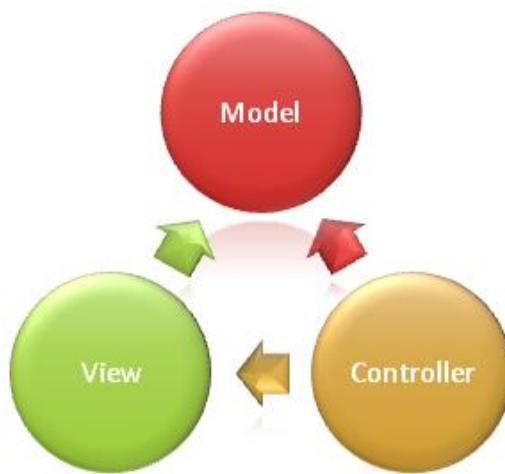


## O que é o padrão MVC?

O padrão de arquitetura MVC (Model-View-Controller) separa um aplicativo em três grupos de componentes principais: Modelos, Exibições e Componentes. 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.

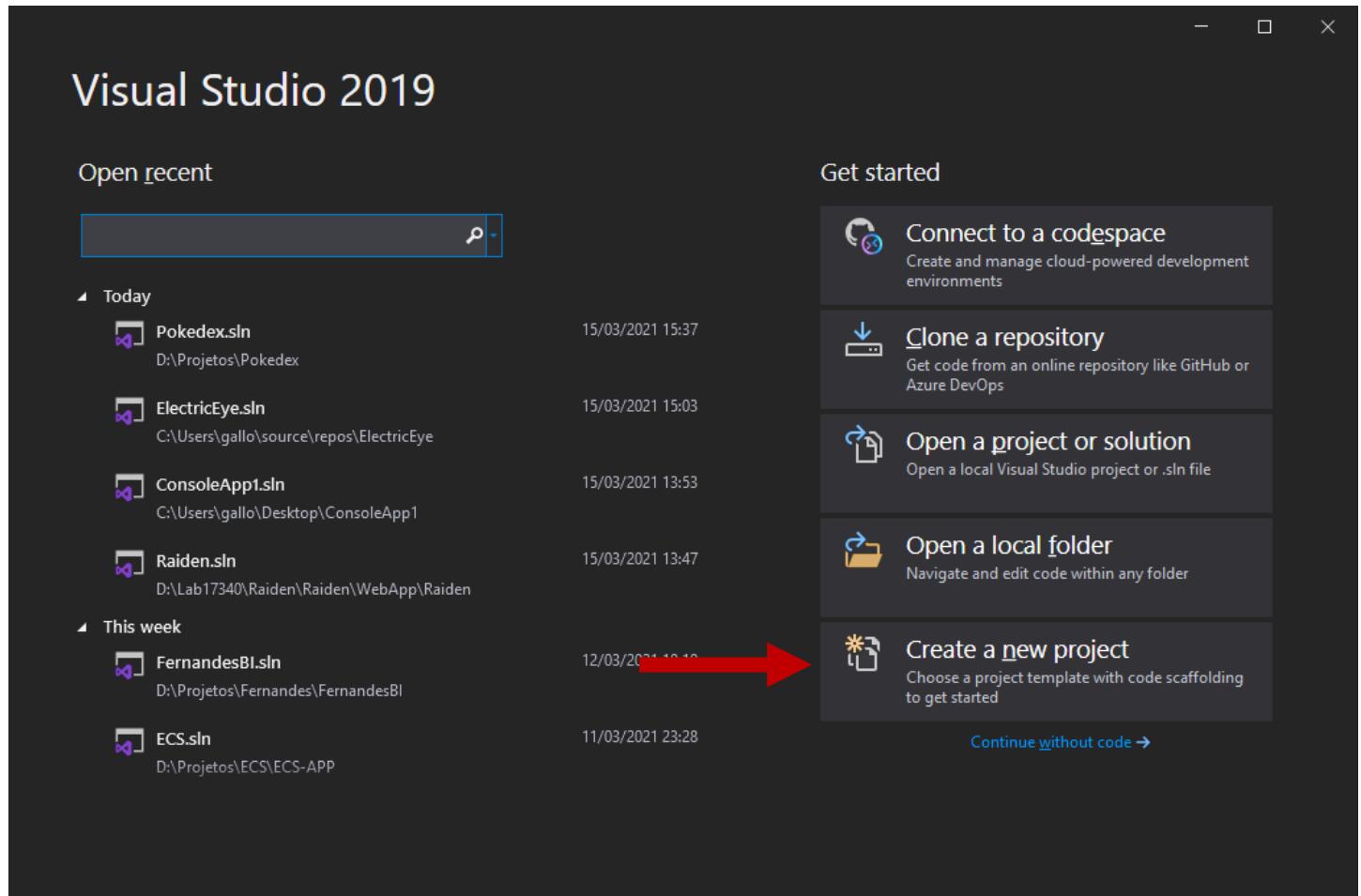
## RESTAURANTEETEC

Iniciaremos o desenvolvimento do projeto RestauranteEtec, pela criação do projeto. Essa criação será apresentada de duas maneiras, usando o Visual Studio Community 2019 e o Visual Studio Code.

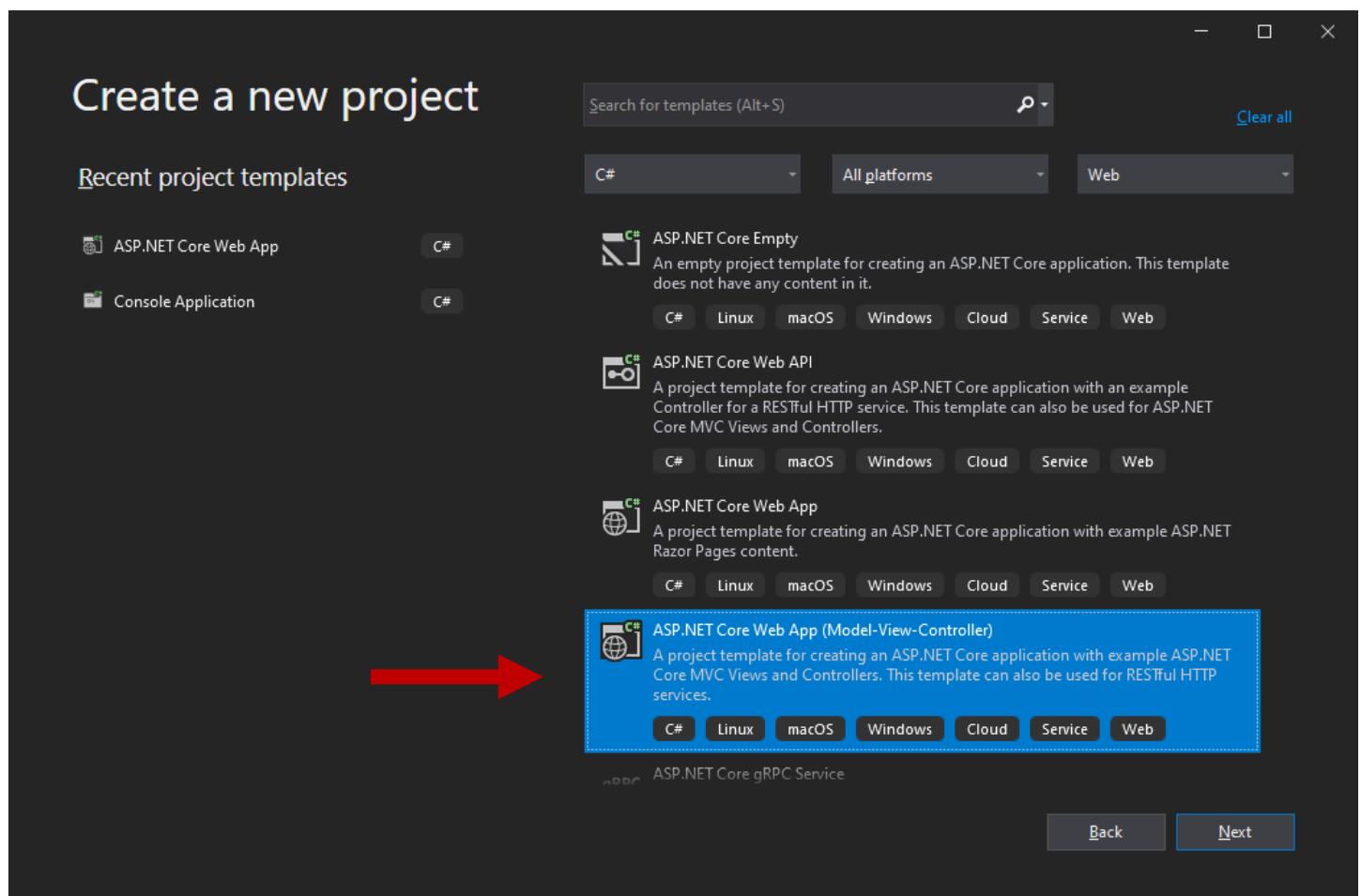
### Criando o projeto no Visual Studio Community 2019:

Clique na opção “Create a new project” ou “Criar um novo projeto”, de acordo com sua instalação.

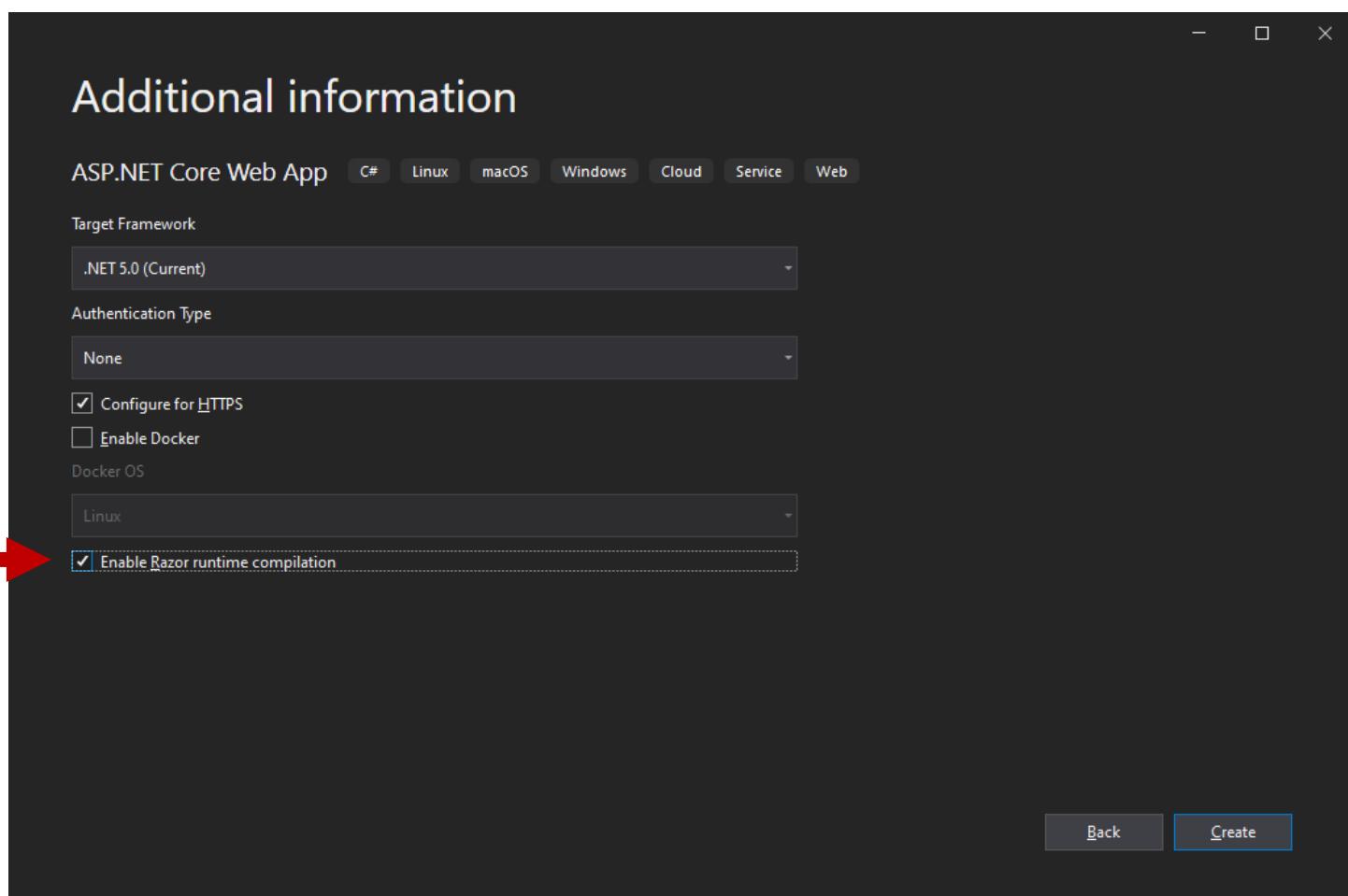
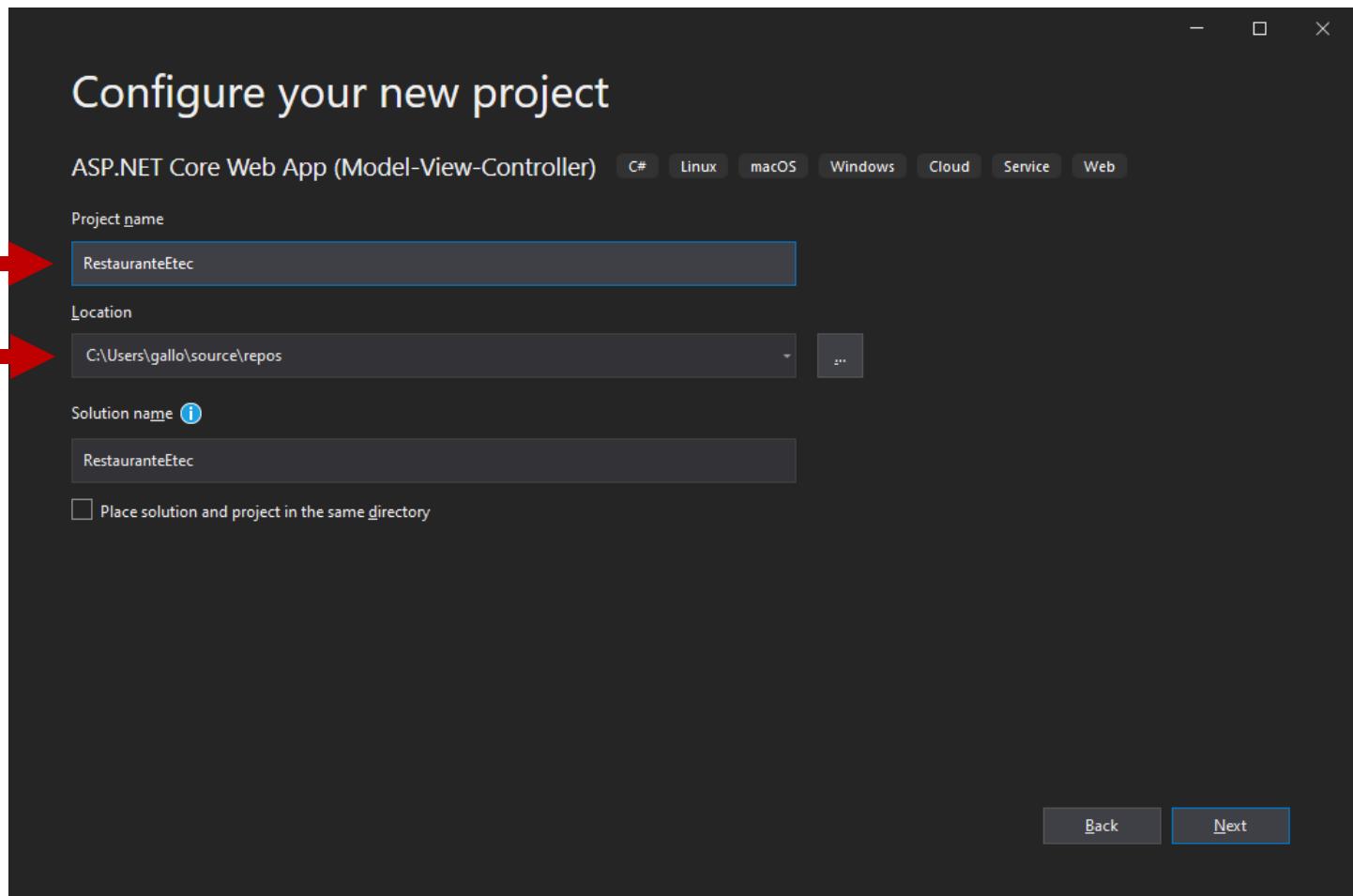
# Visual Studio 2019



Seleciona a opção “ASP.NET Core Web App (Model-View-Controller)” ou “Aplicativo Web ASP.NET Core (Modelo-Exibição-Controlador)”, de acordo com sua instalação, e clique no botão “Next” ou “Próximo”



Em “Project name” ou “Nome do projeto”, digite RestauranteEtec. Mude caso queria a opção “Location” ou “Local” que será onde o projeto ficará salvo e clique no botão “Next” ou “Próximo”.

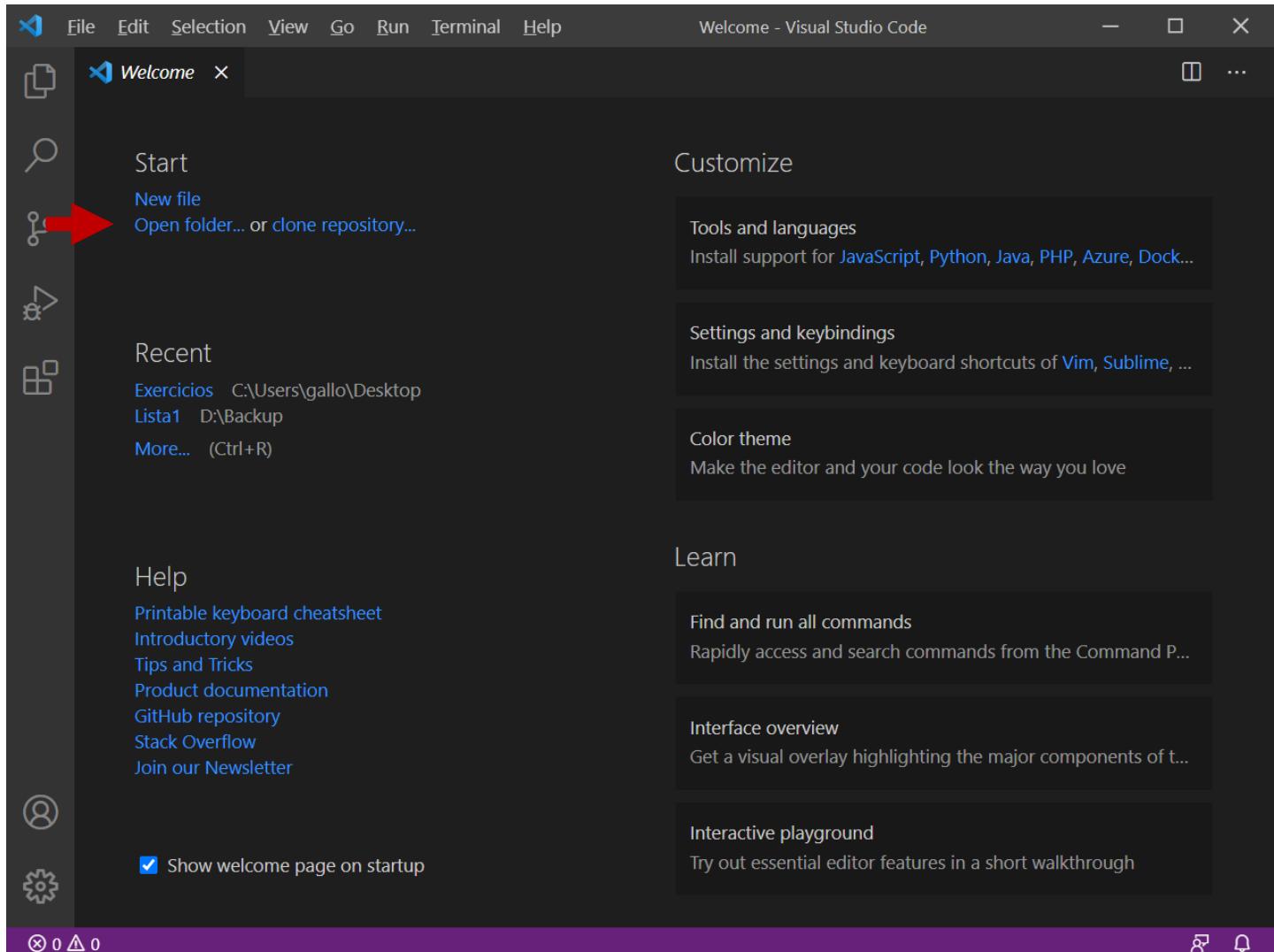


Para finalizar a configuração e criação do projeto, marque a opção “**Enable Razor runtime compilation**” ou “**Habilitar compilação Razor em tempo real**” e clique no botão “**Create**” ou “**Criar**”.

Projeto criado agora é só programar....

## Criando o projeto no Visual Studio Code:

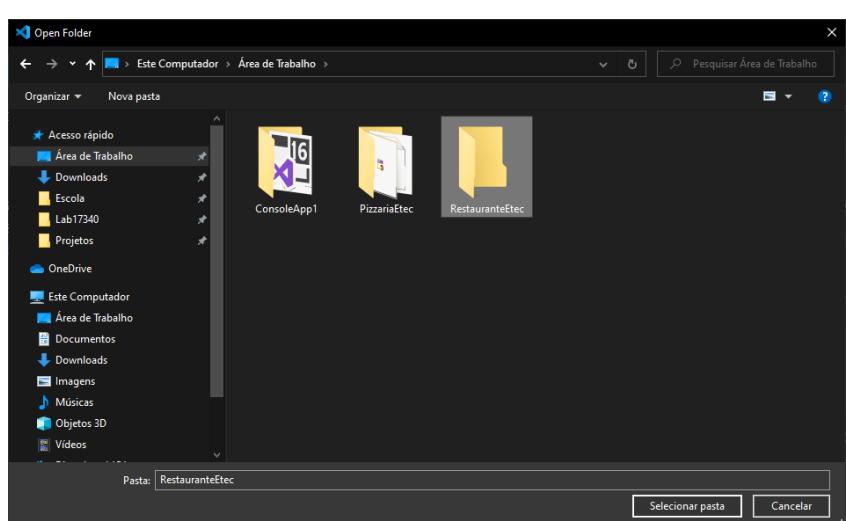
Clique na opção “**Open Folder**”:

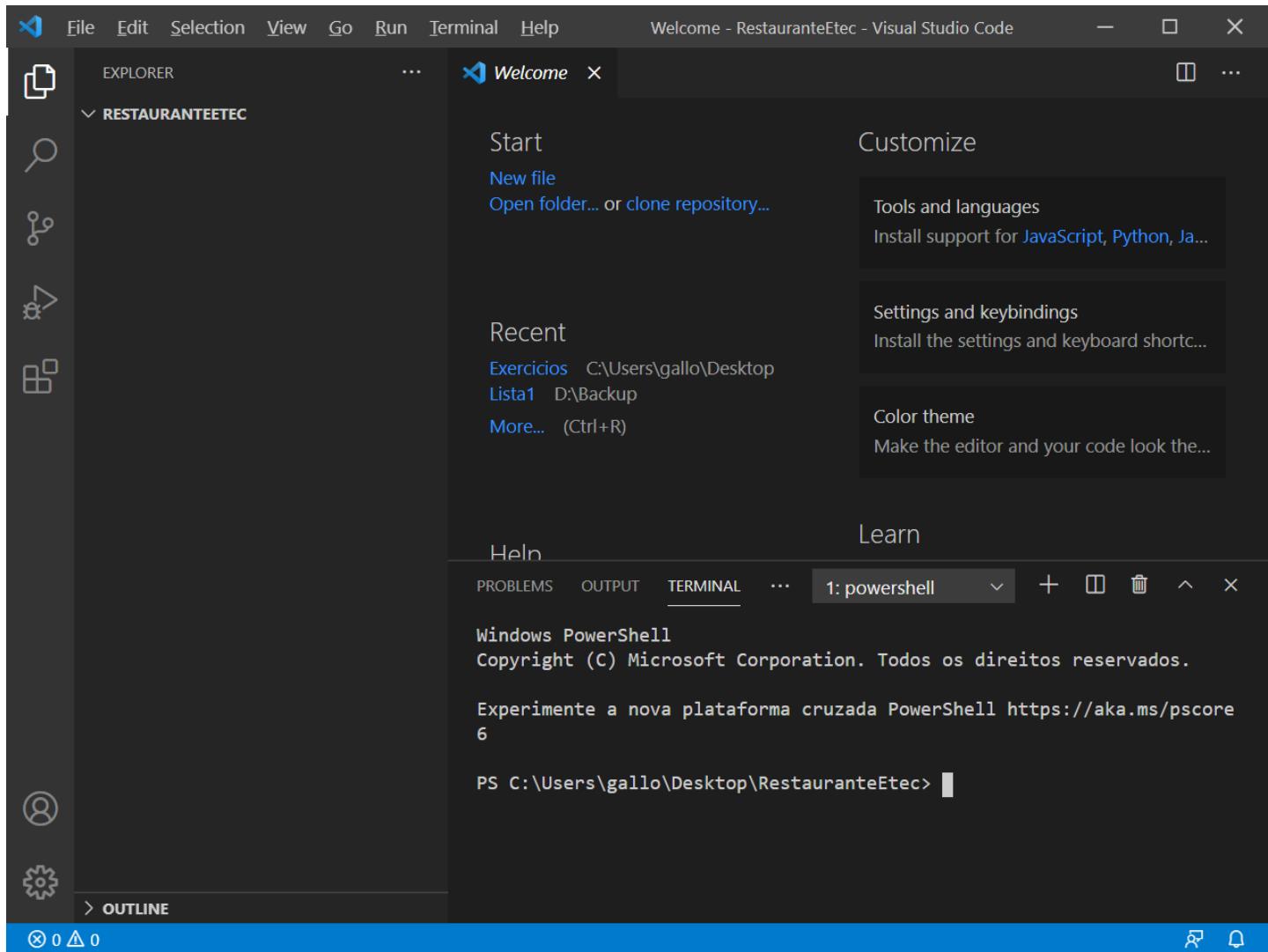


Na janela navega em seu computador, até o local onde deseja salvar seu projeto, e crie uma pasta com o nome **RestauranteEtec**.

Com a pasta selecionada, clique no botão **[Selecionar pasta]**.

O Visual Studio Code, ficará conforme a imagem seguinte, agora vamos usar o terminar. Para abrir o terminal clique: **[Ctrl] + ' (Control + Aspas)**.





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

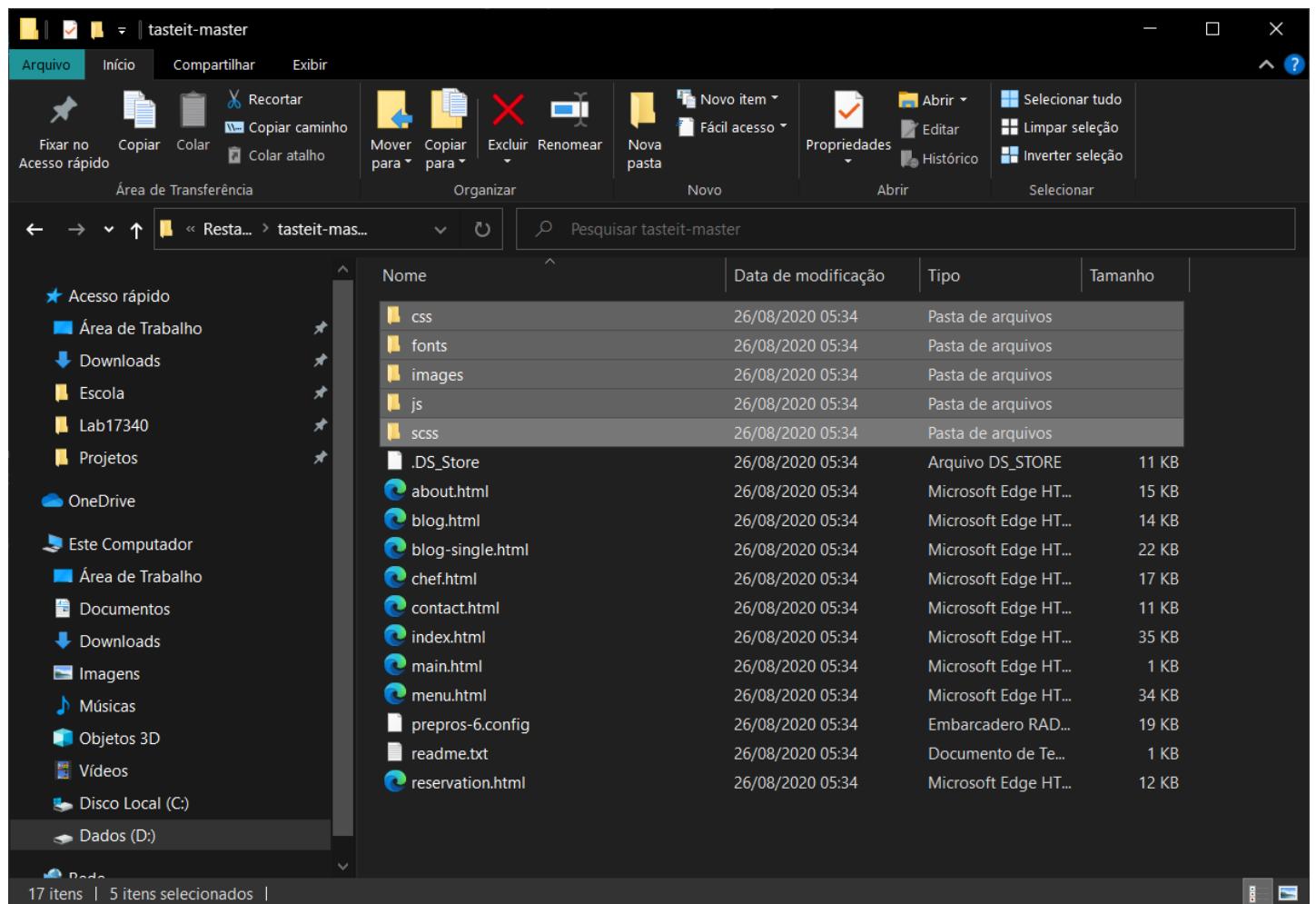
```
dotnet new mvc -lang "C#" -f net5.0
```

No caso aqui, especificamos que o projeto é “MVC” a linguagem de programação é C# e o framework a ser usado o .NET 5.

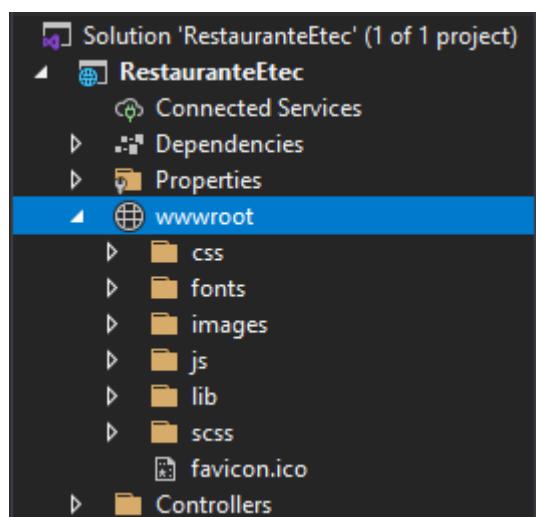
E com isso o projeto está pronto para começar o desenvolvimento.

## 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 pastas e cole na **wwwroot** de nosso projeto.



Caso o Visual Studio exiba uma mensagem que a pasta CSS ou JS já existem no projeto, basta clicar em sobrepor.

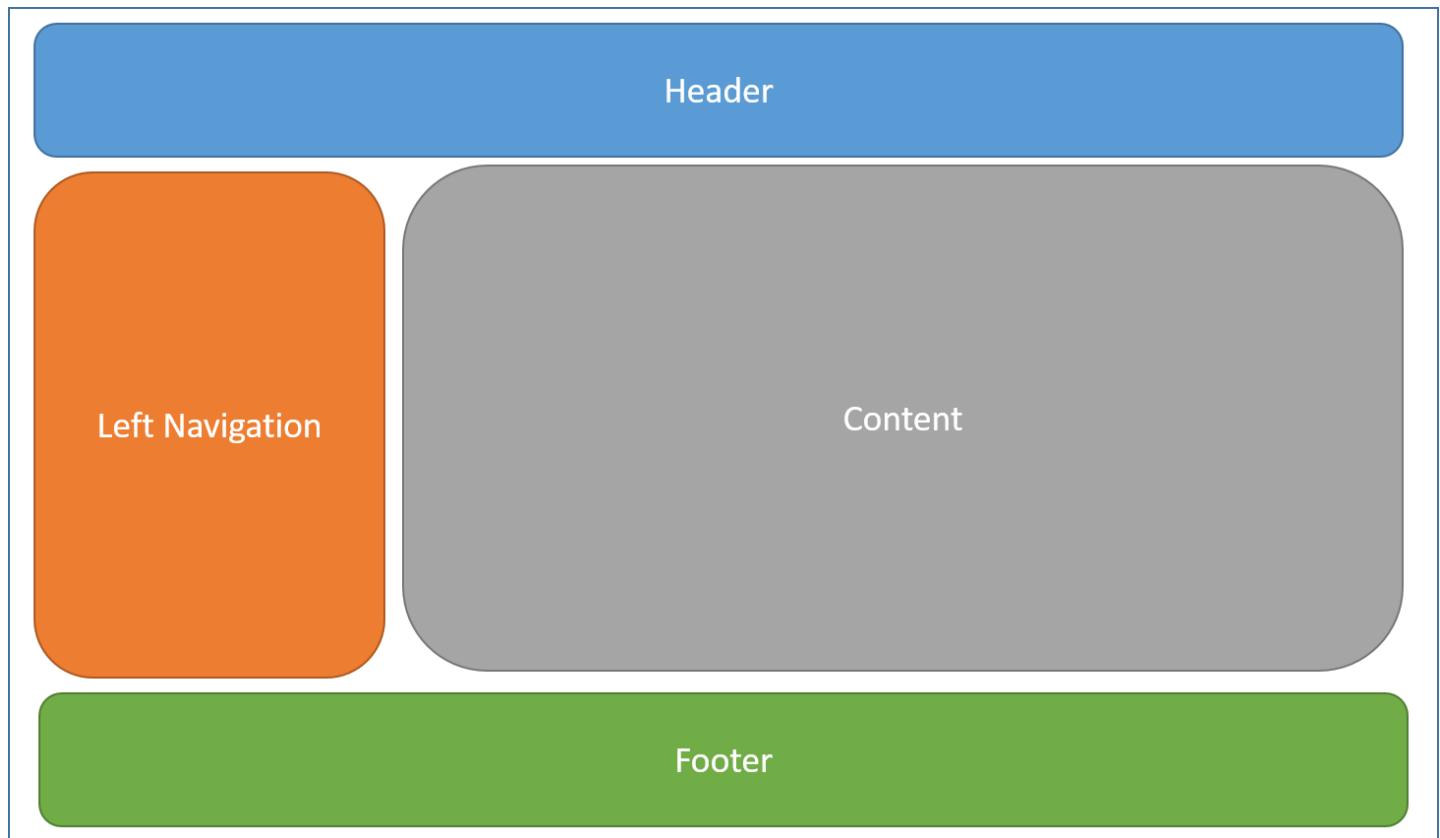


Com isso os arquivos estáticos necessários a correta exibição das páginas estarão disponíveis em seu projeto.

O próximo passo é editar o arquivo **\_Layout.cshtml** que está localizado na pasta **Views\Shared**.

## 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=swap" 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>
        </div>
    </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>
    </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 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>
<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>
    </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 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>
        <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 paradisematic country, in which roasted parts of sentences fly into your
mouth.</p>
                </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" 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>,<br/>
Tomatoe</span></p>
                        </div>
                    </div>
                    <div class="heading-menu text-center ftco-animate">
                        <h3>Beef Roast Source</h3>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>
```

```
        </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" style="background-image: url(@Url.Content("~/images/breakfast-3.jpg"));">
    <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>
    <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" style="background-image: url(@Url.Content("~/images/lunch-1.jpg"))">
            <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 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 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 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>
</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>
<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>,<span>Tomatoe</span></p>
            </div>
        </div>
        <div class="menus d-flex ftco-animate">
            <div class="menu-img img" style="background-image: url(images/dinner-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/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>
    </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 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>
```

```
<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>
        <p><span>Meat</span>, <span>Potatoes</span>, <span>Rice</span>,<br/><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>,<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: 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>,<br/><span>Tomatoe</span></p>
    </div>
</div>
<span class="flat flaticon-wine" style="left: 0;"/><span class="flat flaticon-wine-1" style="right: 0;"/>
</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>
            </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/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>
</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>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </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>

```



```

        <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-
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-
plus"></span></a></li>
                        <li class="ftco-animate"><a href="#"><span class="icon-
instagram"></span></a></li>
                    </ul>
                </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>
```

```
        </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:

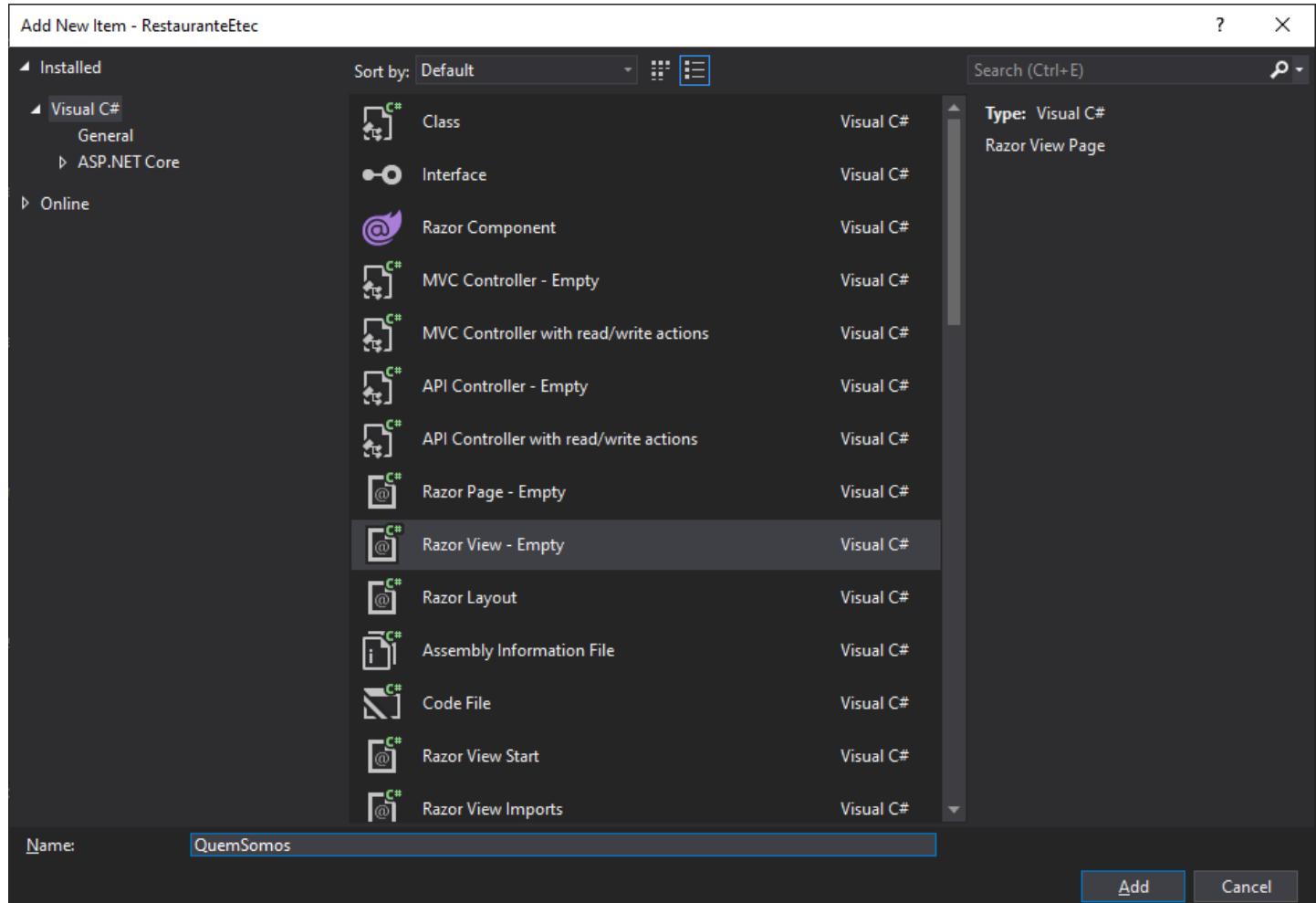
```
22     0 references
23         public IActionResult Index()
24             {
25                 return View();
26             }
27
28     0 references
29         public IActionResult QuemSomos()
30             {
31                 return View();
32             }
33
34     0 references
35        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

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:



Clique com o botão direito do mouse e selecione a opção: **Adicionar (Add) – Exibição (View)**. Escolha então a opção **Exibição Razor Vazia (Razor View – Empty)**. E digite **QuemSomos** em Name.



Vamos começar pela edição da página, abra a **View QuemSomos** e no começo da **View** deixe o código conforme abaixo:

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

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

Também nos links **href** para **asp-action**.

Repita os passos para 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
EmailPessoa	string 100
FonePessoa	string 20
DataCadastro	DateTime
DataReserva	DateTime
Convidados	byte
Status	byte

### Cargo

Id	int
Nome	string 30

### Categoria

Id	int
Nome	string 30

### Funcionario

Id	int
Nome	string 60
Descricao	string 500
Foto	string 200
Cargold	int
ExibirHome	bool
OrdemExibicao	byte

### Produto

Id	int
Nome	string 60
Descricao	string 500
Preco	decimal
Foto	string 200
Categoriald	int
ExibirHome	bool

### Blog

Id	int
DataCadastro	DateTime
Titulo	string 100
Texto	string 8000
Imagen	string 200

### Relato

Id	int
Texto	string 1000
NomePessoa	string 60
FotoPessoa	string 200
TipoPessoa	string 30
ExibirHome	bool
OrdemExibicao	byte

### Contato

Id	int
DataContato	DateTime
NomePessoa	string 60
EmailPessoa	string 100
Assunto	string 100
Mensagem	string 500
Status	byte
Retorno	string 500

Usando essa análise vamos agora criar os nossos modelos. Clique com o botão direito na pasta **Models** e escolha **Adicionar -> Classe (Add -> Class)**:

Em **Name** coloque **Categoria**, e digite o código abaixo:

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

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>.

Agora vamos a criação da classe, **Produto**.

Aqui o processo é o mesmo da Categória.

```
1  using System;
2  using System.ComponentModel.DataAnnotations;
3  using System.ComponentModel.DataAnnotations.Schema;
4
5  namespace RestauranteEtec.Models
6  {
7      [Table("Produto")]
8      public class Produto
9      {
10          [Key]
11          public int Id { get; set; }
12
13          [Required(ErrorMessage = "Campo obrigatório")]
14          [StringLength(60, ErrorMessage = "O Nome da Categoria deve possuir no máximo 60 caracteres")]
15          public string Nome { get; set; }
16
17          [StringLength(500, ErrorMessage = "A Descrição deve possuir no máximo 500 caracteres")]
18          public string Descricao { get; set; }
19
20          [Required(ErrorMessage = "Campo obrigatório")]
21          public int CategoriaId { get; set; }
22          public Categoria Categoria { get; set; }
23
24          [StringLength(200)]
25          public string Foto { get; set; }
26
27          public bool ExibirHome { get; set; }
28
29          public bool Ativo { get; set; }
30      }
31  }
```

## 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](#).

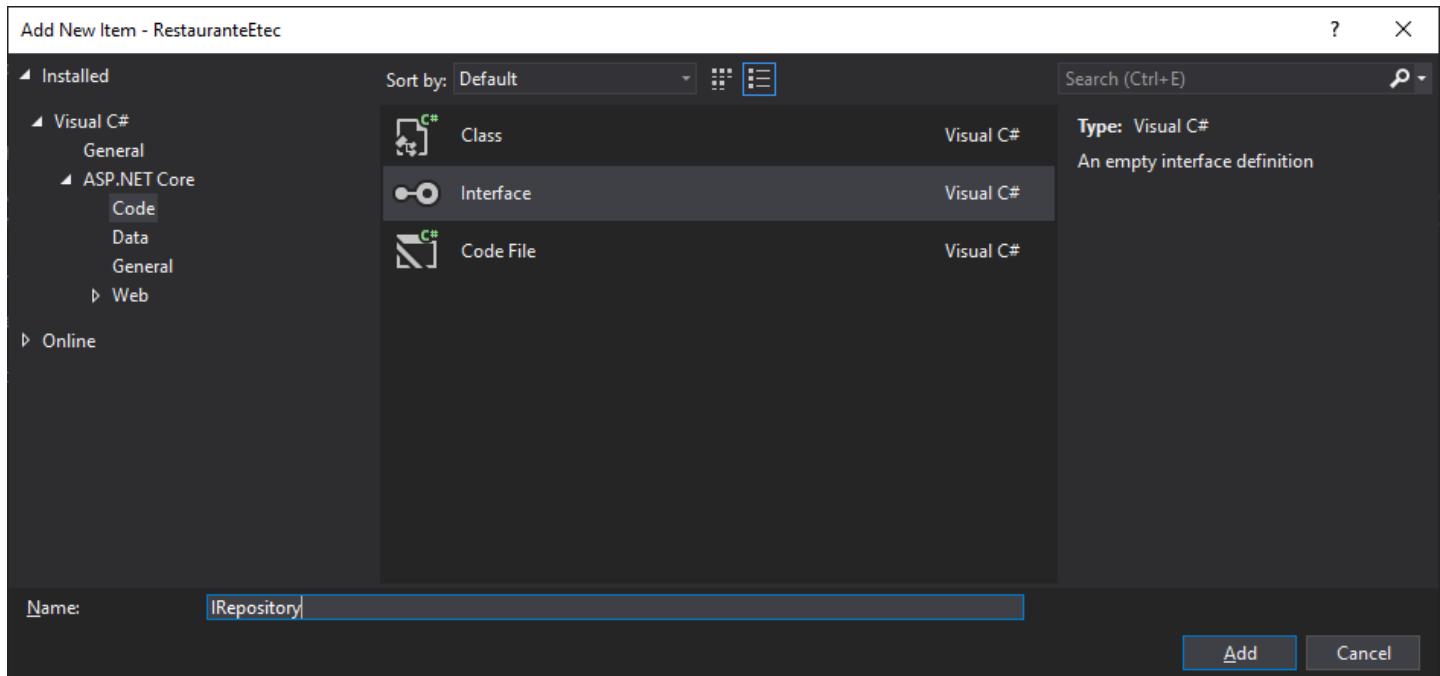
As interfaces podem conter métodos de instância, propriedades, eventos, indexadores ou qualquer combinação desses quatro tipos de membro. As interfaces podem conter construtores, campos, constantes ou operadores estáticos. Para obter links para exemplos, consulte as [seções relacionadas](#). Uma interface 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 com o botão direito no projeto e selecione a opção: **Adicionar -> Nova Pasta (Add -> New Folder)** e nomeiem a pasta como **Interfaces**.

Clique com o botão direito na pasta e selecione a opção de adicionar uma nova classe. Mude a opção acima de **Class (Classe)** para **Interface**, conforme mostra a figura abaixo:



Coloque no nome do arquivo **IRepository** e clique em **Adicionar (Add)**.

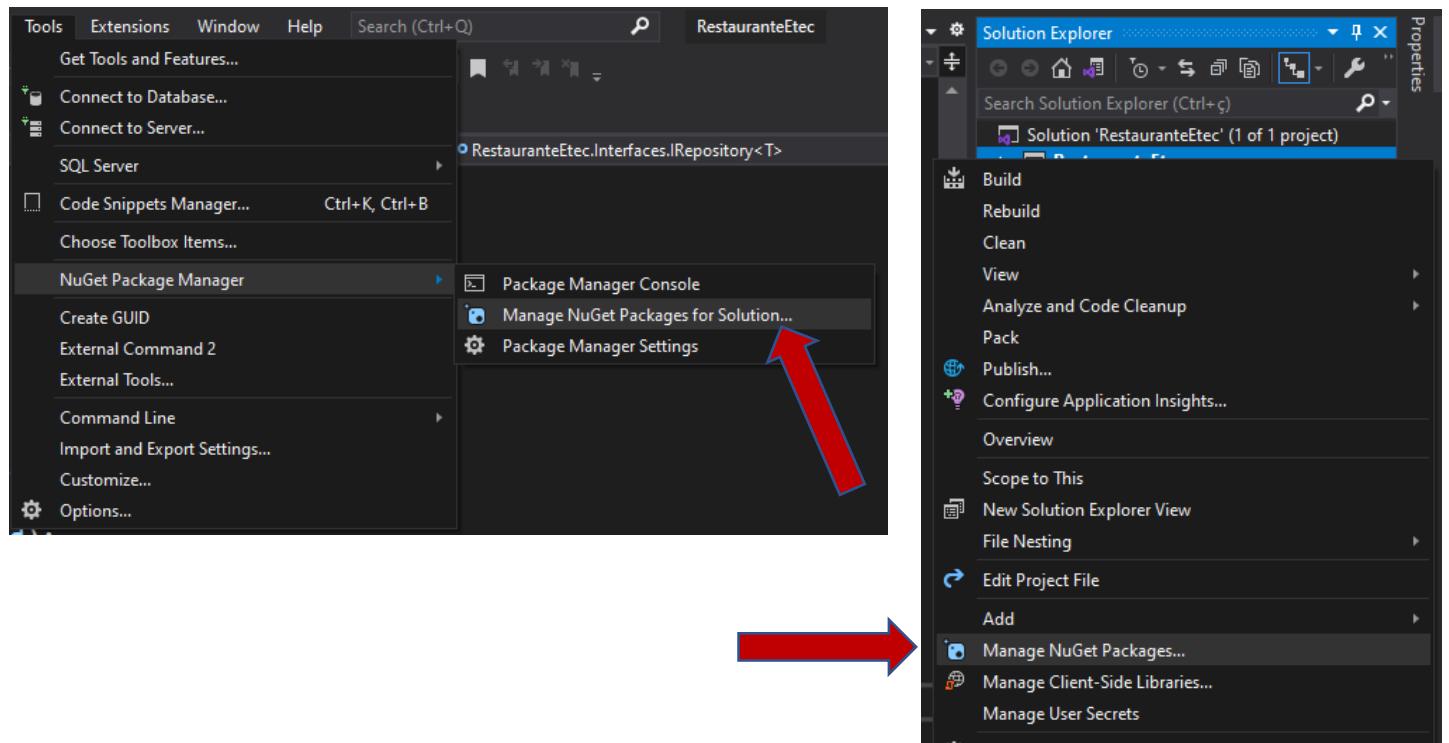
```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5
6  namespace RestauranteEtec.Interfaces
7  {
8      public interface IRepository<T>
9      {
10         I Enumerable<T> GetAll();
11         T GetById(int? id);
12         void Add(T model);
13         void Update(T model);
14         void Delete(int? id);
15     }
16 }
17
```

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.

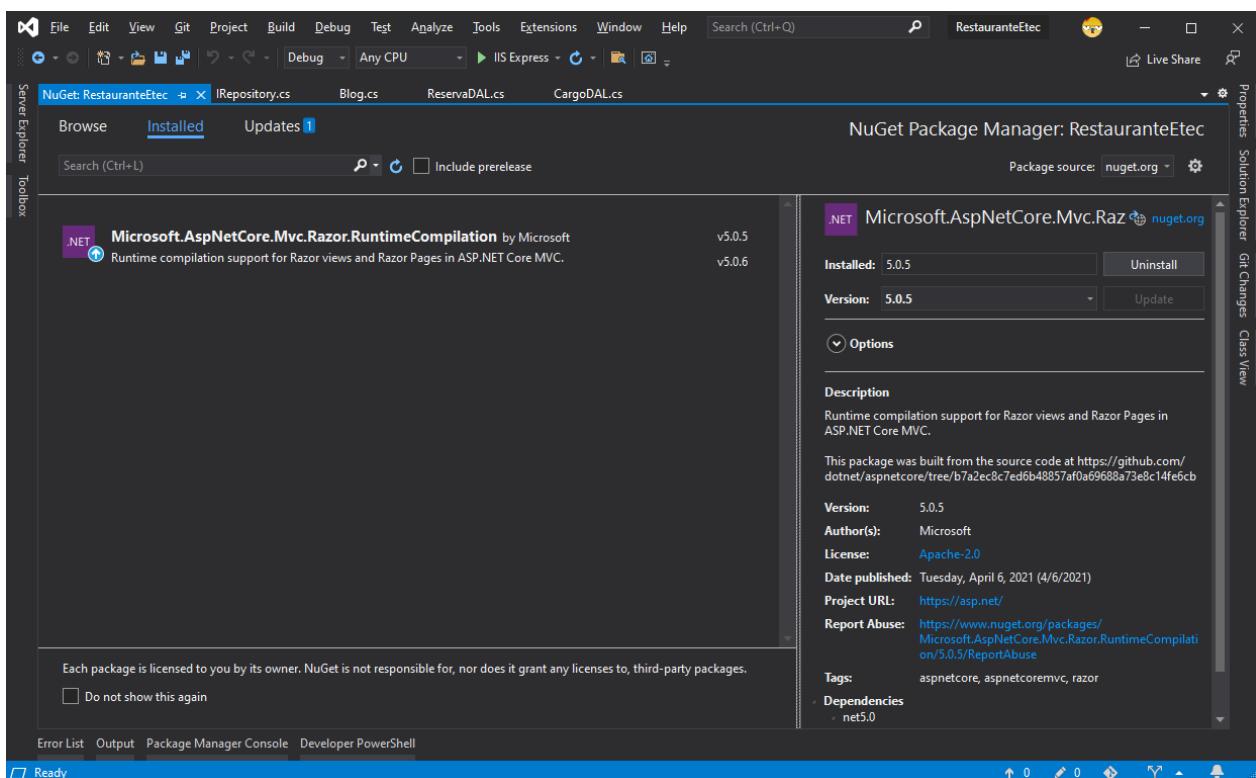
Em nossa aplicação, vamos criar uma pasta para armazenar as classes **DAL**. Clique com o botão direito no projeto e selecione a opção: **Adicionar -> Nova Pasta (Add -> New Folder)** e nomeiem a pasta como **DAL**.

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.

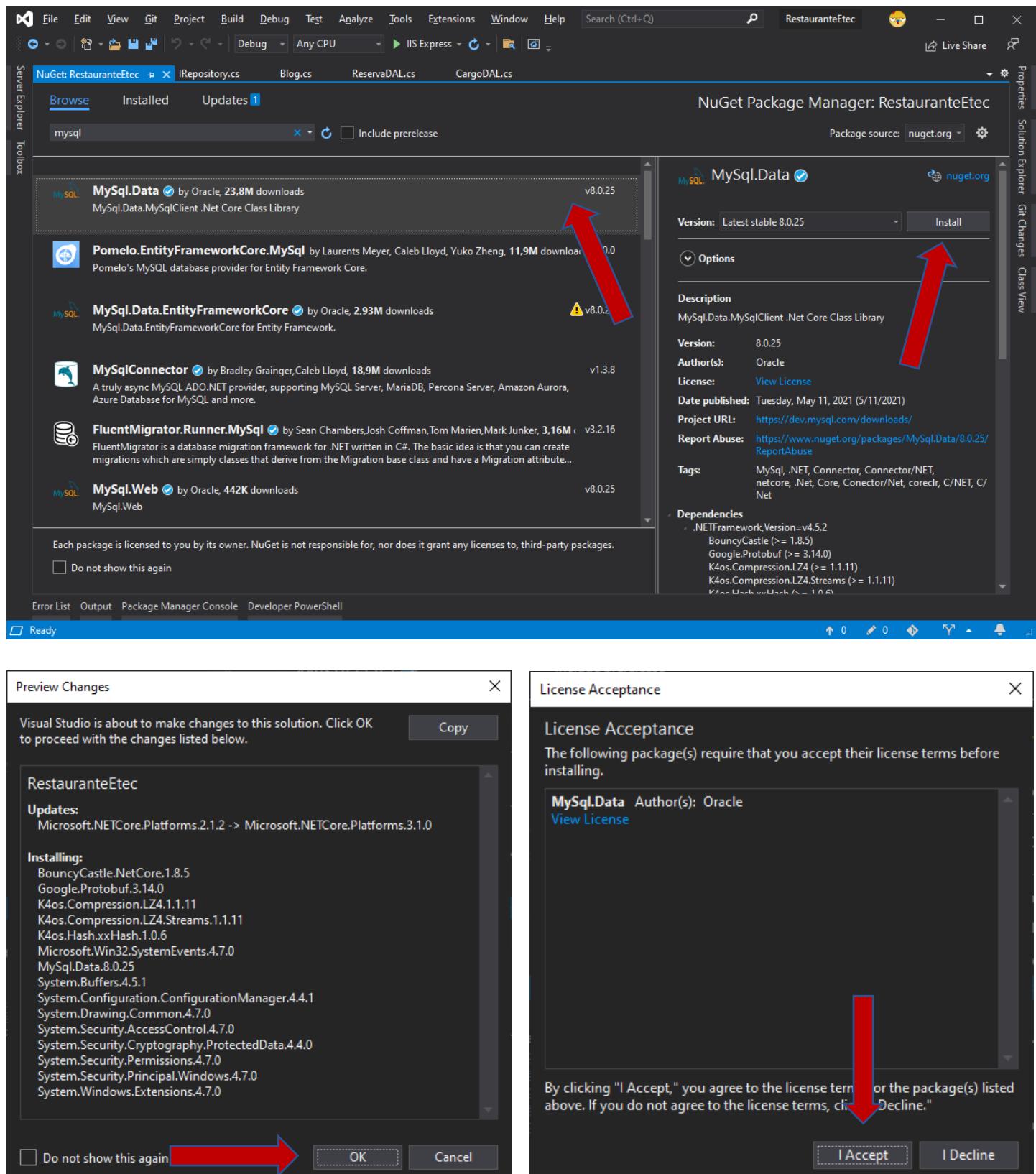
Para incluir pacotes de maneira simples, o Visual Studio Community, possui o **Gerenciador de Pacotes do NuGet (Manage NuGet Packages)**, que pode ser acessado através do menu **Ferramentas (Tools)**, ou clicando com o botão direito no projeto no menu lateral **Explorador de Soluções (Solution Explorer)** e selecionando a opção **Gerenciar Pacotes do NuGet**. Qualquer uma das opções irá exibir a mesma interface.



Nesta primeira aba **Instalados (Installed)**, temos a relação de pacotes já instalados em nosso projeto. Clique na aba **Navegar (Browse)**.



E faça conforme a imagem abaixo, pesquise por “**mysql**” e na lista que será exibida clique no pacote **MySql.Data**, que é fornecido pela **Oracle** (empresa desenvolvedora do **MySQL**, ou seja, esse é o pacote oficial, existem outras opções de conexão). Em seguida clique no botão **Instalar (Install)**.



Depois é só aguardar o término do processo. Nesta fase é interessante que você compile o projeto ao menos uma vez.

Para adicionar esses pacotes através de linha de comando, caso esteja desenvolvendo pelo Visual Studio Code, use: **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**:

```
1  |└ using System;
2  |└ using System.Collections.Generic;
3  |└ using System.Linq;
4  |└ using System.Threading.Tasks;
5
6  |└ namespace RestauranteEtec.DAL
7  |{
8  |└ 2 references
9  |└ public class CargoDAL : IRepository<Cargo>
10 |└ {
11 |└ }
12 |└ }
13 |└ }
```

Para resolver os problemas, precisamos adicionar duas referências a nossa classe:

```
1  |└ using RestauranteEtec.Interfaces;
2  |└ using RestauranteEtec.Models;
3  |└ using System;
4  |└ using System.Collections.Generic;
5  |└ using System.Linq;
6  |└ using System.Threading.Tasks;
7
8  |└ namespace RestauranteEtec.DAL
9  |{
10 |└ 2 references
11 |└ public class CargoDAL : IRepository<Cargo>
12 |└ {
13 |└ }
14 |└ }
15 |└ }
```

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:

The screenshot shows a code editor in Visual Studio with the following code:

```
public class CargoDAL : IRepository<Cargo>
{
}
```

A context menu is open at the end of the class definition, with the "Implement interface" option selected. A tooltip provides two options: "Implement interface" and "Implement all members explicitly". A warning message is displayed: "CS0535 'CargoDAL' does not implement interface member 'IRepository<Cargo>.GetAll()'". Below the code, the implementation of the GetAll() method is shown:Lines 11 to 13
{
 public void Add(Cargo model)
 {
 throw new NotImplementedException();
 }

 public void Delete(int? id)
 {
 throw new NotImplementedException();
 }

 public IEnumerable<Cargo> GetAll()
 {
 throw new NotImplementedException();
 }

 public Cargo GetById(int? id)
 {
 throw new NotImplementedException();
 }

 public void Update(Cargo model)
 {
 throw new NotImplementedException();
 }
}

At the bottom, there are buttons for "Preview changes" and "Fix all occurrences in: Document | Project | Solution".

Clicando em **Implementar interface (Implement interface)**:

The screenshot shows the generated implementation of the IRepository<Cargo> interface in the CargoDAL class:

```
using RestauranteEtec.Interfaces;
using RestauranteEtec.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace RestauranteEtec.DAL
{
    public class CargoDAL : IRepository<Cargo>
    {
        public void Add(Cargo model)
        {
            throw new NotImplementedException();
        }

        public void Delete(int? id)
        {
            throw new NotImplementedException();
        }

        public IEnumerable<Cargo> GetAll()
        {
            throw new NotImplementedException();
        }

        public Cargo GetById(int? id)
        {
            throw new NotImplementedException();
        }

        public void Update(Cargo model)
        {
            throw new NotImplementedException();
        }
    }
}
```

Agora precisamos programar todos os métodos, 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**:

```
2 references
public class CargoDAL : IRepository<Cargo>
{
    string connectionString = @"Server=localhost;port=3306;database=RestauranteEtec;uid=root;pwd=' ";
```

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 no nosso caso é vazia, “ ”.

Antes de programarmos os demais métodos, acrescentes mais dois pacotes ao início do código (linhas 1 e 6):

```
1  using MySql.Data.MySqlClient; ←
2  using RestauranteEtec.Interfaces;
3  using RestauranteEtec.Models;
4  using System;
5  using System.Collections.Generic;
6  using System.Data; ←
7  using System.Linq;
8  using System.Threading.Tasks;
```

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

### Função Add

```
2 references
public void Add(Cargo model)
{
    using (MySqlConnection conexao = new MySqlConnection(connectionString))
    {
        var sql = "insert into Cargo(Nome) values (@Nome)";
        MySqlCommand comando = new MySqlCommand(sql, conexao);
        comando.CommandType = CommandType.Text;
        comando.Parameters.AddWithValue("@Nome", model.Nome);

        conexao.Open();
        comando.ExecuteNonQuery();
        conexao.Close();
    }
}
```

## Função Delete

```
2 references
public void Delete(int? id)
{
    using (MySqlConnection conexao = new MySqlConnection(connectionString))
    {
        var sql = "delete from Cargo where Id = @Id";
        MySqlCommand comando = new MySqlCommand(sql, conexao);
        comando.CommandType = CommandType.Text;

        comando.Parameters.AddWithValue("@Id", id);

        conexao.Open();
        comando.ExecuteNonQuery();
        conexao.Close();
    }
}
```

## Função GetAll

```
public IEnumerable<Cargo> GetAll()
{
    List<Cargo> cargos = new List<Cargo>();
    using (MySqlConnection conexao = new MySqlConnection(connectionString))
    {
        MySqlCommand comando = new MySqlCommand("select * from Cargo", conexao);
        comando.CommandType = CommandType.Text;

        conexao.Open();
        MySqlDataReader leitor = comando.ExecuteReader();
        while (leitor.Read())
        {
            Cargo cargo = new Cargo()
            {
                Id = Convert.ToInt32(leitor["Id"]),
                Nome = leitor["Nome"].ToString()
            };
            cargos.Add(cargo);
        }
        conexao.Close();
    }
    return cargos;
}
```

## Função GetById

```
public Cargo GetById(int? id)
{
    Cargo cargo = new Cargo();
    using (MySqlConnection conexao = new MySqlConnection(connectionString))
    {
        MySqlCommand comando = new MySqlCommand("select * from Cargo where Id = @Id", conexao);
        comando.CommandType = CommandType.Text;
        comando.Parameters.AddWithValue("@Id", id);

        conexao.Open();
        MySqlDataReader leitor = comando.ExecuteReader();
        leitor.Read();
        if (!leitor.HasRows)
        {
            conexao.Close();
            return null;
        }
        cargo.Id = Convert.ToInt32(leitor["Id"]);
        cargo.Nome = leitor["Nome"].ToString();
        conexao.Close();
    }
    return cargo;
}
```

## Função Update

```
public void Update(Cargo model)
{
    using (MySqlConnection conexao = new MySqlConnection(connectionString))
    {
        var sql = "update Cargo set" +
                  " Nome = @Nome" +
                  " where Id = @Id";
        MySqlCommand comando = new MySqlCommand(sql, conexao);
        comando.CommandType = CommandType.Text;

        comando.Parameters.AddWithValue("@Id", model.Id);
        comando.Parameters.AddWithValue("@Nome", model.Nome);

        conexao.Open();
        comando.ExecuteNonQuery();
        conexao.Close();
    }
}
```

## 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

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 VOCÊ PODE REPETIR OS PASSOS DA PÁGINA 35 ATÉ A 39 para criar as classes:

- **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 iniciais das demais classes (páginas 35 e 36), com os códigos conforme as figuras a seguir:

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

Vamos agora a codificação de cada função.

## Função Add:

```
2 references
public void Add(Funcionario model)
{
    using (MySqlConnection conexao = new MySqlConnection(connectionString))
    {
        var sql = "insert into Funcionario(Nome, Descricao, Foto, CargoId, ExibirHome, OrdemExibicao, Ativo)" +
                  " values (@Nome, @Descricao, @Foto, @CargoId, @ExibirHome, @OrdemExibicao, @Ativo)";
        MySqlCommand comando = new MySqlCommand(sql, conexao);
        comando.CommandType = CommandType.Text;
        comando.Parameters.AddWithValue("@Nome", model.Nome);
        comando.Parameters.AddWithValue("@Descricao", model.Descricao);
        comando.Parameters.AddWithValue("@Foto", model.Foto);
        comando.Parameters.AddWithValue("@CargoId", model.CargoId);
        comando.Parameters.AddWithValue("@ExibirHome", model.ExibirHome);
        comando.Parameters.AddWithValue("@OrdemExibicao", model.OrdemExibicao);
        comando.Parameters.AddWithValue("@Ativo", true);

        conexao.Open();
        comando.ExecuteNonQuery();
        conexao.Close();
    }
}
```

## Funções Delete:

```
2 references
public void Delete(int? id)
{
    using (MySqlConnection conexao = new MySqlConnection(connectionString))
    {
        var sql = "delete from Funcionario where Id = @Id";
        MySqlCommand comando = new MySqlCommand(sql, conexao);
        comando.CommandType = CommandType.Text;
        comando.Parameters.AddWithValue("@Id", id);

        conexao.Open();
        comando.ExecuteNonQuery();
        conexao.Close();
    }
}
```

## Função GetAll:

```
4 references
public IEnumerable<Funcionario> GetAll()
{
    List<Funcionario> funcionarios = new List<Funcionario>();
    CargoDAL cargo = new CargoDAL();
    using (MySqlConnection conexao = new MySqlConnection(connectionString))
    {
        MySqlCommand comando = new MySqlCommand("select * from funcionario", conexao);
        comando.CommandType = CommandType.Text;

        conexao.Open();
        MySqlDataReader leitor = comando.ExecuteReader();
        while (leitor.Read())
        {
            Funcionario funcionario = new Funcionario()
            {
                Id = Convert.ToInt32(leitor["Id"]),
                Nome = leitor["Nome"].ToString(),
                Descricao = leitor["Descricao"].ToString(),
                Foto = leitor["Foto"].ToString(),
                CargoId = Convert.ToInt32(leitor["CargoId"].ToString()),
                ExibirHome = Convert.ToInt32(leitor["ExibirHome"]) == 1,
                OrdemExibicao = Convert.ToByte(leitor["OrdemExibicao"]),
                Ativo = Convert.ToInt32(leitor["Ativo"]) == 1
            };
            funcionario.Cargo = cargo.GetById(funcionario.CargoId);

            funcionarios.Add(funcionario);
        }
        conexao.Close();
    }
    return funcionarios;
}
```

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:

```
public Funcionario GetById(int? id)
{
    CargoDAL cargo = new CargoDAL();
    Funcionario funcionario = new Funcionario();
    using (MySqlConnection conexao = new MySqlConnection(connectionString))
    {
        MySqlCommand comando = new MySqlCommand("select * from Funcionario where Id = @Id", conexao);
        comando.CommandType = CommandType.Text;
        comando.Parameters.AddWithValue("@Id", id);

        conexao.Open();
        MySqlDataReader leitor = comando.ExecuteReader();
        leitor.Read();
        if (!leitor.HasRows)
        {
            conexao.Close();
            return null;
        }
        funcionario.Id = Convert.ToInt32(leitor["Id"]);
        funcionario.Nome = leitor["Nome"].ToString();
        funcionario.Descricao = leitor["Descricao"].ToString();
        funcionario.Foto = leitor["Foto"].ToString();
        funcionario.CargoId = Convert.ToInt32(leitor["CargoId"].ToString());
        funcionario.ExibirHome = Convert.ToInt32(leitor["ExibirHome"]) == 1;
        funcionario.OrdemExibicao = Convert.ToByte(leitor["OrdemExibicao"]);
        funcionario.Ativo = Convert.ToInt32(leitor["Ativo"]) == 1;
        funcionario.Cargo = cargo.GetById(funcionario.CargoId);
        conexao.Close();
    }
    return funcionario;
}
```

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:

```
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**.

```
RestauranteEtec
1  using Microsoft.AspNetCore.Mvc;
2  using Microsoft.Extensions.Logging;
3  using RestauranteEtec.DAL;
4  using RestauranteEtec.Models;
5  using System;
6  using System.Collections.Generic;
7  using System.Diagnostics;
8  using System.Linq;
9  using System.Threading.Tasks;
10
```

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**.

```
[HttpGet]
0 references
public IActionResult Index()
{
    var funcionarios = new FuncionarioDAL();
    var chefes = funcionarios.GetAll().Where(f => f.Ativo && f.ExibirHome).OrderBy(f => f.OrdemExibicao).ToList();
    ViewData["Chefes"] = chefes;

    return View();
}
```

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, alteramos a View **Index.cshtml**, localizada em **Views\Home**, substituindo as 4 divs de funcionários (linhas entre 560 e 631, aproximadamente) por um foreach. Para facilitar segue abaixo a seção inteira de Chefs alterada (procure no seu Index por “**Our Chefs**” e altere todo o código da seção:

```
551 551     <section class="ftco-section bg-light">
552 552         <div class="container">
553 553             <div class="row justify-content-center mb-5 pb-2">
554 554                 <div class="col-md-7 text-center heading-section ftco-animate">
555 555                     <span class="subheading">Chefes</span>
556 556                     <h2 class="mb-4">Nossos Mestres</h2>
557 557                 </div>
558 558             </div>
559 559             <div class="row">
560 560                 @foreach (var chefe in ViewBag.Chefes)
561 561                 {
562 562                     <div class="col-md-6 col-lg-3 ftco-animate">
563 563                         <div class="staff">
564 564                             <div class="img" style="background-image: url(@Url.Content(chefe.Foto));"></div>
565 565                             <div class="text px-4 pt-2">
566 566                             <h3>@chefe.Nome</h3>
567 567                             <span class="position mb-2">@chefe.Cargo.Nome</span>
568 568                             <div class="faded">
569 569                                 <p>@chefe.Descricao</p>
570 570                         </div>
571 571                     </div>
572 572                 </div>
573 573             </div>
574 574         </div>
575 575     </div>
576 576 </section>
577 577
```

Execute e verifique o resultado

RestauranteEtec - Home + https://localhost:44349 Não sincronizando ...

RestauranteEtec HOME QUEM SOMOS CHEFES MENU RESERVAS BLOG CONTATOS

## Chefs Nossos Mestres



**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 cardápio, 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 no canal **Semana 17**. Descompacte o arquivo e siga os passos abaixo:

- 1- Copie a pasta **assets** do **template** para a pasta **wwwroot** do projeto **RestauranteEtec**.
- 2- Clique com o botão direito do mouse na pasta **Views\Shared** e escolha a opção **Adicionar (Add)** e no submenu **Novo Item (New Item)**, na lista que será exibida selecione a opção **Layout Razor (Razor Layout)**, mude o nome do arquivo na parte de baixo da tela para **\_LayoutAdmin** e clique no botão **Adicionar (Add)**.
- 3- 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**.
- 4- Digite logo abaixo do código copiado: **@RenderBody()**
- 5- Agora volte ao arquivo **index.html** do **template** e copia as linhas de 443 a 517.
- 6- Desça até a tag **</body>** e inclua a seguinte linha antes da tag: **@await RenderSectionAsync("Scripts", required: false)**
- 7- Altere a linha 2 para: **<html lang="pt-br">**
- 8- Altere a linha 9 para: **<title>RestauranteEtec - @ViewData["Title"]</title>**
- 9- Agora assim como fizemos no primeiro layout, precisamos encontrar todas as referências a pasta **assets** e acrescente antes um **“~/” (til e barra)**.
- 10- 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>
                <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>
            
```

```
        </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>

        <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>
            <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>

```

Agora clique com o botão direito do mouse em **Controllers** e selecione a opção **Adicionar (Add)** e no submenu **Controlador (Controller)**. Na lista que é exibida, escolhe **Controlador MVC – Vazio (MVC Controller – Empty)**, mude o nome para **AdminController** e clique em **Adicionar (Add)**.

Com o controller criado, clique com o botão direito do mouse em **Index** (nome da **action** criada automaticamente) e escolha a opção **Adicionar Exibição (Add View)**. Na lista que for exibida escolha a opção **Exibição Razor – Vazia (Razor View – Empty)**, deixe o nome do arquivo como **Index** e clique em adicionar, que o Visual Studio irá criar uma pasta dentro de **Views** com o nome **Admin** e dentro irá adicionar uma view **Index.cshtml**.

No arquivo Index criado, altere o começo do código da seguinte forma:

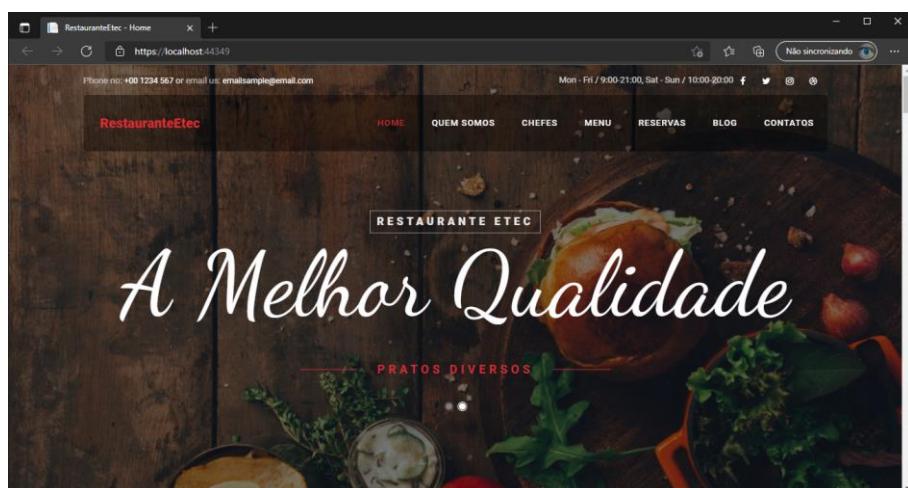
```

@{
    ViewData["Title"] = "Área Administrativa";
    Layout = "~/Views/Shared/_LayoutAdmin.cshtml";
}

```

E pronto, nosso layout administrativo está pronto. Caso você queira, pode copiar as linhas que ficaram de fora da copia 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:



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

The screenshot shows a dark-themed administrative dashboard for "Restaurante ETEC - Área Administrativa". The top navigation bar includes links for Home, Log Out, and a search bar. The main header displays the restaurant's name and the current date.

**Dashboard Statistics:**

- Reservas: 9526 (+4.2% ↑)
- Contatos: 8323 (+1.2% ↑)
- Visitas: 6200 (+5.2% ↑)
- Contatos: 5630 (+2.2% ↑)

**Site Traffic:** A line chart showing visitor trends from January to October. The Y-axis ranges from 0 to 14 visitors. The chart shows two series: New Visitor (light grey) and Old Visitor (dark grey).

Mês	New Visitor	Old Visitor
Jan	~2	~6
Feb	~4	~8
Mar	~6	~10
Apr	~8	~12
May	~10	~13
Jun	~12	~11
Jul	~10	~9
Aug	~8	~7
Sep	~6	~9
Oct	~4	~7

**Weekly sales:** A donut chart showing sales distribution by source.

Fonte	Valor	Crescimento (%)
Direct	\$5856	+55%
Affiliate	\$2602	+25%
E-mail	\$1802	+15%
Other	\$1105	+5%

**Recent Order Tables:** A table listing recent orders.

PRODUTO	FOTO	ID DO PRODUTO	VALOR	DATA	ENVIO
Iphone 5	[Thumbnail]	#9405822	\$ 1250.00	03 Ago 2017	[Progress Bar]
Earphone GL	[Thumbnail]	#9405820	\$ 1500.00	03 Ago 2017	[Progress Bar]
HD Hand Camera	[Thumbnail]	#9405830	\$ 1400.00	03 Ago 2017	[Progress Bar]

## CRIANDO O CRUD DE CARGOS

Vamos agora criar um **controller** para realizar as operações de **CRUD** através de nossa camada **DAL**. Também vamos fazer uso de algumas facilidades que o Visual Studio Community oferece. Clique com o botão direito do mouse em Controllers, e selecione a opção **Adicionar (Add)** e no submenu **Controlador (Controller)**. Na lista que é exibida, escolhe **Controlador MVC com ações de leitura/escrita (MVC Controller with read/write actions)**, mude o nome para **CargosController** (não esqueça de tirar o 1 no final do nome) e clique em **Adicionar (Add)**.

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**:

```
11  [ ] 0 references
12  [ ] public class CargosController : Controller
13  [ ] {
14  [ ]     private readonly CargoDAL cargoDAL = new ...;
```

Esse objeto será utilizado pelas **Actions**, para ter acesso aos métodos criados na DAL de acesso 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  [ ] // GET: CargosController
16  [ ] 3 references
17  [ ] public ActionResult Index()
18  [ ] {
19  [ ]     var cargos = cargoDAL.GetAll();
20  [ ]     return View(cargos);
21  [ ] }
```

**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)**.

```
22 // GET: CargosController/Details/5
23 0 references
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 }
```

**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.

```
29 // GET: CargosController/Create
30 0 references
31 public ActionResult Create()
32 {
33     return View();
34 }
```

**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: CargosController/Create
41 [HttpPost]
42 [ValidateAntiForgeryToken]
43 0 references
44 public ActionResult Create([Bind]Cargo cargo)
45 {
46     if (!ModelState.IsValid)
47         return View(cargo);
48     try
49     {
50         cargoDAL.Add(cargo);
51         return RedirectToAction("Index");
52     }
53     catch
54     {
55         return View(cargo);
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)**.

```
57
58 // GET: CargosController/Edit/5
59 0 references
60 public ActionResult Edit(int? id)
61 {
62     if (id == null)
63         return NotFound();
64     var cargo = cargoDAL.GetById(id);
65     if (cargo == null)
66         return NotFound();
67     return View(cargo);
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**).

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

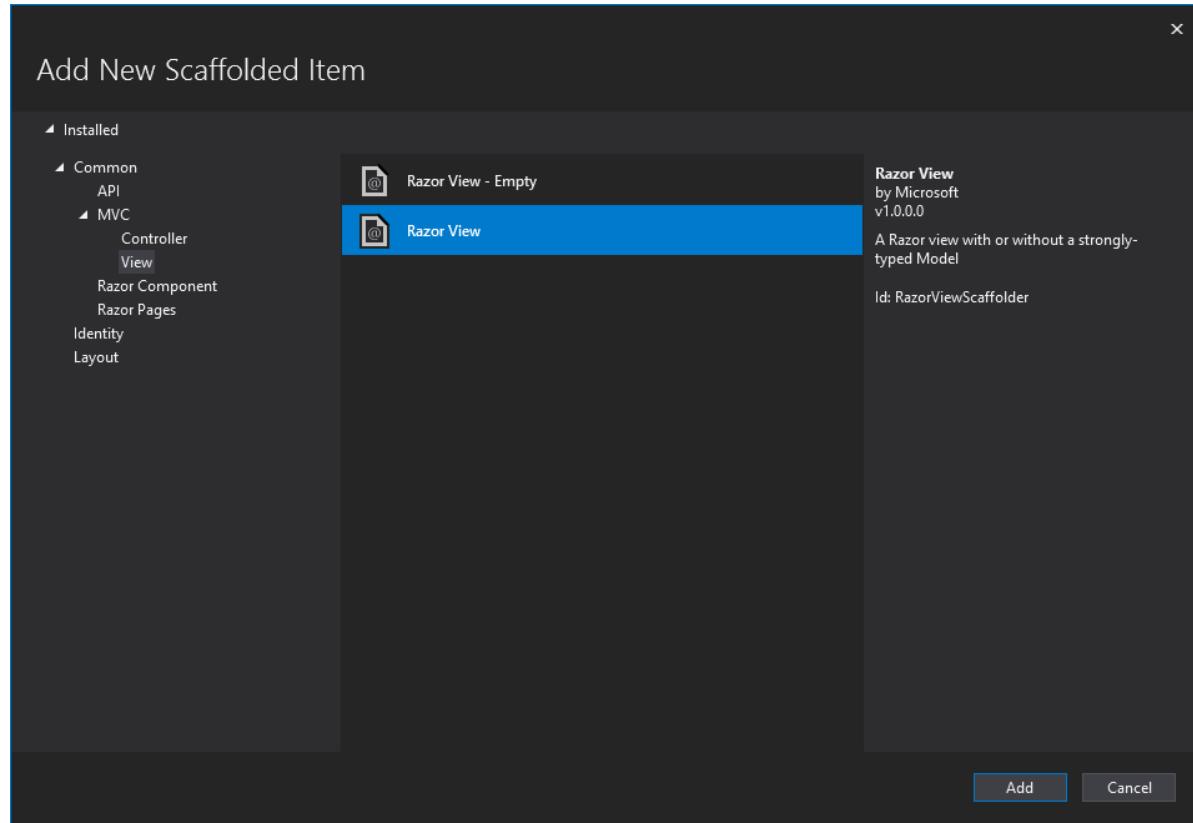
```
91 // GET: CargosController/Delete/5
92 []
93 public ActionResult Delete(int? id)
94 {
95     if (id == null)
96         return NotFound();
97     var cargo = cargoDAL.GetById(id);
98     if (cargo == null)
99         return NotFound();
100    return View(cargo);
101 }
```

**Delete (Post):** Aqui realizamos a execução da chamada do método **Delete** da **CargoDAL**, enviando o **id** do cargo a ser excluído.

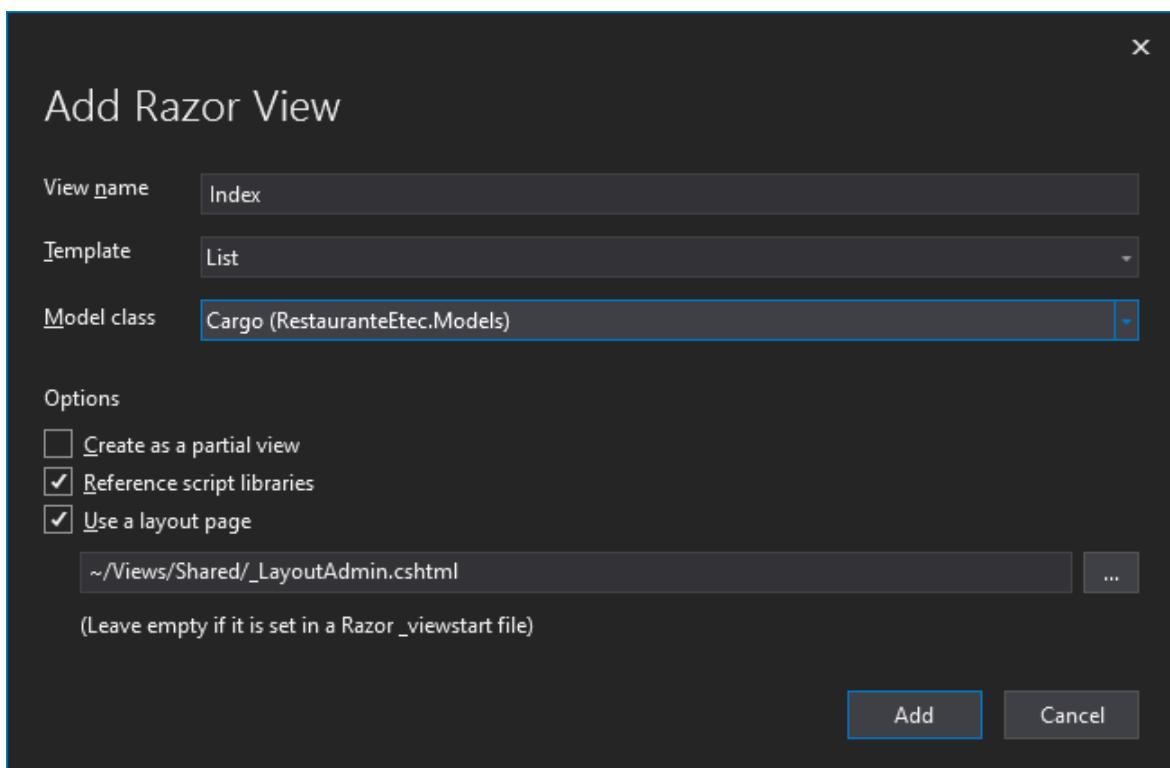
```
102 // POST: CargosController/Delete/5
103 [HttpPost, ActionName("Delete")]
104 [ValidateAntiForgeryToken]
105 public IActionResult DeleteConfirmed(int? id)
106 {
107     if (id == null)
108         return NotFound();
109     cargoDAL.Delete(id);
110     return RedirectToAction("Index");
111 }
```

Agora precisamos criar as **Views** de cada **Action**. Para facilitar vamos novamente fazer uso de funcionalidades do Visual Studio Community. Assim como fizemos no **AdminController**, clique com o botão direito do mouse sobre o nome da **Action Index**, do **CargosController**, e escolha a opção **Adicionar Exibição (Add View)**. Só que não escolha ainda, vamos as diferenças a partir daqui:

1ª Diferença, escolha opção **Exibição Razor (Razor View)**, conforme mostrado na imagem abaixo:



O Visual Studio então irá apresentar a tela abaixo, que você deve alterar para ficar conforme a imagem, antes de clicar em **Adicionar (Add)**:



Esse é o código criado automaticamente pelo Scaffolding do Visual Studio:

```
Index.cshtml + X CargoDAL.cs CargosController.cs RestauranteEtec.csproj AdminController.cs
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.

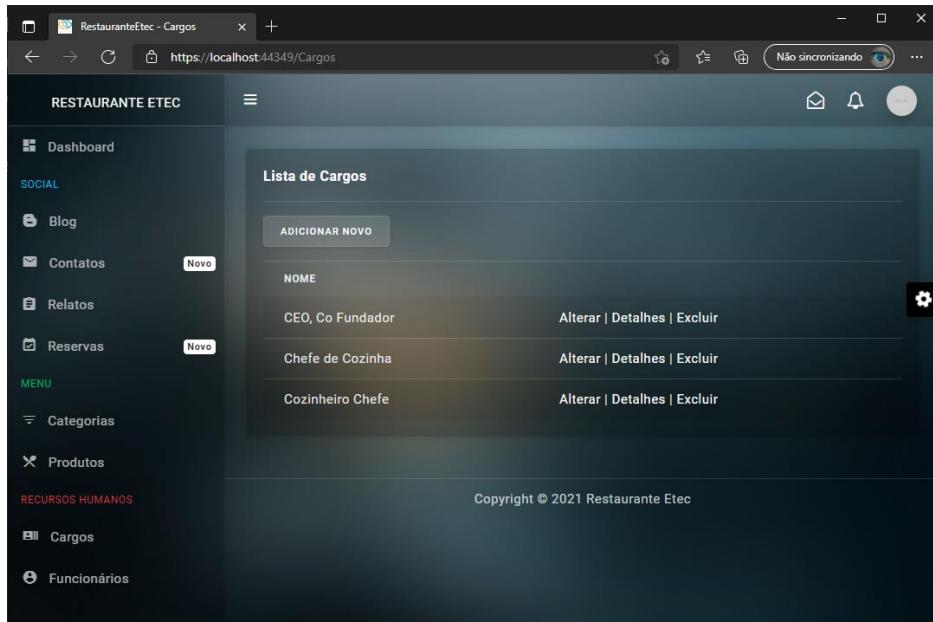
O **foreach**, é usado para criar uma linha na tabela para cada item da lista de **cargos** recebidos pelo **model**.

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 \*/**).

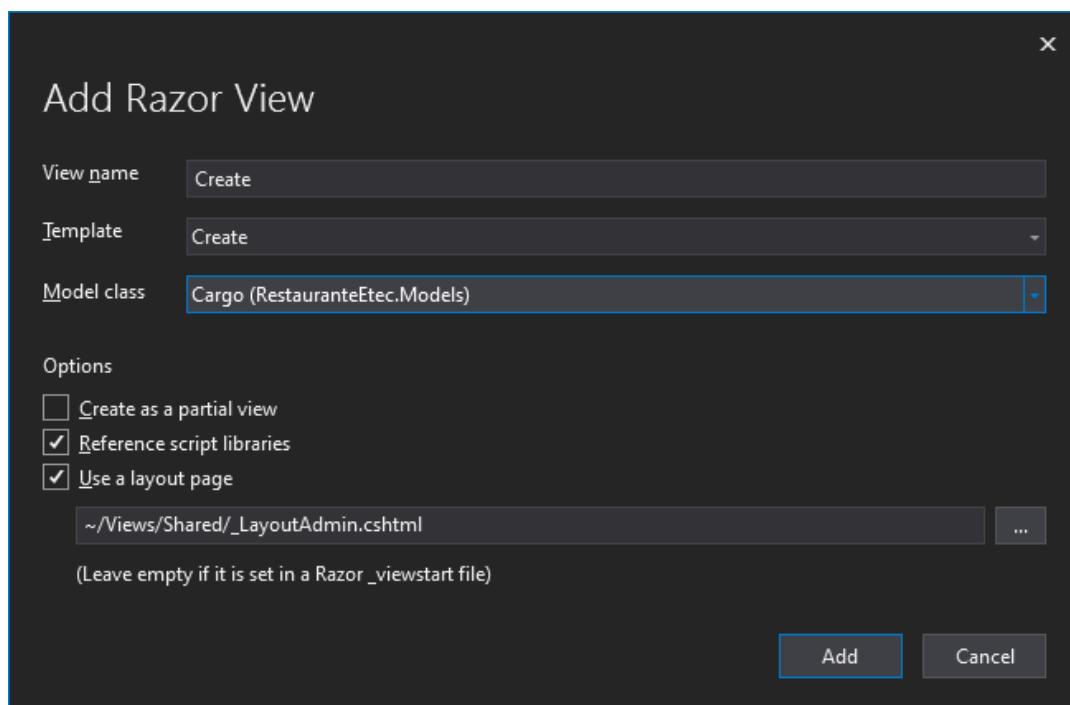
Altere seu código para ficar conforme a imagem abaixo:

```
Index.cshtml* CargoDAL.cs CargosController.cs RestauranteEtec.csproj AdminController.cs
1 @model IEnumerable<RestauranteEtec.Models.Cargo>
2 @{
3     ViewData["Title"] = "Cargos";
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 Cargos</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></th>
23                         </tr>
24                     </thead>
25                     <tbody>
26                         @foreach (var item in Model)
27                         {
28                             <tr>
29                                 <td>
30                                     @Html.DisplayFor(modelItem => item.Nome)
31                                 </td>
32                                 <td>
33                                     @Html.ActionLink("Alterar", "Edit", new { id = item.Id }) |
34                                     @Html.ActionLink("Detalhes", "Details", new { id = item.Id }) |
35                                     @Html.ActionLink("Excluir", "Delete", new { id = item.Id })
36                                 </td>
37                             </tr>
38                         }
39                     </tbody>
40                 </table>
41             </div>
42         </div>
43     </div>
44 </div>
```

Até aqui nosso resultado foi o seguinte:



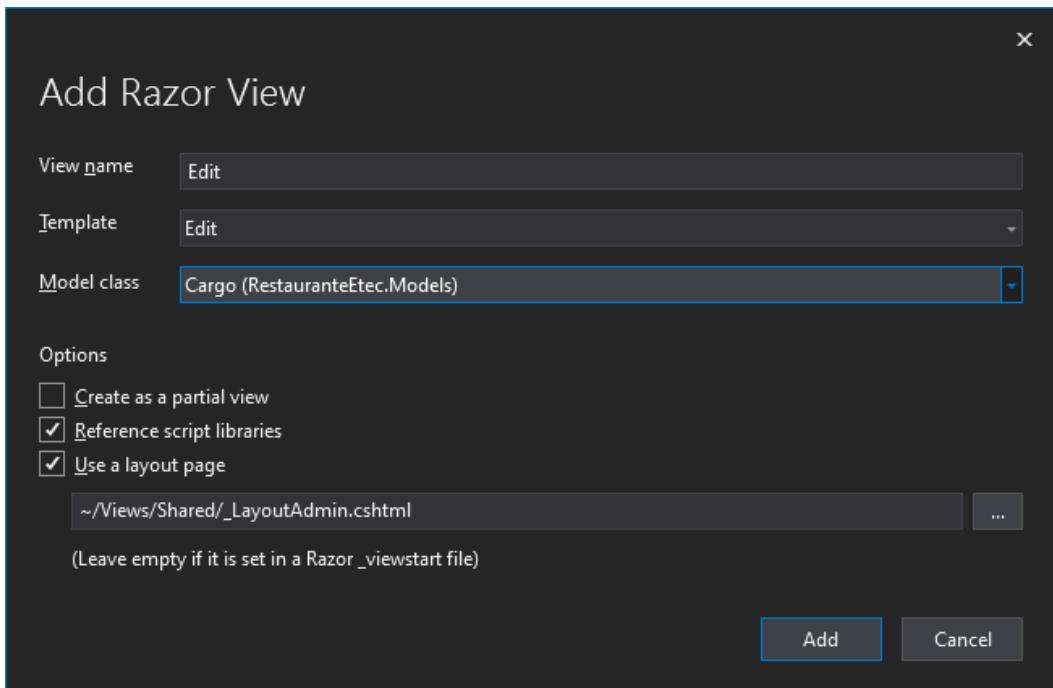
Vamos a próxima View, botão direito em **Create** no **CargosController** e Adicionar Exibição (Add View):



Altere o código gerado conforme a imagem abaixo:

```
1 @model RestauranteEtec.Models.Cargo
2
3 @{
4     ViewData["Title"] = "Cargos";
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 }
```

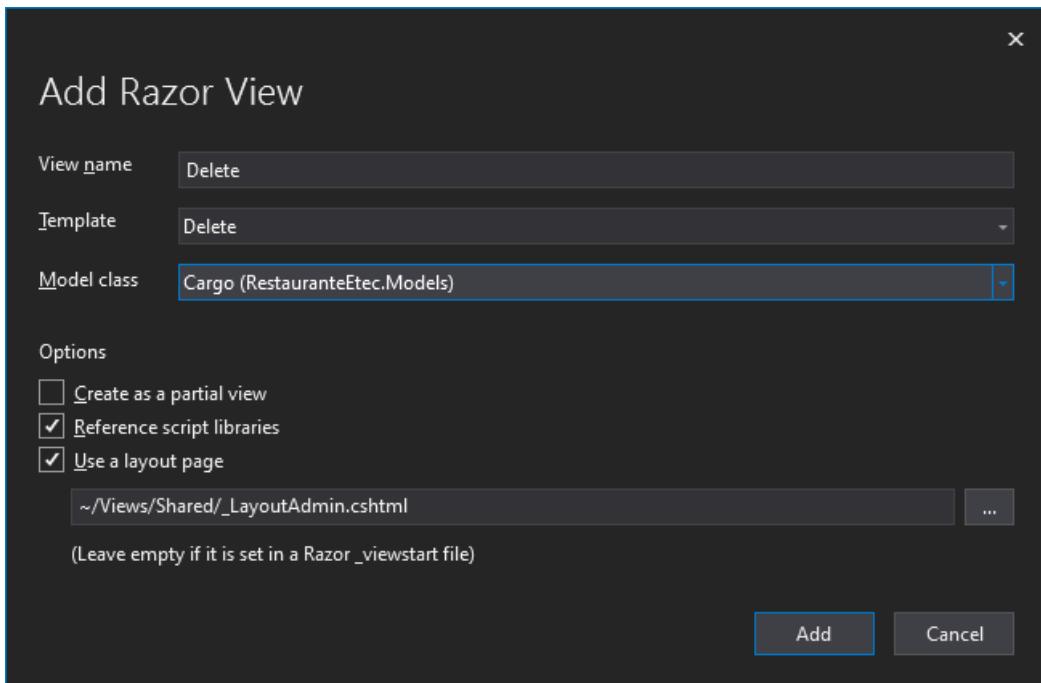
Vamos a próxima **View**, botão direito em **Edit** no **CargosController** e Adicionar Exibição (Add View):



Altere o código gerado conforme a imagem abaixo:

```
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">Editar Cargo</div>
13                 <hr>
14                 <form asp-action="Edit" method="post">
15                     <div asp-validation-summary="ModelOnly" class="text-danger"></div>
16                     <input type="hidden" asp-for="Id" />
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                         <input type="submit" value="Salvar" class="btn btn-success mr-2" />
24                         <a class="btn btn-danger" asp-action="Index">Cancelar</a>
25                     </div>
26                 </form>
27             </div>
28         </div>
29     </div>
30 </div>
31
32 @section Scripts {
33     @await Html.RenderPartialAsync("_ValidationScriptsPartial");
34 }
```

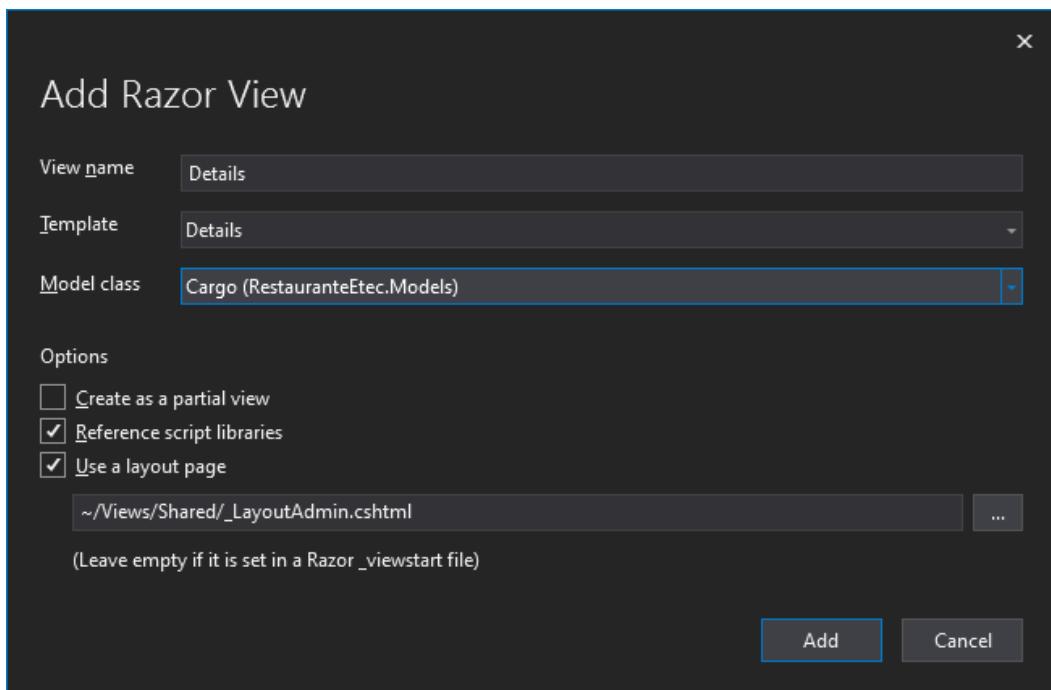
Vamos a próxima **View**, botão direito em **Delete** no **CargosController** e Adicionar Exibição (Add View):



Altere o código gerado conforme a imagem abaixo:

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

Vamos a próxima View, botão direito em **Details** no **CargosController** e Adicionar Exibição (Add View):



Altere o código gerado conforme a imagem abaixo:

```
Details.cshtml ✘ Delete.cshtml Edit.cshtml Create.cshtml Index.cshtml CargosController.cs
1  @model RestauranteEtec.Models.Cargo
2
3  @{
4      ViewData["Title"] = "Details";
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">Detalhes do Cargo</div>
13                 <hr>
14                 <dl class="row">
15                     <dt class="col-sm-2">
16                         @Html.DisplayNameFor(model => model.Nome)
17                     </dt>
18                     <dd class="col-sm-10">
19                         @Html.DisplayFor(model => model.Nome)
20                     </dd>
21                 </dl>
22                 <div>
23                     <a class="btn btn-warning mr-3" asp-action="Edit" asp-route-id="@Model.Id">Alterar</a>
24                     <a class="btn btn-light" asp-action="Index">Voltar a Listagem</a>
25                 </div>
26             </div>
27         </div>
28     </div>
29 </div>
```

Pode executar sem medo, e o resultado será o seguinte:

## Create:

A screenshot of a web application window titled "RestauranteEtec - Cargo". The URL in the address bar is <https://localhost:44349/Cargos/Create>. The page has a dark blue header with the title and a navigation bar with icons for home, notifications, and user profile.

The main content area is titled "Adicionar Cargo" (Add Position). It contains a "NOME" (Name) input field, which is currently empty. Below the input field are two buttons: a green "SALVAR" (Save) button and a red "CANCELAR" (Cancel) button. On the far right of the page is a small gear icon for settings.

The left sidebar, titled "RESTAURANTE ETEC", includes sections for SOCIAL (Dashboard, Blog, Contatos, Relatos, Reservas), MENU (Categorias, Produtos), and RECURSOS HUMANOS (Cargos, Funcionários). The "Cargos" item is highlighted with a blue background.

At the bottom right of the page, the text "Copyright © 2021 Restaurante Etec" is visible.

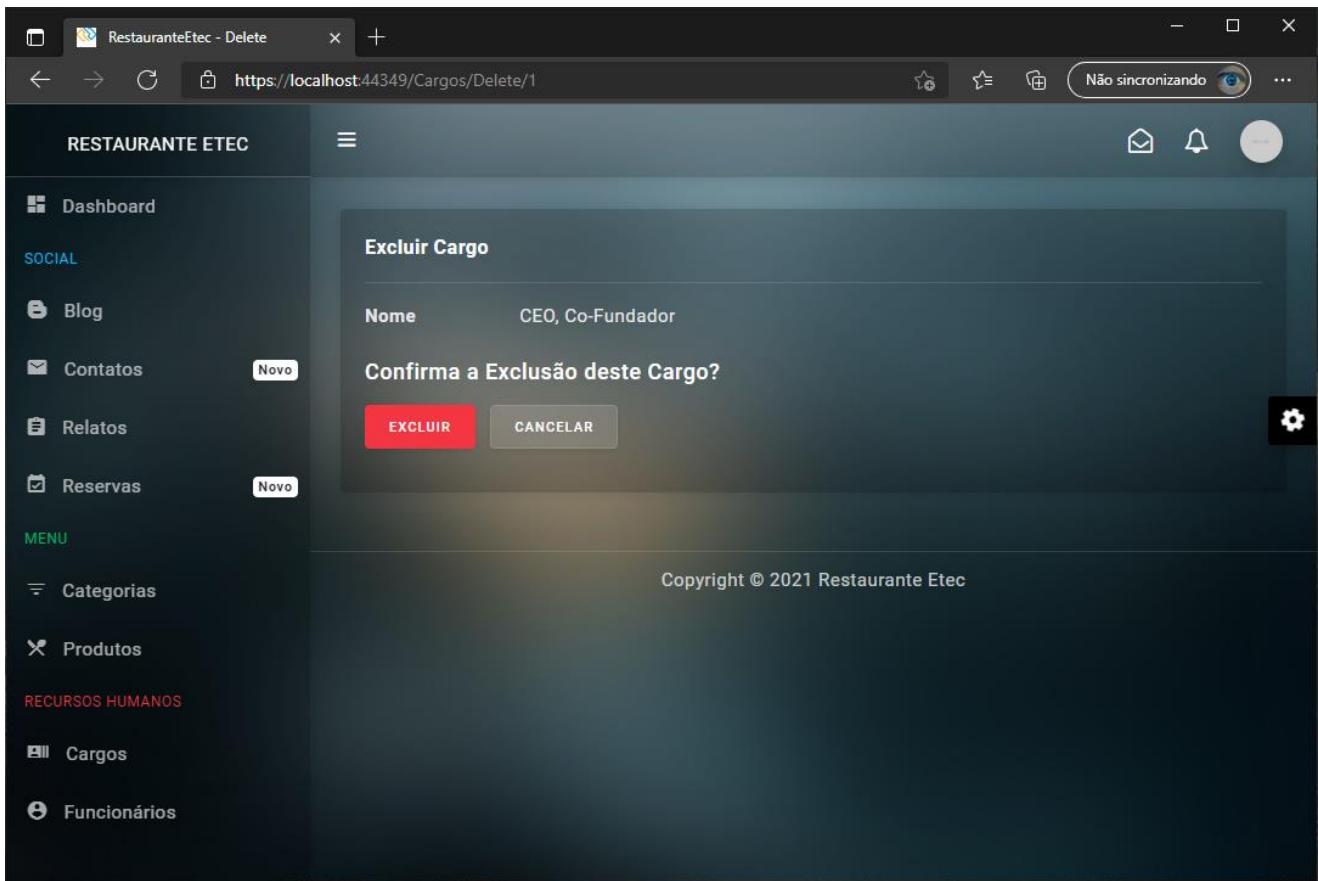
## Edit:

A screenshot of a web application window titled "RestauranteEtec - Cargo". The URL in the address bar is <https://localhost:44349/Cargos/Edit/1>. The interface is identical to the Create screen, showing the same header, sidebar, and footer.

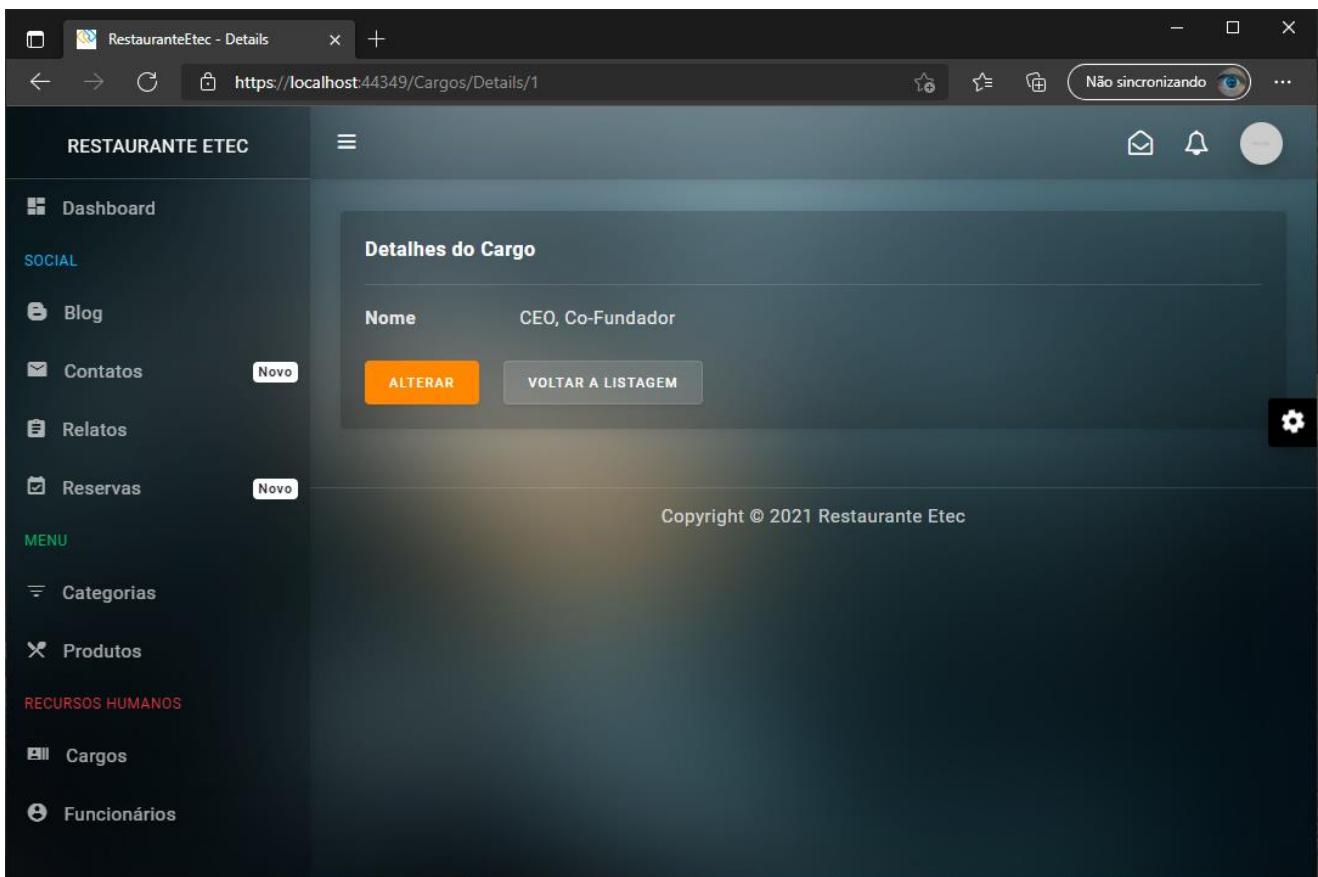
The main content area is titled "Editar Cargo" (Edit Position). It contains a "NOME" (Name) input field with the value "CEO, Co-Fundador". Below the input field are the same green "SALVAR" and red "CANCELAR" buttons. The gear icon for settings is also present on the right.

The left sidebar and footer are identical to the Create screen.

## Delete:



## Details:



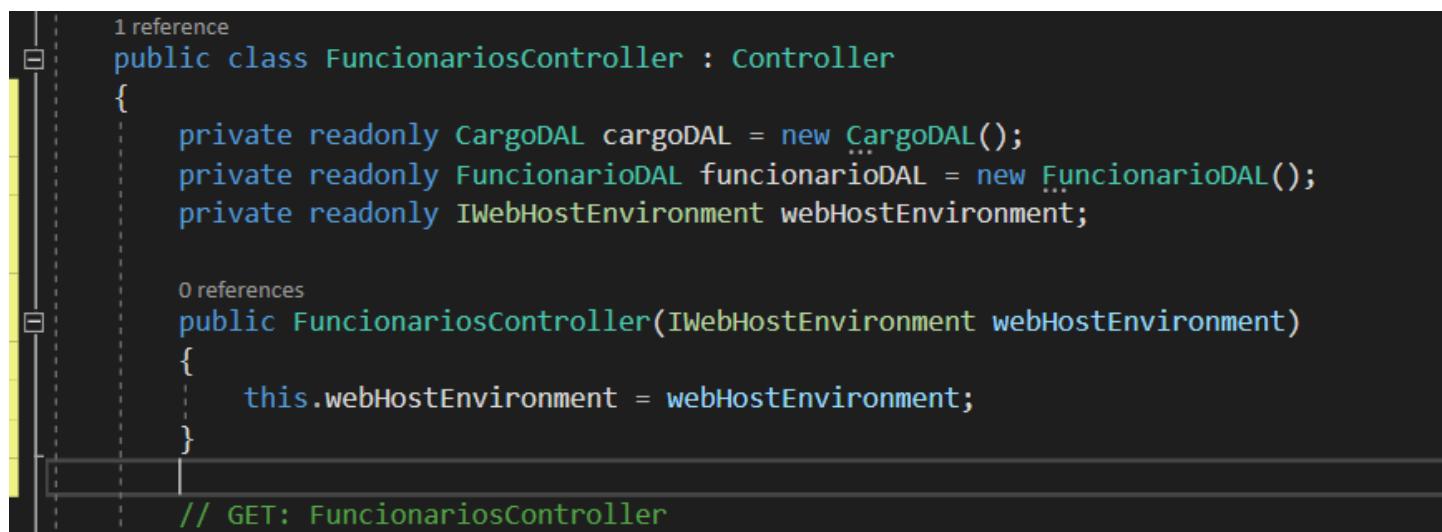
## CRIANDO O CRUD DE FUNCIONÁRIOS

Vamos agora criar um **controller** para realizar as operações de **CRUD** através de nossa camada **DAL**. Também vamos fazer uso de algumas facilidades que o Visual Studio Community oferece. Clique com o botão direito do mouse em Controllers, e selecione a opção **Adicionar (Add)** e no submenu **Controlador (Controller)**. Na lista que é exibida, escolhe **Controlador MVC com ações de leitura/escrita (MVC Controller with read/write actions)**, mude o nome para **FuncionariosController** (não esqueça de tirar o 1 no final do nome) e clique em **Adicionar (Add)**.

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 **FuncionarioDAL** e posteriormente vamos criar as **Views** de cada **Action**. Vale lembrar que cada funcionário possui um cargo, desta forma vamos precisar também de um objeto para pesquisar os dados dos cargos para exibir nas **Views**.

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.

Vamos começar incluindo no começo do **Controller** alguns objetos:



```
1 reference
public class FuncionariosController : Controller
{
    private readonly CargoDAL cargoDAL = new CargoDAL();
    private readonly FuncionarioDAL funcionarioDAL = new FuncionarioDAL();
    private readonly IWebHostEnvironment webHostEnvironment;

    0 references
    public FuncionariosController(IWebHostEnvironment webHostEnvironment)
    {
        this.webHostEnvironment = webHostEnvironment;
    }

    // GET: FuncionariosController
```

Não esqueça que é necessário clicar nos erros que surgirem para adicionar os **using**: `using RestauranteEtec.DAL;` `using Microsoft.AspNetCore.Hosting;`

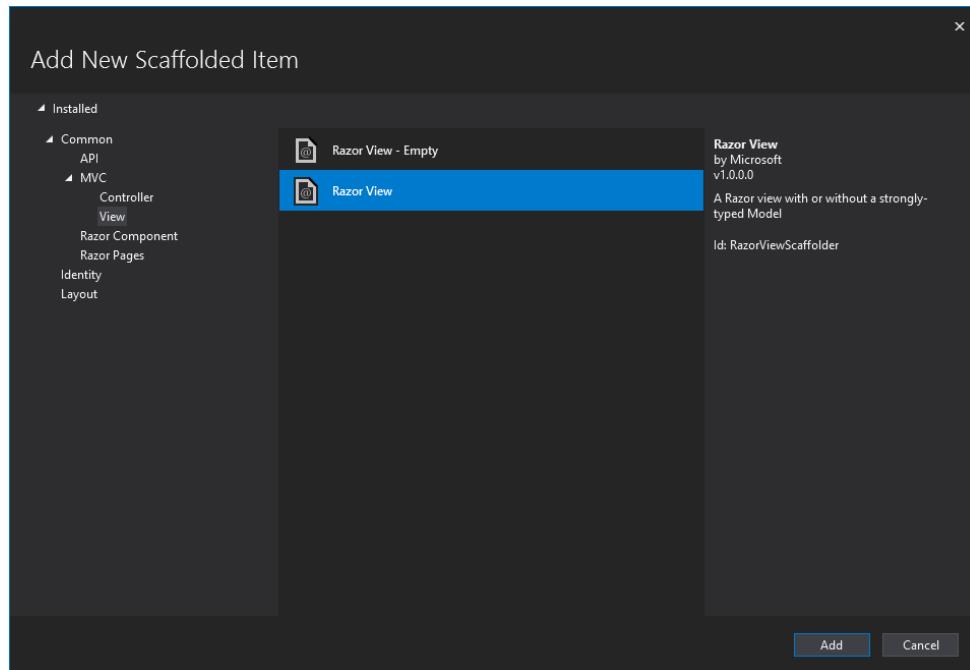
O construtor recebe o **IWebHostEnvironment**, que é o objeto que vai nos permitir ter acesso físico a pasta **wwwroot** no servidor.

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

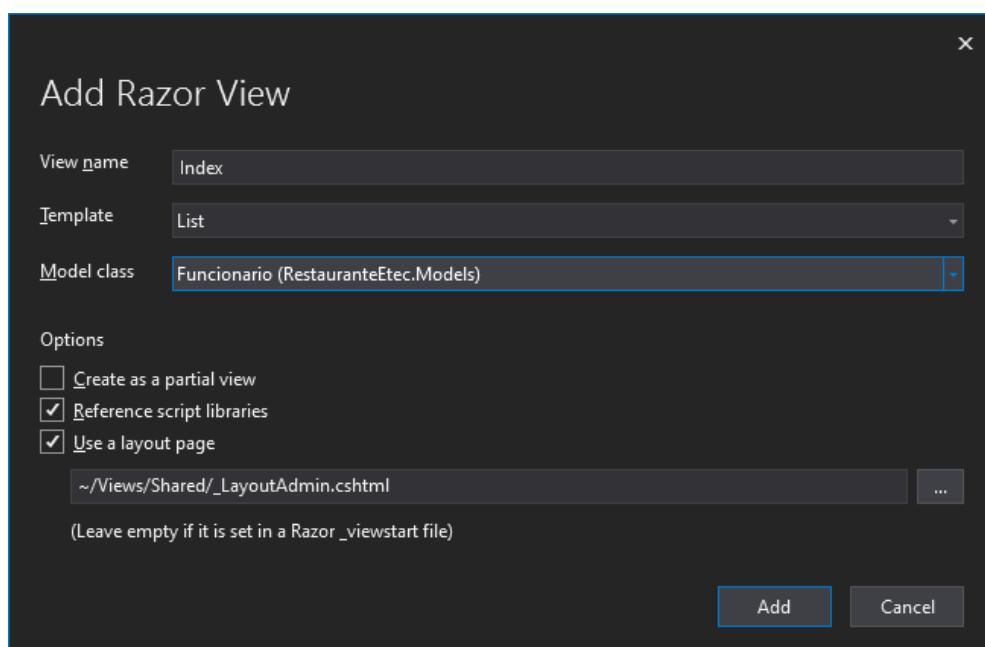
**Index:** Aqui utilizamos o método **GetAll**, programado na **FuncionarioDAL** para retornar todos os funcionários cadastrados a **View**.

```
23 // GET: FuncionariosController
24     public ActionResult Index()
25     {
26         var funcionarios = funcionarioDAL.GetAll();
27         return View(funcionarios);
28     }
29
```

Clique com o botão direito do mouse sobre o nome da **Action Index**, do **FuncionariosController**, e escolha a opção **Adicionar Exibição (Add View)**. Escolha opção **Exibição Razor (Razor View)**, conforme mostrado na imagem abaixo:



O Visual Studio então irá apresentar a tela abaixo, que você deve alterar para ficar conforme a imagem, antes de clicar em **Adicionar (Add)**:



Assim, como fizemos com o cargo, aqui também vamos editar essa **View**, para usar o mesmo layout de apresentação da área administrativa. Altere o código criado conforme o código abaixo:

```
Index.cshtml* ✘ FuncionariosController.cs
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.Nome)
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                                    @Html.DisplayFor(modelItem => item.ExibirHome)
46                                </td>
47                                <td>
48                                    @Html.DisplayFor(modelItem => item.Ativo)
49                                </td>
50
51                                <td>
52                                    <a asp-action="Edit" class="mr-3 text-warning" asp-route-id="@item.Id">
53                                        <i class="zmdi zmdi-edit" title="Alterar"></i>
54                                    </a>
55                                    <a asp-action="Details" class="mr-3 text-white" asp-route-id="@item.Id">
56                                        <i class="zmdi zmdi-zoom-in" title="Detalhes"></i>
57                                    </a>
58                                    <a asp-action="Delete" class="text-danger" asp-route-id="@item.Id">
59                                        <i class="zmdi zmdi-delete" title="Excluir"></i>
60                                    </a>
61                                </td>
62                            </tr>
63                        }
64                    </tbody>
65                </table>
66            </div>
67        </div>
68    </div>
69 </div>
```

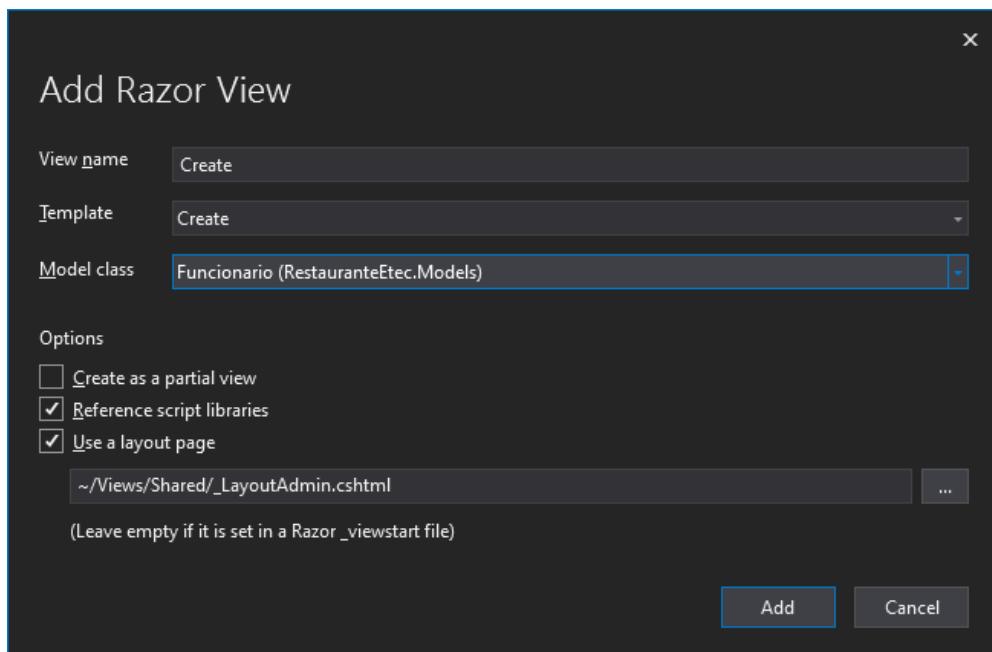
Volte ao **FuncionariosController** e vamos a próxima **action**:

**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 o usuário quando for incluir um novo funcionário, poderá de maneira fácil, apenas selecionar o cargo que aquele funcionário possui no restaurante. Ao digitar o código abaixo, não se esqueça de incluir o **using**: `using Microsoft.AspNetCore.Mvc.Rendering;`

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**).

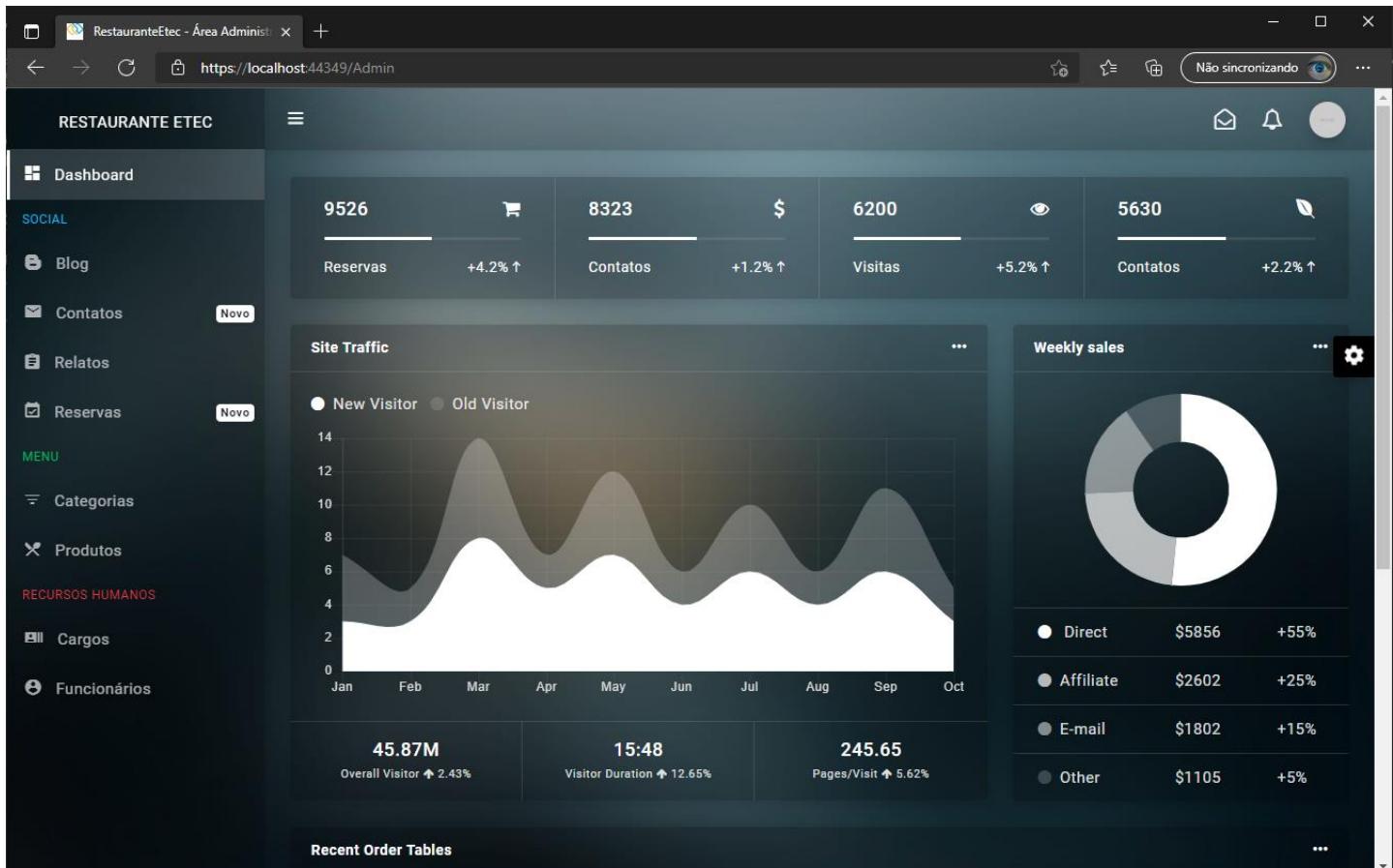
```
37     // GET: FuncionariosController/Create
38     public ActionResult Create()
39     {
40         ViewData["Cargos"] = new SelectList(cargoDAL.GetAll(), "Id", "Nome");
41         return View();
42     }
43
```

Clique com o botão direito do mouse sobre o nome da **Action Create**, e escolha a opção **Adicionar Exibição (Add View)**. Escolha opção **Exibição Razor (Razor View)**, e faça as alterações conforme a imagem abaixo, antes de clicar em **Adicionar (Add)**:



Se executarmos nosso projeto e navegarmos até a área administrativa, posteriormente clicando no link **Funcionários**, será exibido a **View Index** do controlador **FuncionariosController**.

**Área Administrativa: View Index do AdminController**



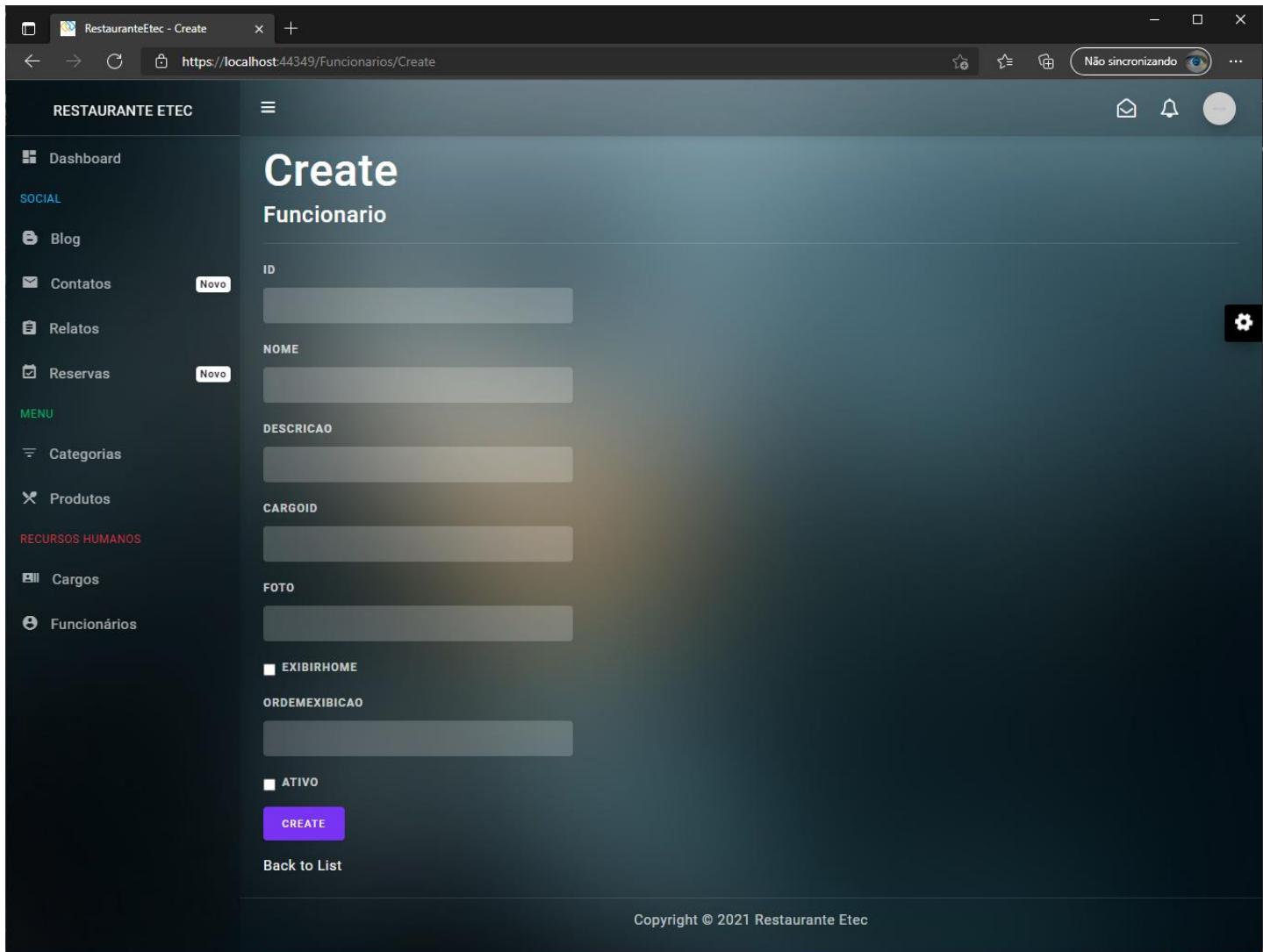
## Lista de Funcionários: View Index do FuncionariosController

The screenshot shows the 'Lista de Funcionários' view. The sidebar includes categories: SOCIAL (Blog, Contatos), MENU (Categorias, Produtos), and RECURSOS HUMANOS (Cargos, Funcionários). The main content displays a table of employees:

NOME	NOME	EXIBIRHOME	ATIVO	AÇÕES
João Gustavo	CEO, Co-Fundador	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<span>editar</span> <span>deletar</span>
Michelle Fraulen	Chefe de Cozinha	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<span>editar</span> <span>deletar</span>
Alfred Smith	Cozinheiro Chefe	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<span>editar</span> <span>deletar</span>
Antonio Santibanez	Cozinheiro Chefe	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<span>editar</span> <span>deletar</span>
Gallo	Chefe de Cozinha	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<span>editar</span> <span>deletar</span>

At the bottom, it says 'Copyright © 2021 Restaurante Etec'.

Clicando no botão **Adicionar Novo**, será exibida a **View Create**, que ainda não alteramos:



Pessoal, o que precisa ficar claro aqui é que a técnica **Scaffolding** facilita a criação dos arquivos, mas ela não irá gerar páginas visualmente interessantes ou que acompanhem o design abordado no projeto. Ou seja, é importante após a criação do arquivo sempre fazer alterações para editar o arquivo Razor (Views .cshtml), colocando-se no lugar do usuário que irá utilizar a página e incluindo elementos que facilitem sua utilização (usabilidade).

Na imagem acima, podemos ver que o **Id** está sendo exibido, o **Cargold** deveria ter no lugar uma lista de cargos, o campo **Foto** não permite subir um arquivo de imagem e as caixas de checagem (**ExibirHome** e **Ativo**) não estão seguindo o layout existente no layout (arquivo **forms.html** do **template** de área administrativa, possui classes para estilizar elementos de formulários).

A proposta que tenho para página é bem diferente. Vamos separar a foto do restante dos campos, colocando na lateral direita da página, além disso, iremos usar um código **javascript** que será acionado toda vez que o elemento **input** do tipo **file** sofrer uma alteração (**change**), ou seja, sempre que o usuário selecionar uma imagem, o **javascript** irá exibir essa imagem na página, como uma visualização prévia. Vamos incluir nosso **select**, exibindo uma lista de cargos, e alteraremos por fim as classes das caixas de checagem para usar o layout do admin.

**enctype="multipart/form-data"** – essa instrução no **form**, é a que permite enviar ao servidor as imagens carregadas, não esqueça dela pois é muito importante.

Vamos as alterações:

```

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-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="2" ></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
33                             <div class="form-row">
34                                 <div class="form-group col-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-4 pt-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-4 pt-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-3">
53                             <div class="row">
54                                 <div class="form-group col-12">
55                                     <img src="" class="img-fluid" id='PreviewImagen' />
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             </div>
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('PreviewImagen');
81                         img.src = URL.createObjectURL(this.files[0]);
82                     }
83                 });
84             });
85         </script>
86     }

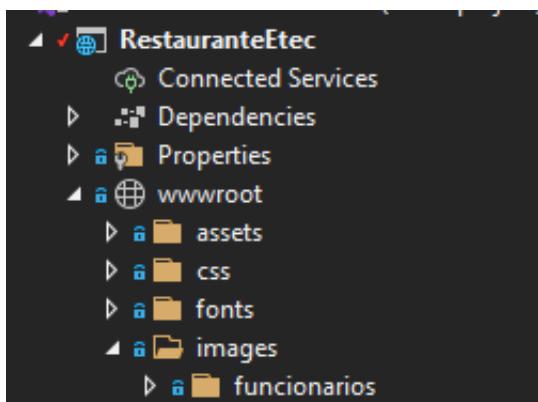
```

Antes de executarmos nosso código, vamos voltar ao **FuncionariosController** e editar o **Post** do **Create**, para receber os dados da **View**, e se tudo estiver correto, gravar no servidor uma cópia da imagem carregada e os dados do novo funcionário no banco de dados.

**Create (Post):** Para o código abaixo funcionar corretamente, precisamos adicionar ao **usings** no começo do controlador duas linhas novas: **using RestauranteEtec.Models;** **using System.IO;** Também 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.

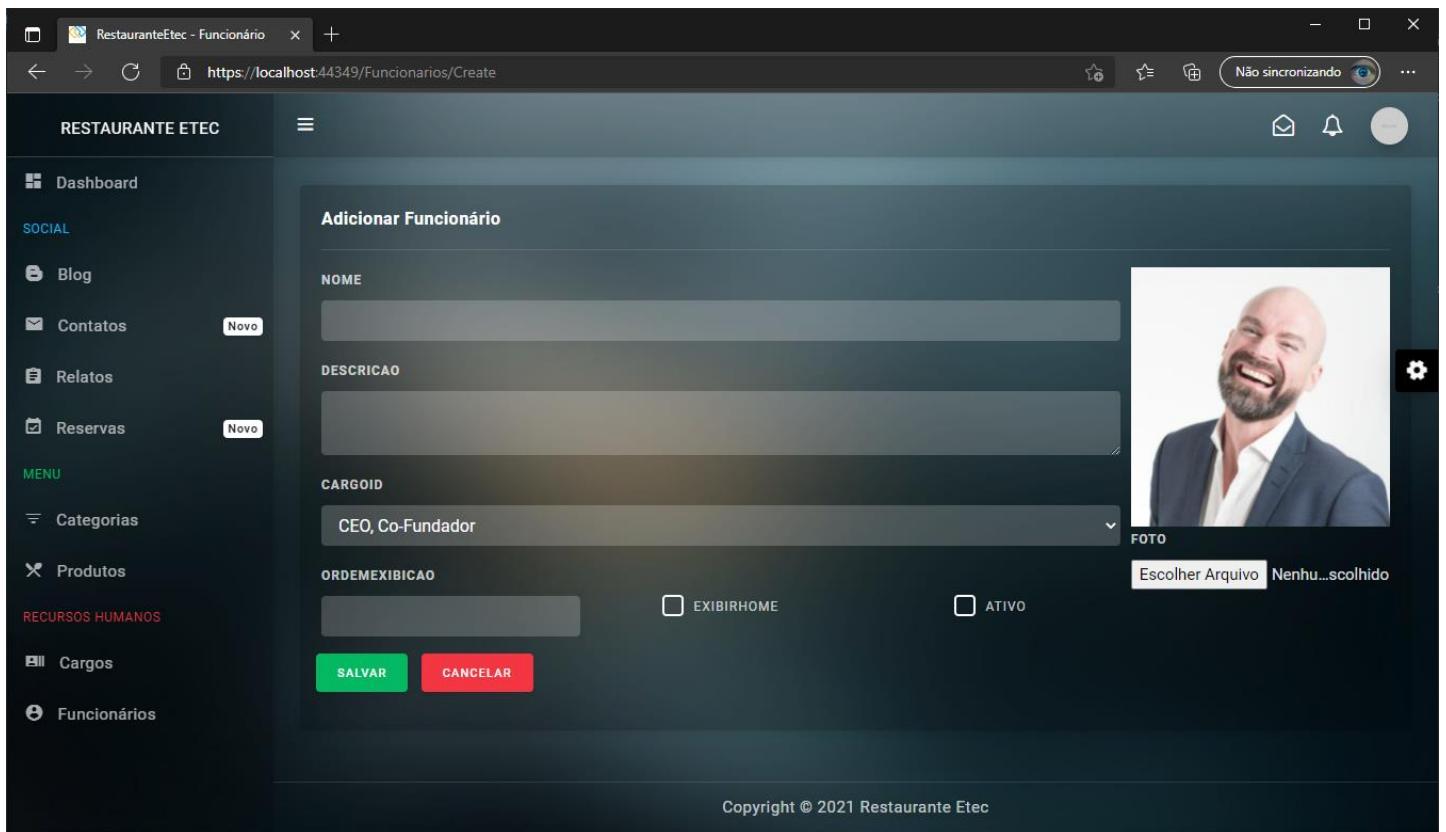
```
46 // POST: FuncionariosController/Create
47 [HttpPost]
48 [ValidateAntiForgeryToken]
49 public async Task<ActionResult> Create([Bind] Funcionario funcionario, IFormFile Foto)
50 {
51     ViewData["Cargos"] = new SelectList(cargoDAL.GetAll(), "Id", "Nome");
52     if (!ModelState.IsValid)
53         return View(funcionario);
54     try
55     {
56         if (Foto != null)
57         {
58             string pasta = Path.Combine(webHostEnvironment.WebRootPath, "images\\funcionarios");
59             var nomeArquivo = Guid.NewGuid().ToString() + "_" + Foto.FileName;
60             string caminhoArquivo = Path.Combine(pasta, nomeArquivo);
61             using (var stream = new FileStream(caminhoArquivo, FileMode.Create))
62             {
63                 await Foto.CopyToAsync(stream);
64             };
65             funcionario.Foto = "/images/funcionarios/" + nomeArquivo;
66         }
67         funcionarioDAL.Add(funcionario);
68         return RedirectToAction("Index");
69     }
70     catch
71     {
72         return View(funcionario);
73     }
74 }
75 }
```

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 será 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 recebidas e processadas 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.

Agora sim, executando novamente nosso projeto e fazendo toda a navegação a página de Adicionar Novo funcionário, temos então um layout moderno e simples, onde após clicar no botão para selecionar uma foto, já conseguimos ter uma visualização dela.



Vamos para a execução, voltar ao nosso **FuncionariosController**, e codificar a **Action Edit (Get)**.

**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 84, 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.

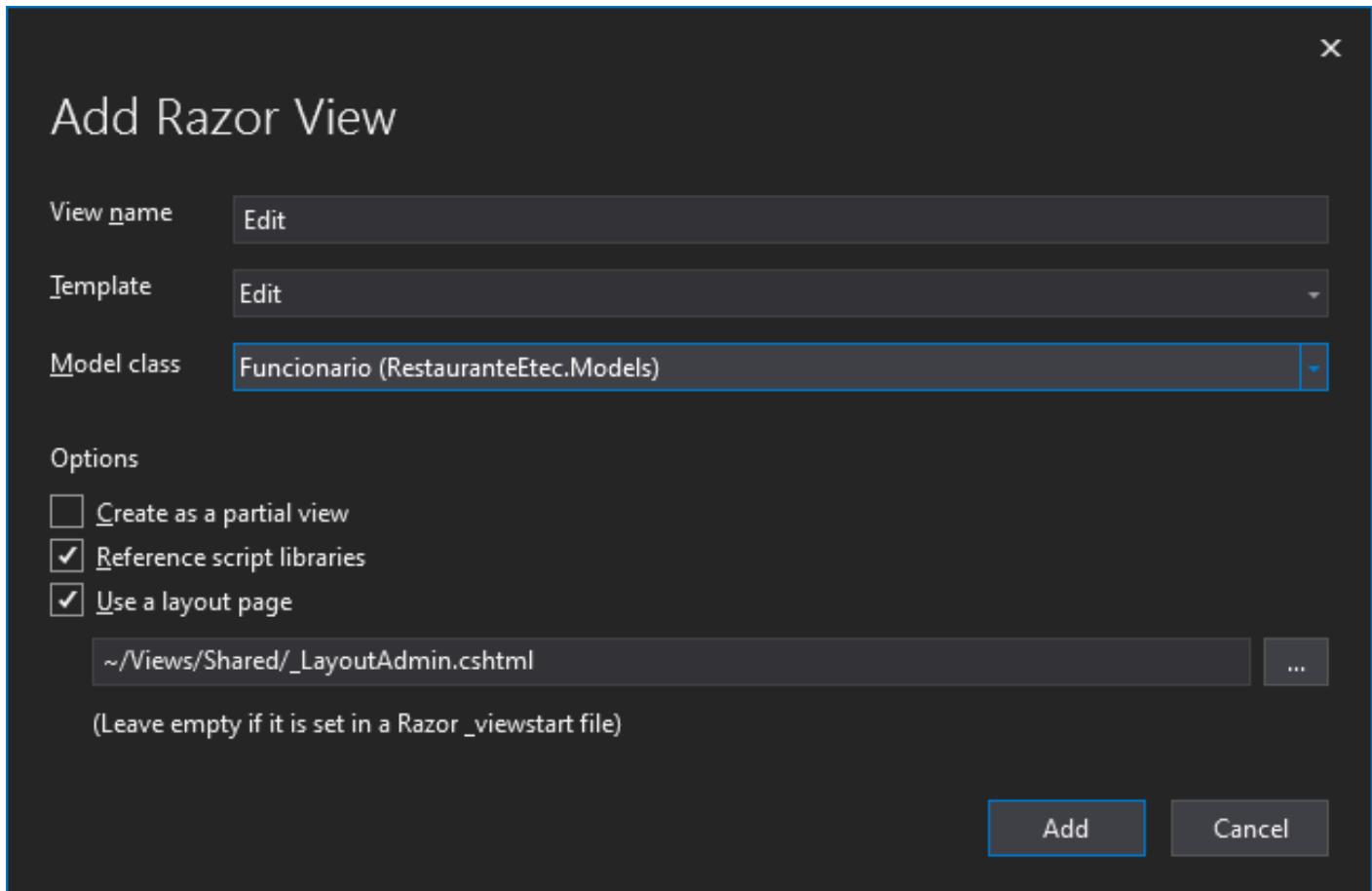
```
76 // GET: FuncionariosController/Edit/5
77 public ActionResult Edit(int? id)
78 {
79     if (id == null)
80         return NotFound();
81     var funcionario = funcionarioDAL.GetById(id);
82     if (funcionario == null)
83         return NotFound();
84     ViewData["wwwroot"] = webHostEnvironment.WebRootPath;
85     ViewData["Cargos"] = new SelectList(cargoDAL.GetAll(), "Id", "Nome");
86     return View(funcionario);
87 }
```

Assim, como mencionado acima, vamos precisar de uma imagem que informe ao usuário que aquele funcionário ainda não tem foto ou que sua foto não está mais disponível no servidor. Para isso você precisa fazer o download no canal **Semana 17**, do arquivo “**sem\_foto.png**” e copiar esse arquivo na pasta **images** dentro de **wwwroot**.



FOTO INDISPONÍVEL

Clique com o botão direito do mouse sobre o nome da **Action Edit**, e escolha a opção **Adicionar Exibição (Add View)**. Escolha opção **Exibição Razor (Razor View)**, e faça as alterações conforme a imagem abaixo, antes de clicar em **Adicionar (Add)**:



O código que vamos alterar aqui é bem semelhante ao que fizemos no **Create.cshtml**.

O que precisa ficar bem claro, é que semelhante, nem sempre é igual. Temos alguns pontos bem distintos:

- A **Action** do **form** (linha 12).
- A linha 13, que é responsável por guardar o valor do **Id** do funcionário que está em edição.
- E as linhas 56 a 61 onde:
  - A linha 56, guarda a **foto** atual do funcionário;
  - A linha 57/58/59 exibe a foto atual ou a **sem\_foto.png**;
  - A linha 60/61 cria **um input file** que permite ao usuário escolher outra foto.

Sem mais delongas, vamos ao código:

```

1  @model RestauranteEtec.Models.Funcionario
2  @{
3      ViewData["Title"] = "Funcionário";
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 Cargo</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                                     
59                                     <input type="file" id="NovaFoto" name="NovaFoto" class="form-control-file" 
60                                         accept=".jpg,.jpeg,.png,.gif" />
61                                     <span asp-validation-for="Foto" class="text-danger"></span>
62                                 </div>
63                             </div>
64                         </div>
65                         <div class="form-group">
66                             <input type="submit" value="Salvar" class="btn btn-success mr-2" />
67                             <a class="btn btn-danger" asp-action="Index">Cancelar</a>
68                         </div>
69                     </div>
70                 </div>
71             </form>
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('PreviewImagen');
83                     img.src = URL.createObjectURL(this.files[0]);
84                 }
85             });
86         });
87     </script>
88 }

```

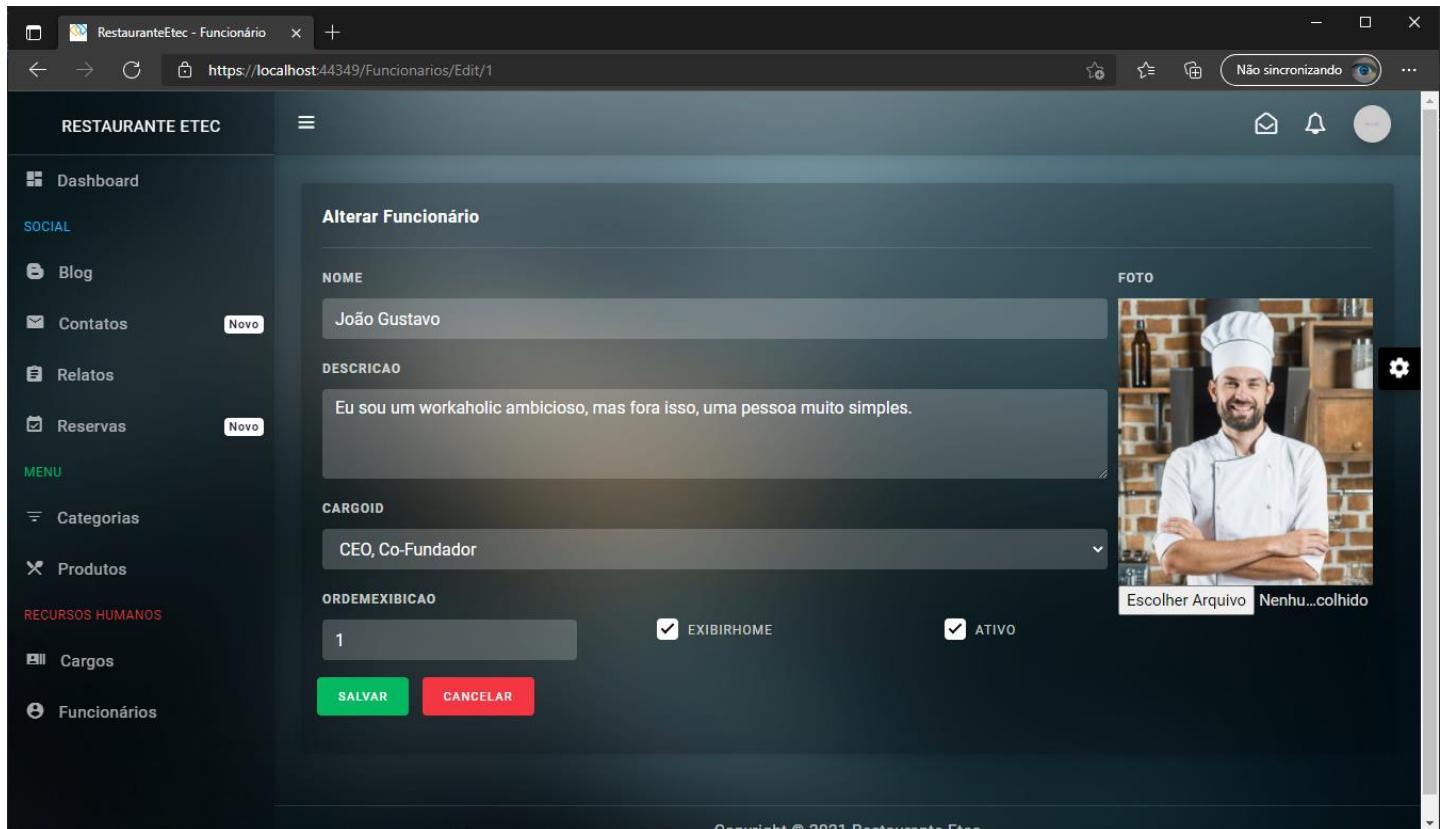
Antes de executarmos nosso código, vamos voltar ao **FuncionariosController** e editar o **Post** do **Edit**, para receber os dados da **View**, e se tudo estiver correto, gravar os dados alterados do funcionário no banco de dados, e se uma nova foto foi enviada ao servidor, gravar uma cópia dessa foto.

**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**.

```
89 // POST: FuncionariosController/Edit/5
90 [HttpPost]
91 [ValidateAntiForgeryToken]
92 public async Task<ActionResult> Edit(int id, [Bind] Funcionario funcionario, IFormFile NovaFoto)
93 {
94     if (id != funcionario.Id)
95         return NotFound();
96     ViewData["wwwroot"] = webHostEnvironment.WebRootPath;
97     ViewData["Cargos"] = new SelectList(cargoDAL.GetAll(), "Id", "Nome");
98     if (!ModelState.IsValid)
99         return View(funcionario);
100    try
101    {
102        if (NovaFoto != null)
103        {
104            string pasta = Path.Combine(webHostEnvironment.WebRootPath, "images\\funcionarios");
105            var nomeArquivo = Guid.NewGuid().ToString() + "_" + NovaFoto.FileName;
106            string caminhoArquivo = Path.Combine(pasta, nomeArquivo);
107            using (var stream = new FileStream(caminhoArquivo, FileMode.Create))
108            {
109                await NovaFoto.CopyToAsync(stream);
110            };
111            funcionario.Foto = "/images/funcionarios/" + nomeArquivo;
112        }
113        funcionarioDAL.Update(funcionario);
114        return RedirectToAction("Index");
115    }
116    catch
117    {
118        return View(funcionario);
119    }
120 }
```

Agora sim, executando novamente nosso projeto e fazendo toda a navegação a página de que lista nossos funcionários, podemos clicar no ícone de **Lápis** na última coluna, para abrir a página de edição desse funcionário. Como em nosso banco de dados, o caminho das imagens dos funcionários, tem um “~”, ocorra a exibição do **sem\_foto.png**.

Clique no botão **Escolher Arquivo** e localize a foto na pasta **images** do projeto, salve a alteração e entre novamente na página de edição do funcionário.



Vamos para a execução, voltar ao nosso **FuncionariosController**, e codificar a **Action Delete (Get)**.

**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 136, 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.

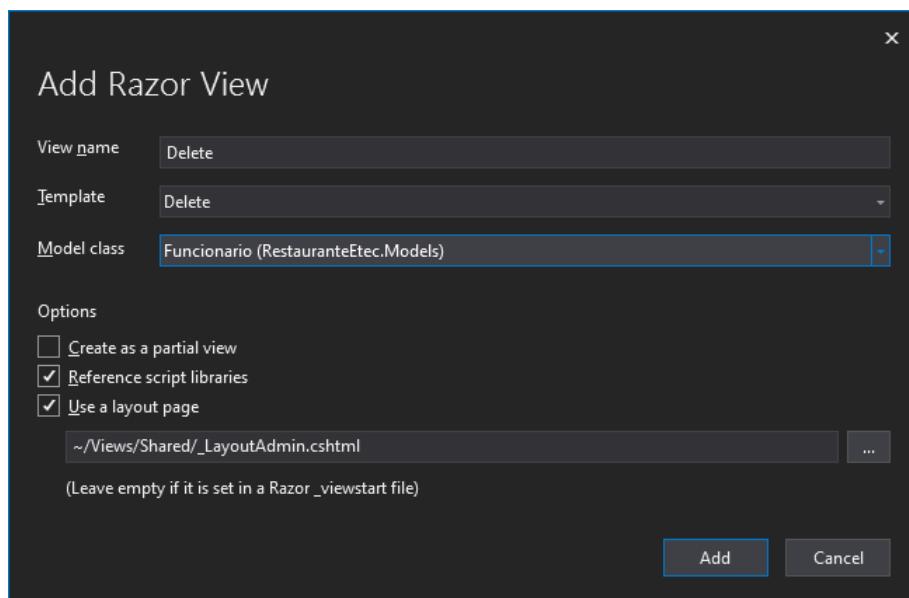
```

128 // GET: FuncionariosController/Delete/5
129 0 references
130 public ActionResult Delete(int? id)
131 {
132     if (id == null)
133         return NotFound();
134     var funcionario = funcionarioDAL.GetById(id);
135     if (funcionario == null)
136         return NotFound();
137     ViewData["wwwroot"] = webHostEnvironment.WebRootPath;
138     return View(funcionario);
139 }
```

Vamos aproveitar que estamos no **controller** e vamos alterar a **Action Delete Post**:

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

Clique com o botão direito do mouse sobre o nome da **Action Delete**, e escolha a opção **Adicionar Exibição (Add View)**. Escolha opção **Exibição Razor (Razor View)**, e faça as alterações conforme a imagem abaixo, antes de clicar em **Adicionar (Add)**:



Agora vamos editar o layout para melhor apresentação dos dados.

```

1 @model RestauranteEtec.Models.Funcionario
2
3 @{
4     ViewData["Title"] = "Funcionário";
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">Excluir Funcionário</div>
13                 <hr>
14                 <div class="row">
15                     <div class="col-3">
16                         
17                         class="img-fluid" id='PreviewImagem' />
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                             <label class="col-3" asp-for="ExibirHome"></label>
34                             <label class="col-9"[@Html.DisplayFor(model => model.ExibirHome)</label>
35                         </div>
36                         <div class="row">
37                             <label class="col-3" asp-for="Ativo"></label>
38                             <label class="col-9"[@Html.DisplayFor(model => model.Ativo)</label>
39                         </div>
40                         <div class="row">
41                             <label class="col-3" asp-for="OrdemExibicao"></label>
42                             <label class="col-9"[@Html.DisplayFor(model => model.OrdemExibicao)</label>
43                         </div>
44                         </div>
45                     </div>
46
47
48                 <form asp-action="Delete" method="post">
49                     <div asp-validation-summary="ModelOnly" class="text-danger"></div>
50                     <input type="hidden" asp-for="Id" />
51                     <h5 class="my-3">Confirma a Exclusão deste Funcionário?</h5>
52                     <div class="form-group">
53                         <input type="submit" value="Excluir" class="btn btn-danger mr-2" />
54                         <a class="btn btn-light" asp-action="Index">Cancelar</a>
55                     </div>
56                 </form>
57             </div>
58         </div>
59     </div>
60 </div>
61

```

Agora sim, executando novamente nosso projeto e fazendo toda a navegação a página de que lista nossos funcionários, podemos clicar no ícone de lixeira na última coluna, para abrir a página de exclusão deste funcionário.

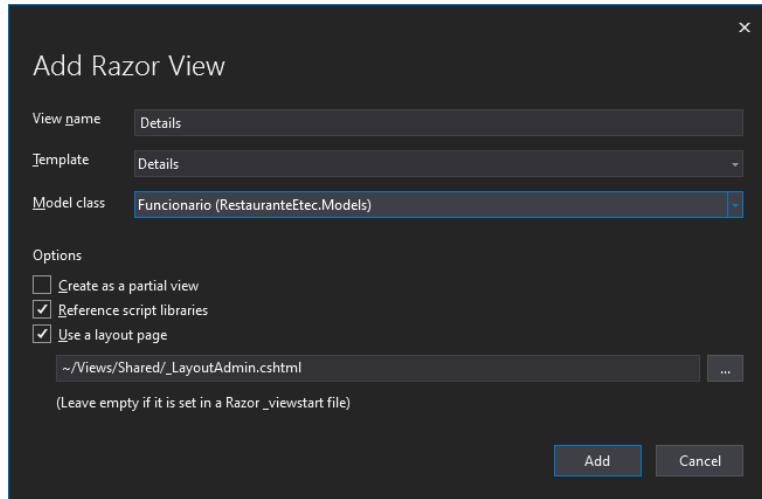
Para finalizar só falta a **Action Details**:

```

33 // GET: FuncionariosController/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);
}

```

Clique com o botão direito do mouse sobre o nome da **Action Delete**, e escolha a opção **Adicionar Exibição (Add View)**. Escolha opção **Exibição Razor (Razor View)**, e faça as alterações conforme a imagem abaixo, antes de clicar em **Adicionar (Add)**:



```

1  @model RestauranteEtec.Models.Funcionario
2
3  @{
4      ViewData["Title"] = "Funcionário";
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">Detalhes do Funcionário</div>
13                 <hr>
14                 <div class="row">
15                     <div class="col-3">
16                         
19                     </div>
20                     <div class="col-9">
21                         <div class="row">
22                             <label class="col-3" asp-for="Nome"></label>
23                             <label class="col-9">@Html.DisplayFor(model => model.Nome)</label>
24                         </div>
25                         <div class="row">
26                             <label class="col-3" asp-for="Descricao"></label>
27                             <label class="col-9">@Html.DisplayFor(model => model.Descricao)</label>
28                         </div>
29                         <div class="row">
30                             <label class="col-3" asp-for="CargoId"></label>
31                             <label class="col-9">@Html.DisplayFor(model => model.Cargo.Nome)</label>
32                         </div>
33                         <div class="row">
34                             <label class="col-3" asp-for="ExibirHome"></label>
35                             <label class="col-9">@Html.DisplayFor(model => model.ExibirHome)</label>
36                         </div>
37                         <div class="row">
38                             <label class="col-3" asp-for="Ativo"></label>
39                             <label class="col-9">@Html.DisplayFor(model => model.Ativo)</label>
40                         </div>
41                         <div class="row">
42                             <label class="col-3" asp-for="OrdemExibicao"></label>
43                             <label class="col-9">@Html.DisplayFor(model => model.OrdemExibicao)</label>
44                         </div>
45                     </div>
46                 </div>
47
48                 <div class="mt-3">
49                     <a class="btn btn-warning mr-3" asp-action="Edit" asp-route-id="@Model.Id">Alterar</a>
50                     <a class="btn btn-light" asp-action="Index">Voltar a Listagem</a>
51                 </div>
52             </div>
53         </div>
54     </div>
55 </div>
56

```

E só executar

RestauranteEtec - Funcionário

https://localhost:44349/Funcionarios/Details/1

RESTAURANTE ETEC

Dashboard

SOCIAL

- Blog
- Contatos Novo
- Relatos
- Reservas Novo

MENU

- Categorias
- Produtos

RECURSOS HUMANOS

- Cargos
- Funcionários

Detalhes do Funcionário



NOME	JOÃO GUSTAVO
DESCRICAO	EU SOU UM WORKAHOLIC AMBICIOSO, MAS FORA ISSO, UMA PESSOA MUITO SIMPLES.
CARGOID	CEO, CO-FUNDADOR
EXIBIRHOME	<input checked="" type="checkbox"/>
ATIVO	<input checked="" type="checkbox"/>
ORDEMEXIBICAO	1

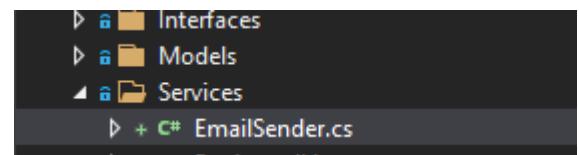
[ALTERAR](#) [VOLTAR A LISTAGEM](#)

Copyright © 2021 Restaurante Etec

## ADICIONANDO UM SERVIÇO DE DISPARO DE E-MAIL

Existem várias formas de criação de um disparado 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**.



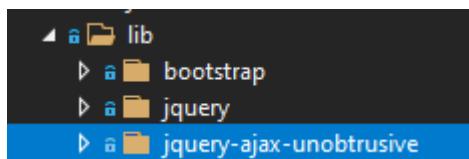
```
1  using System;
2  using System.Net;
3  using System.Net.Mail;
4  using System.Threading.Tasks;
5
6  namespace RestauranteEtec.Services
7  {
8      public class EmailSender
9      {
10         public async Task<bool> Mail(string Para, string De, string Assunto, string Mensagem)
11         {
12 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

Nossa classe é bem simples, criamos um **Mail Message**, que representa uma mensagem em formato **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 no canal da Semana 18, 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 EmailSender.cs* Index.cshtml FuncionariosController.cs Edit.cshtml BlogController.cs Answ
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, podemos incluir esta **PartialView** nas páginas que precisamos usar requisições com **AJAX**.

Vamos agora criar o nosso **Controller**, assim como fizemos com os demais. Adicione na pasta **Controllers** o **ContatosController**.

Modifique o começo do código adicionando os seguintes **usings**:

```
1 using Microsoft.AspNetCore.Http;
2 using Microsoft.AspNetCore.Mvc;
3 using RestauranteEtec.DAL;
4 using RestauranteEtec.Models;
5 using RestauranteEtec.Services;
6 using System;
7 using System.Collections.Generic;
8 using System.Linq;
9 using System.Threading.Tasks;
```

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.

```
0 references
13 public class ContatosController : Controller
14 {
15     private readonly ContatoDAL contatoDAL = new ContatoDAL();
16 }
```

Como comentei acima, não vamos ter uma página para o usuário cadastrar “**Create**” um contato, então vamos apagar o método **Create (GET)**, criado automaticamente. Vamos alterar o **Create (POST)** para receber nossa requisição **AJAX**, que vamos criar logo mais na página de **Contatos**. Dessa forma, apague o código da imagem abaixo:

```
// GET: ContatosController/Create
0 references
public ActionResult Create()
{
    return View();
}
```

E altere o código do **Create (POST)**, conforme a imagem:

```
// POST: ContatosController/Create
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<JsonResult> Create([Bind] Contato contato)
{
    if (ModelState.IsValid)
    {
        contatoDAL.Add(contato);
        var email = new EmailSender();
        var assunto = "Contato feito no RestauranteEtec";
        var mensagem = "<b>Pessoa: </b> " + contato.NomePessoa;
        mensagem += "<br><b>E-mail: </b> " + contato.EmailPessoa;
        mensagem += "<br><b>Assunto: </b> " + contato.Assunto;
        mensagem += "<br><b>Mensagem: </b> " + contato.Mensagem;
        await email.Mail("gallojunior@gmail.com", contato.EmailPessoa, assunto, mensagem);
        return Json(data: "done");
    }
    return Json(data: "erro");
}
```

O que está fazendo aqui é conferindo 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.

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**:

Na imagem abaixo, você precisa adicionar a linha 1 e em seguida fazer as alterações nas linhas a partir da seção que está linha 49:

```

Contatos.cshtml*  ContatosController.cs      Index.cshtml      Edit.cshtml      BlogController.cs      Answer.cshtml      _Grid.cshtml      FuncionariosController.cs
1  @model RestauranteEtec.Models.Contato
2  @{
3      ViewData["Title"] = "Contatos";
4  }
5
6  <section class="hero-wrap hero-wrap-2">...</section>
7
8  <section class="ftco-section contact-section bg-light">...</section>
9
10 <section class="ftco-section ftco-no-pt contact-section">
11     <div class="container">
12         <div class="row d-flex align-items-stretch no-gutters">
13             <div class="col-md-6 p-5 order-md-last">
14                 <h2 class="h4 mb-5 font-weight-bold">Contact Us</h2>
15                 <form asp-action="Create" asp-controller="Contatos" method="post" data-ajax="true"
16                     data-ajax-method="post" data-ajax-complete="completed" data-ajax-failure="failed">
17                     <div asp-validation-summary="ModelOnly" class="text-danger hidden"></div>
18                     <div class="form-group">
19                         <input asp-for="NomePessoa" class="form-control" placeholder="Seu Nome" />
20                         <span asp-validation-for="NomePessoa" class="text-danger"></span>
21                     </div>
22                     <div class="form-group">
23                         <input asp-for="EmailPessoa" class="form-control" placeholder="Seu E-mail" />
24                         <span asp-validation-for="EmailPessoa" class="text-danger"></span>
25                     </div>
26                     <div class="form-group">
27                         <input asp-for="Assunto" class="form-control" placeholder="Assunto" />
28                         <span asp-validation-for="Assunto" class="text-danger"></span>
29                     </div>
30                     <div class="form-group">
31                         <textarea asp-for="Mensagem" cols="30" rows="7" class="form-control" placeholder="Mensagem"></textarea>
32                         <span asp-validation-for="Mensagem" class="text-danger"></span>
33                     </div>
34                     <div class="form-group">
35                         <input type="submit" value="Enviar Mensagem" class="btn btn-primary py-3 px-5">
36                     </div>
37                 </form>
38             </div>
39             <div class="col-md-6 d-flex align-items-stretch">
40                 <div id="map">
41                     <iframe src="https://www.google.com/maps/embed?pb=!m14!1m8!1m3!1d14745.589106937443!2d-48.5461716!3d-22.489277" style="width:100%; height:100%; border:none; margin-bottom:10px;"></iframe>
42                 </div>
43             </div>
44         </div>
45     </div>
46 </section>
47
48 @section Scripts{
49     @await Html.RenderPartialAsync("_ValidationScriptsPartial");
50     <script src="https://cdnjs.cloudflare.com/ajax/libs/sweetalert/2.1.2/sweetalert.min.js"></script>
51
52     <script type="text/javascript">
53         completed = function (xhr) {
54             swal({
55                 title: "Obrigado!",
56                 text: "Agradecemos seu contato e retornaremos em breve!!",
57                 icon: "success",
58                 button: "OK"
59             });
60             document.getElementById("formContato").reset();
61         }
62         failed = function (xhr) {
63             swal({
64                 title: "Problemas ao Enviar!",
65                 text: "Ocorreu um problema no envio de sua mensagem! Por favor, tente novamente em alguns instantes!",
66                 icon: "warning",
67                 button: "OK"
68             });
69         }
70     </script>
71 }

```

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

Screenshot of the RestauranteEtec website's contact page. The map shows the location of Etec Comendador João Rays in Barra Bonita, SP. The contact form is visible on the right.

**RestauranteEtec**

HOME QUEM SOMOS CHEFES MENU RESERVAS BLOG CONTATOS

**Etec Comendador João Rays**  
Rua Ludovico Victório, 2140 - Vila Hab., Barra Bonita - SP, 17340-000  
4,8 ★★★★ 12 comentários  
[Visualizar mapa ampliado](#)

**Contact Us**

Seu Nome

Seu E-mail

Assunto

Mensagem

**Enviar Mensagem**

Mapa: Google Maps showing the location of Etec Comendador João Rays in Barra Bonita, SP. The map includes labels for Rua Ludovico Victório, Coplacana, COLINA DA BARRA, PORTAL SÃO JOSÉ DA BARRA, VILA NARCISA, and various neighborhoods like JARDIM DRACENA, JARDIM CAMPOS SALLES, and RECENTO DA BARRA.

Screenshot of the RestauranteEtec website's contact page after a message has been sent. A confirmation modal is displayed.

**RestauranteEtec**

HOME QUEM SOMOS CHEFES MENU RESERVAS BLOG CONTATOS

**Etec Comendador João Rays**  
Rua Ludovico Victório, 2140 - Vila Hab., Barra Bonita - SP, 17340-000  
4,8 ★★★★ 12 comentários  
[Visualizar mapa ampliado](#)

**Contact Us**

Agradecemos seu contato e retornaremos em breve!!

Obrigado!

OK

**Enviar Mensagem**

Mapa: Google Maps showing the location of Etec Comendador João Rays in Barra Bonita, SP. The map includes labels for Rua Ludovico Victório, Coplacana, COLINA DA BARRA, PORTAL SÃO JOSÉ DA BARRA, VILA NARCISA, and various neighborhoods like JARDIM DRACENA, JARDIM CAMPOS SALLES, and RECENTO DA BARRA.

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



gallo.forcode@gmail.com

para mim ▾

18:38 (há 0 minuto)



Pessoa: José Antonio Gallo Junior  
E-mail: [jose.junior818@etec.sp.gov.br](mailto:jose.junior818@etec.sp.gov.br)

Assunto: Excelente Aula

Mensagem: Achei o projeto RestauranteEtec MARAVILHOSOOOOOOOO.

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:

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4
5  namespace RestauranteEtec.Services
6  {
7      5 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         2 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:

```

13  0 references
14  public class ContatosController : Controller
15  {
16      private readonly ContatoDAL contatoDAL = new ContatoDAL();
17      private readonly int pageSize = 3;
18
19      // GET: ContatosController
20      2 references
21      public ActionResult Index()
22      {
23          var contatos = PaginatedList<Contato>.Create(contatoDAL.GetAll().AsQueryable(), 1, pageSize);
24          return View(contatos);
25      }
26
27 }

```

A linha 16, define que cada página de nossa tabela deve mostrar 3 linhas, esse número posteriormente deve ser aumentado, estamos usando 3 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 apenas 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
public PartialViewResult GetGrid(string sortOrder, string searchString, int? pageNumber)
{
    ViewData["NameSortParm"] = String.IsNullOrEmpty(sortOrder) ? "name_desc" : "";
    ViewData["DateSortParm"] = sortOrder == "Date" ? "date_desc" : "Date";
    ViewData["CurrentFilter"] = searchString;

    var contatos = contatoDAL.GetAll().AsQueryable();

    if (!String.IsNullOrEmpty(searchString))
    {
        contatos = contatos.Where(s => s.NomePessoa.Contains(searchString)
                            || s.Assunto.Contains(searchString));
    }

    switch (sortOrder)
    {
        case "name_desc":
            contatos = contatos.OrderByDescending(s => s.NomePessoa);
            break;
        case "Date":
            contatos = contatos.OrderBy(s => s.DataCadastro);
            break;
        case "date_desc":
            contatos = contatos.OrderByDescending(s => s.DataCadastro);
            break;
        default:
            contatos = contatos.OrderBy(s => s.NomePessoa);
            break;
    }
    var retorno = PaginatedList<Contato>.Create(contatos, pageNumber ?? 1, pageSize);
    return PartialView("_Grid", retorno);
}
```

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.

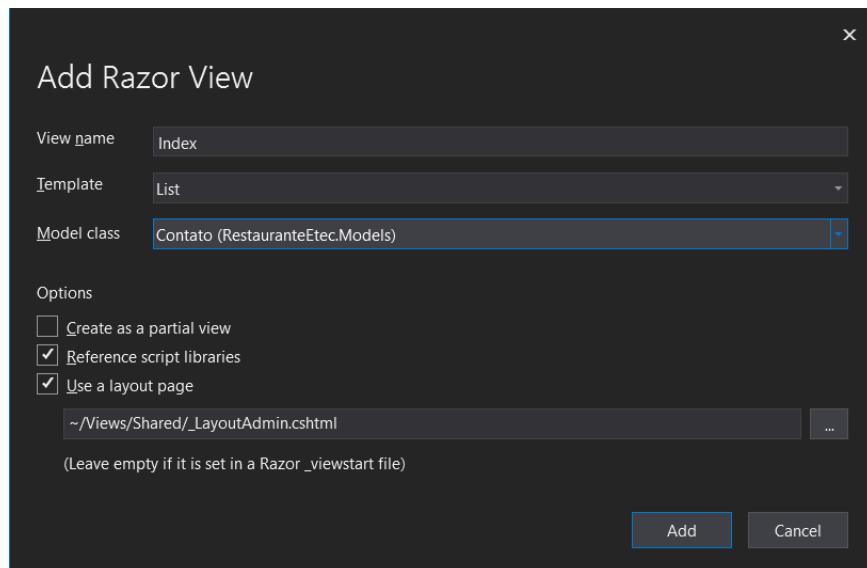
Após preencher os **ViewDatas**, com os valores da requisição, cria uma lista de contatos, usando o **GetAll()**.

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, mais uma **PartialView**, de nome “**\_Grid**”, que é o arquivo que vamos logo após o **Index**.

Clique com o botão direito do mouse sobre a **Action Index** e escolha a opção **Adicionar View (Add View)**.



Vamos alterar um pouquinho o código criado automaticamente.

```
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 }
```

Agora adicione uma **View Vazia**, na pasta **Contatos** (botão direito do mouse na pasta, **Adicionar, Exibição**) com o nome **\_Grid**, e insira o código abaixo:

```
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-asc"></i>}
12                     @if (ViewBag.DateSortParm == "date_desc")
13                         {<i class="zmdi zmdi-sort-amount-desc"></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 == null)
22                         {<i class="zmdi zmdi-sort-amount-asc"></i>}
23                     @if (ViewBag.NameSortParm == "name_desc")
24                         {<i class="zmdi zmdi-sort-amount-desc"></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.DataCadastro)
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, só 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. As linhas 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.

O resultado é o seguinte:

DATA DO CONTATO	NOME DO CONTATO	ASSUNTO	AÇÕES
15/06/2021 22:22:00	Gallo	Testando o envio de mensagem pelo Restaurante	
15/06/2021 22:28:00	Gallo	Testando o envio de mensagem pelo Restaurante	
15/06/2021 22:42:00	Gallo	Testando	