** um objeto do tipo **CargoDAL**:



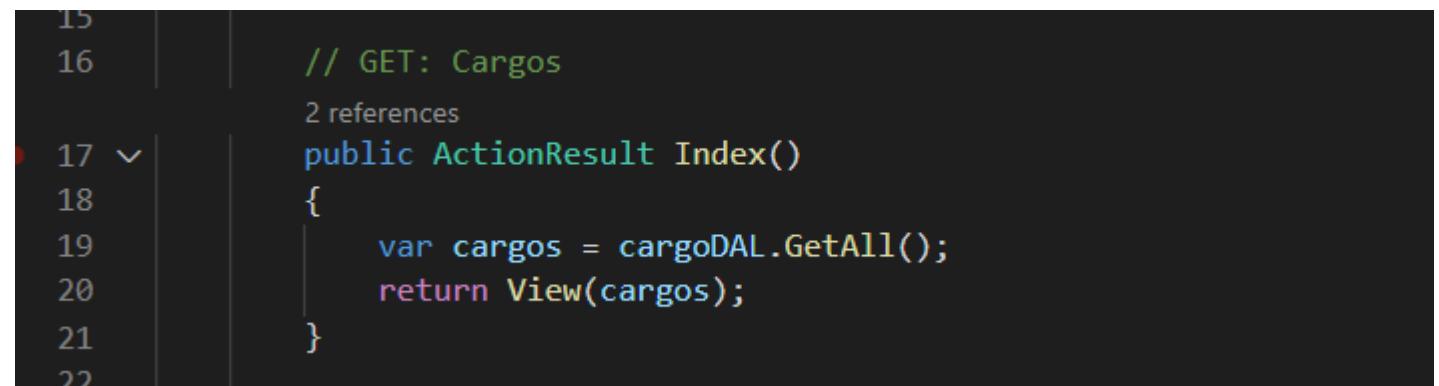
```
C# CargosController.cs X

Controllers > C# CargosController.cs > {} RestauranteEtec.Controllers > RestauranteEtec.Controllers.CargosController >
6   using Microsoft.AspNetCore.Mvc;
7   using RestauranteEtec.DAL; ←
8   using RestauranteEtec.Models;
9
10  namespace RestauranteEtec.Controllers
11  {
12      0 references
13      public class CargosController : Controller
14      {
15          3 references
16          // GET: Cargos
17          2 references
18          public ActionResult Index()
19          {
20              1 reference
21          }
22      }
23  }
```

Esse objeto será utilizado pelas **Actions**, para ter acesso aos métodos criados na **DAL** de recuperação e modificação de dados dos **cargos cadastrados no banco de dados**.

Vamos agora alterar os métodos conforme as imagens abaixo:

**Index:** Aqui utilizamos o método **GetAll**, programado na **CargoDAL** para retornar todos os cargos cadastrados a **View**:



```
15
16      // GET: Cargos
17      2 references
18      public ActionResult Index()
19      {
20          var cargos = cargoDAL.GetAll();
21          return View(cargos);
22      }
23  }
```

**Detail:** Aqui utilizamos o método **.GetById**, programado na **CargoDAL** para retornar apenas o cargo com **Id** igual ao **parâmetro id**, recebido pela **action**, e enviado a **View**. Caso o **id** for **nulo** ou o **Cargo** não for encontrado, o sistema retorna uma página de **Não Encontrado (NotFound)**. Lembre-se de colocar o "?" na linha 24, conforme a imagem a seguir:

```
22
23         // GET: Cargos/Details/5
24     public ActionResult Details(int? id)
25     {
26         if (id == null)
27             return NotFound();
28         var cargo = cargoDAL.GetById(id);
29         if (cargo == null)
30             return NotFound();
31         return View(cargo);
32     }
33 }
```

**Create (Get):** Este método tem a função de exibir ao usuário a **View** que irá permitir que um novo **Cargo**, possa ser cadastrado. Por isso, ele não precisa de acesso ao banco e sua função é simples.

```
33
34         // GET: Cargos/Create
35     public ActionResult Create()
36     {
37         return View();
38     }
39 }
```

**Create(Post):** Aqui podemos ver o polimorfismo em ação, o método **Create**, em sua forma **Post**, recebe os dados do novo **Cargo** inserido no formulário da **View**, e faz a chamada do método **Add** do **CargoDAL**, para enviar esses dados ao banco de dados, posteriormente retornando o usuário a página **Index** dos **Cargos**. Isso se os dados forem válidos (**ModelState.IsValid**).

```
40         // POST: Cargos/Create
41         [HttpPost]
42         [ValidateAntiForgeryToken]
43     public ActionResult Create([Bind]Cargo cargo)
44     {
45         if (!ModelState.IsValid)
46             return View(cargo);
47         try
48         {
49             cargoDAL.Add(cargo);
50             return RedirectToAction("Index");
51         }
52         catch
53         {
54             return View(cargo);
55         }
56     }
57 }
```

**Edit (Get):** Aqui utilizamos o método  **GetById**, programado na **CargoDAL** para retornar apenas o cargo com **Id** igual ao **parâmetro id**, recebido pela **action**, e enviado a **View**. Caso o **id** for **nulo** ou o **Cargo** não for encontrado, o sistema retorna uma página de **Não Encontrado (NotFound)**.

```
58     // GET: Cargos/Edit/5
      0 references
59     public ActionResult Edit(int? id)
60     {
61         if (id == null)
62             return NotFound();
63         var cargo = cargoDAL.GetById(id);
64         if (cargo == null)
65             return NotFound();
66         return View(cargo);
67     }
68 }
```

**Edit (Post):** Aqui podemos ver o polimorfismo em ação, o método **Edit**, em sua forma **Post**, recebe os dados de alteração do **Cargo** inserido no formulário da **View**, e faz a chamada do método **Update** do **CargoDAL**, para enviar esses dados ao banco de dados, posteriormente retornando o usuário a página **Index** dos **Cargos**. Isso se os dados forem válidos (**ModelState.IsValid**).

```
68
69     // POST: Cargos/Edit/5
70     [HttpPost]
71     [ValidateAntiForgeryToken]
      0 references
72     public ActionResult Edit(int id, [Bind]Cargo cargo)
73     {
74         if (id != cargo.Id)
75         {
76             return NotFound();
77         }
78         if (!ModelState.IsValid)
79             return View(cargo);
80         try
81         {
82             cargoDAL.Update(cargo);
83             return RedirectToAction("Index");
84         }
85         catch
86         {
87             return View(cargo);
88         }
89     }
90 }
```

**Delete (Get):** Este método tem a função de exibir ao usuário a **View** que irá exibir os dados do **Cargo** solicitando uma confirmação para exclusão. Sendo assim sua funcionalidade é semelhante ao **Edit**.

```
90
91         // GET: Cargos/Delete/5
92
93         public ActionResult Delete(int? id)
94     {
95         if (id == null)
96             return NotFound();
97         var cargo = cargoDAL.GetById(id);
98         if (cargo == null)
99             return NotFound();
100        return View(cargo);
101    }
```

**Delete (Post):** Aqui realizamos a execução da chamada do método **Delete** da **CargoDAL**, enviando o **id** do cargo a ser excluído. Note que mudamos o nome do método para **DeleteConfirmed**, isso acontece porque o **GET** e **POST** do **Delete** possuem a mesma assinatura, ou seja, os dois utilizam apenas o **id** para confirmar a exclusão, dessa forma, usamos a anotação **ActionName** para informar que a ação em si tem o nome **Delete**.

```
101
102         // POST: Cargos/Delete/5
103         [HttpPost, ActionName("Delete")]
104         [ValidateAntiForgeryToken]
105
106         public ActionResult DeleteConfirmed(int? id)
107     {
108         if (id == null)
109             return NotFound();
110         cargoDAL.Delete(id);
111         return RedirectToAction("Index");
112     }
```

Agora precisamos criar as **Views** de cada **Action**. Lembre-se que o **aspnet-codegenerator** possui em seus argumentos de utilização, a obrigatoriedade de informar o **template** de página, dessa forma, para cada **View** vamos utilizar um argumento diferente.

Vamos começar pela **View Index** do **CargosController**, abra seu terminal e execute:

```
dotnet-aspnet-codegenerator view Index List -l
~/Views/Shared/_LayoutAdmin.cshtml" -outDir "Views\Cargos" -m Cargo
```

No comando acima, utilizamos o **template List**, para criar uma **View** com uma lista do **Model Cargo** (especificado pela opção **-m Cargo**), com o layout **\_LayoutAdmin**.

```

@ Index.cshtml ×
Views > Cargos > @ Index.cshtml
1  @model IEnumerable<RestauranteEtec.Models.Cargo>
2
3  @{
4      ViewData["Title"] = "Index";
5      Layout = "~/Views/Shared/_LayoutAdmin.cshtml";
6  }
7
8  <h1>Index</h1>
9
10 <p>
11     <a asp-action="Create">Create New</a>
12 </p>
13 <table class="table">
14     <thead>
15         <tr>
16             <th>
17                 @Html.DisplayNameFor(model => model.Id)
18             </th>
19             <th>
20                 @Html.DisplayNameFor(model => model.Nome)
21             </th>
22             <th></th>
23         </tr>
24     </thead>
25     <tbody>
26     @foreach (var item in Model) {
27         <tr>
28             <td>
29                 @Html.DisplayFor(modelItem => item.Id)
30             </td>
31             <td>
32                 @Html.DisplayFor(modelItem => item.Nome)
33             </td>
34             <td>
35                 @Html.ActionLink("Edit", "Edit", new { /* id=item.PrimaryKey */ }) |
36                 @Html.ActionLink("Details", "Details", new { /* id=item.PrimaryKey */ }) |
37                 @Html.ActionLink("Delete", "Delete", new { /* id=item.PrimaryKey */ })
38             </td>
39         </tr>
40     }
41     </tbody>
42 </table>

```

A 1ª linha informa a **View** que ela é **fortemente tipada**; isso significa que ela tem um **model**, no caso uma lista (**IEnumerable**) de uma classe do projeto; especificamente uma **lista de cargos**.

Esta página está programada para exibir uma tabela, com duas colunas: a primeira mostrando o **id** do cargo e a segunda mostrando o **nome** do cargo.

O **foreach**, é usado para criar uma linha na tabela para cada item da lista de **cargos** recebidos pelo **model**.

Vamos fazer algumas alterações nesse código principalmente, pelo fato de estarmos usando um **layout** diferente e porque não queremos exibir o **id** na página, ele será usado nos **links** criados com a função **ActionLink** para enviar ao **controller** qual o **cargo clicado** (**/\* id=item.PrimaryKey \*/**).

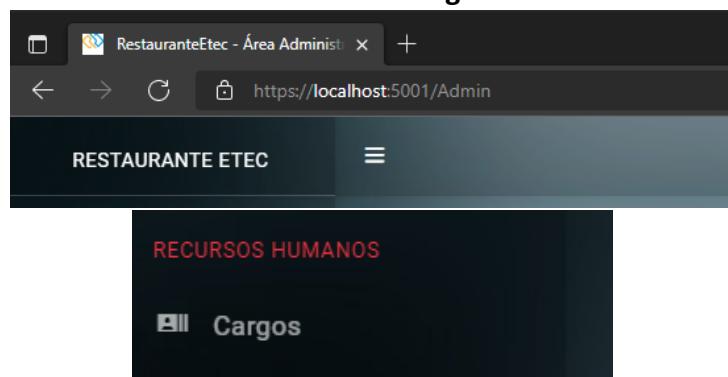
Altere seu código para ficar conforme a imagem abaixo:

## @ Index.cshtml X

Views > Cargos > @ Index.cshtml

```
1  @model IEnumerable<RestauranteEtec.Models.Cargo>
2  @{
3      ViewData["Title"] = "Cargos";
4      Layout = "~/Views/Shared/_LayoutAdmin.cshtml";
5  }
6  <div class="row mt-3">
7      <div class="col-lg-12">
8          <div class="card">
9              <div class="card-body">
10                 <div class="card-title">Lista de Cargos</div>
11                 <hr>
12                 <p class="mb-3">
13                     <a class="btn btn-light" asp-action="Create">Adicionar Novo</a>
14                 </p>
15                 <table class="table table-striped table-hover">
16                     <thead>
17                         <tr>
18                             <th>
19                                 @Html.DisplayNameFor(model => model.Nome)
20                             </th>
21                             <th>Ações</th>
22                         </tr>
23                     </thead>
24                     <tbody>
25                         @foreach (var item in Model)
26                         {
27                             <tr>
28                                 <td>
29                                     @Html.DisplayFor(modelItem => item.Nome)
30                                 </td>
31                                 <td>
32                                     <a asp-action="Edit" class="mr-3 text-warning" asp-route-id="@item.Id">
33                                         <i class="zmdi zmdi-edit" title="Alterar"></i>
34                                     </a>
35                                     <a asp-action="Details" class="mr-3 text-white" asp-route-id="@item.Id">
36                                         <i class="zmdi zmdi-zoom-in" title="Detalhes"></i>
37                                     </a>
38                                     <a asp-action="Delete" class="text-danger" asp-route-id="@item.Id">
39                                         <i class="zmdi zmdi-delete" title="Excluir"></i>
40                                     </a>
41                                 </td>
42                             </tr>
43                         }
44                     </tbody>
45                 </table>
46             </div>
47         </div>
48     </div>
49 </div>
```

Salve seu projeto, execute no terminal um “**dotnet run**”, na barra de navegação digite “**/Admin**” no final do código e navegue pelo menu lateral até o cadastro de **Cargos**:



The screenshot shows a dark-themed web application interface for "Restaurante ETEC". On the left, a sidebar lists various menu items under categories like SOCIAL, MENU, and RECURSOS HUMANOS. The "Cargos" item is selected, indicated by a red box. The main content area is titled "Lista de Cargos" and contains a table with three rows of position names. A "Novo" button is visible next to the "Contatos" item in the sidebar.

NOME	AÇÕES
CEO, Co Fundador	
Chefe de Cozinha	
Cozinheiro Chefe	

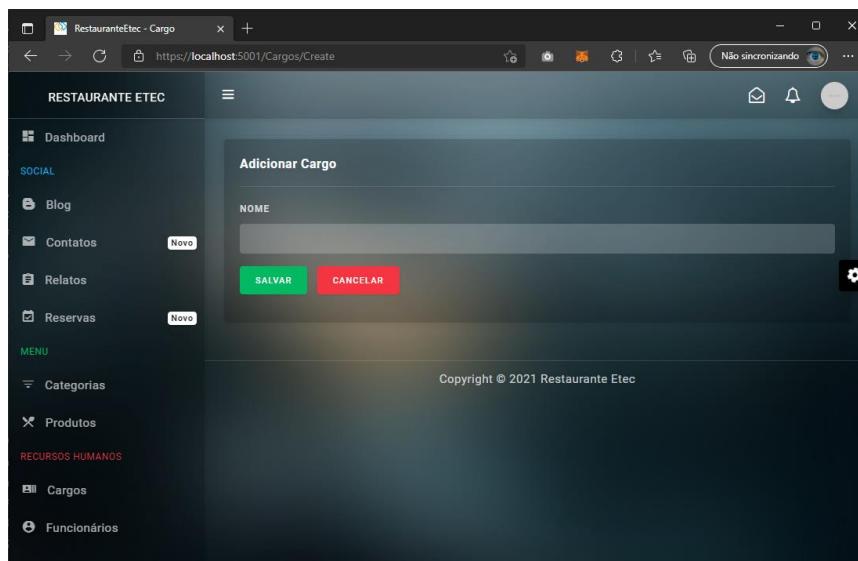
Copyright © 2021 Restaurante Etec

Vamos para as próximas **Views**, execute os comandos das próximas páginas, seguindo da alteração do código gerado conforme as imagens posteriores aos comandos.

Aqui estamos criando a **View Create**, utilizando o **template** de mesmo nome **Create**. Adicionar Cargo.

```
dotnet-aspnet-codegenerator view Create Create -l
~/Views/Shared/_LayoutAdmin.cshtml" -outDir "Views\Cargos" -m Cargo
```

```
@ Create.cshtml X
Views > Cargos > @ Create.cshtml
1 @model RestauranteEtec.Models.Cargo
2
3 @{
4     ViewData["Title"] = "Cargo";
5     Layout = "~/Views/Shared/_LayoutAdmin.cshtml";
6 }
7
8 <div class="row mt-3">
9     <div class="col-lg-12">
10         <div class="card">
11             <div class="card-body">
12                 <div class="card-title">Adicionar Cargo</div>
13                 <hr>
14                 <form asp-action="Create" method="post">
15                     <div asp-validation-summary="ModelOnly" class="text-danger"></div>
16                     <div class="form-group">
17                         <label asp-for="Nome" class="control-label"></label>
18                         <input asp-for="Nome" class="form-control" />
19                         <span asp-validation-for="Nome" class="text-danger"></span>
20                     </div>
21                     <div class="form-group">
22                         <input type="submit" value="Salvar" class="btn btn-success mr-2" />
23                         <a class="btn btn-danger" asp-action="Index">Cancelar</a>
24                     </div>
25                 </form>
26             </div>
27         </div>
28     </div>
29 </div>
30 @section Scripts {
31     @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
32 }
```

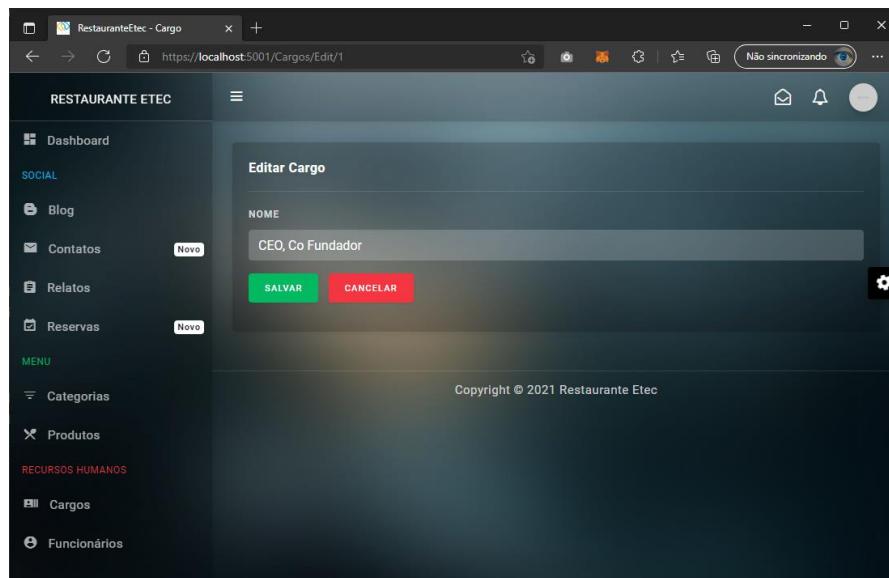


Agora vamos criar a **View Edit**, utilizando o **template** de mesmo nome **Edit**. Alterar Cargo.

```
dotnet-aspnet-codegenerator view Edit Edit -l
~/Views/Shared/_LayoutAdmin.cshtml" -outDir "Views\Cargos" -m Cargo
```

```
@ Edit.cshtml X

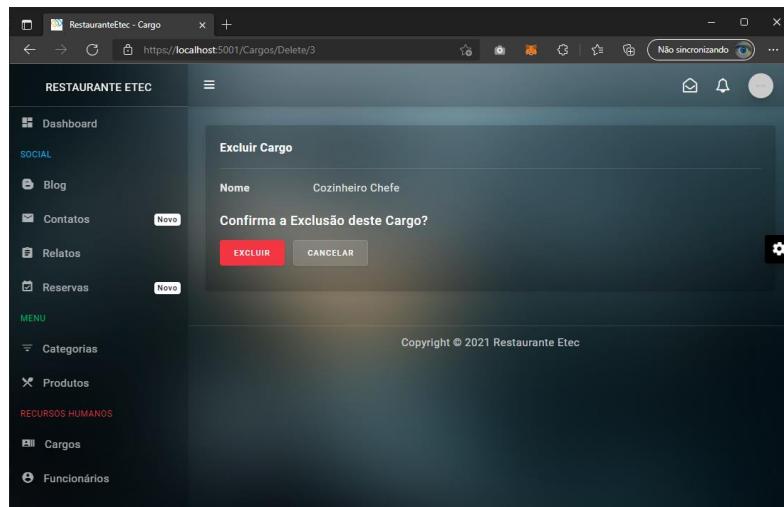
Views > Cargos > @ Edit.cshtml
1  @model RestauranteEtec.Models.Cargo
2  @{
3      ViewData["Title"] = "Cargo";
4      Layout = "~/Views/Shared/_LayoutAdmin.cshtml";
5  }
6  <div class="row mt-3">
7      <div class="col-lg-12">
8          <div class="card">
9              <div class="card-body">
10                 <div class="card-title">Editar Cargo</div>
11                 <hr>
12                 <form asp-action="Edit" method="post">
13                     <div asp-validation-summary="ModelOnly" class="text-danger"></div>
14                     <input type="hidden" asp-for="Id" />
15                     <div class="form-group">
16                         <label asp-for="Nome" class="control-label"></label>
17                         <input asp-for="Nome" class="form-control" />
18                         <span asp-validation-for="Nome" class="text-danger"></span>
19                     </div>
20                     <div class="form-group">
21                         <input type="submit" value="Salvar" class="btn btn-success mr-2" />
22                         <a class="btn btn-danger" asp-action="Index">Cancelar</a>
23                     </div>
24                 </form>
25             </div>
26         </div>
27     </div>
28 </div>
29 @section Scripts {
30     @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
31 }
```



Agora vamos criar a **View Delete**, utilizando o **template de mesmo nome Delete**. Excluir Cargo.

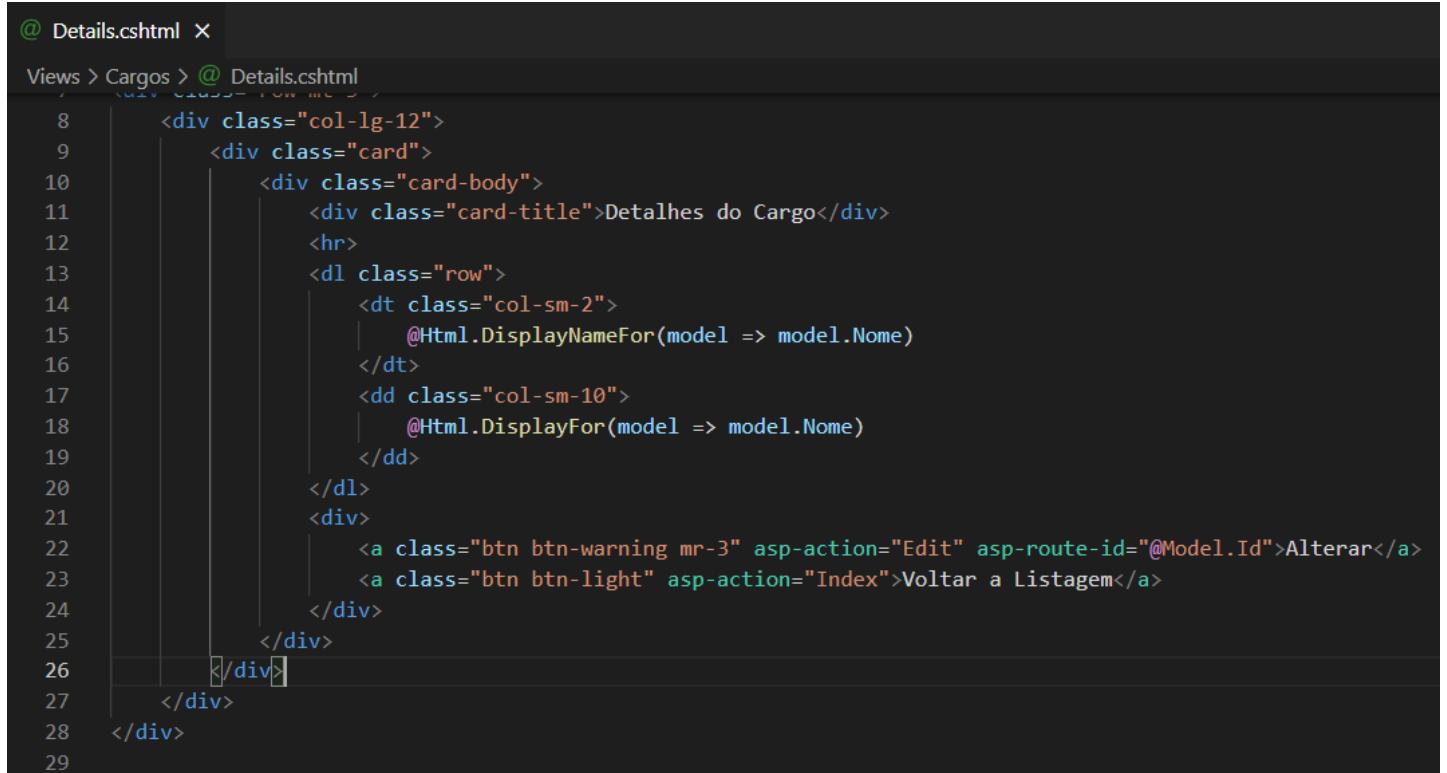
```
dotnet-aspnet-codegenerator view Delete Delete -l
~/Views/Shared/_LayoutAdmin.cshtml" -outDir "Views\Cargos" -m Cargo
```

```
@ Delete.cshtml X
Views > Cargos > @ Delete.cshtml
1 @model RestauranteEtec.Models.Cargo
2 @{
3     ViewData["Title"] = "Cargo";
4     Layout = "~/Views/Shared/_LayoutAdmin.cshtml";
5 }
6
7 <div class="row mt-3">
8     <div class="col-lg-12">
9         <div class="card">
10            <div class="card-body">
11                <div class="card-title">Excluir Cargo</div>
12                <hr>
13                <dl class="row">
14                    <dt class="col-sm-2">
15                        @Html.DisplayNameFor(model => model.Nome)
16                    </dt>
17                    <dd class="col-sm-10">
18                        @Html.DisplayFor(model => model.Nome)
19                    </dd>
20                </dl>
21                <form asp-action="Delete" method="post">
22                    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
23                    <input type="hidden" asp-for="Id" />
24                    <h5 class="my-3">Confirma a Exclusão deste Cargo?</h5>
25                    <div class="form-group">
26                        <input type="submit" value="Excluir" class="btn btn-danger mr-2" />
27                        <a class="btn btn-light" asp-action="Index">Cancelar</a>
28                    </div>
29                </form>
30            </div>
31        </div>
32    </div>
33 </div>
```



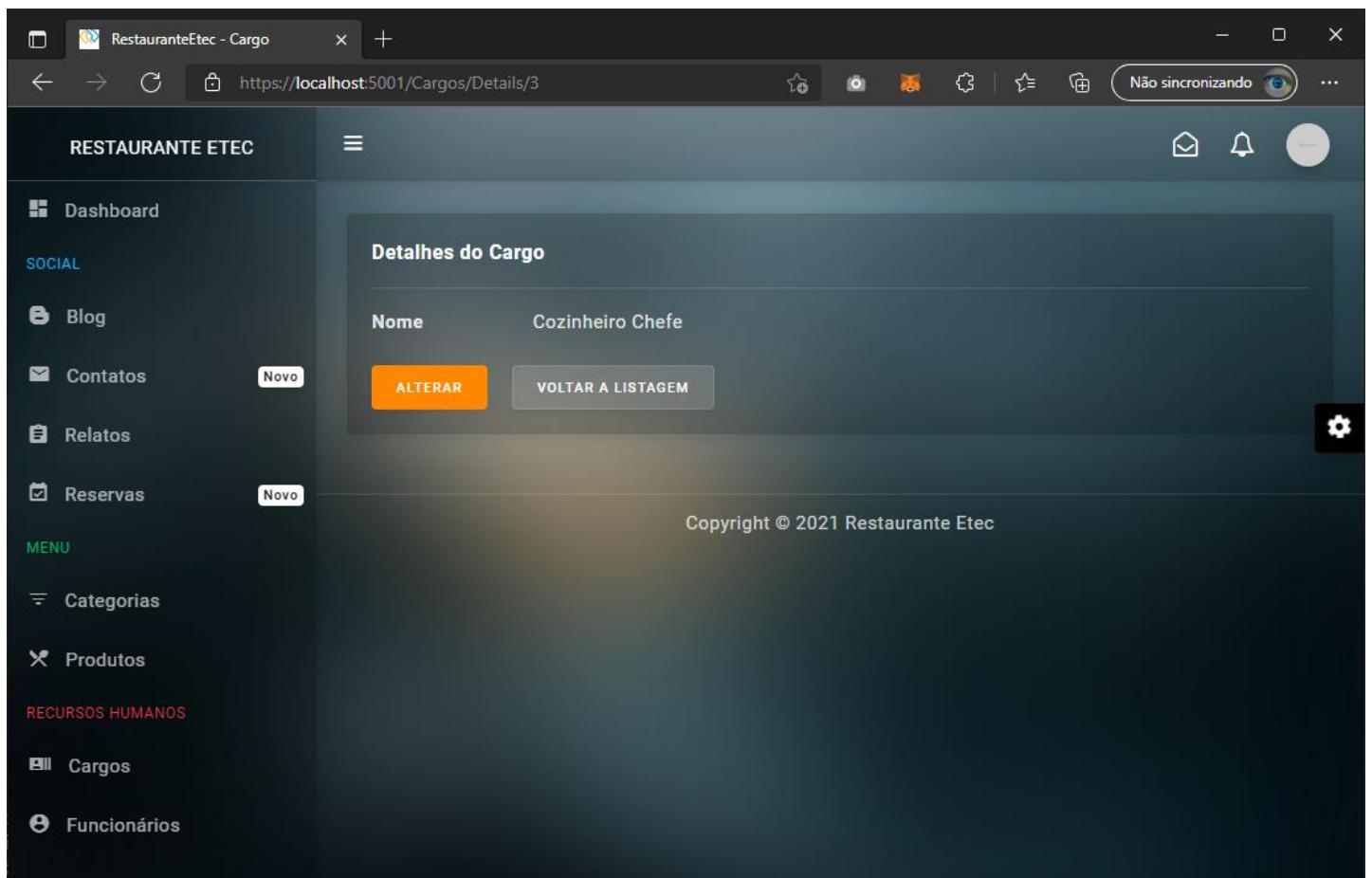
Agora vamos criar a **View Details**, utilizando o **template** de mesmo nome **Details**. Detalhes do Cargo

```
dotnet-aspnet-codegenerator view Details Details -l  
"~/Views/Shared/_LayoutAdmin.cshtml" -outDir "Views\Cargos" -m Cargo
```



The screenshot shows a code editor with the file `Details.cshtml` open. The code is a standard ASP.NET Core view for displaying details of a cargo. It uses Bootstrap classes for layout and includes links to edit or return to the index.

```
@ Details.cshtml X  
Views > Cargos > @ Details.cshtml  
8   <div class="col-lg-12">  
9     <div class="card">  
10       <div class="card-body">  
11         <div class="card-title">Detalhes do Cargo</div>  
12         <hr>  
13         <dl class="row">  
14           <dt class="col-sm-2">  
15             @Html.DisplayNameFor(model => model.Nome)  
16           </dt>  
17           <dd class="col-sm-10">  
18             @Html.DisplayFor(model => model.Nome)  
19           </dd>  
20         </dl>  
21         <div>  
22           <a class="btn btn-warning mr-3" asp-action="Edit" asp-route-id="@Model.Id">Alterar</a>  
23           <a class="btn btn-light" asp-action="Index">Voltar a Listagem</a>  
24         </div>  
25       </div>  
26     </div>  
27   </div>  
28 </div>  
29
```

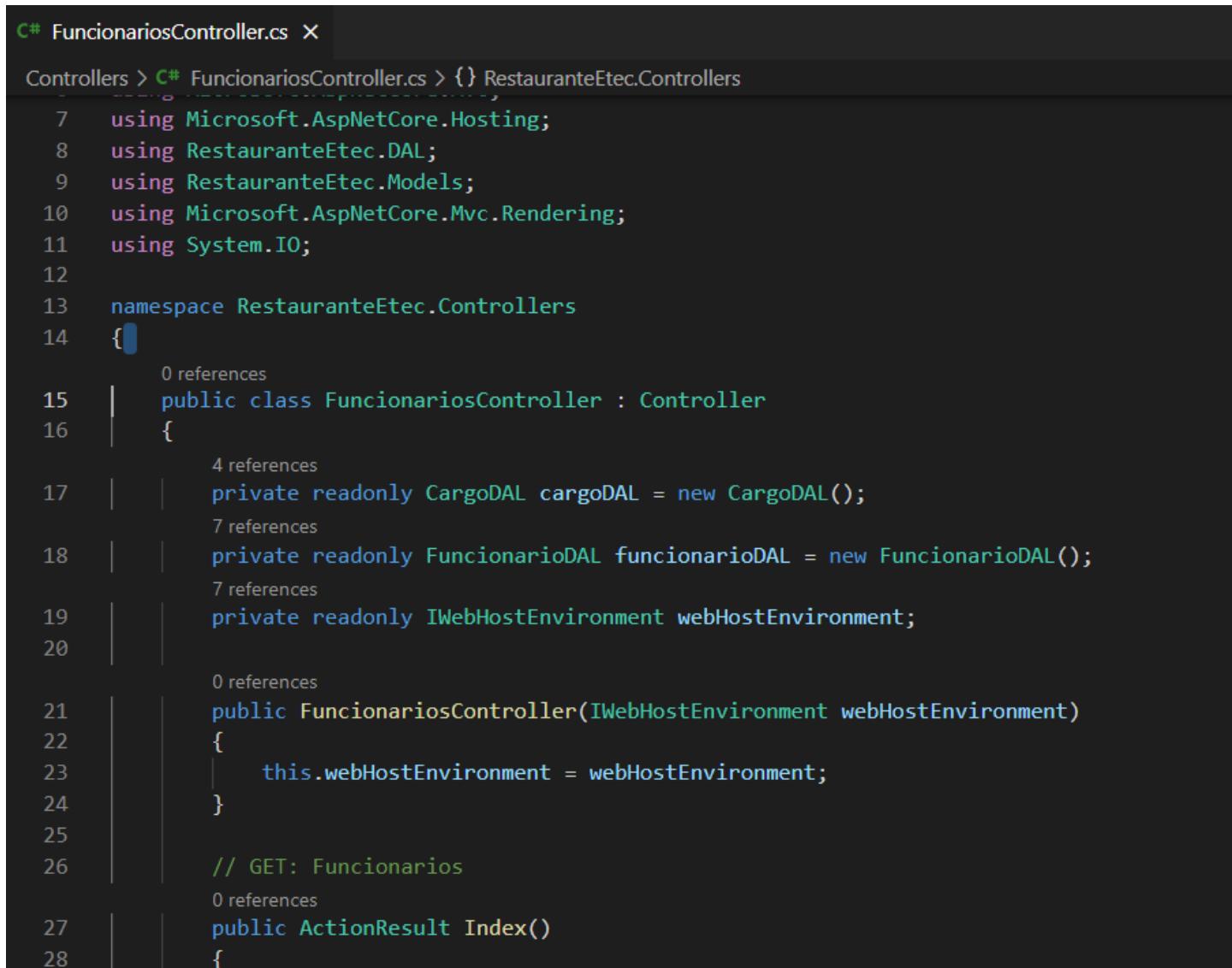


## CRIANDO O CRUD DE FUNCIONÁRIOS COM UPLOAD DE IMAGEM

Vamos agora criar um **Controller** para realizar as operações de **CRUD** através de nossa camada **DAL** para Funcionários. Também vamos precisar fazer uma injeção de dependência, isso significa que vamos adicionar ao controlador de funcionários, um objeto do servidor que permite que ele consiga ter acesso a arquivos de configuração do projeto, mais especificamente, esse objeto terá a função de nos permitir ter acesso físico a pasta de imagens do nosso servidor, para que com isso, nosso código consiga fazer upload das fotos dos funcionários.

```
dotnet-aspnet-codegenerator controller -name FuncionariosController  
-outDir Controllers -actions
```

Vamos começar incluindo no começo do **Controller** alguns objetos, **linhas 17, 18 e 19**, e não se esqueça de incluir os **usings** das **linhas 7, 8, 9, 10 e 11**, e por fim, o método **construtor linhas 21 a 24**:



The screenshot shows a code editor with the file 'FuncionariosController.cs' open. The code is as follows:

```
C# FuncionariosController.cs X  
  
Controllers > C# FuncionariosController.cs > {} RestauranteEtec.Controllers  
7   using Microsoft.AspNetCore.Hosting;  
8   using RestauranteEtec.DAL;  
9   using RestauranteEtec.Models;  
10  using Microsoft.AspNetCore.Mvc.Rendering;  
11  using System.IO;  
12  
13 namespace RestauranteEtec.Controllers  
14 {  
15     public class FuncionariosController : Controller  
16     {  
17         private readonly CargoDAL cargoDAL = new CargoDAL();  
18         private readonly FuncionarioDAL funcionarioDAL = new FuncionarioDAL();  
19         private readonly IWebHostEnvironment webHostEnvironment;  
20  
21         public FuncionariosController(IWebHostEnvironment webHostEnvironment)  
22         {  
23             this.webHostEnvironment = webHostEnvironment;  
24         }  
25  
26         // GET: Funcionarios  
27         public ActionResult Index()  
28     }
```

O método construtor (linhas 21 a 24) recebe o **IWebHostEnvironment**, que é um serviço **middleware** que vai nos permitir ter acesso físico a pasta **wwwroot** no servidor.

Vamos agora alterar as **Actions** do **FuncionariosController** conforme as imagens abaixo:

**Index:** Aqui utilizamos o método **GetAll**, programado na **FuncionarioDAL** para retornar todos os funcionários cadastrados a **View**.

```
25
26     // GET: Funcionarios
27     0 references
28     public ActionResult Index()
29     {
30         var funcionarios = funcionarioDAL.GetAll();
31         return View(funcionarios);
32     }
```

**Detail:** Aqui utilizamos o método  **GetById**, programado na **FuncionarioDAL** para retornar apenas o funcionário com **Id** igual ao **parâmetro id**, recebido pela **action**, e enviado a **View**. Caso o **id** for **nulo** ou o **Funcionario** não for encontrado, o sistema retorna uma página de **Não Encontrado (NotFound)**. Também estamos utilizando Lembre-se de colocar o "?" na linha 34. Na linha 41, criamos uma **ViewData["wwwroot"]** que será utilizada na **View**, para verificarmos se a imagem existe no servidor, desta forma podemos exibir outra imagem se esta não for localizada.

```
32
33     // GET: Funcionarios/Details/5
34     0 references
35     public ActionResult Details(int? id)
36     {
37         if (id == null)
38             return NotFound();
39         var funcionario = funcionarioDAL.GetById(id);
40         if (funcionario == null)
41             return NotFound();
42         ViewData["wwwroot"] = webHostEnvironment.WebRootPath;
43         return View(funcionario);
44     }
```

**Create (Get):** Este método tem a função de exibir ao usuário a **View** que irá permitir que um novo **Funcionário** possa ser cadastrado. Apesar de ser uma página simples, vamos adicionar a **View**, um elemento do **HTML** chamado **SELECT**, que exibe uma lista de seleção (um combo) com os **Cargos cadastrados**, desta forma quando o usuário for incluir um novo funcionário, poderá de maneira fácil, apenas selecionar o cargo que aquele funcionário possui no restaurante. Estamos utilizando uma **ViewData** para enviar um **SelectList**, que é uma lista de objetos, carregado com os cargos cadastrados no banco de dados, consultados através do **cargoDAL.GetAll()**, seguidos duas informações, o **campo chave primária (Id)** e o campo que queremos que apareça no **Select** da página (**Nome**). Para finalizar o método **GET** do **CREATE**, vamos criar um funcionário, ajustar algumas propriedades e enviar a **View** para que o usuário possa preencher os demais valores.

```

44
45     // GET: Funcionarios/Create
46     0 references
47     public ActionResult Create()
48     {
49         ViewData["Cargos"] = new SelectList(cargoDAL.GetAll(), "Id", "Nome");
50         var funcionario = new Funcionario();
51         funcionario.Ativo = true;
52         funcionario.ExibirHome = false;
53         return View(funcionario);
54     }

```

**Create (Post):** Vamos precisar mudar a **Action Create** para funcionar de forma **Assíncrona**. A programação assíncrona é um poderoso recurso da linguagem C# que permite que você continue com a execução do seu programa no **thread** principal enquanto uma tarefa de longa duração é executada no seu próprio thread separadamente do thread principal.

```

55     // POST: Funcionarios/Create
56     [HttpPost]
57     [ValidateAntiForgeryToken]
58     0 references
59     public async Task<ActionResult> Create([Bind] Funcionario funcionario, IFormFile Foto)
60     {
61         ViewData["Cargos"] = new SelectList(cargoDAL.GetAll(), "Id", "Nome");
62         if (!ModelState.IsValid)
63             return View(funcionario);
64         try
65         {
66             if (Foto != null)
67             {
68                 string pasta = Path.Combine(webHostEnvironment.WebRootPath, "images\\funcionarios");
69                 var nomeArquivo = Guid.NewGuid().ToString() + "_" + Foto.FileName;
70                 string caminhoArquivo = Path.Combine(pasta, nomeArquivo);
71                 using (var stream = new FileStream(caminhoArquivo, FileMode.Create))
72                 {
73                     await Foto.CopyToAsync(stream);
74                 };
75                 funcionario.Foto = "/images/funcionarios/" + nomeArquivo;
76                 funcionarioDAL.Add(funcionario);
77                 return RedirectToAction("Index");
78             }
79             catch
80             {
81                 return View(funcionario);
82             }
83         }

```

Antes de executarmos nosso projeto, deve ficar claro que em nosso código, estamos considerando que dentro da pasta **images**, na **wwwroot**, exista uma pasta **funcionarios**, onde serão armazenadas as fotos dos **funcionários**. Ou seja, precisamos antes de executar nosso projeto, criar a pasta mencionada.

Cada arquivo de imagem enviado ao servidor, serão recebidos e processados pelo nosso código, usamos um objeto **Guid**, para criar uma chave única (**Guid** criar chaves alfanuméricos), e definimos o nome da imagem no servidor, composta por esse **Guid** somado o nome atual do arquivo. Isso é necessário, porque caso o usuário faça o upload de uma imagem como mesmo nome de outra já existente no servidor, a primeira imagem seria substituída pela nova imagem. Desta forma, sempre teremos nomes únicos.

**Edit (Get):** Aqui utilizamos o método **GetById**, programado no **FuncionarioDAL** para retornar apenas o funcionário com **Id** igual ao **parâmetro id**, recebido pela **Action**, e enviado a **View**. Caso o **id** for **nulo** ou o **Funcionário** não for encontrado, o sistema retorna uma página de **Não Encontrado (NotFound)**. Na linha 93, criamos uma **ViewData["wwwroot"]** que será utilizada na **View**, para verificarmos se a imagem existe no servidor, desta forma podemos exibir outra imagem se esta não for localizada. Também precisamos criar a **ViewData["Cargos"]** igual fizemos no **Create**, para alimentar o **Select** de **Cargos**. Por fim, o método apresenta ao usuário a **View**, com o objeto **funcionario** carregado no código.

```

85     // GET: Funcionarios/Edit/5
86     0 references
87     public ActionResult Edit(int? id)
88     {
89         if (id == null)
90             return NotFound();
91         var funcionario = funcionarioDAL.GetById(id);
92         if (funcionario == null)
93             return NotFound();
94         ViewData["wwwroot"] = webHostEnvironment.WebRootPath;
95         ViewData["Cargos"] = new SelectList(cargoDAL.GetAll(), "Id", "Nome");
96         return View(funcionario);
97     }

```

**Edit (Post):** Assim como fizemos no **Create Post**, vamos precisar mudar a **Action Edit** para funcionar de forma **Assíncrona**. O restante do código é bem semelhante ao **Edit** do **CargosController**, com a inclusão do **NovaFoto**.

```

101    public async Task<ActionResult> Edit(int id, [Bind] Funcionario funcionario, IFormFile NovaFoto)
102    {
103        if (id != funcionario.Id)
104            return NotFound();
105        ViewData["wwwroot"] = webHostEnvironment.WebRootPath;
106        ViewData["Cargos"] = new SelectList(cargoDAL.GetAll(), "Id", "Nome");
107        if (!ModelState.IsValid)
108            return View(funcionario);
109        try
110        {
111            if (NovaFoto != null)
112            {
113                string pasta = Path.Combine(webHostEnvironment.WebRootPath, "images\\funcionarios");
114                var nomeArquivo = Guid.NewGuid().ToString() + "_" + NovaFoto.FileName;
115                string caminhoArquivo = Path.Combine(pasta, nomeArquivo);
116                using (var stream = new FileStream(caminhoArquivo, FileMode.Create))
117                {
118                    await NovaFoto.CopyToAsync(stream);
119                };
120                funcionario.Foto = "/images/funcionarios/" + nomeArquivo;
121            }
122            funcionarioDAL.Update(funcionario);
123            return RedirectToAction("Index");
124        }
125        catch
126        {
127            return View(funcionario);
128        }
129    }

```

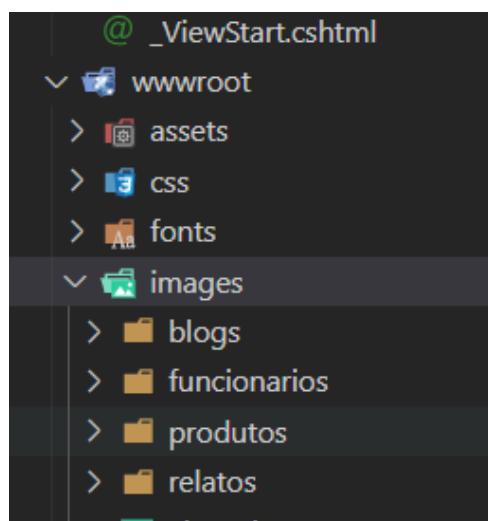
**Delete (Get):** Aqui utilizamos o método **GetById**, programado no **FuncionarioDAL** para retornar apenas o funcionário com **Id** igual ao **parâmetro id**, recebido pela **Action**, e enviado a **View**. Caso o **id** for **nulo** ou o **Funcionário** não for encontrado, o sistema retorna uma página de **Não Encontrado (NotFound)**. Na linha 139, criamos uma **ViewData["wwwroot"]** que será utilizada na **View**, para verificarmos se a imagem existe no servidor, desta forma podemos exibir outra imagem se esta não for localizada. Por fim, o método apresenta ao usuário a **View**, com o objeto **funcionario** carregado no código.

```
131     // GET: Funcionarios/Delete/5
132     0 references
133     public ActionResult Delete(int? id)
134     {
135         if (id == null)
136             return NotFound();
137         var funcionario = funcionarioDAL.GetById(id);
138         if (funcionario == null)
139             return NotFound();
140         ViewData["wwwroot"] = webHostEnvironment.WebRootPath;
141         return View(funcionario);
142     }
```

**Delete (Post):** Aqui temos a mesma estrutura de código que o **CargoControllers**.

```
142
143     // POST: Funcionarios/Delete/5
144     [HttpPost, ActionName("Delete")]
145     [ValidateAntiForgeryToken]
146     0 references
147     public IActionResult DeleteConfirmed(int? id)
148     {
149         if (id == null)
150             return NotFound();
151         funcionarioDAL.Delete(id);
152         return RedirectToAction("Index");
```

Adicione algumas pastas dentro da pasta **wwwroot**, conforme a imagem abaixo:



Vamos precisar de uma imagem que informe ao usuário que um funcionário ainda não tem foto ou que sua foto não está mais disponível no servidor. Para isso você precisa fazer o download do arquivo “[sem\\_foto.png](#)” e copiar esse arquivo na pasta **images** dentro de **wwwroot**.



FOTO INDISPONÍVEL

Vamos agora criar as **Views** do **FuncionariosController** e fazer a alterações dos códigos gerados.

**View Index** do **FuncionariosController**, abra seu terminal e execute:

```
dotnet-aspnet-codegenerator view Index List -m Funcionario -l  
"~/Views/Shared/_LayoutAdmin.cshtml" -outDir "Views\Funcionarios"
```

**View Create** do **FuncionariosController**, abra seu terminal e execute:

```
dotnet-aspnet-codegenerator view Create Create -m Funcionario -l  
"~/Views/Shared/_LayoutAdmin.cshtml" -outDir "Views\Funcionarios"
```

**View Edit** do **FuncionariosController**, abra seu terminal e execute:

```
dotnet-aspnet-codegenerator view Edit Edit -m Funcionario -l  
"~/Views/Shared/_LayoutAdmin.cshtml" -outDir "Views\Funcionarios"
```

**View Delete** do **FuncionariosController**, abra seu terminal e execute:

```
dotnet-aspnet-codegenerator view Delete Delete -m Funcionario -l  
"~/Views/Shared/_LayoutAdmin.cshtml" -outDir "Views\Funcionarios"
```

**View Details** do **FuncionariosController**, abra seu terminal e execute:

```
dotnet-aspnet-codegenerator view Details Details -m Funcionario -l  
"~/Views/Shared/_LayoutAdmin.cshtml" -outDir "Views\Funcionarios"
```

Neste ponto, temos as **Views** criadas, se executar seu código irá ver que as páginas estão geradas e funcionais. O detalhe **aqui** é o design, uma vez que os **templates** gerados automaticamente, não seguem o mesmo padrão que o estilo que estamos utilizando, portanto, vamos a edição.

As imagens abaixo, mostram como deve ficar os códigos das diferentes **Views**, após suas edições.

## @ Index.cshtml ×

Views > Funcionarios > @ Index.cshtml

```
1  @model IEnumerable<RestauranteEtec.Models.Funcioario>
2  @{
3      ViewData["Title"] = "Funcionários";
4      Layout = "~/Views/Shared/_LayoutAdmin.cshtml";
5  }
6
7  <div class="row mt-3">
8      <div class="col-lg-12">
9          <div class="card">
10              <div class="card-body">
11                  <div class="card-title">Lista de Funcionários</div>
12                  <hr>
13                  <p class="mb-3">
14                      <a class="btn btn-light" asp-action="Create">Adicionar Novo</a>
15                  </p>
16                  <table class="table table-striped table-hover">
17                      <thead>
18                          <tr>
19                              <th>
20                                  @Html.DisplayNameFor(model => model.Nome)
21                              </th>
22                              <th>
23                                  @Html.DisplayNameFor(model => model.Cargo)
24                              </th>
25                              <th>
26                                  @Html.DisplayNameFor(model => model.ExibirHome)
27                              </th>
28                              <th>
29                                  @Html.DisplayNameFor(model => model.Ativo)
30                              </th>
31                              <th>Ações</th>
32                          </tr>
33                      </thead>
34                      <tbody>
35                          @foreach (var item in Model)
36                          {
37                              <tr>
38                                  <td>
39                                      @Html.DisplayFor(modelItem => item.Nome)
40                                  </td>
41                                  <td>
42                                      @Html.DisplayFor(modelItem => item.Cargo.Nome)
43                                  </td>
44                                  <td>
45                                      <div class="icheck-material-white">
46                                          <input type="checkbox" asp-for="@item.ExibirHome" disabled/>
47                                          <label asp-for="@item.ExibirHome">Exibir</label>
48                                      </div>
49                                  </td>
50                                  <td>
51                                      <div class="icheck-material-white">
52                                          <input type="checkbox" asp-for="@item.Ativo" disabled />
53                                          <label asp-for="@item.Ativo">Ativo</label>
54                                      </div>
55                                  </td>
56
57                                  <td>
58                                      <a asp-action="Edit" class="mr-3 text-warning" asp-route-id="@item.Id">
59                                          <i class="zmdi zmdi-edit" title="Alterar"></i>
60                                      </a>
61                                      <a asp-action="Details" class="mr-3 text-white" asp-route-id="@item.Id" style="color: white; background-color: #007bff; border-radius: 50%; padding: 2px 5px; font-size: small;">
62                                          <i class="zmdi zmdi-zoom-in" title="Detalhes"></i>
63                                      </a>
64                                      <a asp-action="Delete" class="text-danger" asp-route-id="@item.Id">
65                                          <i class="zmdi zmdi-delete" title="Excluir"></i>
66                                      </a>
67                                  </td>
68                              </tr>
69                          }
70                      </tbody>
71                  </table>
72              </div>
73          </div>
74      </div>
75  </div>
```

```

@ Create.cshtml X
Views > Funcionarios > @ Create.cshtml
1  @model RestauranteEtec.Models.Funcioario
2  @{
3      ViewData["Title"] = "Funcionário";
4      Layout = "~/Views/Shared/_LayoutAdmin.cshtml";
5  }
6
7  <div class="row mt-3">
8      <div class="col-lg-12">
9          <div class="card">
10              <div class="card-body">
11                  <div class="card-title">Adicionar Funcionário</div>
12                  <hr>
13                  <form asp-action="Create" enctype="multipart/form-data">
14                      <div class="form-row">
15                          <div asp-validation-summary="ModelOnly" class="text-danger"></div>
16                          <div class="col-md-9 col-sm-12">
17                              <div class="form-group">
18                                  <label asp-for="Nome" class="control-label"></label>
19                                  <input asp-for="Nome" class="form-control" />
20                                  <span asp-validation-for="Nome" class="text-danger"></span>
21                              </div>
22                              <div class="form-group">
23                                  <label asp-for="Descrição" class="control-label"></label>
24                                  <textarea asp-for="Descrição" class="form-control" rows="2" > </textarea>
25                                  <span asp-validation-for="Descrição" class="text-danger"></span>
26                              </div>
27                              <div class="form-group">
28                                  <label asp-for="CargoId" class="control-label"></label>
29                                  <select asp-for="CargoId" class="form-control" asp-items="ViewBag.Cargos"></select>
30                                  <span asp-validation-for="CargoId" class="text-danger"></span>
31                              </div>
32
33                              <div class="form-row">
34                                  <div class="form-group col-md-4">
35                                      <label asp-for="OrdemExibicao" class="control-label"></label>
36                                      <input asp-for="OrdemExibicao" class="form-control" />
37                                      <span asp-validation-for="OrdemExibicao" class="text-danger"></span>
38                                  </div>
39                                  <div class="form-group col-md-4 pt-md-4 text-center">
40                                      <div class="icheck-material-white">
41                                          <input type="checkbox" asp-for="ExibirHome" />
42                                          <label asp-for="ExibirHome"></label>
43                                      </div>
44                                  </div>
45                                  <div class="form-group col-md-4 pt-md-4 text-center">
46                                      <div class="icheck-material-white">
47                                          <input type="checkbox" asp-for="Ativo" />
48                                          <label asp-for="Ativo"></label>
49                                      </div>
50                                  </div>
51                              </div>
52                          <div class="col-md-3 col-sm-12">
53                              <div class="row">
54                                  <div class="form-group col-12">
55                                      <img src="" class="img-fluid" id='PreviewImagem' />
56                                      <label asp-for="Foto" class="control-label"></label>
57                                      <input type="file" asp-for="Foto" class="form-control-file" accept=".jpg,.jpeg,.png,.gif" />
58                                      <span asp-validation-for="Foto" class="text-danger"></span>
59                                  </div>
60                              </div>
61                          </div>
62                      </div>
63                      <div class="form-group">
64                          <input type="submit" value="Salvar" class="btn btn-success mr-2" />
65                          <a class="btn btn-danger" asp-action="Index">Cancelar</a>
66                      </div>
67                  </div>
68              </form>
69          </div>
70      </div>
71  </div>
72
73
74  @section Scripts {
75      @await Html.RenderPartialAsync("_ValidationScriptsPartial");
76      <script type="text/javascript">
77          window.addEventListener('load', function () {
78              document.querySelector('input[type="file"]').addEventListener('change', function () {
79                  if (this.files && this.files[0]) {
80                      var img = document.getElementById('PreviewImagem');
81                      img.src = URL.createObjectURL(this.files[0]);
82                  }
83              });
84          });
85      </script>
86  }

```

```

@ Edits.cshtml X
Views > Funcionarios > @ Edit.cshtml
1  @model RestauranteEtec.Models.Funcioario
2  @{
3      ViewData["Title"] = "Funcionario";
4      Layout = "~/Views/Shared/_LayoutAdmin.cshtml";
5  }
6  <div class="row mt-3">
7      <div class="col-lg-12">
8          <div class="card">
9              <div class="card-body">
10                 <div class="card-title">Alterar Funcionario</div>
11                 <hr>
12                 <form asp-action="Edit" enctype="multipart/form-data">
13                     <input type="hidden" asp-for="Id" />
14                     <div class="form-row">
15                         <div asp-validation-summary="ModelOnly" class="text-danger"></div>
16                         <div class="col-9">
17                             <div class="form-group">
18                                 <label asp-for="Nome" class="control-label"></label>
19                                 <input asp-for="Nome" class="form-control" />
20                                 <span asp-validation-for="Nome" class="text-danger"></span>
21                             </div>
22                             <div class="form-group">
23                                 <label asp-for="Descricao" class="control-label"></label>
24                                 <textarea asp-for="Descricao" class="form-control" rows="3" > </textarea>
25                                 <span asp-validation-for="Descricao" class="text-danger"></span>
26                             </div>
27                             <div class="form-group">
28                                 <label asp-for="CargoId" class="control-label"></label>
29                                 <select asp-for="CargoId" class="form-control" asp-items="ViewBag.Cargos"></select>
30                                 <span asp-validation-for="CargoId" class="text-danger"></span>
31                             </div>
32                             <div class="form-row">
33                                 <div class="form-group col-4">
34                                     <label asp-for="OrdemExibicao" class="control-label"></label>
35                                     <input asp-for="OrdemExibicao" class="form-control" />
36                                     <span asp-validation-for="OrdemExibicao" class="text-danger"></span>
37                                 </div>
38                                 <div class="form-group col-4 pt-4 text-center">
39                                     <div class="icheck-material-white">
40                                         <input type="checkbox" asp-for="ExibirHome" />
41                                         <label asp-for="ExibirHome"></label>
42                                     </div>
43                                 </div>
44                                 <div class="form-group col-4 pt-4 text-center">
45                                     <div class="icheck-material-white">
46                                         <input type="checkbox" asp-for="Ativo" />
47                                         <label asp-for="Ativo"></label>
48                                     </div>
49                                 </div>
50                             </div>
51                             <div class="col-3">
52                                 <div class="row">
53                                     <div class="form-group col-12">
54                                         <label asp-for="Foto" class="control-label"></label>
55                                         <input type="hidden" asp-for="Foto" />
56                                         
57                                         <input type="file" id="NovaFoto" name="NovaFoto" class="form-control-file" accept=".jpg,.jpeg,.png,.gif" />
58                                         <span asp-validation-for="Foto" class="text-danger"></span>
59                                     </div>
60                                 </div>
61                             </div>
62                         </div>
63                     </div>
64                 </div>
65             </div>
66             <div class="form-group">
67                 <input type="submit" value="Salvar" class="btn btn-success mr-2" />
68                 <a class="btn btn-danger" asp-action="Index">Cancelar</a>
69             </div>
70         </div>
71     </div>
72 </div>
73 </div>
74 </div>
75 </div>
76 @section Scripts {
77     @await Html.RenderPartialAsync("_ValidationScriptsPartial");
78     <script type="text/javascript">
79         window.addEventListener('load', function () {
80             document.querySelector('input[type="file"]').addEventListener('change', function () {
81                 if (this.files && this.files[0]) {
82                     var img = document.getElementById('PreviewImagem');
83                     img.src = URL.createObjectURL(this.files[0]);
84                 }
85             });
86         });
87     </script>
88 }

```

@ Delete.cshtml X

Views > Funcionarios > @ Delete.cshtml

```
1  @model RestauranteEtec.Models.Funcioario
2  @{
3      ViewData["Title"] = "Funcionário";
4      Layout = "~/Views/Shared/_LayoutAdmin.cshtml";
5  }
6
7  <div class="row mt-3">
8      <div class="col-lg-12">
9          <div class="card">
10             <div class="card-body">
11                 <div class="card-title">Excluir Funcionário</div>
12                 <hr>
13                 <div class="row">
14                     <div class="col-3">
15                         
18                     </div>
19                     <div class="col-9">
20                         <div class="row">
21                             <label class="col-3" asp-for="Nome"></label>
22                             <label class="col-9">@Html.DisplayFor(model => model.Nome)</label>
23                         </div>
24                         <div class="row">
25                             <label class="col-3" asp-for="Descricao"></label>
26                             <label class="col-9">@Html.DisplayFor(model => model.Descricao)</label>
27                         </div>
28                         <div class="row">
29                             <label class="col-3" asp-for="CargoId"></label>
30                             <label class="col-9">@Html.DisplayFor(model => model.Cargo.Nome)</label>
31                         </div>
32                         <div class="row">
33                             <div class="col-3">
34                                 <div class="icheck-material-white">
35                                     <input type="checkbox" asp-for="ExibirHome" disabled />
36                                     <label asp-for="ExibirHome"></label>
37                                 </div>
38                             </div>
39                             <div class="col-3">
40                                 <div class="icheck-material-white">
41                                     <input type="checkbox" asp-for="Ativo" disabled />
42                                     <label asp-for="Ativo"></label>
43                                 </div>
44                             </div>
45                             <div class="col-6 pt-1">
46                                 <label asp-for="OrdemExibicao"></label>
47                                 <label>@Html.DisplayFor(model => model.OrdemExibicao)</label>
48                             </div>
49                         </div>
50                     </div>
51                 </div>
52
53             <form asp-action="Delete" method="post">
54                 <div asp-validation-summary="ModelOnly" class="text-danger"></div>
55                 <input type="hidden" asp-for="Id" />
56                 <h5 class="my-3">Confirma a Exclusão deste Funcionário?</h5>
57                 <div class="form-group">
58                     <input type="submit" value="Excluir" class="btn btn-danger mr-2" />
59                     <a class="btn btn-light" asp-action="Index">Cancelar</a>
60                 </div>
61             </form>
62         </div>
63     </div>
64 </div>
65 </div>
```

## @ Details.cshtml X

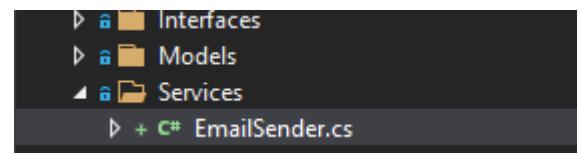
Views > Funcionarios > @ Details.cshtml

```
1  @model RestauranteEtec.Models.Funcionario
2  @{
3      ViewData["Title"] = "Funcionário";
4      Layout = "~/Views/Shared/_LayoutAdmin.cshtml";
5  }
6
7  <div class="row mt-3">
8      <div class="col-lg-12">
9          <div class="card">
10             <div class="card-body">
11                 <div class="card-title">Detalhes do Funcionário</div>
12                 <hr>
13                 <div class="row">
14                     <div class="col-3">
15                         
18                     </div>
19                     <div class="col-9">
20                         <div class="row">
21                             <label class="col-3" asp-for="Nome"></label>
22                             <label class="col-9">@Html.DisplayFor(model => model.Nome)</label>
23                         </div>
24                         <div class="row">
25                             <label class="col-3" asp-for="Descricao"></label>
26                             <label class="col-9">@Html.DisplayFor(model => model.Descricao)</label>
27                         </div>
28                         <div class="row">
29                             <label class="col-3" asp-for="CargoId"></label>
30                             <label class="col-9">@Html.DisplayFor(model => model.Cargo.Nome)</label>
31                         </div>
32                         <div class="row">
33                             <div class="col-3">
34                                 <div class="icheck-material-white">
35                                     <input type="checkbox" asp-for="ExibirHome" disabled />
36                                     <label asp-for="ExibirHome"></label>
37                                 </div>
38                             </div>
39                             <div class="col-3">
40                                 <div class="icheck-material-white">
41                                     <input type="checkbox" asp-for="Ativo" disabled />
42                                     <label asp-for="Ativo"></label>
43                                 </div>
44                             </div>
45                             <div class="col-6 pt-1">
46                                 <label asp-for="OrdemExibicao"></label>
47                                 <label>@Html.DisplayFor(model => model.OrdemExibicao)</label>
48                             </div>
49                         </div>
50                     </div>
51                 </div>
52
53                 <div class="mt-3">
54                     <a class="btn btn-warning mr-3" asp-action="Edit" asp-route-id="@Model.Id">Alterar</a>
55                     <a class="btn btn-light" asp-action="Index">Voltar a Listagem</a>
56                 </div>
57             </div>
58         </div>
59     </div>
60 </div>
```

## ADICIONANDO UM SERVIÇO DE DISPARO DE E-MAIL

Existem várias formas de criação de um disparo de e-mails. Uma pesquisa rápida pela internet, pode mostrar como criar um projeto separado, que será consumido pelos demais para realizar a ação de disparo. Em nosso projeto, vamos fazer uma abordagem mais simples criando uma classe que será responsável por enviar e-mails usando um serviço **SMTP** (Protocolo de Transferência de Correio Simples é o protocolo padrão de envio de mensagens de correio eletrônico através da Internet entre dois dispositivos computacionais, definido na **RFC 821**) com a biblioteca **.Net.Mail**.

Comece adicionando uma pasta ao projeto com o nome **Services**. Adicione uma classe na pasta **Services** com o nome **EmailSender**.



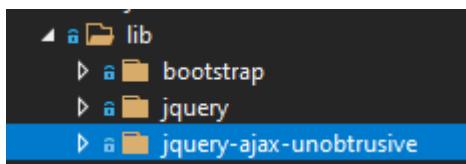
```
C# EmailSender.cs X
Services > C# EmailSender.cs > ...
1  using System.Net;
2  using System.Net.Mail;
3  using System.Threading.Tasks;
4
5  namespace RestauranteEtec.Services
6  {
7      0 references
8      public class EmailSender
9      {
10          0 references
11          public async Task<bool> Mail(string Para, string De, string Assunto, string Mensagem)
12          {
13              var m = new MailMessage()
14              {
15                  Subject = Assunto,
16                  Body = Mensagem,
17                  IsBodyHtml = true
18              };
19              MailAddress para = new MailAddress(Para);
20              m.To.Add(para);
21              m.From = new MailAddress(De);
22              m.Sender = para;
23              var smtp = new SmtpClient
24              {
25                  Host = "smtp.gmail.com",
26                  Port = 587,
27                  Credentials = new NetworkCredential("gallozord@gmail.com", "@Etec123#"),
28                  EnableSsl = true
29              };
30              try
31              {
32                  await smtp.SendMailAsync(m);
33                  return true;
34              }
35              catch
36              {
37                  return false;
38              }
39          }
40      }
```

Nossa classe é bem simples, criamos um **Mail Message**, que representa uma mensagem em formato de **HTML**. Usando um servidor **SMTP**, no meu caso estou usando o **Gmail**, fazemos o disparo do e-mail.

Aqui fica uma **observação**, para usar isso, é necessário que o seu e-mail esteja configurado para permitir “App menos Seguros”, esse procedimento é explicado nesse vídeo bem curto: <https://www.youtube.com/watch?v=V4escC0wRYM&t=57s>.

Agora vamos adicionar ao projeto uma biblioteca **jquery** que nos permite fazer chamadas **AJAX** simplificadas, **ajax.unobtrusive**. Mais informações sobre a biblioteca em: <https://www.learnrazorpages.com/razor-pages/ajax/unobtrusive-ajax>.

**AJAX** é um acrônimo para **Asynchronous Javascript and XML**, ou **JavaScript e XML Assíncronos**. É uma técnica utilizada pelos desenvolvedores para criar aplicações capazes de manter a comunicação assíncrona com o servidor. Em nosso caso, podemos fazer chamadas a ações do **Controller**, sem que a **View** seja recarregada.



Faça o download do **zip** disponibilizado [aqui](#), descompacte a pasta do **zip**, faça uma cópia e cole dentro da pasta **wwwroot\lib** no projeto.

Agora vamos adicionar o **jquery** no arquivo **\_ValidationScriptsPartial.cshtml**, que está na pasta **ViewsShared**:

```
@/_ValidationScriptsPartial.cshtml X
Views > Shared > @_ValidationScriptsPartial.cshtml
1 <script src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>
2 <script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.min.js"></script>
3 <script src="~/lib/jquery-ajax-unobtrusive/jquery.unobtrusive-ajax.js"></script>
```

Com isso, basta incluir esta **PartialView** nas páginas que precisamos usar requisições com **AJAX**.

Vamos agora criar o nosso **ContatosController**, assim como fizemos com os demais. A grande diferença dessa controller, é que ele não irá possuir todos os métodos de um **CRUD**, já que não existe um “Cadastro de Contatos”, já que dados inseridos na tabela de contatos depende de clientes usarem essa opção (“Contato”) diretamente no site.

```
dotnet-aspnet-codegenerator controller -name ContatosController -outDir Controllers
```

Agora vamos incluir um objeto do tipo **contatoDAL**, para realizar as operações de **CRUD** com a tabela de **Contatos**. Só para esclarecimento, não teremos todas as operações de **CRUD**, pois um contato não pode ser cadastro no banco por um usuário, ele será incluído, quando um visitante do site, preencher o formulário de contato contido na página.

Como comentei acima, não vamos ter uma página para o usuário cadastrar “**Create**” um contato. Vamos criar um **Create (POST)** para receber nossa requisição **AJAX**, que vamos criar logo mais na página de **Contatos**.

Modifique o código do **Controller**, conforme apresentado na imagem abaixo:

```

C# ContatosController.cs X
Controllers > C# ContatosController.cs > ...
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using Microsoft.AspNetCore.Mvc;
6  using RestauranteEtec.DAL;
7  using RestauranteEtec.Models;
8  using RestauranteEtec.Services;
9
10 < namespace RestauranteEtec.Controllers
11 {
12     0 references
13     public class ContatosController : Controller
14     {
15         1 reference
16         private readonly ContatoDAL contatoDAL = new ContatoDAL();
17
18         0 references
19         public IActionResult Index()
20         {
21             return View();
22         }
23
24         // POST: ContatosController/Create
25         [HttpPost]
26         [ValidateAntiForgeryToken]
27         0 references
28         public async Task<JsonResult> Create([Bind] Contato contato)
29         {
30             if (ModelState.IsValid)
31             {
32                 contatoDAL.Add(contato);
33                 var email = new EmailSender();
34                 var assunto = "Contato feito no RestauranteEtec";
35                 var mensagem = "<b>Pessoa: </b> " + contato.NomePessoa;
36                 mensagem += "<br><b>E-mail: </b> " + contato.EmailPessoa;
37                 mensagem += "<br><b>Assunto: </b> " + contato.Assunto;
38                 mensagem += "<br><b>Mensagem: </b> " + contato.Mensagem;
39                 await email.Mail("gallojunior@gmail.com", contato.EmailPessoa, assunto, mensagem);
40                 return Json(data: "done");
41             }
42         }
43     }
44 }

```



No método **Create** é conferido o **Modelo (ModelState.IsValid)**, então adicionamos o contato no banco (**contatoDAL.add(contato)**) e por fim, criamos um objeto do tipo **EmailSender**, criamos uma mensagem contendo o nome da pessoa, o **email**, assunto e a mensagem que o visitante do site preencheu, e usamos o **objeto email** para enviar esta mensagem ao responsável do restaurante, no meu caso, enviei para outro e-mail meu. Caso tudo ocorra corretamente, a função devolve um **Json**, informando que tudo ocorreu como devia, se ocorrer algum problema, retorna um erro. Na linha 35, coloque um e-mail próprio e existente, que receberá os contados do site.

É importante notar em dentro da variável mensagem foram incluídos elementos de **HTML**, isso ocorre porque codificamos o disparador de **email** para enviar mensagens com texto formatado dessa forma, ou seja, você poderia criar uma página inteira e enviá-la de forma mais elegante ao usuário.

Agora precisamos alterar o código da página **Contatos** que está dentro da pasta **Views\Home**:

```

@ Contatos.cshtml x
Views > Home > @ Contatos.cshtml
1  @model RestauranteEtec.Models.Contato
2  @{
3      ViewData["Title"] = "Contatos";
4  }
5
6  <section class="hero-wrap hero-wrap-2" style="background-image: url(@Url.Content("~/images/bg_5.jpg")); data-stellar-background-ratio="0.5">
7      <div class="overlay"></div>
8      <div class="container">
9          <div class="row no-gutters slider-text align-items-end justify-content-center">
10             <div class="col-md-9 ftco-animate text-center mb-5">
11                 <h1 class="mb-2 bread">Entre em Contato</h1>
12                 <p class="breadcrumbs"><span class="mr-2"><a asp-action="Index">Home <i class="fa fa-chevron-right"></i></a></span>
13                     <span>Contatos <i class="fa fa-chevron-right"></i></span></p>
14             </div>
15         </div>
16     </div>
17 </section>
18
19 <section class="ftco-section contact-section bg-light">
20     <div class="containen">
21         <div class="row d-flex contact-info">
22             <div class="col-md-12">
23                 <h2 class="h4 font-weight-bold">Informações de Contatos</h2>
24                 <div>
25                     <div class="w-100"></div>
26                     <div class="col-md-3 d-flex">
27                         <div class="dbox">
28                             <p><span>Endereço:</span> 198 West 21th Street, Suite 721 New York NY 10016</p>
29                         </div>
30                     </div>
31                     <div class="col-md-3 d-flex">
32                         <div class="dbox">
33                             <p><span>Phone:</span> <a href="tel://1234567920">+ 1235 2355 98</a></p>
34                         </div>
35                     </div>
36                     <div class="col-md-3 d-flex">
37                         <div class="dbox">
38                             <p><span>Email:</span> <a href="mailto:info@yoursite.com">info@yoursite.com</a></p>
39                         </div>
40                     </div>
41                     <div class="col-md-3 d-flex">
42                         <div class="dbox">
43                             <p><span>Website:</span> <a href="#">yoursite.com</a></p>
44                         </div>
45                     </div>
46                 </div>
47             </div>
48         </div>
49     </section>
50     <section class="ftco-section ftco-no-pt contact-section">
51         <div class="container">
52             <div class="row d-flex align-items-stretch no-gutters">
53                 <div class="col-md-6 p-5 order-md-last">
54                     <h2 class="h4 mb-5 font-weight-bold">Entre em Contato Conosco:</h2>
55                     <form asp-action="Create" asp-controller="Contatos" method="post" data-ajax="true"
56                         data-ajax-method="post" data-ajax-complete="completed" data-ajax-failure="failed">
57                         <div asp-validation-summary="ModelOnly" class="text-danger" hidden></div>
58                         <div class="form-group">
59                             <input asp-for="NomePessoa" class="form-control" placeholder="Seu Nome" />
60                             <span asp-validation-for="NomePessoa" class="text-danger"></span>
61                         </div>
62                         <div class="form-group">
63                             <input asp-for="EmailPessoa" class="form-control" placeholder="Seu E-mail" />
64                             <span asp-validation-for="EmailPessoa" class="text-danger"></span>
65                         </div>
66                         <div class="form-group">
67                             <input asp-for="Assunto" class="form-control" placeholder="Assunto" />
68                             <span asp-validation-for="Assunto" class="text-danger"></span>
69                         </div>
70                         <div class="form-group">
71                             <textarea asp-for="Mensagem" cols="30" rows="7" class="form-control" placeholder="Mensagem"></textarea>
72                             <span asp-validation-for="Mensagem" class="text-danger"></span>
73                         </div>
74                         <div class="form-group">
75                             <input type="submit" value="Enviar Mensagem" class="btn btn-primary py-3 px-5" />
76                         </div>
77                     </form>
78                 </div>
79                 <div class="col-md-6 d-flex align-items-stretch">
80                     <div id="map">
81                         <iframe src="https://www.google.com/maps/embed?pb=!m1!1m8!1m3!1d14745.589106937443!2d-48.5461716!3d-22.489273!3m2!1i1024!2i
82                         </div>
83                 </div>
84             </div>
85         </div>
86     </section>
87
88     @section Scripts{
89         @await Html.RenderPartialAsync("_ValidationScriptsPartial");
90         <script src="https://cdnjs.cloudflare.com/ajax/libs/sweetalert/2.1.2/sweetalert.min.js"></script>
91
92         <script type="text/javascript">
93             completed = function (xhr) {
94                 swal({
95                     title: "Obrigado!",
96                     text: "Agradecemos seu contato e retornaremos em breve!!",
97                     icon: "success",
98                     button: "OK"
99                 });
100                 document.getElementById("formContato").reset();
101             }
102             failed = function (xhr) {
103                 swal({
104                     title: "Problemas ao Enviar!",
105                     text: "Ocorreu um problema no envio de sua mensagem! Por favor, tente novamente em alguns instantes!",
106                     icon: "warning",
107                     button: "OK"
108                 });
109             }
110         </script>
111     }

```

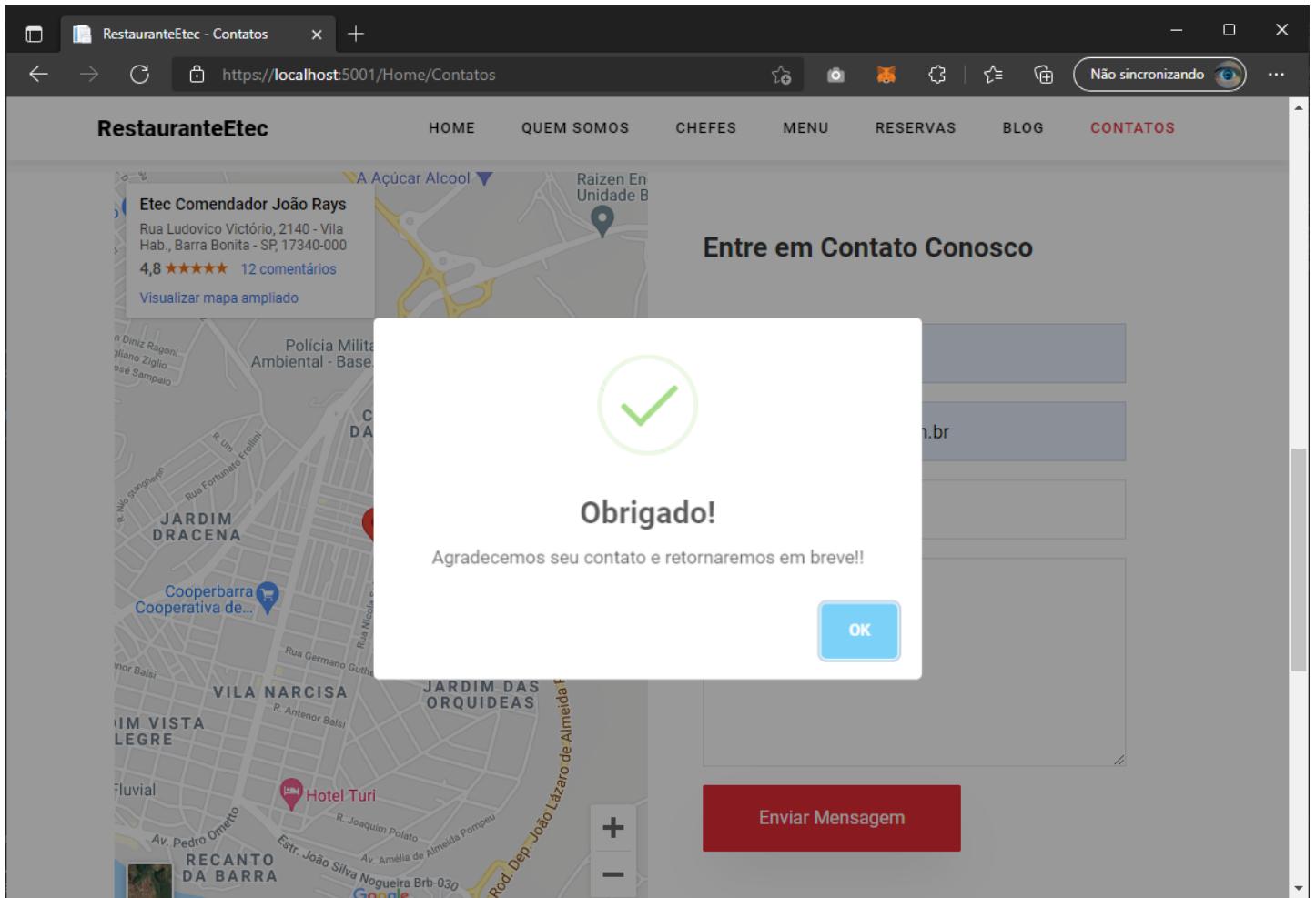
A linha 81, é um **iframe**, que pode ser adquirido no site do GoogleMaps e integrado a qualquer site. Como consiste de uma linha muito grande para o print, segue abaixo seu código para copiar e colar:

```
<iframe src="https://www.google.com/maps/embed?pb=!1m14!1m8!1m3!1d14745.589106937443!2d-48.5461716!3d-22.489273!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x0%3A0xd1b954a115160672!2sEtec%20Comendador%20Jo%C3%A3o%20Rays!5e0!3m2!1spt-BR!2sbr!4v1620350764544!5m2!1spt-BR!2sbr" width="100%" height="100%" style="border:0;" allowfullscreen="" loading="lazy"></iframe>
```

Agora é só executar e ir até a página de Contatos:

The screenshot shows a web page for 'RestauranteEtec - Contatos'. At the top, there's a navigation bar with links to HOME, QUEM SOMOS, CHEFES, MENU, RESERVAS, BLOG, and CONTATOS. Below the navigation is a map of a neighborhood in São Paulo. A red pin marks the location of 'Etec Comendador João Rays' at Rua Ludovico Victório, 2140 - Vila Hab., Barra Bonita - SP, 17340-000. The map also shows other landmarks like 'Pólicia Militar Ambiental - Base...', 'COLINA DA BARRA', and 'PORTAL SÃO JOSÉ DA BARRA'. To the right of the map is a contact form with the heading 'Entre em Contato Conosco'. It contains four input fields: 'Galloman', 'lab17340@lab17340.com.br', 'Testando', and 'Amei seu site'. At the bottom right of the form is a large red button labeled 'Enviar Mensagem'.

Preencha o formulário com alguns dados e pressione o botão **[Enviar Mensagem]**



Agora acessando o e-mail que foi configurado no **ContatosController** como responsável pelo Restaurante, vamos ter:

### Contato feito no RestauranteEtec ➔ Caixa de entrada x

gallozord@gmail.com  
para mim ▾ 18:20 (há 1 minuto) ⭐ ↗ :

Pessoa: Galloman  
E-mail: [lab17340@lab17340.com.br](mailto:lab17340@lab17340.com.br)  
Assunto: Testando  
Mensagem: Amei seu site

Responder Encaminhar

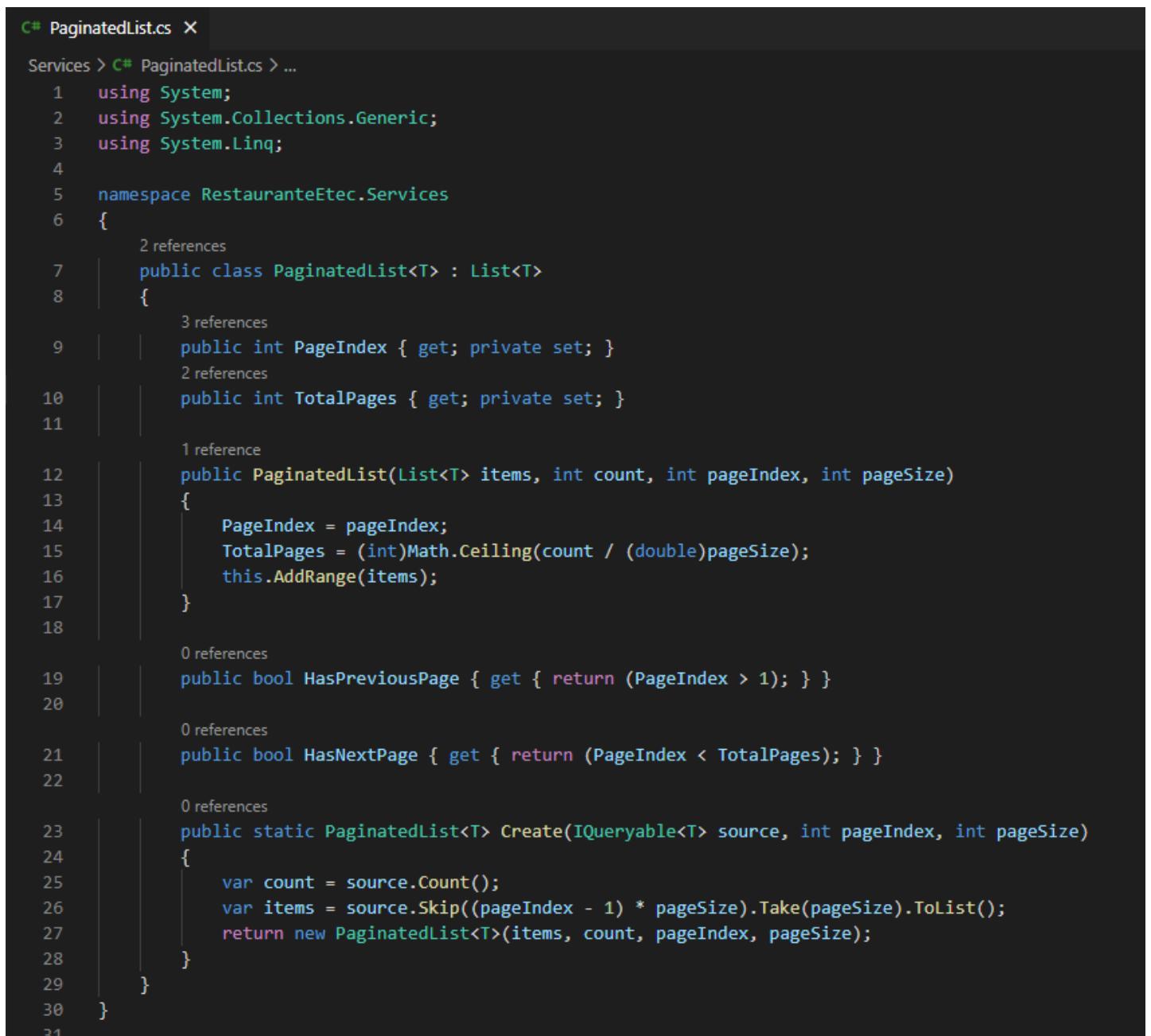
O próximo passo, será a alteração da **Action Index** do **ContatosController**, para permitir ao administrador do Restaurante, verificar os contatos realizados e respondê-los pelo próprio site.

Porém, como podemos ter um número grande de contatos, vamos adicionar um recurso de paginação, ou seja, vamos separar os registros por páginas, e adicionar a tabela de exibição dois botões de navegação, além da possibilidade de pesquisar os contatos pelo nome da pessoa ou assunto.

Para que seja possível esse procedimento, poderíamos utilizar diversos recursos prontos, como o **X.PaginatedList** ou mesmo um **DataTable.js**, porém vamos fazer usando o material de apoio da Microsoft, que orienta ao desenvolvimento de uma classe auxiliar e alguns códigos de aplicação

(<https://docs.microsoft.com/pt-br/aspnet/core/data/ef-mvc/sort-filter-page?view=aspnetcore-5.0>). Além disso, vamos também utilizar nossa biblioteca **AJAX** para que a cada requisição de página, apenas a tabela que exibe os dados seja atualizada, e não a página inteira, para isso vamos precisar usar uma **PartialView** (Exibição Parcial).

Vamos começar adicionando a pasta **Services** uma nova classe com o nome **PaginatedList**, que você deve codificar conforme a imagem abaixo:



The screenshot shows a code editor window with the file 'PaginatedList.cs' open. The code defines a generic class 'PaginatedList<T>' that implements 'List<T>'. It includes properties for 'PageIndex' and 'TotalPages', and a constructor that takes a list of items, a count, a page index, and a page size. It also includes methods for checking if there is a previous or next page, and a static method for creating a paginated list from a queryable source.

```
C# PaginatedList.cs X
Services > C# PaginatedList.cs > ...
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4
5  namespace RestauranteEtec.Services
6  {
7      2 references
8      public class PaginatedList<T> : List<T>
9      {
10          3 references
11          public int PageIndex { get; private set; }
12          2 references
13          public int TotalPages { get; private set; }
14
15          1 reference
16          public PaginatedList(List<T> items, int count, int pageIndex, int pageSize)
17          {
18              PageIndex = pageIndex;
19              TotalPages = (int)Math.Ceiling(count / (double)pageSize);
20              this.AddRange(items);
21          }
22
23          0 references
24          public bool HasPreviousPage { get { return (PageIndex > 1); } }
25
26          0 references
27          public bool HasNextPage { get { return (PageIndex < TotalPages); } }
28
29          0 references
30          public static PaginatedList<T> Create(IQueryable<T> source, int pageIndex, int pageSize)
31          {
32              var count = source.Count();
33              var items = source.Skip((pageIndex - 1) * pageSize).Take(pageSize).ToList();
34              return new PaginatedList<T>(items, count, pageIndex, pageSize);
35          }
36      }
37  }
```

Agora vamos voltar ao **ContatosController** e fazer algumas mudanças:

```

0 references
12     public class ContatosController : Controller
13     {
14         3 references
15         private readonly ContatoDAL contatoDAL = new ContatoDAL();
16         2 references
17         private readonly int pageSize = 5;
18         // GET: Contatos
19         0 references
20         public ActionResult Index()
21         {
22             var contatos = PaginatedList<Contato>.Create(contatoDAL.GetAll().AsQueryable(), 1, pageSize);
23             return View(contatos);

```

A linha 15, define que cada página de nossa tabela deve mostrar 5 linhas, esse número posteriormente deve ser aumentado, estamos usando 5 apenas para poder ver a paginação em funcionamento, eu recomendo os valores 10 ou 25, a seu critério.

No código do **Index**, usamos nosso serviço de **paginação** com o **model Contato**, pesquisando com o **GetAll** do **contatoDAL** todos os contados cadastrados, como estamos na página **Index**, o **número 1**, define a exibição da primeira página e o **pageSize** o número de linhas por página. No caso o **PaginatedList**, irá retornar apenas o resultado de **3 contatos do GetAll()**.

Precisamos agora criar um método, que irá ser chamado através de uma requisição **AJAX**, e que deve retornar apenas os dados de contatos, de acordo com uma pesquisa ou pedido de ordenação ou mesmo movimentação de páginas, lembrado que no **Index** estamos exibindo inicialmente a página 1, o que ocorrerá se o usuário quiser ver os dados que ficaram nas outras páginas?

Dessa forma, vamos criar uma **PartialView**, começando pelo lado do **Controller** e posteriormente o arquivo na **View**. Vamos adicionar abaixo do código da **Index**, a **Action** da imagem:

```

0 references
24     public PartialViewResult GetGrid(string sortOrder, string searchString, int? pageNumber)
25     {
26         ViewData["NameSortParm"] = String.IsNullOrEmpty(sortOrder) ? "name_desc" : "";
27         ViewData["DateSortParm"] = sortOrder == "Date" ? "date_desc" : "Date";
28         ViewData["CurrentFilter"] = searchString;
29
30         var contatos = contatoDAL.GetAll().AsQueryable();
31
32         if (!String.IsNullOrEmpty(searchString))
33         {
34             contatos = contatos.Where(s => s.NomePessoa.Contains(searchString)
35             || s.Assunto.Contains(searchString));
36         }
37         switch (sortOrder)
38         {
39             case "name_desc":
40                 contatos = contatos.OrderByDescending(s => s.NomePessoa);
41                 break;
42             case "Date":
43                 contatos = contatos.OrderBy(s => s.DataContato);
44                 break;
45             case "date_desc":
46                 contatos = contatos.OrderByDescending(s => s.DataContato);
47                 break;
48             default:
49                 contatos = contatos.OrderBy(s => s.NomePessoa);
50                 break;
51         }
52         var retorno = PaginatedList<Contato>.Create(contatos, pageNumber ?? 1, pageSize);
53         return PartialView("_Grid", retorno);
54     }

```

Nesse código definimos um filtro de ordenação por nome e um de data, e filtro de pesquisa corrente (atual). A **Action**, recebe a cada requisição um texto de ordenação, um texto de pesquisa e o número de página que é opcional.

Após preencher os **ViewDatas**, com os valores da requisição, cria uma lista de contatos, usando o **GetAll()**.

Se o filtro de pesquisa veio preenchido, aplica uma pesquisa nos contatos, procurando o texto informado nos campos **NomePessoa** ou **Assunto**.

Em seguida, usando um condicional **switch**, verifica se o filtro de ordenação está preenchido, e ordena os dados pesquisados, por **NomePessoa** ou **DataCadastro**, em ordem crescente ou decrescente.

E por fim, usa o **PaginatedList**, com os dados de contatos filtrados, para recuperar apenas a página que o usuário solicitou ou a página 1, caso ele não tenha solicitado mudança de página. E devolve não uma página completa, mas uma **PartialView**, de nome “**\_Grid**”, que é o arquivo que vamos logo após o **Index**.

Agora vamos criar a **View Index** e a **PartialView \_Grid**, dentro de **Views\Contatos**. Abra seu terminal e execute os comandos abaixo:

Para criar a **Index**:

```
dotnet-aspnet-codegenerator view Index List -m Contato -l  
"~/Views/Shared/_LayoutAdmin.cshtml" -outDir "Views\Contatos"
```

Para criar a **\_Grid**:

```
dotnet-aspnet-codegenerator view _Grid Empty -partial -outDir  
"Views\Contatos"
```

Vamos alterar um pouquinho o código criado automaticamente.

## @ Index.cshtml ×

Views > Contatos > @ Index.cshtml

```
1  @{
2      ViewData["Title"] = "Contatos";
3      Layout = "~/Views/Shared/_LayoutAdmin.cshtml";
4  }
5
6  <div class="row mt-3">
7      <div class="col-lg-12">
8          <div class="card">
9              <div class="card-body">
10                 <div class="card-title">Lista de Contatos</div>
11                 <hr>
12                 <form class="mb-3" data-ajax="true" data-ajax-method="get" data-ajax-update="#Grid"
13                     data-ajax-mode='replace' data-ajax-url="@Url.Action("GetGrid", "Contatos")">
14                     <div class="form-row">
15                         <div class="input-group col-6">
16                             <input name="searchString" type="text" class="form-control"
17                                 placeholder="Digite um nome ou assunto " aria-label=""
18                                 aria-describedby="basic-addon1" value="@ViewData["CurrentFilter"]">
19                             <div class="input-group-append">
20                                 <button class="btn btn-light" type="submit">Pesquisar</button>
21                             </div>
22                         </div>
23                         <a class="btn btn-info" asp-action="Index">Exibir Tudo</a>
24                     </div>
25                 </form>
26
27                 <div id="Grid">
28                     <partial name="_Grid" />
29                 </div>
30             </div>
31         </div>
32     </div>
33 </div>
34
35 @section Scripts{
36     @await Html.RenderPartialAsync("_ValidationScriptsPartial");
37 }
38
```

Agora vamos alterar a **\_Grid**, e insira o código abaixo:

```

@_Grid.cshtml ×
Views > Contatos > @_Grid.cshtml
1 @model RestauranteEtec.Services.PaginatedList<RestauranteEtec.Models.Contato>
2 <table class="table table-striped table-hover">
3   <thead>
4     <tr>
5       <th>
6         <a asp-action="Index" asp-route-sortOrder="@ViewData["DateSortParm"]"
7           data-ajax="true" data-ajax-method="get" data-ajax-update="#Grid" data-ajax-mode='replace'
8           data-ajax-url="@Url.Action("GetGrid","Contatos", new { sortOrder = ViewBag.DateSortParm, searchString = ViewBag.CurrentFilter })">
9           Data do Contato
10          @if (ViewBag.DateSortParm == "Date")
11            {<i class="zmdi zmdi-sort-amount-desc"></i>}
12          @if (ViewBag.DateSortParm == "date_desc")
13            {<i class="zmdi zmdi-sort-amount-asc"></i>}
14        </a>
15      </th>
16      <th>
17        <a asp-action="Index" asp-route-sortOrder="@ViewData["NameSortParm"]"
18          data-ajax="true" data-ajax-method="get" data-ajax-update="#Grid" data-ajax-mode='replace'
19          data-ajax-url="@Url.Action("GetGrid","Contatos", new { sortOrder = ViewBag.NameSortParm, searchString = ViewBag.CurrentFilter })">
20           Nome do Contato
21          @if (ViewBag.NameSortParm == "") 
22            {<i class="zmdi zmdi-sort-desc"></i>}
23          @if (ViewBag.NameSortParm == "name_desc")
24            {<i class="zmdi zmdi-sort-asc"></i>}
25        </a>
26      </th>
27      <th>Assunto</th>
28      <th>Ações</th>
29    </tr>
30  </thead>
31  <tbody>
32    @foreach (var item in Model)
33    {
34      <tr>
35        <td>
36          @Html.DisplayFor(modelItem => item.DataContato)
37        </td>
38        <td>
39          @Html.DisplayFor(modelItem => item.NomePessoa)
40        </td>
41        <th>@Html.DisplayFor(modelItem => item.Assunto)</th>
42        <td>
43          <a asp-action="Details" class="mr-3 text-white" asp-route-id="@item.Id">
44            <i class="zmdi zmdi-zoom-in" title="Detalhes"></i>
45          </a>
46          <a asp-action="Answer" class="mr-3 text-warning" asp-route-id="@item.Id">
47            <i class="zmdi zmdi-edit" title="Responder"></i>
48          </a>
49          <a asp-action="Delete" class="text-danger" asp-route-id="@item.Id">
50            <i class="zmdi zmdi-delete" title="Excluir"></i>
51          </a>
52        </td>
53      </tr>
54    }
55  </tbody>
56 </table>
57 <div class="text-center mt-3">
58  @{
59    var prevDisabled = !Model.HasPreviousPage ? "disabled" : "";
60    var nextDisabled = !Model.HasNextPage ? "disabled" : "";
61  }
62  <a asp-action="Index" asp-route-pageNumber="@((ModelPageIndex - 1))"
63    data-ajax="true" data-ajax-method="get" data-ajax-update="#Grid" data-ajax-mode='replace'
64    data-ajax-url="@Url.Action("GetGrid","Contatos", new { pageNumber = ModelPageIndex - 1, searchString = ViewBag.CurrentFilter })"
65    class="btn btn-light @prevDisabled">
66    Anterior
67  </a>
68  <a asp-action="Index" asp-route-pageNumber="@((ModelPageIndex + 1))"
69    data-ajax="true" data-ajax-method="get" data-ajax-update="#Grid" data-ajax-mode='replace'
70    data-ajax-url="@Url.Action("GetGrid","Contatos", new { pageNumber = ModelPageIndex + 1, searchString = ViewBag.CurrentFilter })"
71    class="btn btn-light @nextDisabled">
72    Próxima
73  </a>
74 </div>

```

Apesar do código ser extenso, é bem simples.

As linhas entre 6 e 14, estão o cabeçalho da coluna **Data do Contato**, o texto apresentado é um link que ao ser clicado solicita os dados da **Action GetGrid**, passando que o usuário quer ordenar a tabela pela **Data**; essa ordenação pode ser crescente ou decrescente. A linha 10 e 12, servem apenas para exibir um ícone ao lado do texto da **Data do Contato**, informando que os dados foram ordenados.

As linhas entre 17 e 25, são o mesmo que as linhas entre 6 e 14, mas para o campo **Nome do Contato**.

O **foreach** da linha 32, é o mesmo criado automaticamente na Index, que foi transferido aqui para o **Grid**, apenas com a alteração dos ícones e detalhes de informação nas linhas de 43 a 51.

As linhas entre 57 e 74, servem para criar os botões de paginação, mais especificamente:

- As linhas entre 62 e 67, criam um botão que permite ao usuário retornar a página anterior, esse botão é um link, que faz uma chamada **AJAX**, ao **GetGrid**, passando o **número da página – 1**, e o **texto de pesquisa** aplicado.
- As linhas entre 68 e 73, são iguais as linhas 62 e 67, mas criam um botão para solicitar a próxima página de dados, somando 1 a página.

Executando nosso projeto temos seguinte resultado:

The screenshot shows a web application interface for a restaurant management system. The left sidebar has a dark theme with categories like SOCIAL, MENU, and RECURSOS HUMANOS. The main content area is titled 'Lista de Contatos' (Contact List) and displays a table of contacts. The table columns are: DATA DO CONTATO, NOME DO CONTATO, ASSUNTO, and AÇÕES. The data in the table is as follows:

DATA DO CONTATO	NOME DO CONTATO	ASSUNTO	AÇÕES
15/11/2021 07:50	Aline	Precisa de Foto?	[Search, Edit, Delete]
15/11/2021 07:49	Edriano	Show	[Search, Edit, Delete]
15/11/2021 06:08	Galloman	Teste	[Search, Edit, Delete]
15/11/2021 06:12	Galloman	Teste	[Search, Edit, Delete]
15/11/2021 06:20	Galloman	Testando	[Search, Edit, Delete]

At the bottom of the main content area, there are 'ANTERIOR' (Previous) and 'PRÓXIMA' (Next) buttons. The footer of the page says 'Copyright © 2021 Restaurante Etec'.

RestauranteEtec - Contatos X +

https://localhost:5001/Contatos

Não sincronizando

RESTAURANTE ETEC

Dashboard

SOCIAL

Blog

Contatos Novo

Relatos

Reservas Novo

MENU

Categorias

Produtos

RECURSOS HUMANOS

Cargos

Funcionários

**Lista de Contatos**

Digite um nome ou assunto PESQUISAR EXIBIR TUDO

DATA DO CONTATO	NOME DO CONTATO	ASSUNTO	AÇÕES
15/11/2021 07:49	Valdemir	Feriado	

ANTERIOR PRÓXIMA

Copyright © 2021 Restaurante Etec