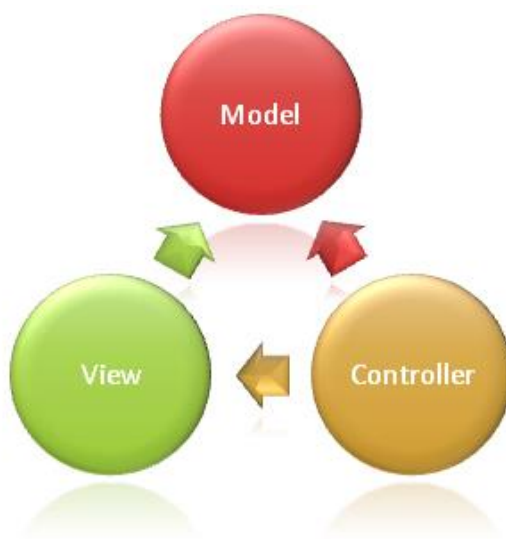


O que é o padrão MVC?

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, mova 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.

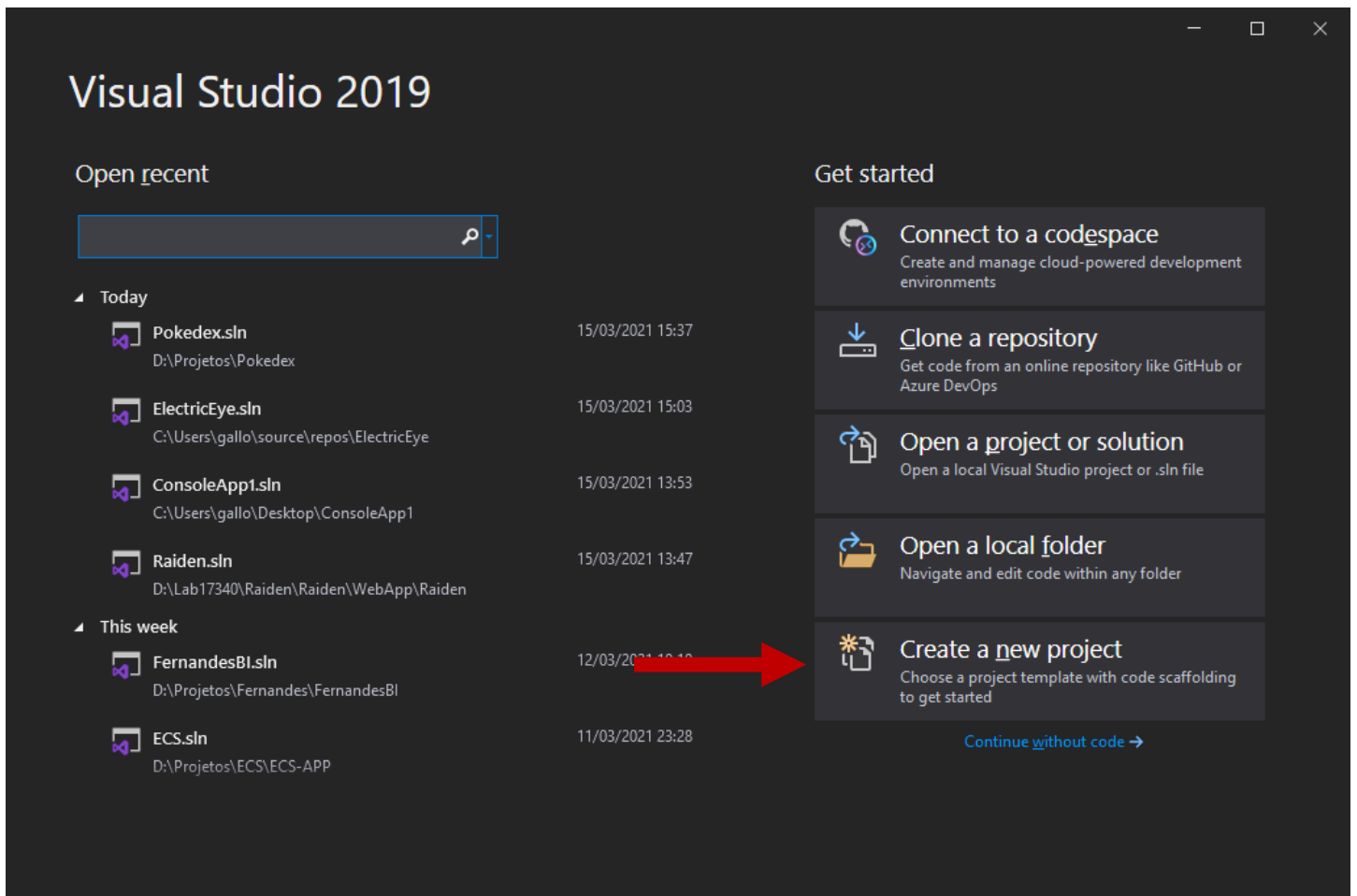
ASP.NET Core MVC fornece uma maneira com base em padrões para criar sites dinâmicos que habilitam uma separação limpa de preocupações. Ele lhe dá controle total sobre a marcação, dá suporte ao desenvolvimento amigável a TDD e usa os padrões da web mais recentes.

RESTAURANTEETEC

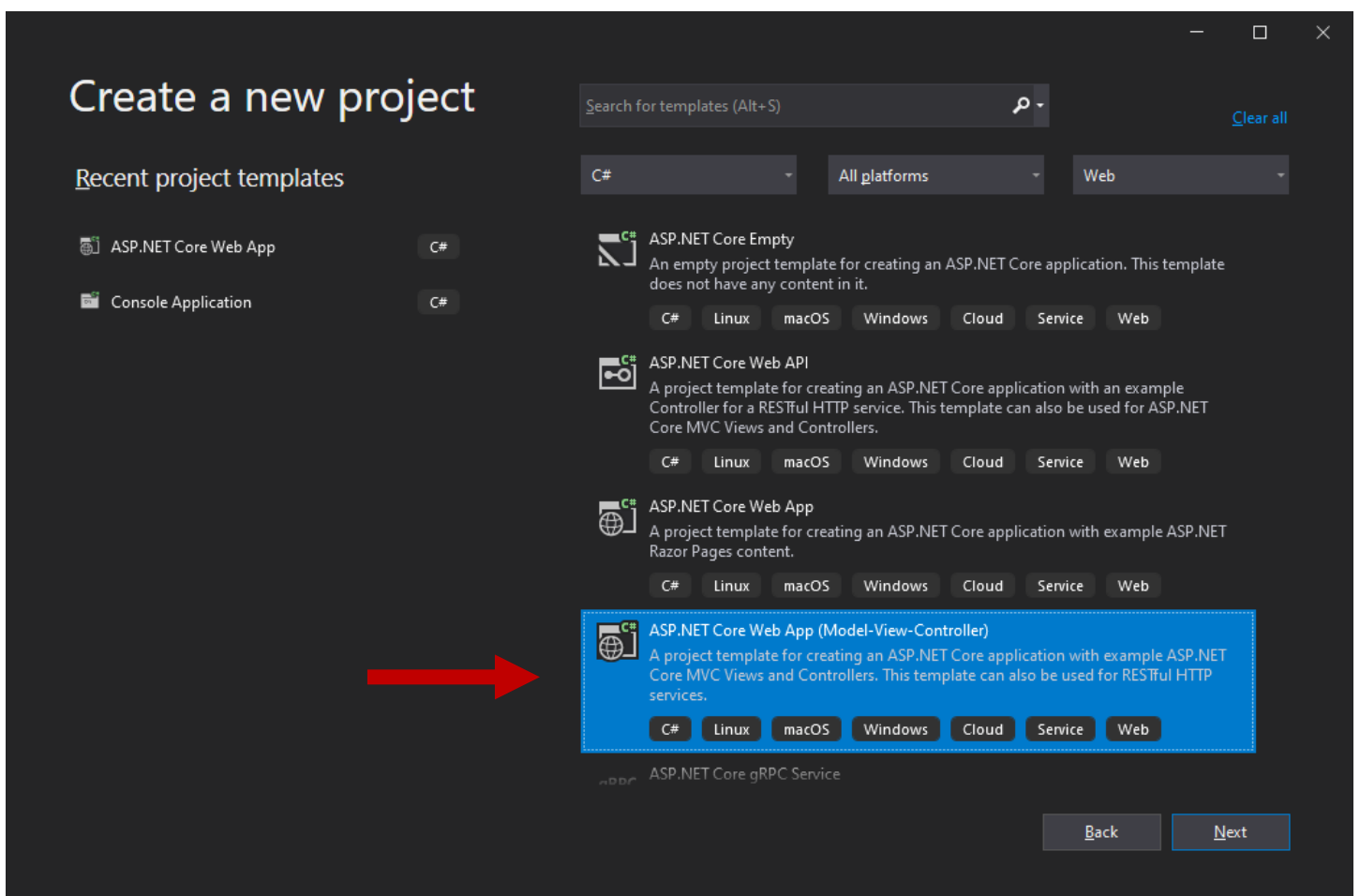
Iniciaremos o desenvolvimento do projeto RestauranteEtec, pela criação do projeto. Essa criação será apresentada de duas maneiras, usando o Visual Studio Community 2019 e o Visual Studio Code.

Criando o projeto no Visual Studio Community 2019:

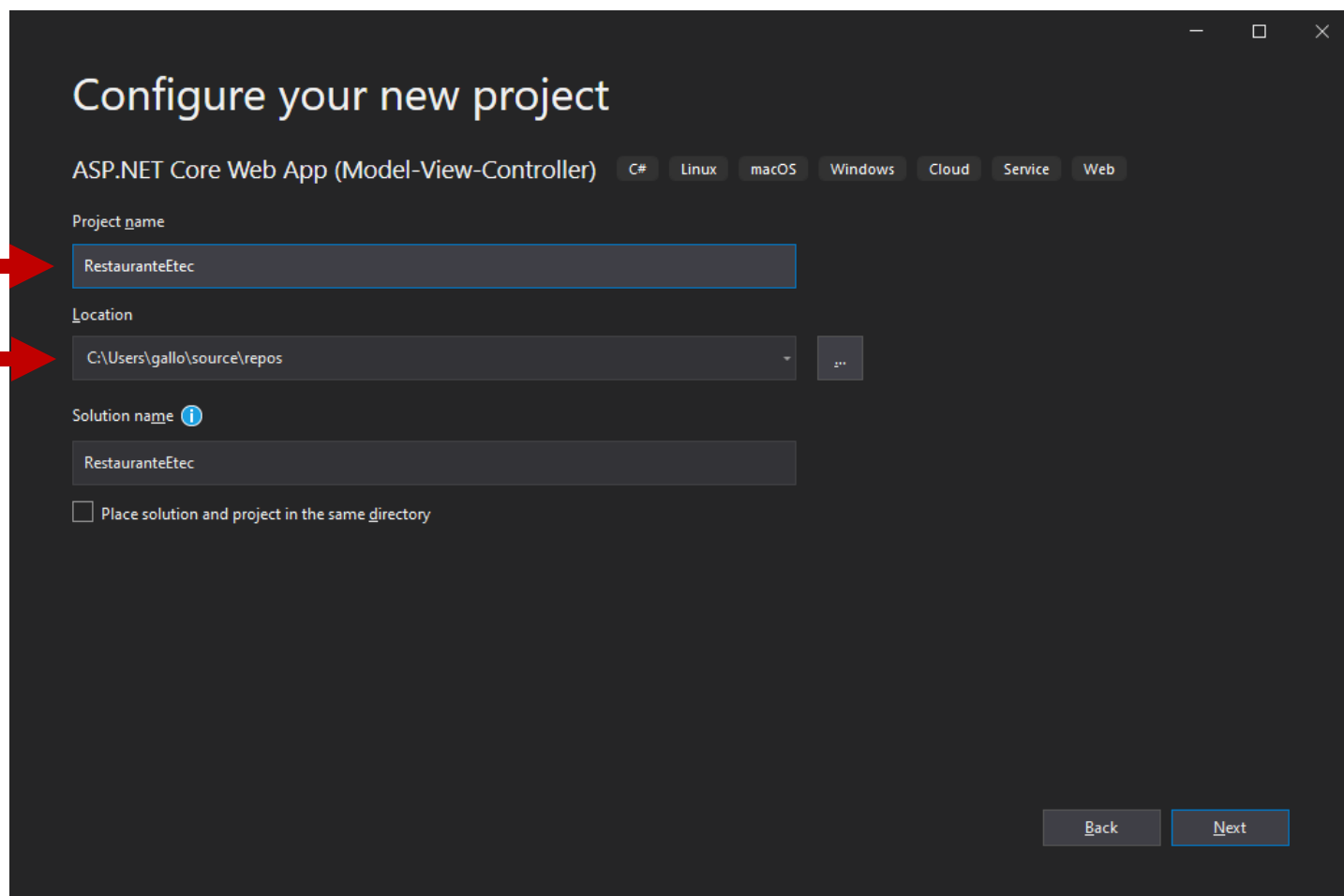
Clique na opção “**Create a new project**” ou “**Criar um novo projeto**”, de acordo com sua instalação.



Selecione 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**”.



Configure your new project

ASP.NET Core Web App (Model-View-Controller) C# Linux macOS Windows Cloud Service Web

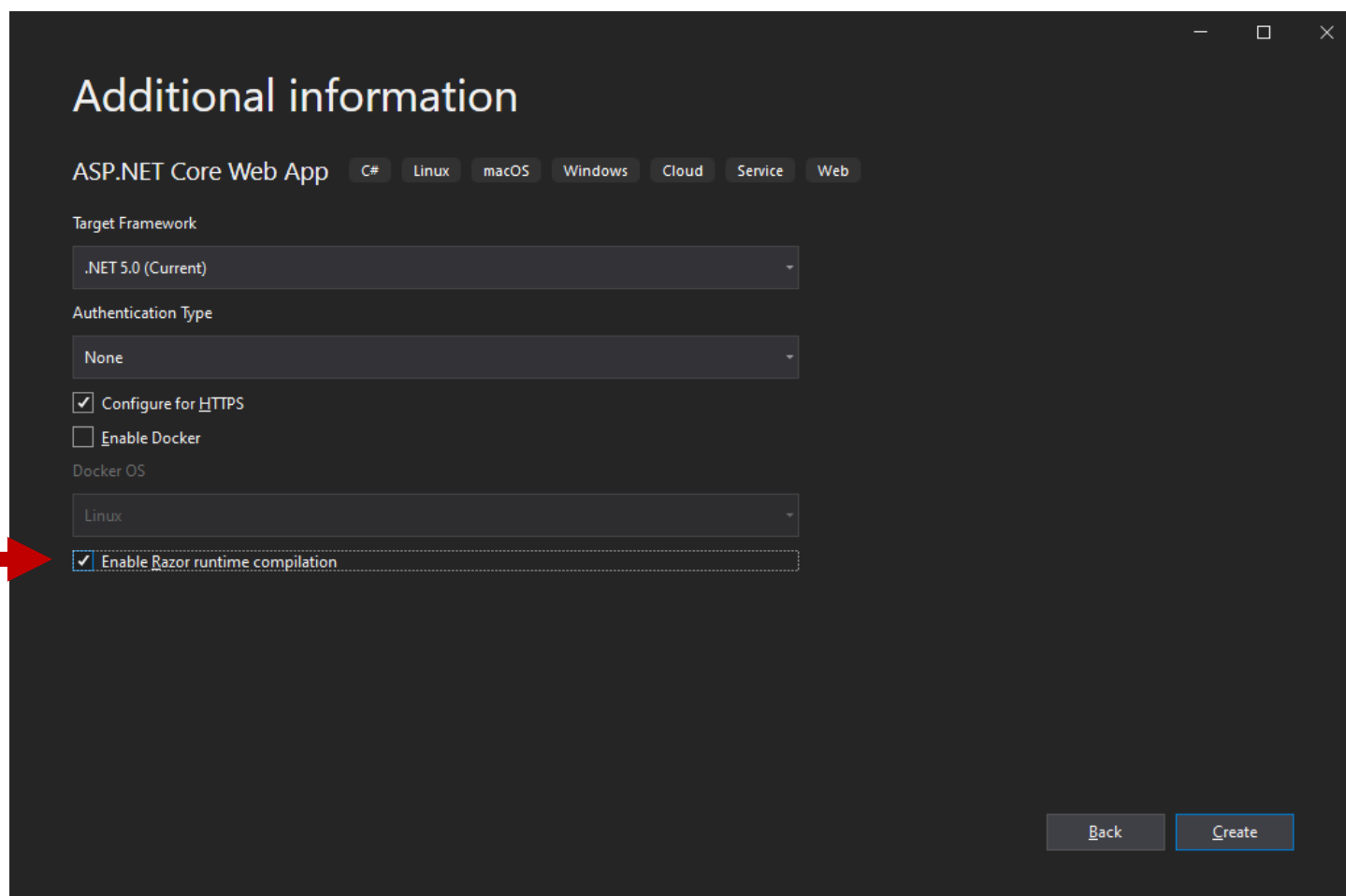
Project name
RestauranteEtec

Location
C:\Users\gallo\source\repos

Solution name ⓘ
RestauranteEtec

☐ Place solution and project in the same directory

Back Next



Additional information

ASP.NET Core Web App C# Linux macOS Windows Cloud Service Web

Target Framework
.NET 5.0 (Current)

Authentication Type
None

☒ Configure for HTTPS

☐ Enable Docker

Docker OS
Linux

☒ Enable Razor runtime compilation

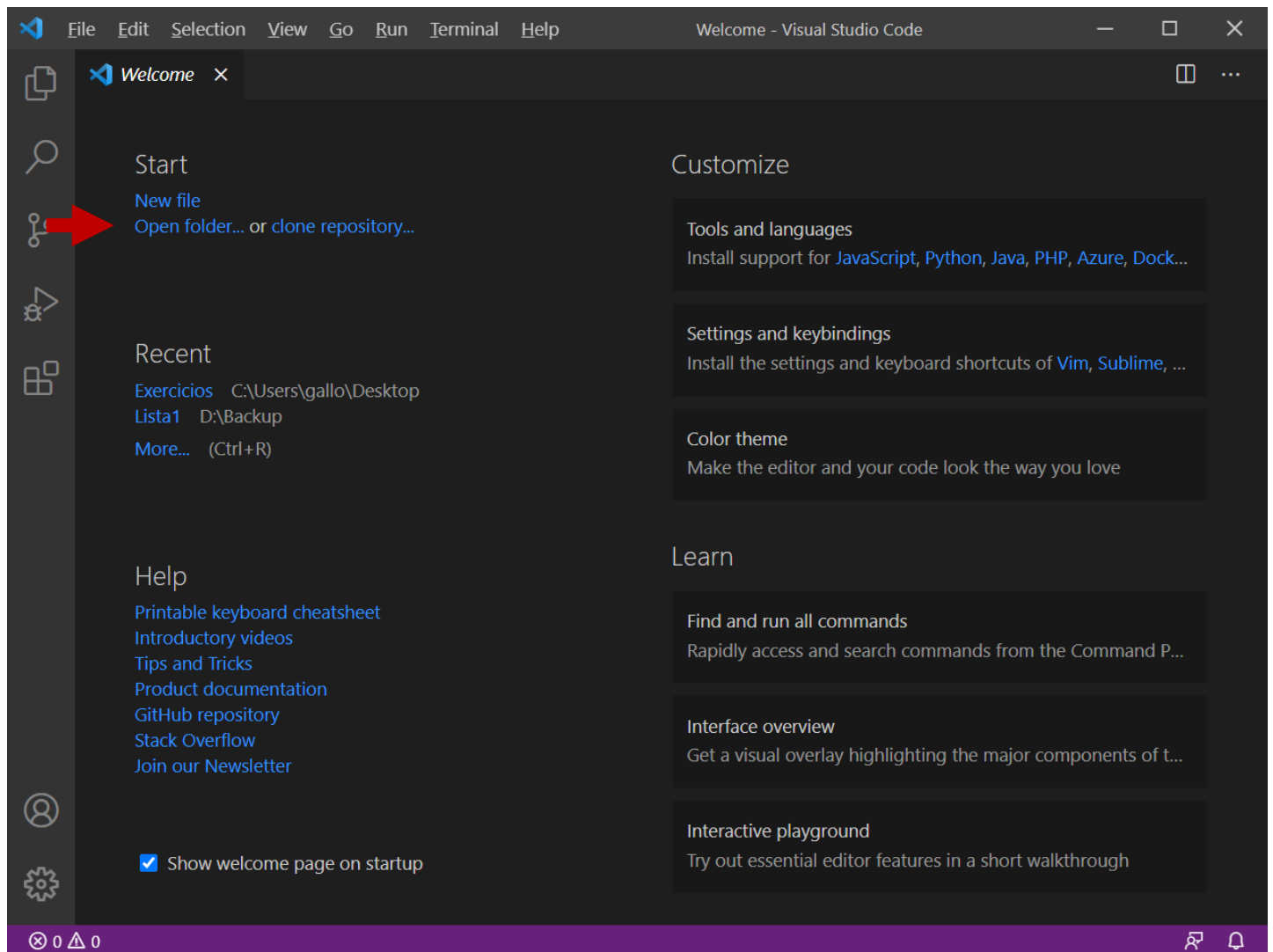
Back Create

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:

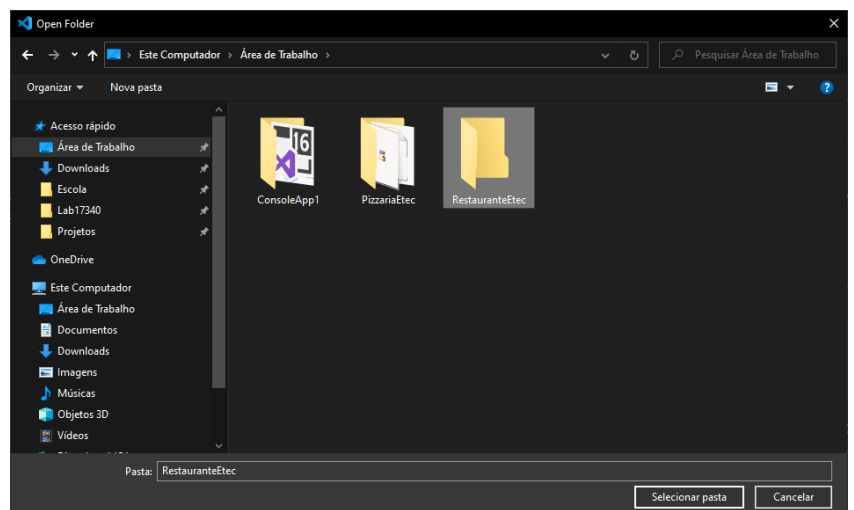
Clique na opção “**Open Folder**”:

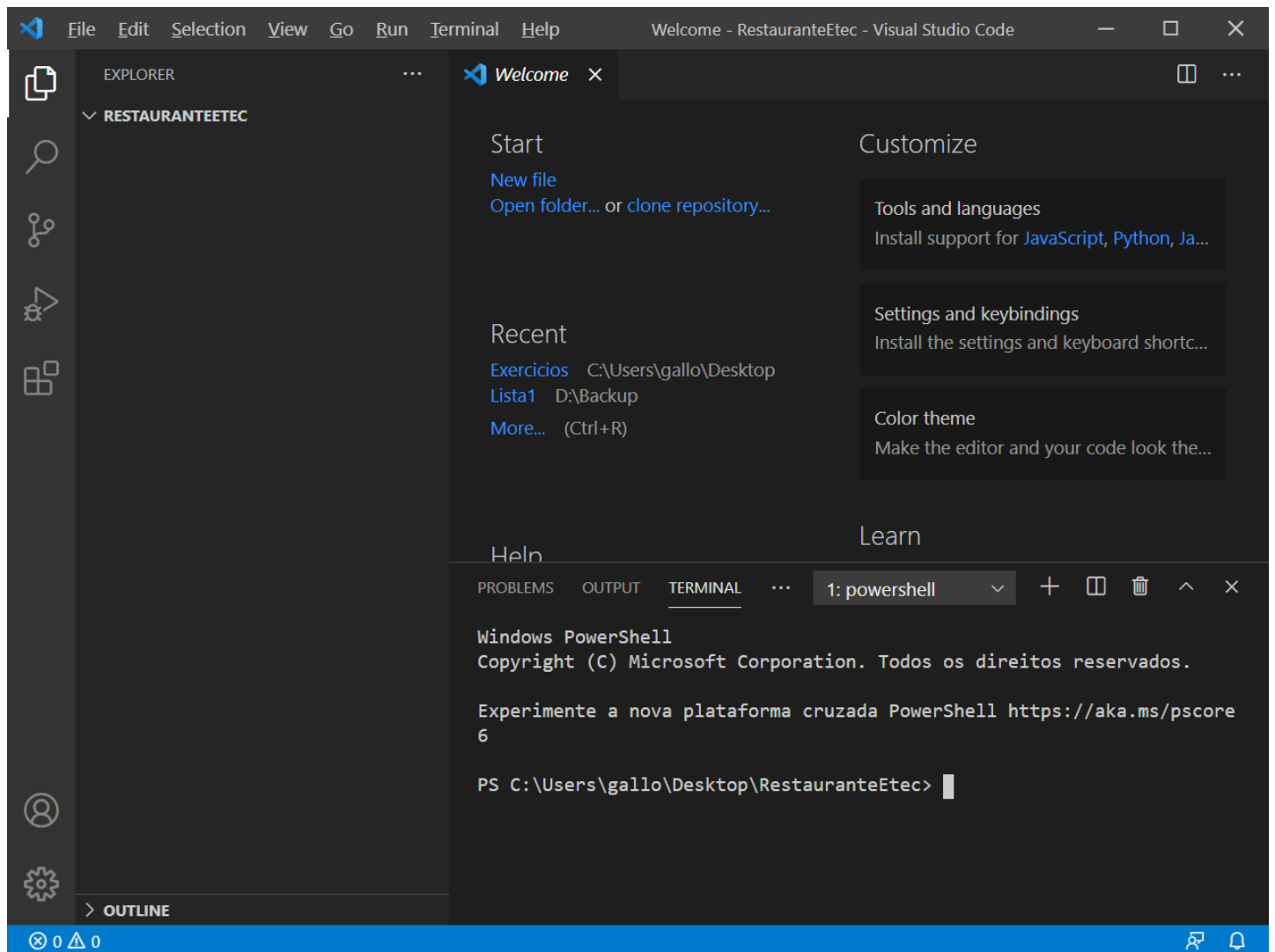


Na janela navega em seu computador, até o local onde deseja salvar seu projeto, e crie uma pasta com o nome **RestauranteEtec**.

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

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





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

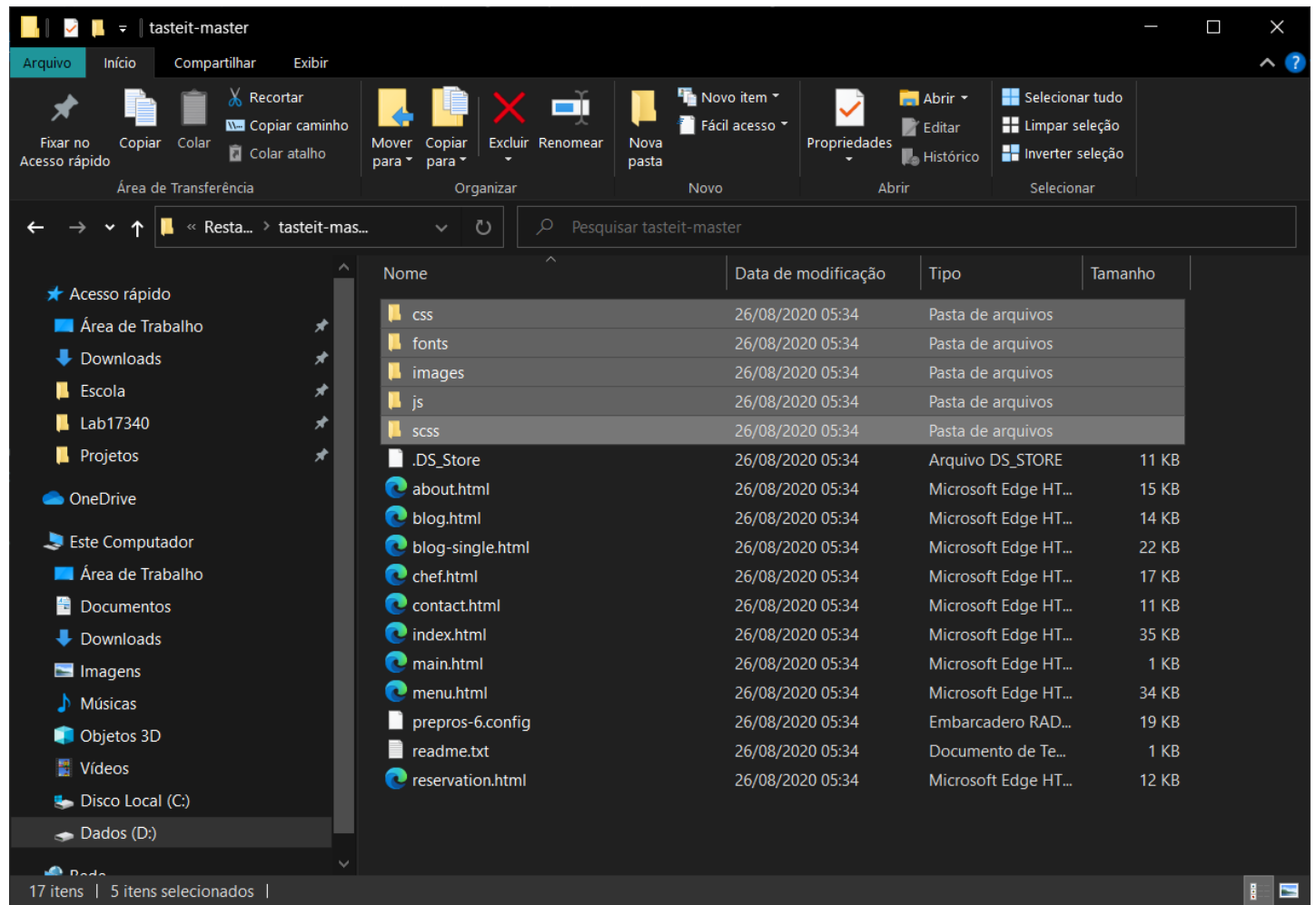
dotnet new mvc -lang "C#" -f net5.0

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

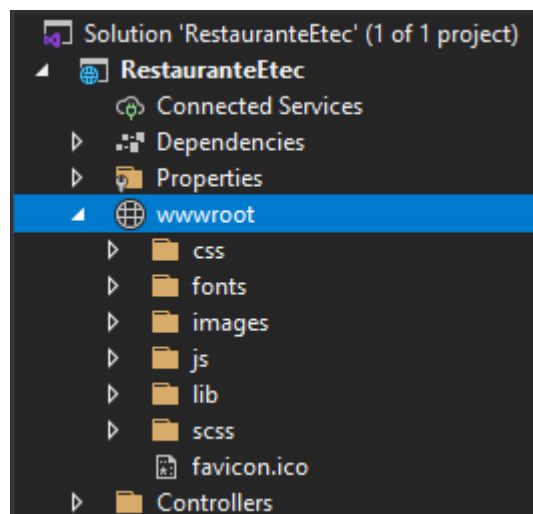
E com isso o projeto está pronto para começar o desenvolvimento.

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 sobrescrever.

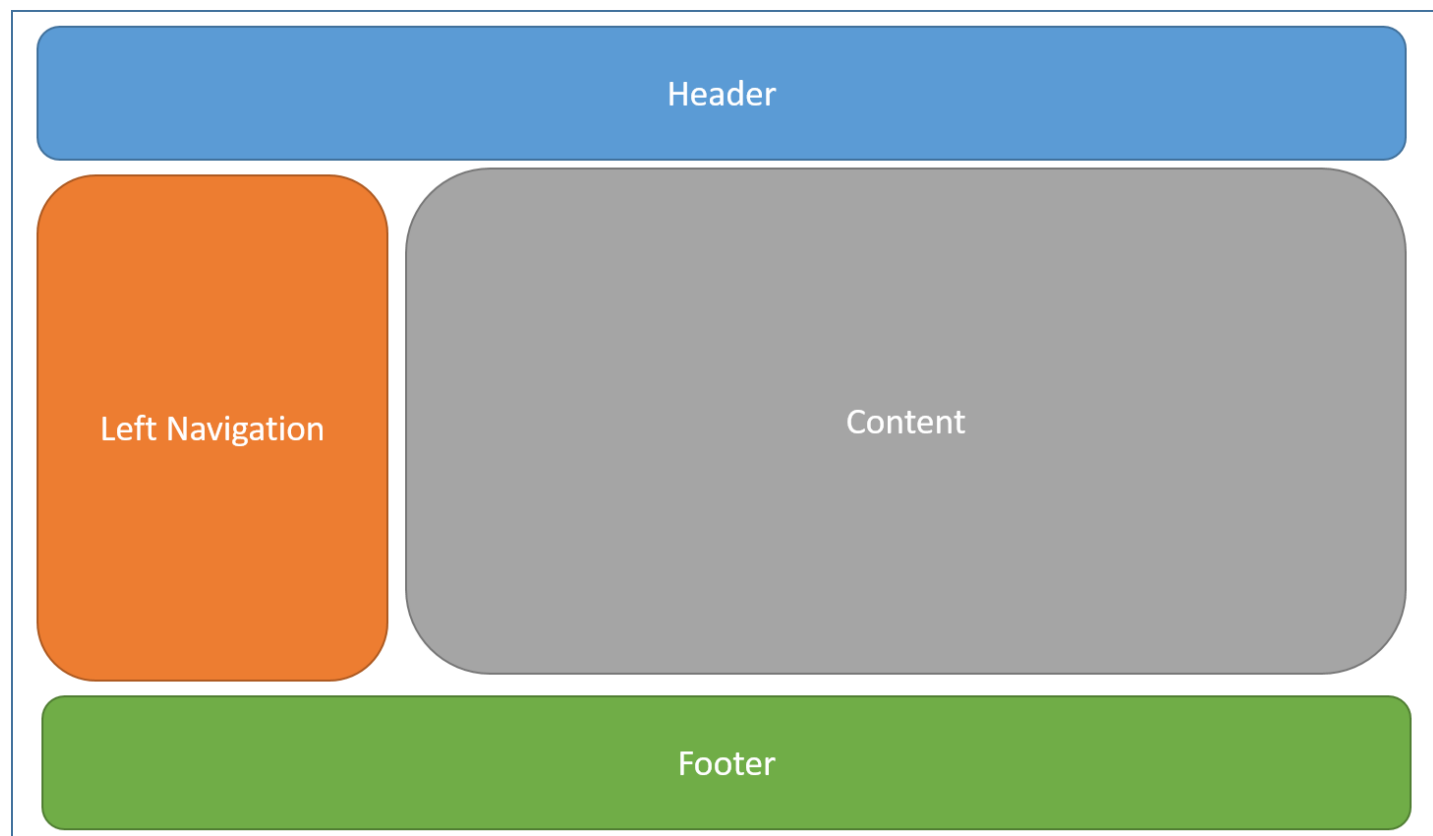


Com isso os arquivos estáticos necessários a correta exibição das páginas estarão disponíveis em seu projeto.

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>
        <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>
<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>
</div>
</section>

<section class="ftco-section ftco-intro" style="background-image:
url(@Url.Content("~/images/bg_3.jpg"));">
  <div class="overlay">
  </div>
  <div class="container">
    <div class="row">
      <div class="col-md-12 text-center">
        <span>Agendamentos para</span>
        <h2>Jantares Particulares & Happy Hours</h2>
      </div>
    </div>
  </div>
</section>

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

```

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

<div class="col-md-6 col-lg-4">
    <div class="menu-wrap">
        <div class="heading-menu text-center ftco-animate">
            <h3>Almoço</h3>
        </div>
        <div class="menus d-flex ftco-animate">
            <div class="menu-img" style="background-image:
url(@Url.Content("~/images/lunch-1.jpg"));"></div>
            <div class="text">
                <div class="d-flex">
                    <div class="one-half">
                        <h3>Beef Roast Source</h3>
                    </div>
                    <div class="one-forth">
                        <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" style="background-image:
url(@Url.Content("~/images/lunch-2.jpg"));"></div>
            <div class="text">
                <div class="d-flex">
                    <div class="one-half">
                        <h3>Beef Roast Source</h3>
                    </div>
                    <div class="one-forth">
                        <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 class="menus border-bottom-0 d-flex ftco-animate">
        <div class="menu-img" style="background-image:
url(@Url.Content("~/images/lunch-3.jpg"));"></div>

```

```

        <div class="text">
            <div class="d-flex">
                <div class="one-half">
                    <h3>Beef Roast Source</h3>
                </div>
                <div class="one-forth">
                    <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-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-forth">
                        <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-forth">
                        <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-forth">
                        <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-forth">
                        <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-forth">
                        <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-forth">
                        <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-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>

```

```

1.jpg);"></div>
    <div class="text">
        <div class="d-flex">
            <div class="one-half">
                <h3>Beef Roast Source</h3>
            </div>
            <div class="one-forth">
                <span class="price">$29</span>
            </div>
        </div>
        <p><span>Meat</span>,          <span>Potatoes</span>,          <span>Rice</span>,
<span>Tomatoe</span></p>
    </div>
</div>
2.jpg);"></div>
    <div class="text">
        <div class="d-flex">
            <div class="one-half">
                <h3>Beef Roast Source</h3>
            </div>
            <div class="one-forth">
                <span class="price">$29</span>
            </div>
        </div>
        <p><span>Meat</span>,          <span>Potatoes</span>,          <span>Rice</span>,
<span>Tomatoe</span></p>
    </div>
</div>
3.jpg);"></div>
    <div class="text">
        <div class="d-flex">
            <div class="one-half">
                <h3>Beef Roast Source</h3>
            </div>
            <div class="one-forth">
                <span class="price">$29</span>
            </div>
        </div>
        <p><span>Meat</span>,          <span>Potatoes</span>,          <span>Rice</span>,
<span>Tomatoe</span></p>
    </div>
</div>
    <span class="flat flaticon-wine" style="left: 0;"></span>
    <span class="flat flaticon-wine-1" style="right: 0;"></span>
</div>
</div>
<div class="col-md-6 col-lg-4">
    <div class="menu-wrap">
        <div class="heading-menu text-center ftco-animate">
            <h3>Bebidas & Chá</h3>
        </div>
        <div class="menus d-flex ftco-animate">
            <div class="menu-img img" style="background-image: url(images/drink-
1.jpg);"></div>
            <div class="text">
                <div class="d-flex">
                    <div class="one-half">
                        <h3>Beef Roast Source</h3>
                    </div>
                    <div class="one-forth">
                        <span class="price">$29</span>
                    </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-forth">
                        <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-forth">
                        <span class="price">$29</span>
                    </div>
                </div>
                <p><span>Meat</span>,          <span>Potatoes</span>,          <span>Rice</span>,
<span>Tomatoe</span></p>
            </div>
        </div>
        <span class="flat flaticon-wine" style="left: 0;"></span>
        <span class="flat flaticon-wine-1" style="right: 0;"></span>
    </div>
</div>
</div>
</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>
<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>

```

```
</div>
</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">Chefes</span>
        <h2 class="mb-4">Nossos Mestres</h2>
      </div>
    </div>
    <div class="row">
      <div class="col-md-6 col-lg-3 ftco-animate">
        <div class="staff">
          <div class="img" style="background-image: url(images/chef-4.jpg);"></div>
          <div class="text px-4 pt-2">
            <h3>John Gustavo</h3>
            <span class="position mb-2">CEO, Co Founder</span>
            <div class="faded">
              <p>I am an ambitious workaholic, but apart from that, pretty simple person.</p>
              <ul class="ftco-social d-flex">
                <li class="ftco-animate"><a href="#"><span class="icon-twitter"></span></a></li>
                <li class="ftco-animate"><a href="#"><span class="icon-facebook"></span></a></li>
                <li class="ftco-animate"><a href="#"><span class="icon-google-plus"></span></a></li>
                <li class="ftco-animate"><a href="#"><span class="icon-instagram"></span></a></li>
              </ul>
            </div>
          </div>
        </div>
      </div>
      <div class="col-md-6 col-lg-3 ftco-animate">
        <div class="staff">
          <div class="img" style="background-image: url(images/chef-2.jpg);"></div>
          <div class="text px-4 pt-2">
            <h3>Michelle Fraulen</h3>
            <span class="position mb-2">Head Chef</span>
            <div class="faded">
              <p>I am an ambitious workaholic, but apart from that, pretty simple person.</p>
              <ul class="ftco-social d-flex">
                <li class="ftco-animate"><a href="#"><span class="icon-twitter"></span></a></li>
                <li class="ftco-animate"><a href="#"><span class="icon-facebook"></span></a></li>
                <li class="ftco-animate"><a href="#"><span class="icon-google-plus"></span></a></li>
                <li class="ftco-animate"><a href="#"><span class="icon-instagram"></span></a></li>
              </ul>
            </div>
          </div>
        </div>
      </div>
      <div class="col-md-6 col-lg-3 ftco-animate">
        <div class="staff">
          <div class="img" style="background-image: url(images/chef-3.jpg);"></div>
          <div class="text px-4 pt-2">
            <h3>Alfred Smith</h3>
```

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

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

Blog

<h2 class="mb-4">Recntes</h2>

</div>

</div>

<div class="row">

<div class="col-md-4 ftco-animate">

<div class="blog-entry">

<div class="text px-4 pt-3 pb-4">

<div class="meta">

<div>August 3, 2020</div>

<div>Admin</div>

</div>

<h3 class="heading">Even the all-powerful Pointing has no control about the blind texts</h3>

<p class="clearfix">

Read more

 3

</p>

</div>

</div>

</div>

<div class="col-md-4 ftco-animate">

<div class="blog-entry">

<div class="text px-4 pt-3 pb-4">

<div class="meta">

<div>August 3, 2020</div>

<div>Admin</div>

</div>

<h3 class="heading">Even the all-powerful Pointing has no control about the blind texts</h3>

<p class="clearfix">

Read more

 3

</p>

</div>

</div>

</div>

<div class="col-md-4 ftco-animate">

<div class="blog-entry">

<div class="text px-4 pt-3 pb-4">

<div class="meta">

<div>August 3, 2020</div>

<div>Admin</div>

</div>

<h3 class="heading">Even the all-powerful Pointing has no control about the blind texts</h3>

<p class="clearfix">

Read more

 3

</p>

</div>

</div>

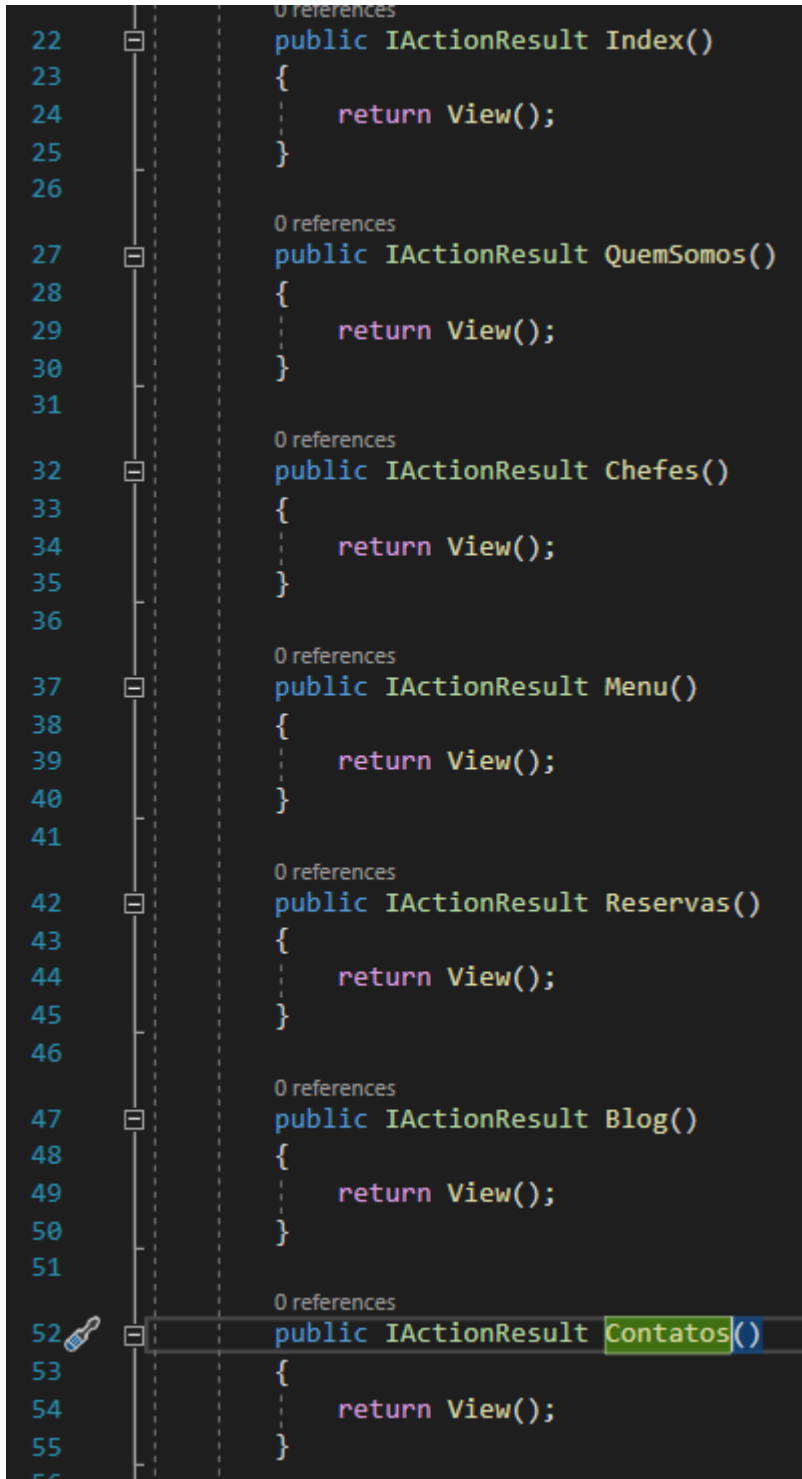
```
        </div>
    </div>
</div>
</section>

<section class="ftco-section ftco-no-pt ftco-no-pb ftco-intro bg-primary">
    <div class="container py-5">
        <div class="row py-2">
            <div class="col-md-12 text-center">
                <h2>Fazemos Deliciosas & 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**.

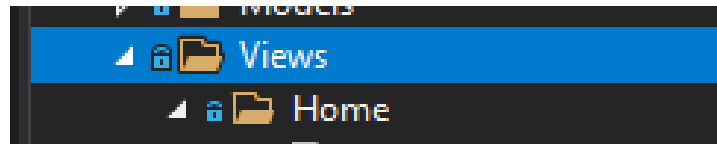
1º - No HomeController, vamos adicionar os códigos abaixo:



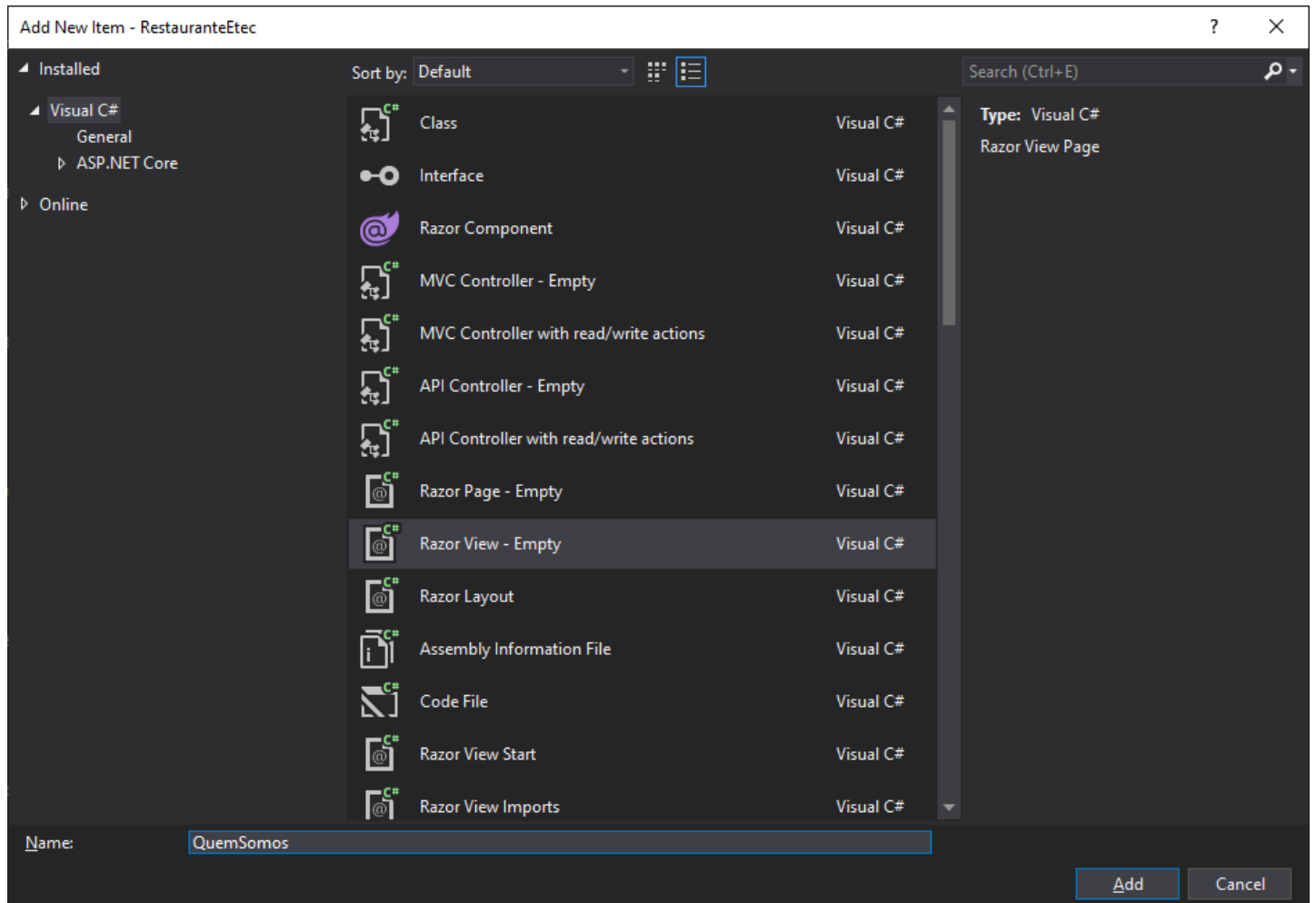
```
22 0 references
23 public IActionResult Index()
24 {
25     return View();
26 }
27
28 0 references
29 public IActionResult QuemSomos()
30 {
31     return View();
32 }
33
34 0 references
35 public IActionResult Chefes()
36 {
37     return View();
38 }
39
40 0 references
41 public IActionResult Menu()
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 Contatos()
54 {
55     return View();
56 }
```

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

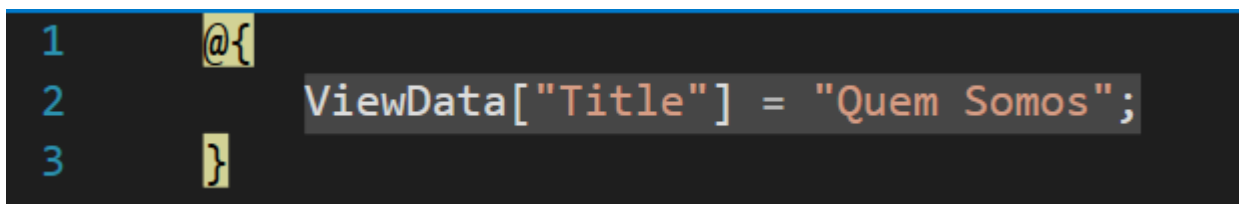
Agora para cada ação (**com exceção da Index**) que já foi desenvolvida, vamos adicionar na pasta **Views\Home** uma **Razor Page** com o mesmo nome, conforme a figura abaixo:



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:



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**.

`style="background-image: url(@Url.Content("~/images/bg_5.jpg"));"`

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
QuemSomos	Quem Somos	about.html	70 – 244
Chefes	Chefes	chef.html	70 – 266
Menu	Menu	menu.html	70 – 810
Reservas	Reservas	reservation.html	70 – 168
Blog	Blog	blog.html	70 – 205
Contatos	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

Categoria

Id	int
Nome	string 30

Produto

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

Relato

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

Cargo

Id	int
Nome	string 30

Funcionario

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

Blog

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

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 Categoria.

```
1  using System;
2  using System.ComponentModel.DataAnnotations;
3  using System.ComponentModel.DataAnnotations.Schema;
4
5  namespace RestauranteEtec.Models
6  {
7      [Table("Produto")]
8      0 references
9      public class Produto
10     {
11         [Key]
12         0 references
13         public int Id { get; set; }
14
15         [Required(ErrorMessage = "Campo obrigatório")]
16         [StringLength(60, ErrorMessage = "O Nome da Categoria deve possuir no máximo 60 caracteres")]
17         0 references
18         public string Nome { get; set; }
19
20         [StringLength(500, ErrorMessage = "A Descrição deve possuir no máximo 500 caracteres")]
21         0 references
22         public string Descricao { get; set; }
23
24         [Required(ErrorMessage = "Campo obrigatório")]
25         0 references
26         public int CategoriaId { get; set; }
27         0 references
28         public Categoria Categoria { get; set; }
29
30         [StringLength(200)]
31         0 references
32         public string Foto { get; set; }
33
34         0 references
35         public bool ExibirHome { get; set; }
36
37         0 references
38         public bool Ativo { get; set; }
39     }
40 }
```

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 especifica. 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 abstratos e lacrados](#).

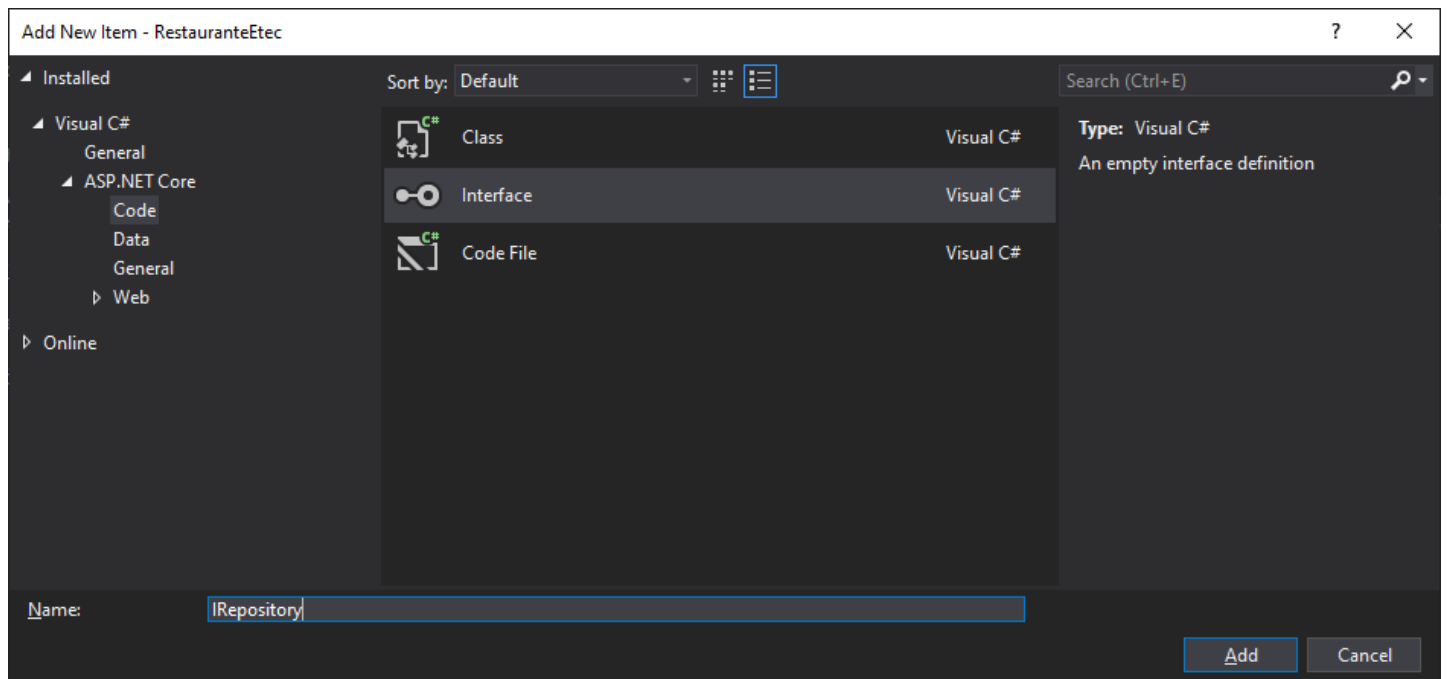
As interfaces podem conter métodos de instância, propriedades, eventos, indexadores ou qualquer combinação desses quatro tipos de membro. As interfaces podem conter construtores, campos, constantes ou operadores estáticos. Para obter links para exemplos, consulte as [seções relacionadas](#). Uma interface 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** 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)**.

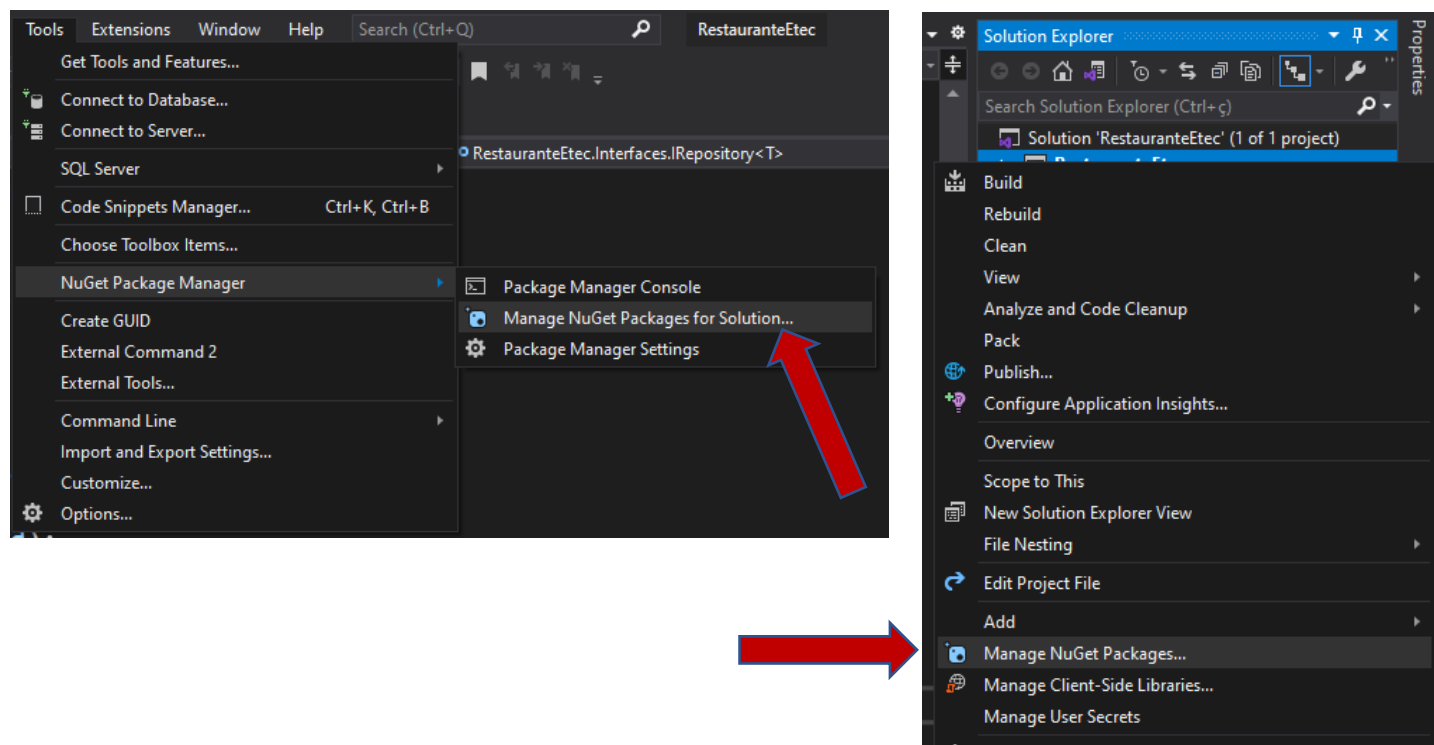
```
1  using System;
2      using System.Collections.Generic;
3      using System.Linq;
4      using System.Threading.Tasks;
5
6  namespace RestauranteEtec.Interfaces
7  {
8      1 reference
8      public interface IRepository<T>
9      {
10         1 reference
10         IEnumerable<T> GetAll();
11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

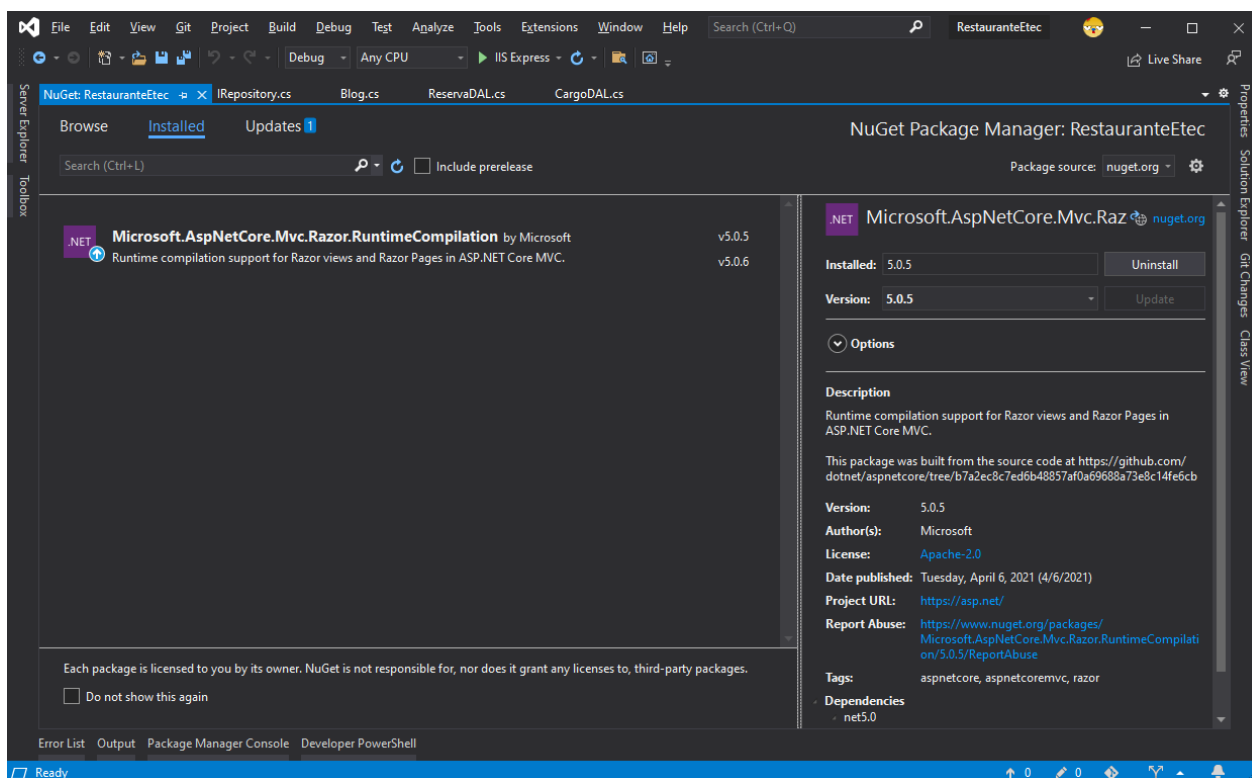
Em nossa aplicação, vamos criar uma pasta para armazenar as classes **DAL**. Clique com o botão direito no projeto e selecione a opção: **Adicionar -> Nova Pasta (Add -> New Folder)** e nomeie a pasta como **DAL**.

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.

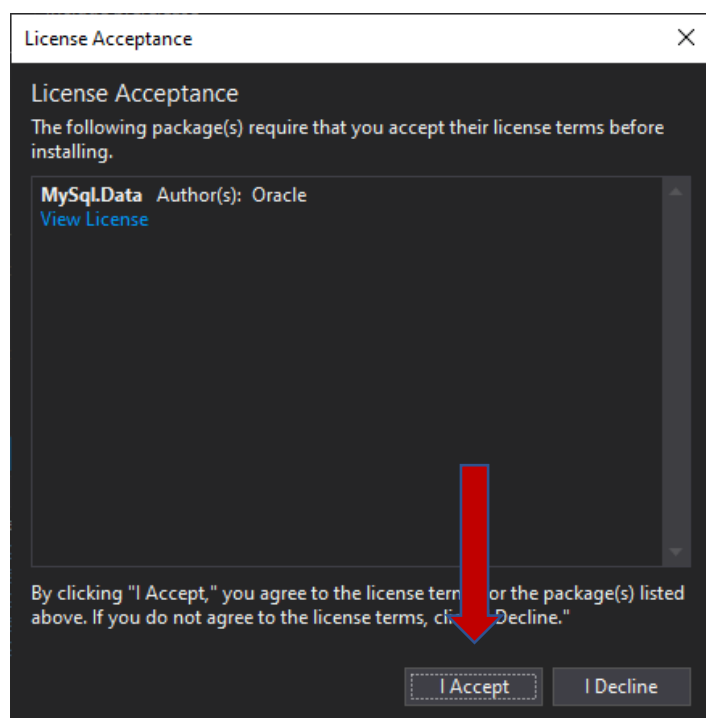
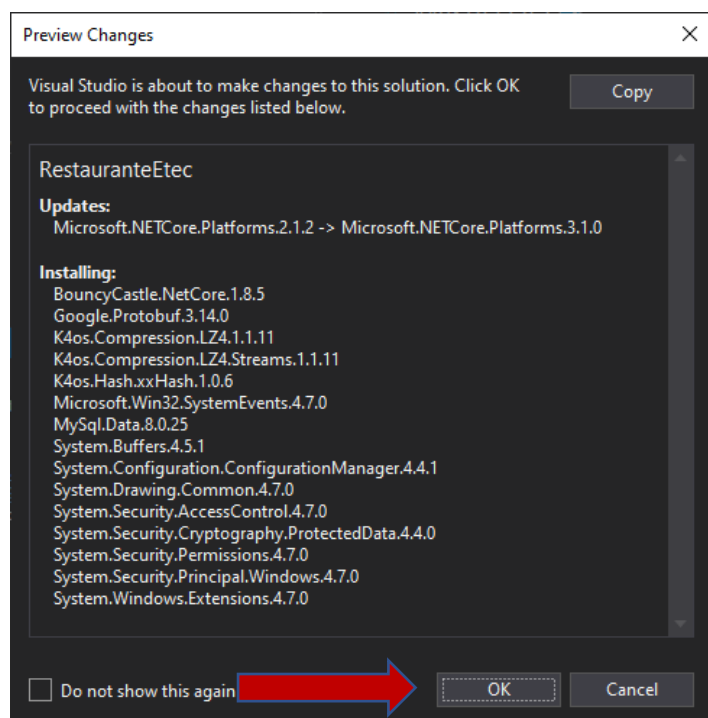
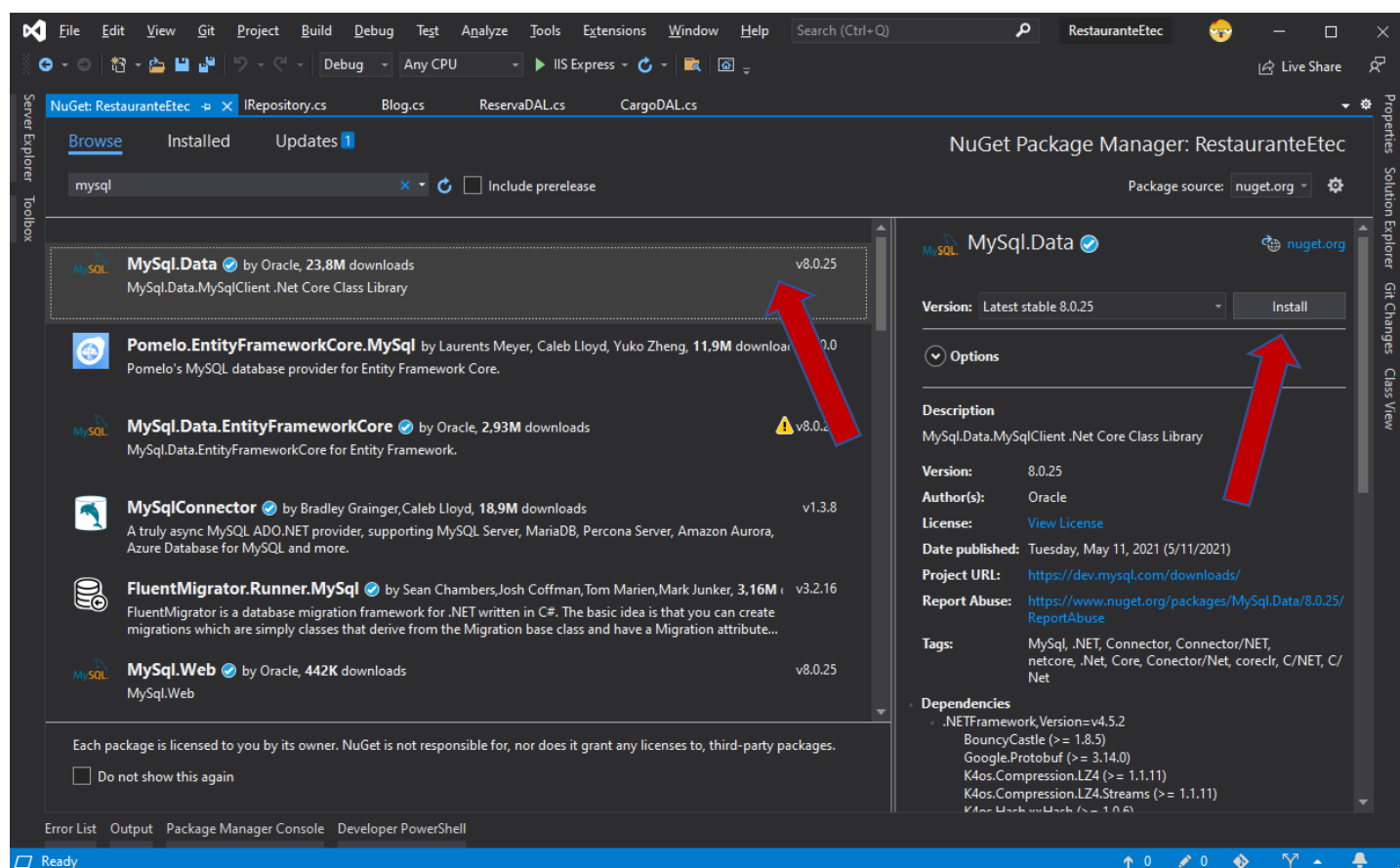
Para incluir pacotes de maneira simples, o Visual Studio Community, possui o **Gerenciador de Pacotes do NuGet (Manage NuGet Packages)**, que pode ser acessado através do menu **Ferramentas (Tools)**, ou clicando com o botão direito no projeto no menu lateral **Explorador de Soluções (Solution Explorer)** 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**:

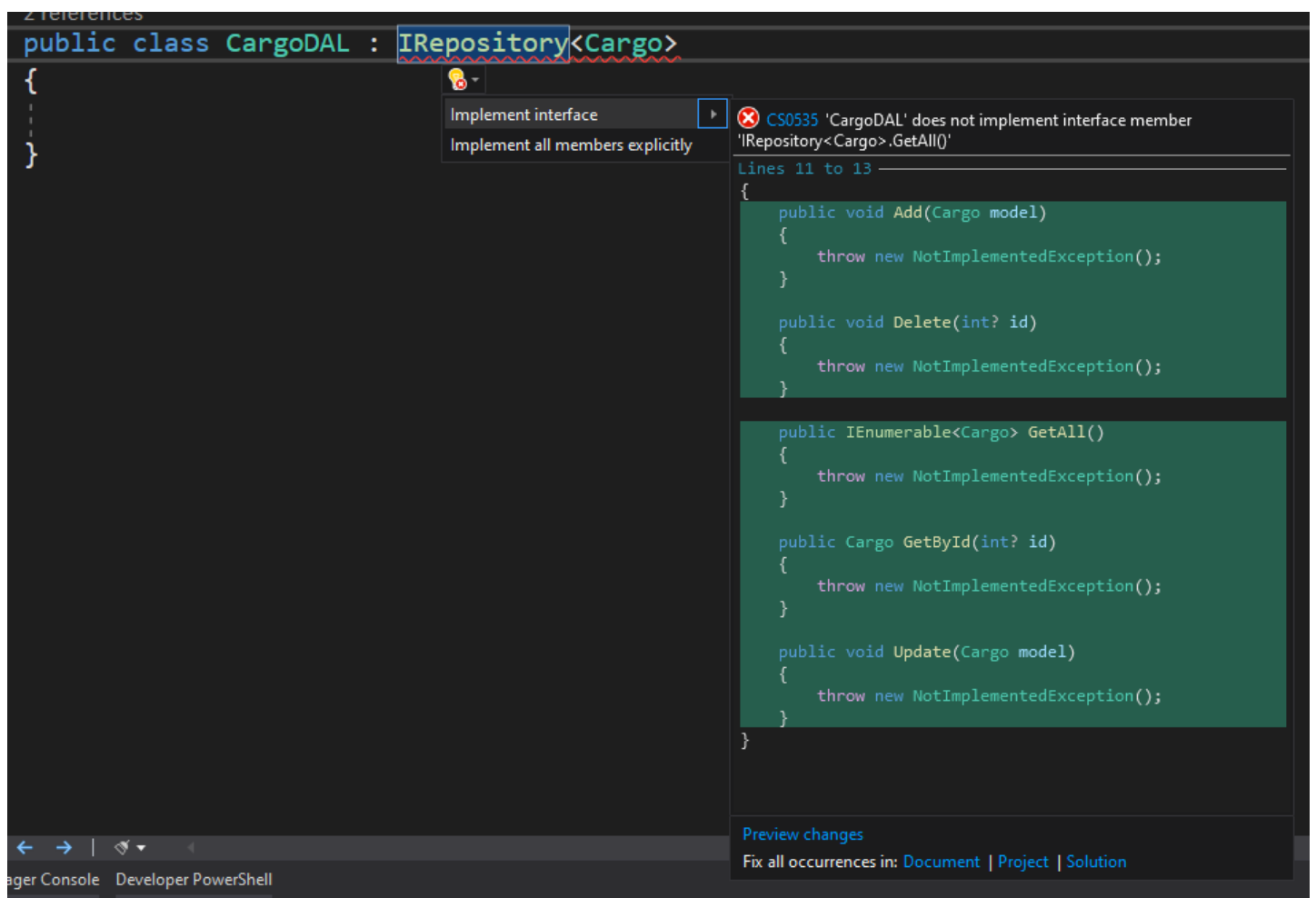
```
1  using System;
2      using System.Collections.Generic;
3  using System.Linq;
4      using System.Threading.Tasks;
5
6  namespace RestauranteEtec.DAL
7  {
8      public class CargoDAL : IRepository<Cargo>
9      {
10     }
11 }
12
13
```

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

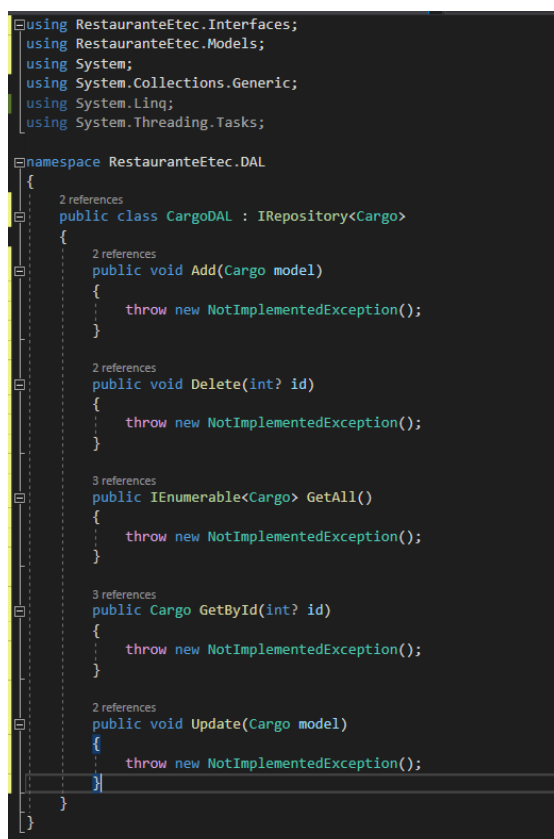
```
1  using RestauranteEtec.Interfaces;
2      using RestauranteEtec.Models;
3  using System;
4      using System.Collections.Generic;
5  using System.Linq;
6      using System.Threading.Tasks;
7
8  namespace RestauranteEtec.DAL
9  {
10     public class CargoDAL : IRepository<Cargo>
11     {
12    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



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



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

```
2 references
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:

int	<code>Convert.ToInt32(leitor["nomeCampo"])</code>
double	<code>Convert.ToDouble(leitor["nomeCampo"])</code>
decimal	<code>Convert.ToDecimal(leitor["nomeCampo"])</code>
byte	<code>Convert.ToByte(leitor["nomeCampo"])</code>
string	<code>Leitor["nomeCampo"].ToString()</code>
bool	Get: <code>Convert.ToInt32(leitor["nomeCampo"]) == 1</code> Add: <code>true</code> ou <code>false</code> ou valor da propriedade Update: <code>model.Ativo ? 1 : 0</code>

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     2 references
13     public class FuncionarioDAL : IRepository<Funcionario>
14     {
15         string connectionString = @"Server=localhost;port=3306;database=RestauranteEtec;uid=root;pwd='';";
16
17         2 references
18         public void Add(Funcionario model)
19         {
20             throw new NotImplementedException();
21         }
22
23         2 references
24         public void Delete(int? id)
25         {
26             throw new NotImplementedException();
27         }
28
29         4 references
30         public IEnumerable<Funcionario> GetAll()
31         {
32             throw new NotImplementedException();
33         }
34
35         2 references
36         public Funcionario GetById(int? id)
37         {
38             throw new NotImplementedException();
39         }
40
41         2 references
42         public void Update(Funcionario model)
43         {
44             throw new NotImplementedException();
45         }
46     }
47 }
```

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:

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

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

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

```
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();
}
```

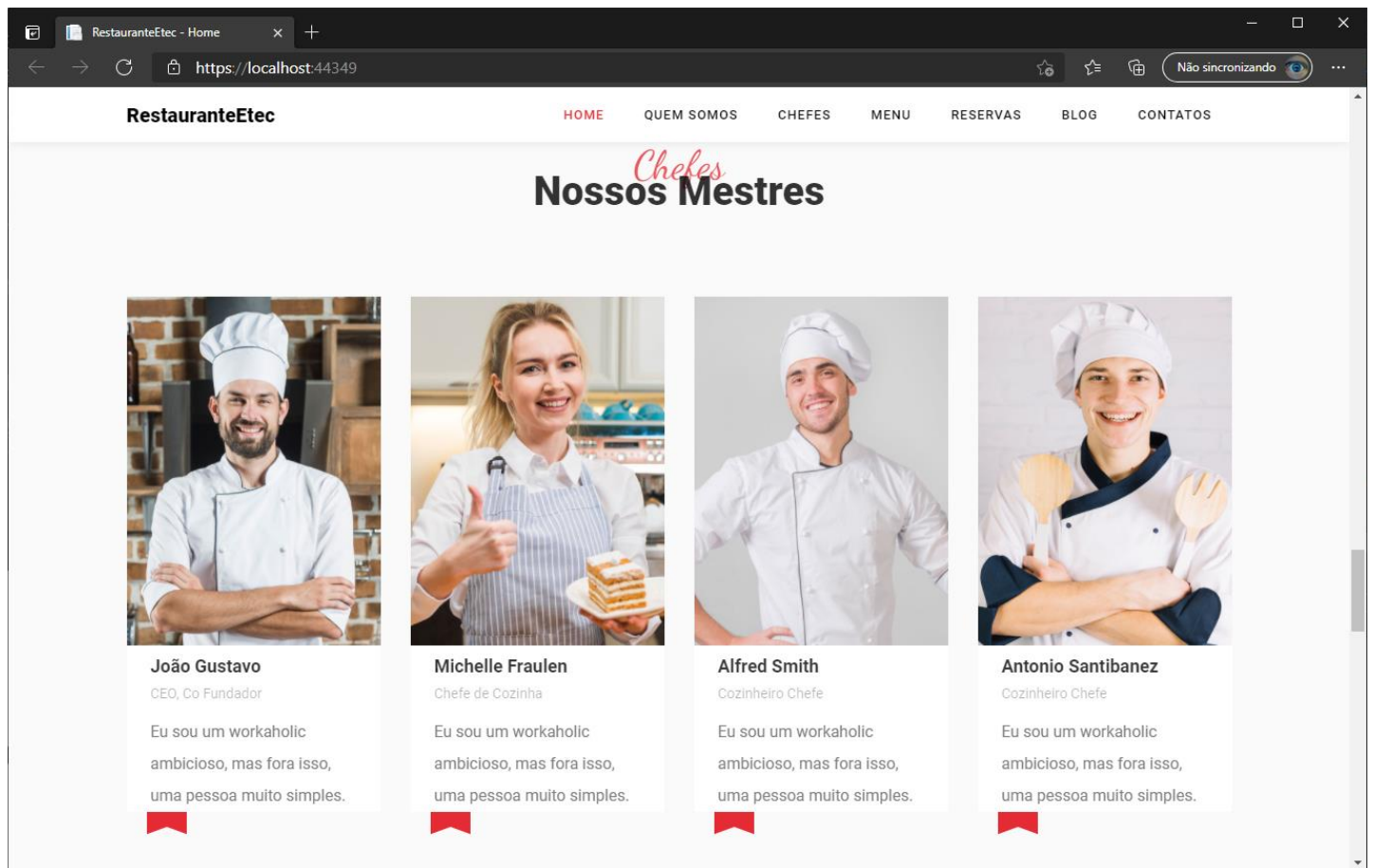
O objeto **funcionarios**, nos permite executar os acessos e alterações de dados programados nos métodos (**Add**, **Delete**, **GetAll**, **GetById** e **Update**).

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 Chefes alterada (procure no seu Index por **"Our Chefs"** e altere todo o código da seção:

```
551 <section class="ftco-section bg-light">
552     <div class="container">
553         <div class="row justify-content-center mb-5 pb-2">
554             <div class="col-md-7 text-center heading-section ftco-animate">
555                 <span class="subheading">Chefes</span>
556                 <h2 class="mb-4">Nossos Mestres</h2>
557             </div>
558         </div>
559         <div class="row">
560             @foreach (var chefe in ViewBag.Chefes)
561             {
562                 <div class="col-md-6 col-lg-3 ftco-animate">
563                     <div class="staff">
564                         <div class="img" style="background-image: url('@Url.Content(chefe.Foto)');"></div>
565                         <div class="text px-4 pt-2">
566                             <h3>@chefe.Nome</h3>
567                             <span class="position mb-2">@chefe.Cargo.Nome</span>
568                             <div class="faded">
569                                 <p>@chefe.Descricao</p>
570                             </div>
571                         </div>
572                     </div>
573                 </div>
574             }
575         </div>
576     </div>
577 </section>
578
```

Execute e verifique o resultado



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.