Arquitetura de Software

MVC

Model – View – Controller

José Motta Lopes
josemotta@bampli.com

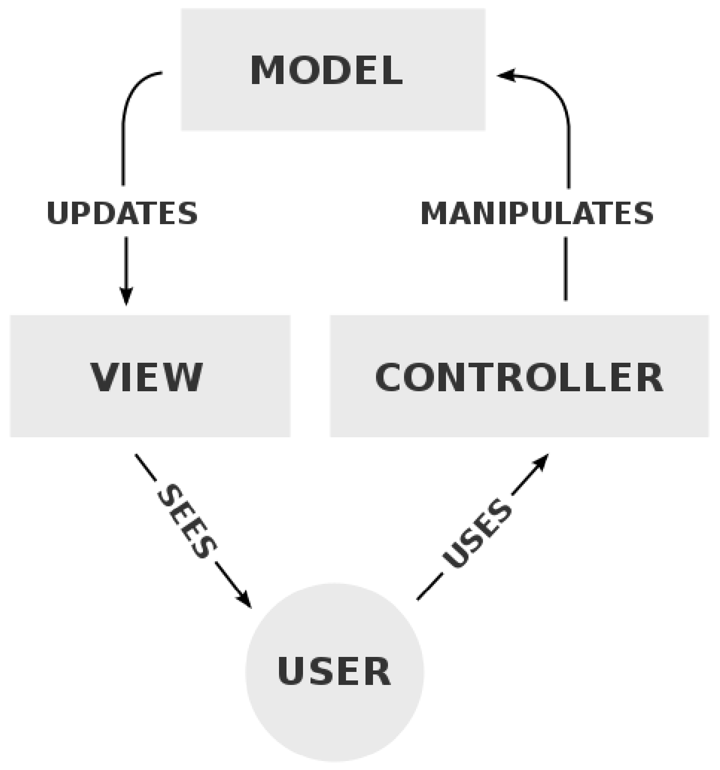
Agenda

- MVC
  - Utilização
  - Framework Java
  - Model, View & Controller
- ASP.NET MVC
  - Características
  - Roteamento
  - Validação do Modelo
  - Dependency Injection
  - Filtros
  - Unit Tests
- Exemplo
  - Cria DB, Aplicação & Projetos
  - Instala SQL e Entity Framework
  - Projetos Client, Server e Shared Model
  - Criação do Web API Controller
  - Adiciona views
  - Adiciona ordenação por coluna

MVC

ARCHITECTURAL PATTERN

- **Padrão de arquitetura usado na web em interfaces do usuário.**
- **Divide aplicação em três partes interconectadas:**
  - **Model**
  - **View**
  - **Controller**
- **Desacoplamento de componentes**
  - **Eficiente reuso do código**
  - **Desenvolvimento paralelo**
  - **Automação de testes**

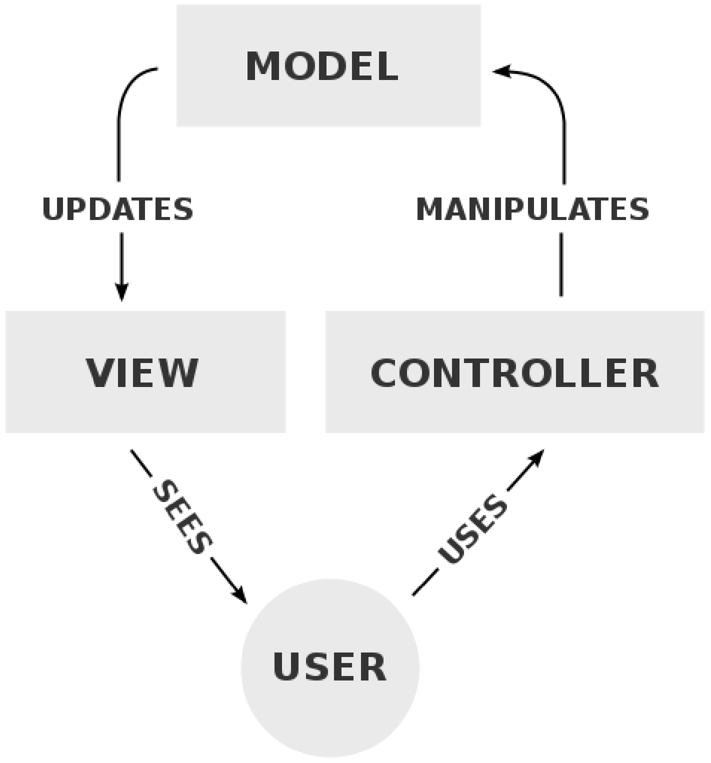


MVC

UTILIZAÇÃO

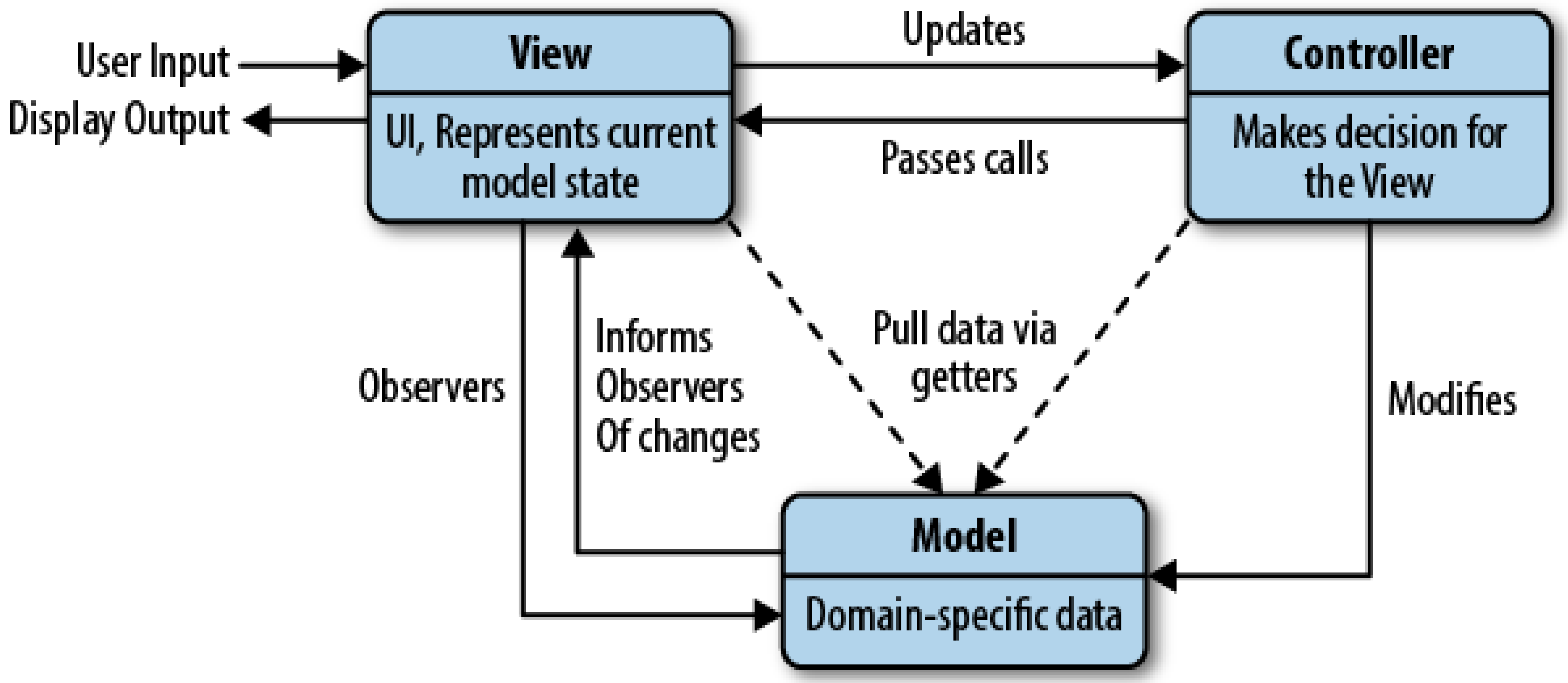
- **Interfaces gráficas (GUI)**
  - **Desktop**
  - **Web**
  - **Mobile**

- **MVC frameworks:**
  - **Java**
  - **C#**
  - **Ruby**
  - **PHP**
  - **Entre outros**



MVC

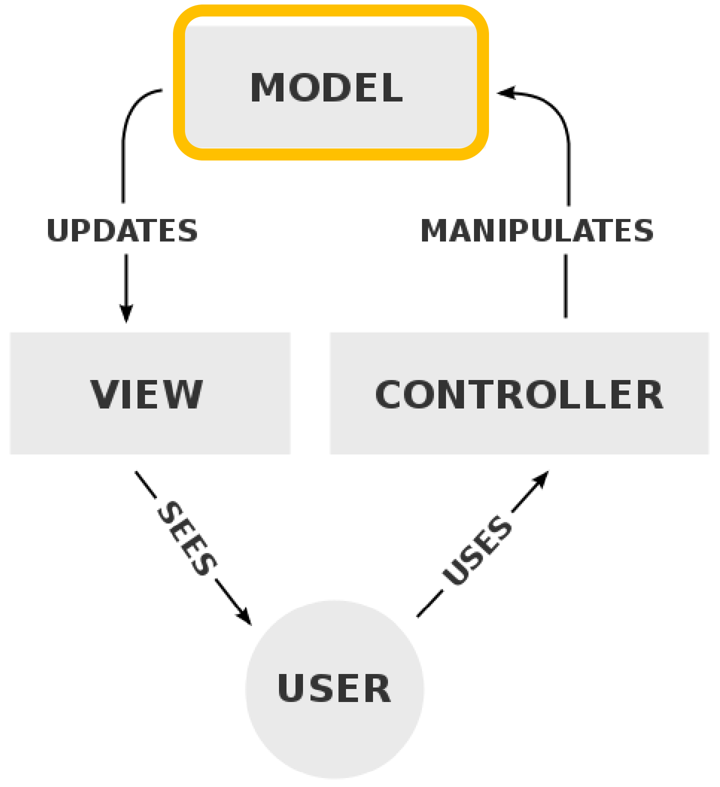
FRAMEWORK JAVA



MVC

MODEL

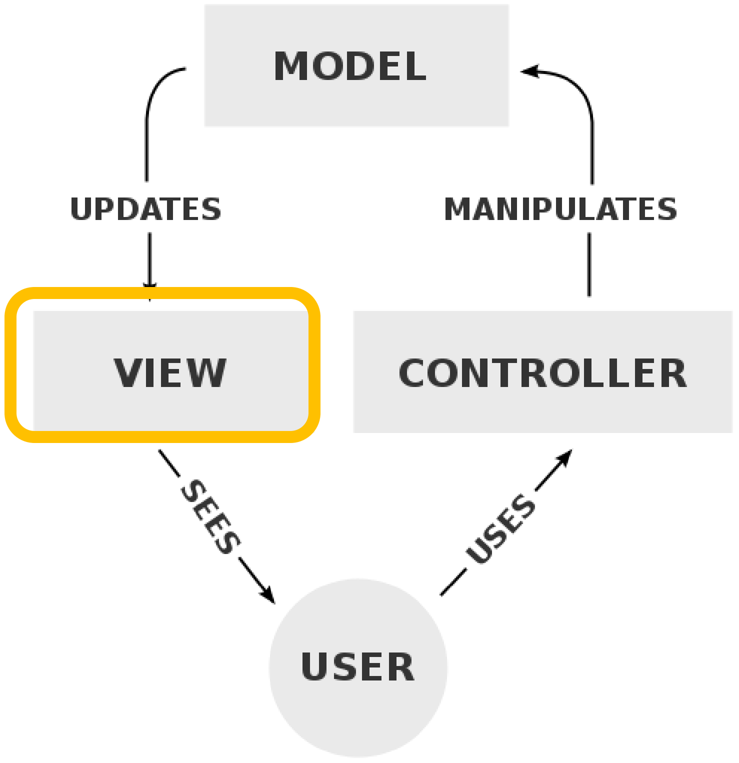
- Componente central do padrão
- Estrutura de dados dinâmica
- Independe da UI
- Gerencia regras, lógica e dados
- Representa o estado da aplicação
- Modelo do domínio a bordo!
- Lógica de negócios encapsulada
- Lógica de persistência de dados
- Notifica as *views* (observers)



MVC

VIEW

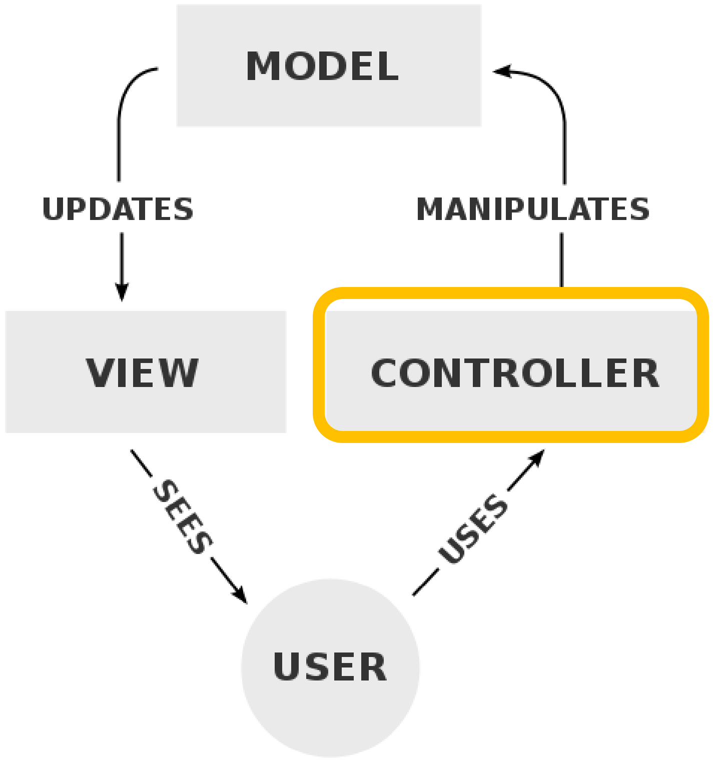
- Representa a saída de informação
- Utilizado pela lógica da UI
- Renderiza UI para mostrar info
- Informação pode ter várias *views*
    - Gráfico de barras
    - Tabela
    - Filtros
- *View* observa model
- View é notificado de mudança
- Recebe informação do *controller*



MVC

CONTROLLER

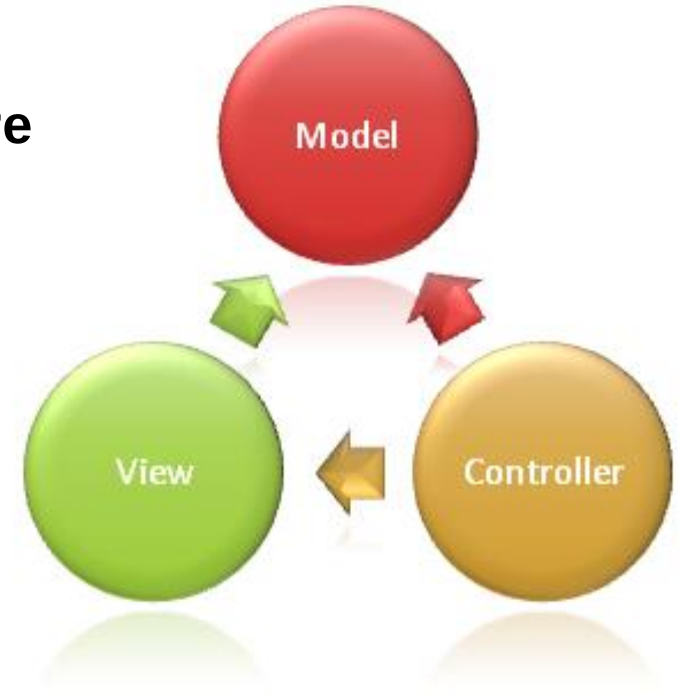
- Intermediário entre *model* e *view*
- Facilita a estratégia MVC
- Atualiza *model* quando usuário manipula a *view*
- Cria instâncias do ViewModel
- View pode delegar gerenciamento de eventos para o *controller*
- Não é sua função tratar mudanças do *model*



ASP.NET MVC

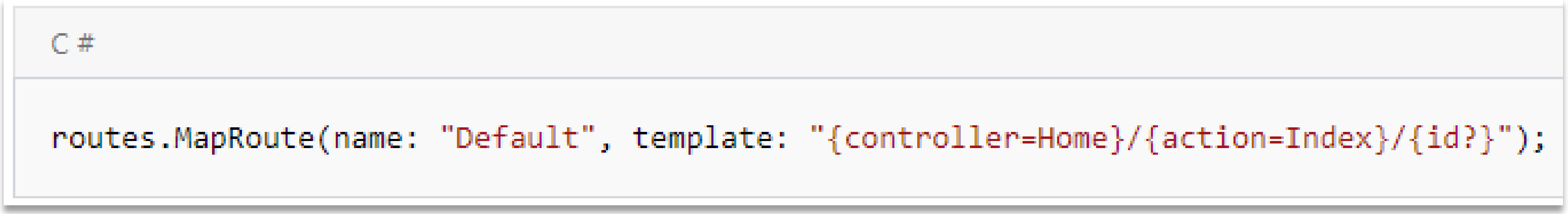
CARACTERÍSTICAS

- **Framework**
  - **Leve**
  - **Código aberto e**
  - **Altamente testável**
- **Otimizado para uso com ASP.NET Core**
- **Integrado com ASP.NET:**
  - **Autenticação**
  - **Roteamento**
  - **Validação do modelo**
  - **Injeção de dependência**
  - **Testes**
  - **Master Pages**
  - **APIs**



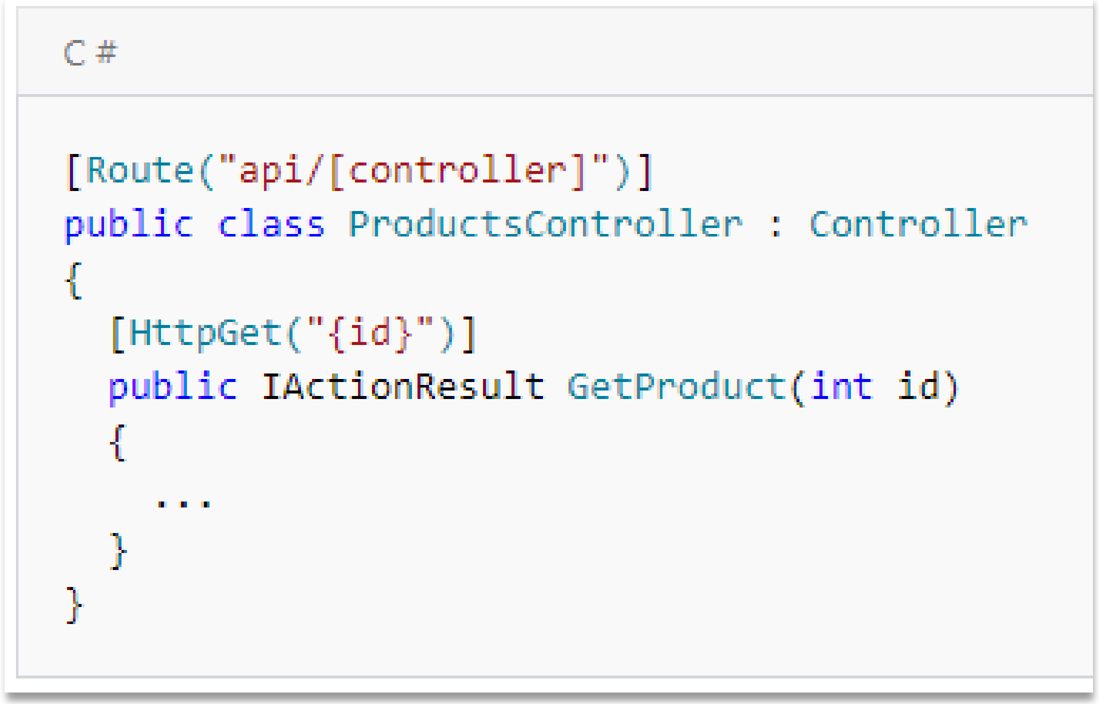
ASP.NET MVC

ROTEAMENTO



C#

```
routes.MapRoute(name: "Default", template: "{controller=Home}/{action=Index}/{id?}");
```

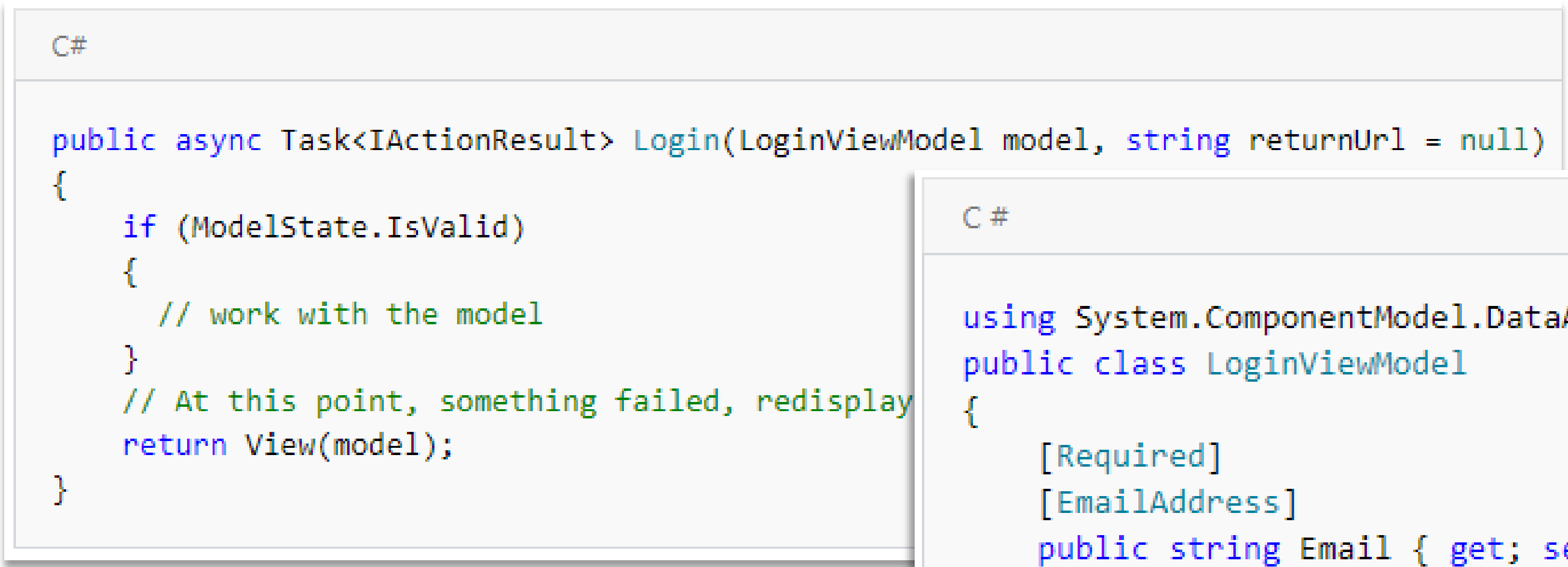


C#

```
[Route("api/[controller]")]
public class ProductsController : Controller
{
   [HttpGet("{id}")]
   public IActionResult GetProduct(int id)
   {
      ...
   }
}
```

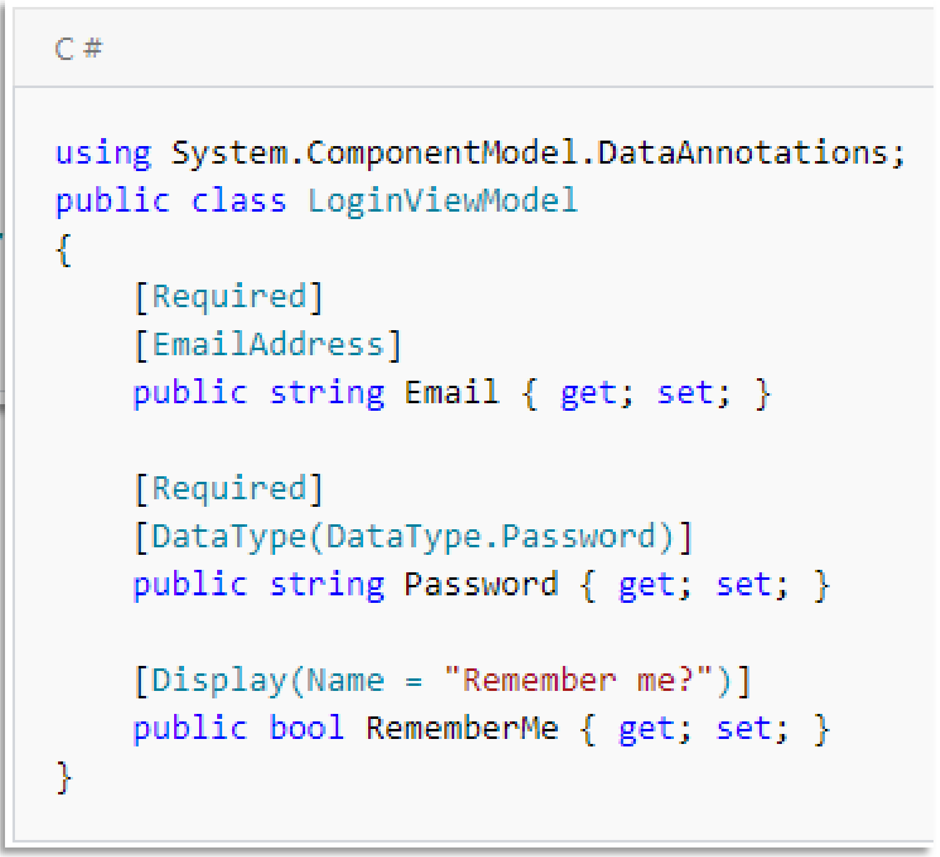
ASP.NET MVC

VALIDAÇÃO DO MODELO



C#

```csharp
public async Task<IActionResult> Login(LoginViewModel model, string returnUrl = null)
{
    if (ModelState.IsValid)
    {
      // work with the model
    }
    // At this point, something failed, redisplay
    return View(model);
}
```

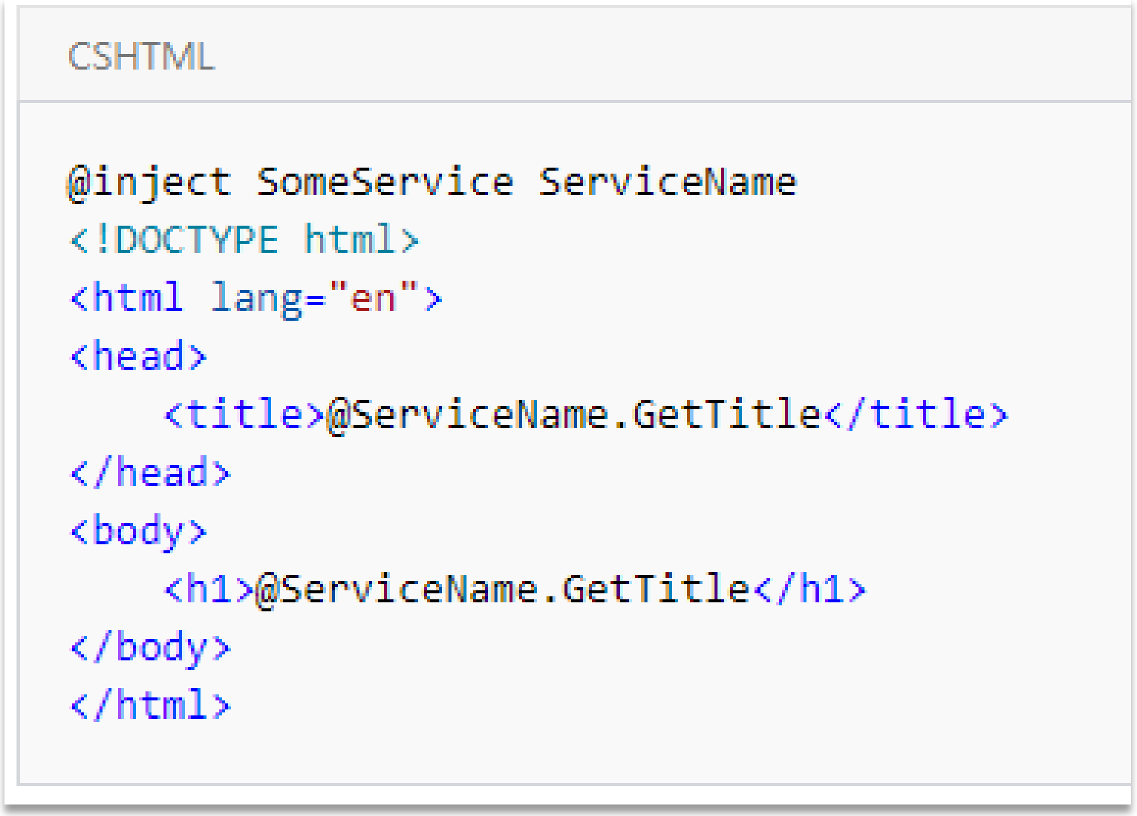


C#

```csharp
using System.ComponentModel.DataAnnotations;
public class LoginViewModel
{
    [Required]
    [EmailAddress]
    public string Email { get; set; }

    [Required]
    [DataType(DataType.Password)]
    public string Password { get; set; }

    [Display(Name = "Remember me?")]
    public bool RememberMe { get; set; }
}
```

ASP.NET MVC

DEPENDENCY INJECTION

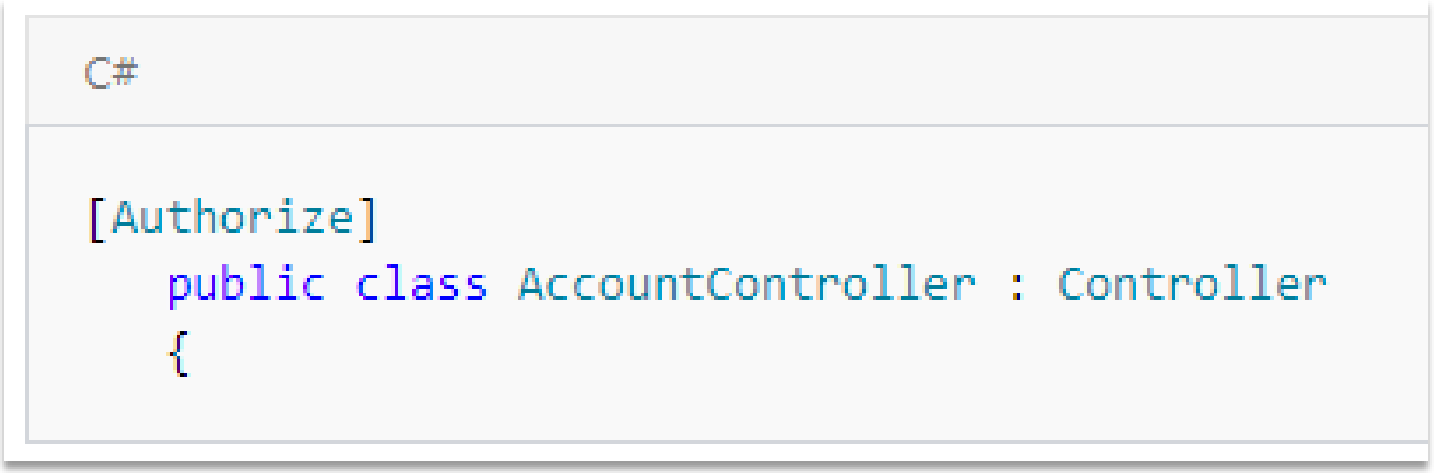


CSHTML

```
@inject SomeService ServiceName
<!DOCTYPE html>
<html lang="en">
<head>
    <title>@ServiceName.GetTitle</title>
</head>
<body>
    <h1>@ServiceName.GetTitle</h1>
</body>
</html>
```

ASP.NET MVC

FILTROS

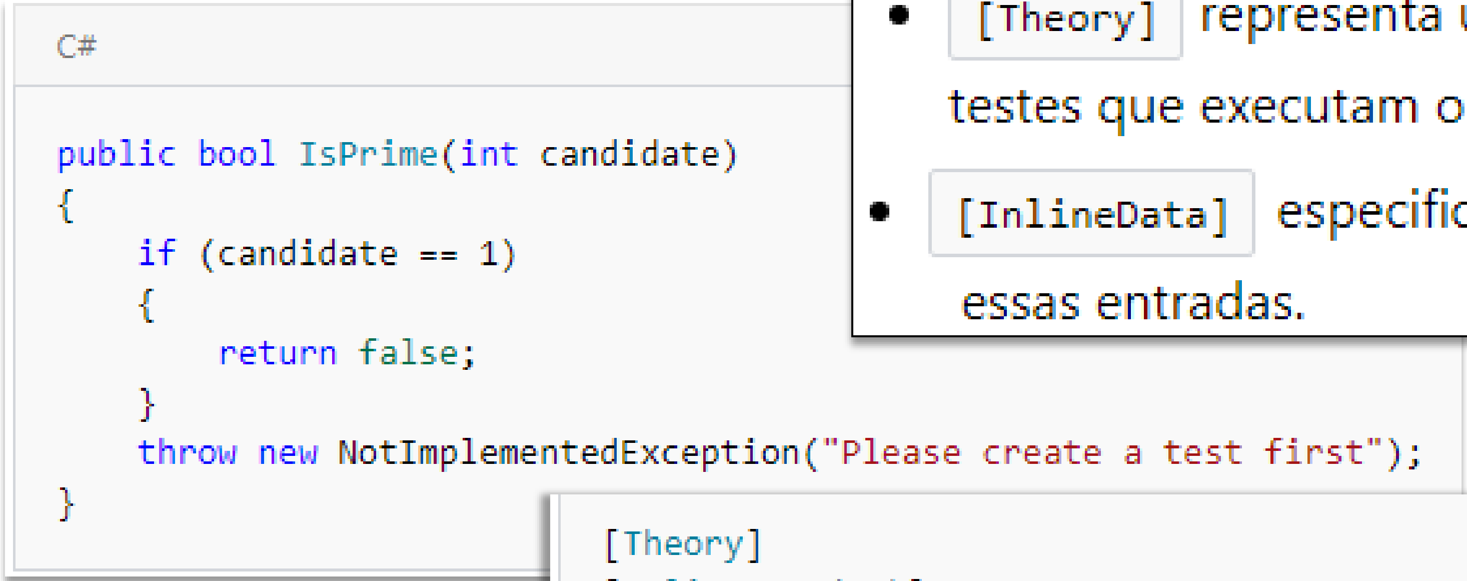


```
C#

[Authorize]
    public class AccountController : Controller
    {
```

- Execução de lógica personalizada
- Pré e pós-processamento
- Filtros embutidos na estrutura

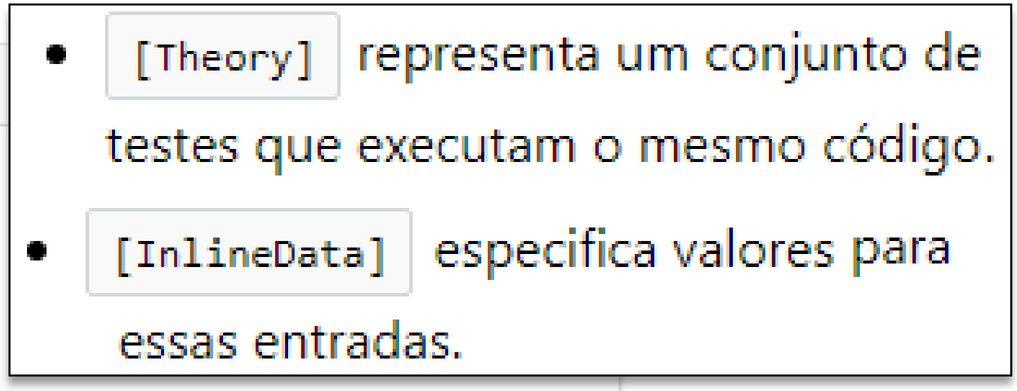
ASP.NET MVC

UNIT TESTS

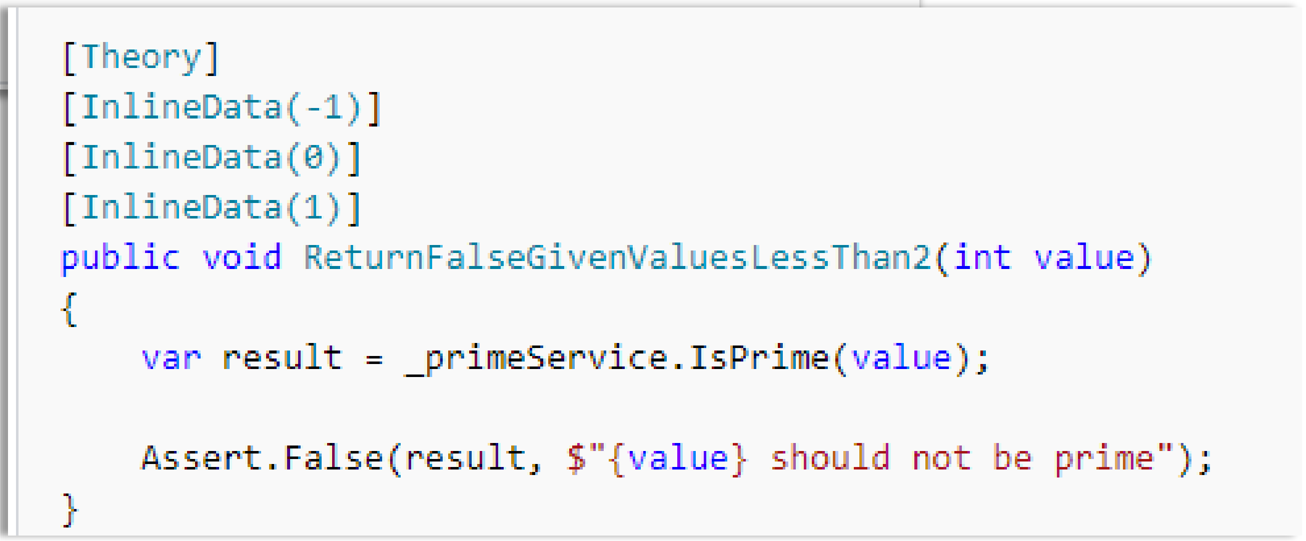


C#

```
public bool IsPrime(int candidate)
{
    if (candidate == 1)
    {
        return false;
    }
    throw new NotImplementedException("Please create a test first");
}
```



- `[Theory]` representa um conjunto de testes que executam o mesmo código.
- `[InlineData]` especifica valores para essas entradas.



```
[Theory]
[InlineData(-1)]
[InlineData(0)]
[InlineData(1)]
public void ReturnFalseGivenValuesLessThan2(int value)
{
    var result = _primeService.IsPrime(value);

    Assert.False(result, $"{value} should not be prime");
}
```

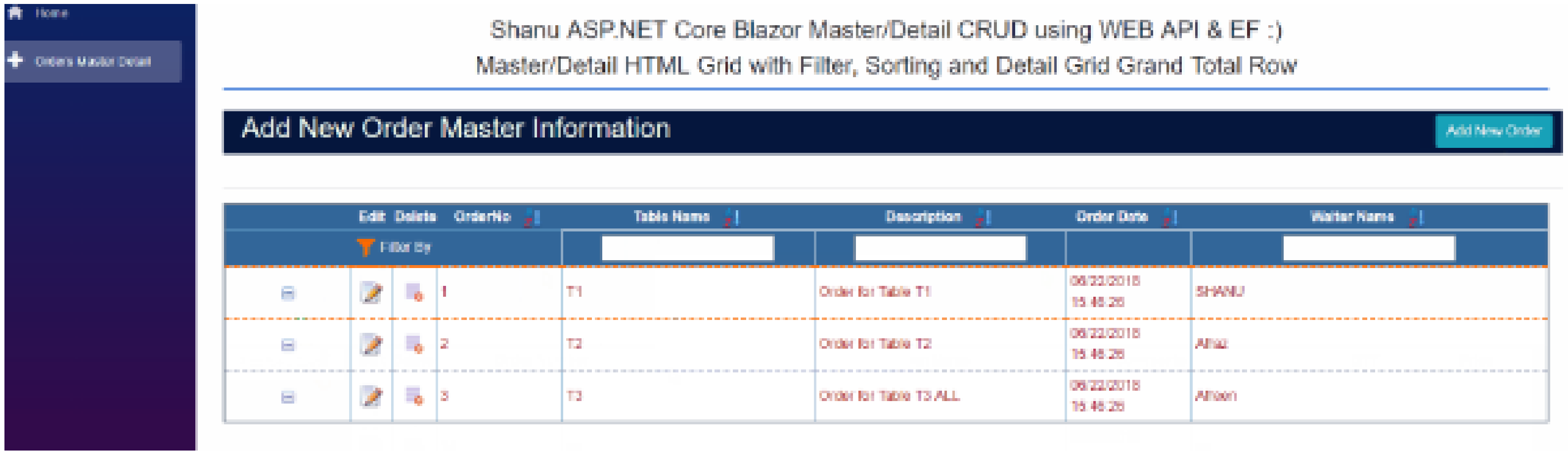
Exemplo

ASP.NET Core Blazor Master/Detail CRUD with Filtering and Sorting using EF and Web API

syed shanu, 11 Jul 2018

★★★★★ 4.90 (13 votes) Rate: ☆☆☆☆☆

In this article, let's see how to create our own ASP.NET Core Blazor Master Detail HTML CRUD (Insert, Update, Select and Delete) for both Master and Detail Grid with Sorting and Filtering using Entity Framework, and Web API.



Shanu ASP.NET Core Blazor Master/Detail CRUD using WEB API & EF :)
Master/Detail HTML Grid with Filter, Sorting and Detail Grid Grand Total Row

Add New Order Master Information

	Edit	Delete	OrderNo	Table Name	Description	Order Date	Waiter Name
Filter By							
⊟			1	T1	Order for Table T1	06/22/2018 15:46:26	SHANU
⊟			2	T2	Order for Table T2	06/22/2018 15:46:26	Afraz
⊟			3	T3	Order for Table T3 ALL	06/22/2018 15:46:26	Afreen

codeproject.com/Articles/1251603/ASP-NET-Core-Blazor-Master-Detail-CRUD-with-Filter

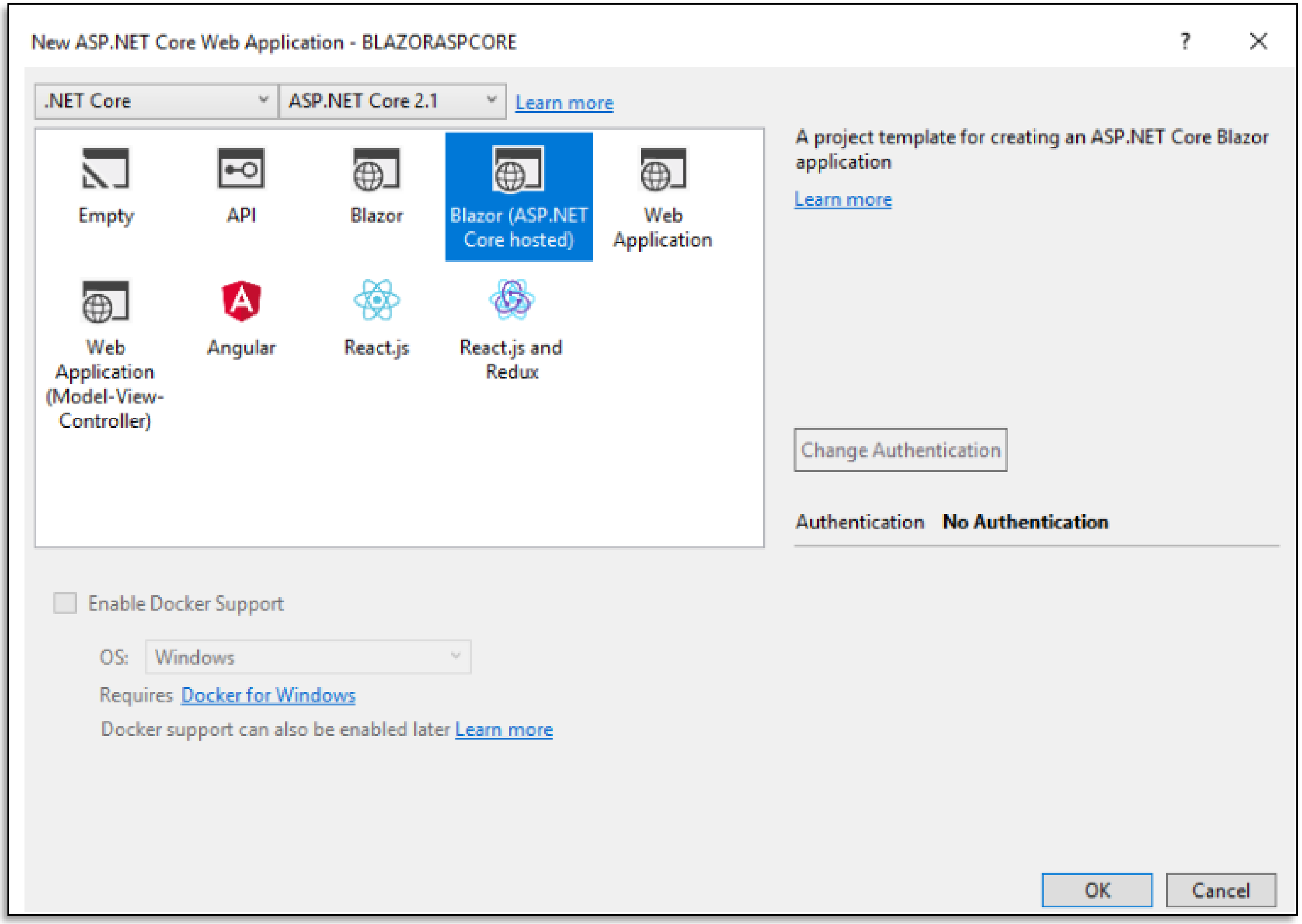
Exemplo

- Cria DB

```
CREATE DATABASE OrderManagement
GO

USE OrderManagement
GO

-- Create Table

CREATE TABLE [dbo].[OrderMasters](
[Order_No] INT IDENTITY PRIMARY KEY,
[Table_ID] [varchar](20) NOT NULL,
[Description] [varchar](200) NOT NULL,
[Order_DATE] [datetime] NOT NULL,
[Waiter_Name] [varchar](20) NOT NULL
)


CREATE TABLE [dbo].[OrderDetails](
  [Order_Detail_No] INT IDENTITY PRIMARY KEY,
 [Order_No] INT,
 [Item_Name] [varchar](20) NOT NULL,
 [Notes] [varchar](200) NOT NULL,
[QTY]  INT NOT NULL,
 [Price] INT NOT NULL
 )
```

Exemplo

- *Cria Aplicação*



New ASP.NET Core Web Application - BLAZORASPCORE

.NET Core | ASP.NET Core 2.1 | Learn more

Empty
API
Blazor
Blazor (ASP.NET Core hosted)
Web Application
Web Application (Model-View-Controller)
Angular
React.js
React.js and Redux

A project template for creating an ASP.NET Core Blazor application

Learn more

Authentication **No Authentication**

Enable Docker Support

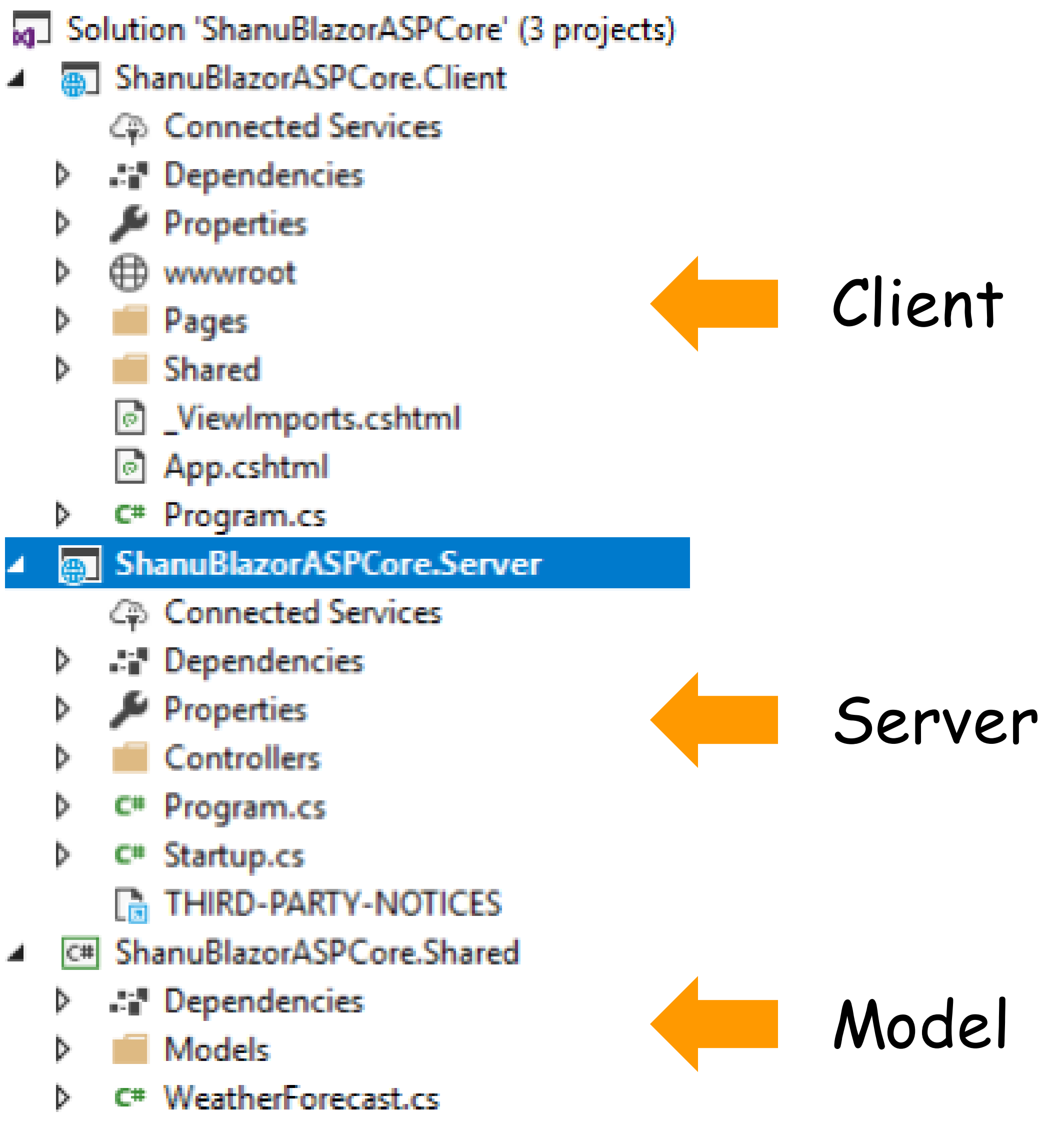
OS: Windows

Requires Docker for Windows

Docker support can also be enabled later Learn more

Exemplo

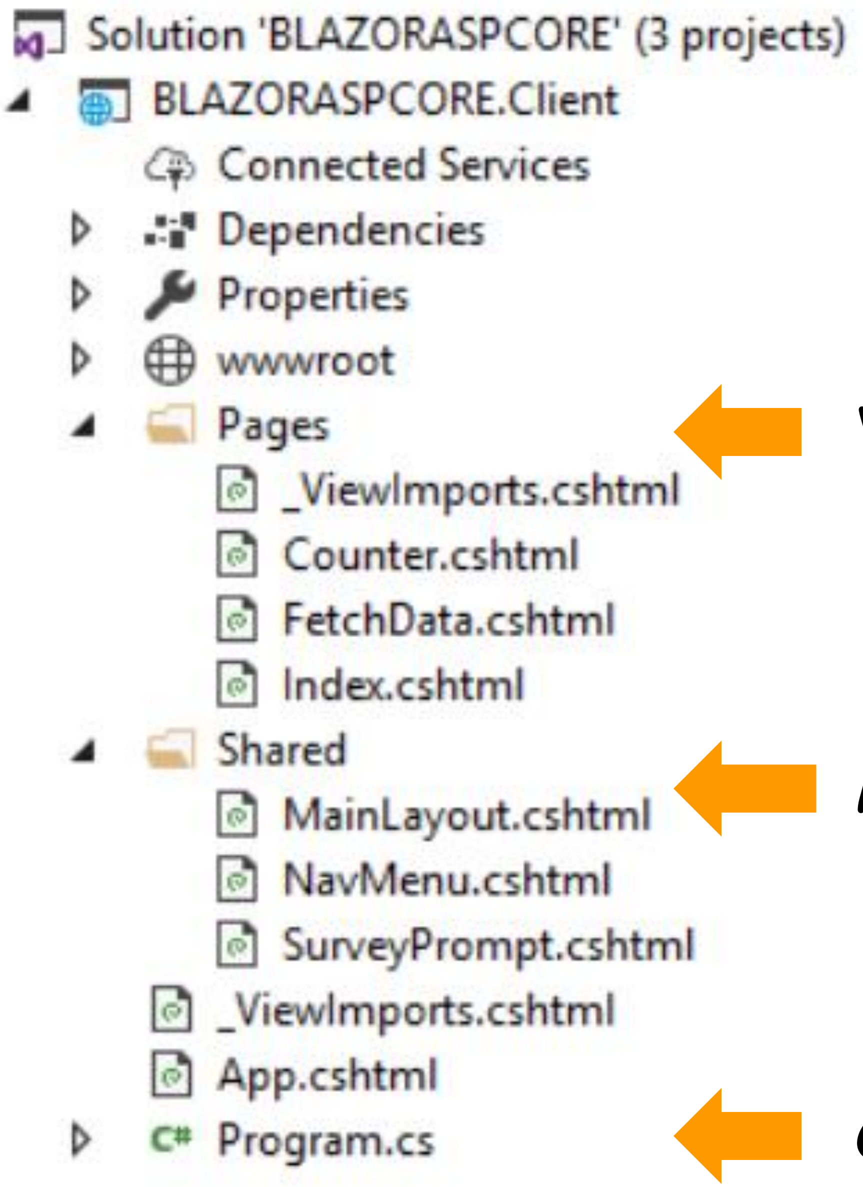
Criação automática de projetos



```
Solution 'ShanuBlazorASPCore' (3 projects)
◢ ShanuBlazorASPCore.Client
      Connected Services
   ▷  Dependencies
   ▷  Properties
   ▷  wwwroot
   ▷  Pages
   ▷  Shared
      _ViewImports.cshtml
      App.cshtml
   ▷  Program.cs
◢ ShanuBlazorASPCore.Server
      Connected Services
   ▷  Dependencies
   ▷  Properties
   ▷  Controllers
   ▷  Program.cs
   ▷  Startup.cs
      THIRD-PARTY-NOTICES
◢ ShanuBlazorASPCore.Shared
   ▷  Dependencies
   ▷  Models
   ▷  WeatherForecast.cs
```

Exemplo

Client Project



Solution 'BLAZORASPCORE' (3 projects)
- BLAZORASPCORE.Client
  - Connected Services
  - Dependencies
  - Properties
  - wwwroot
  - Pages
    - _ViewImports.cshtml
    - Counter.cshtml
    - FetchData.cshtml
    - Index.cshtml
  - Shared
    - MainLayout.cshtml
    - NavMenu.cshtml
    - SurveyPrompt.cshtml
  - _ViewImports.cshtml
  - App.cshtml
  - Program.cs

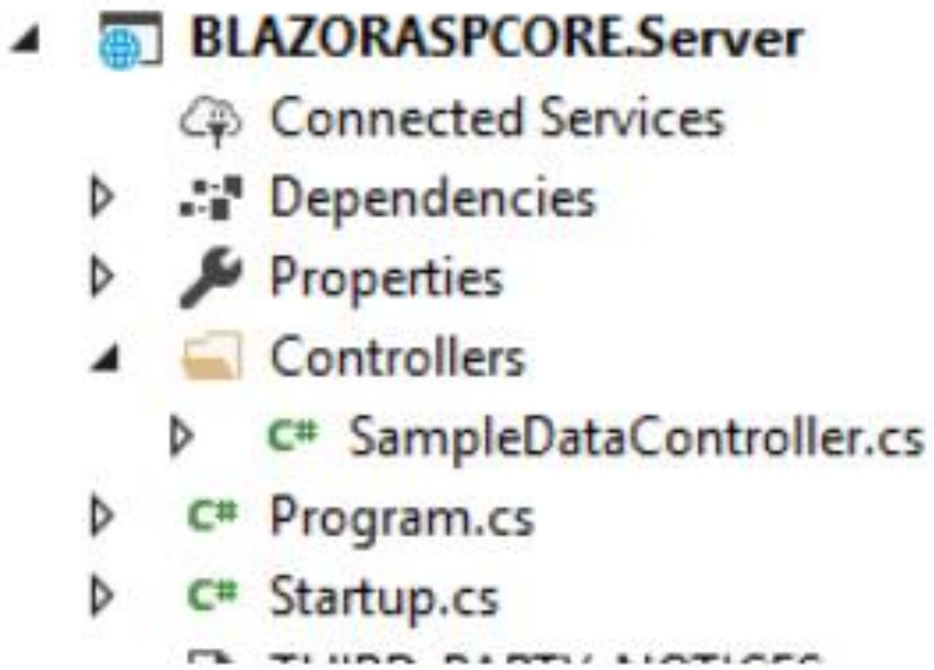
Views

Master Pages

Client UI

Exemplo

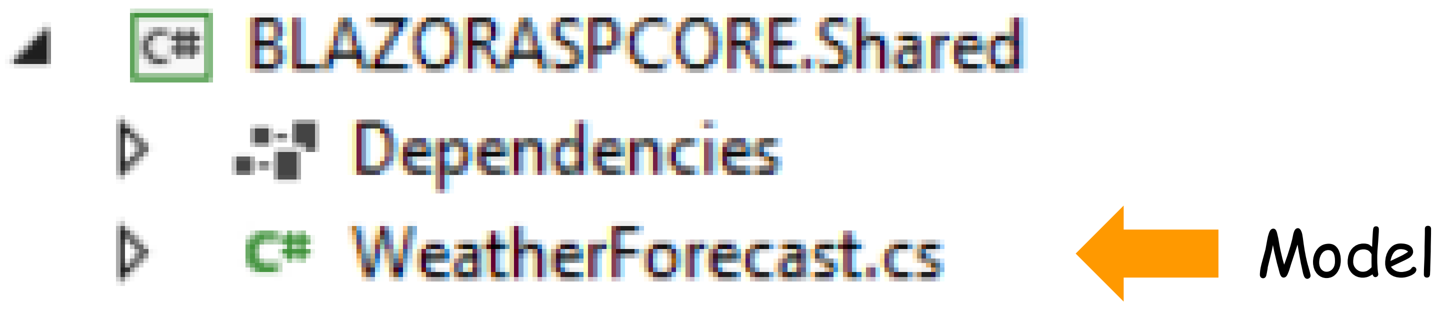
Server Project



BLAZORASPCORE.Server
Connected Services
Dependencies
Properties
Controllers
SampleDataController.cs
Program.cs
Startup.cs

Exemplo

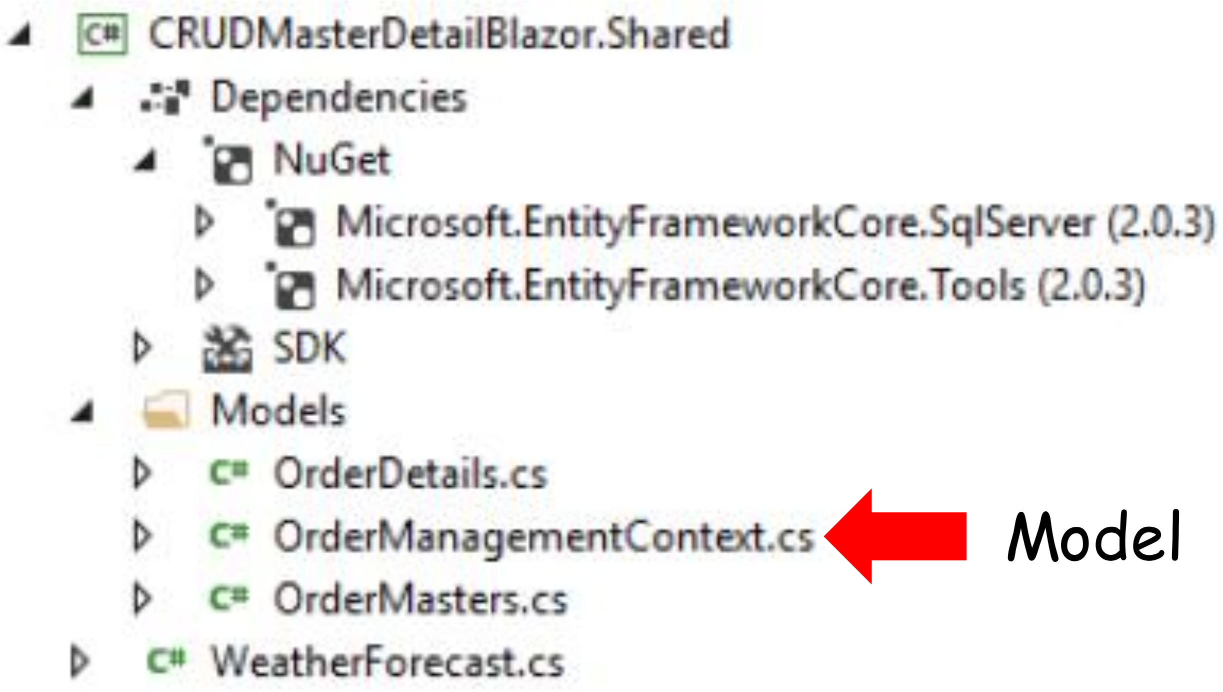
- Modelo compartilhado entre os projetos
  - Client Project
  - Server Project


BLAZORASPCORE.Shared
Dependencies
WeatherForecast.cs

Exemplo

- Instala Packages
  - Microsoft.EntityFrameworkCore.SqlServer
  - Microsoft.EntityFrameworkCore.Tools

```
Scaffold-DbContext "Server= SQLServerName;Database=OrderManagement;
user id=SQLID;password=SQLPWD;Trusted_Connection=True;MultipleActiveResultSets=true"
Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -Tables OrderMasters ,OrderDetails
```

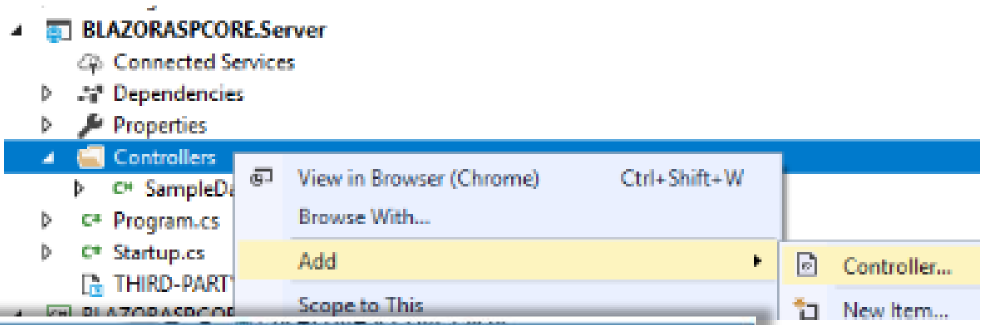


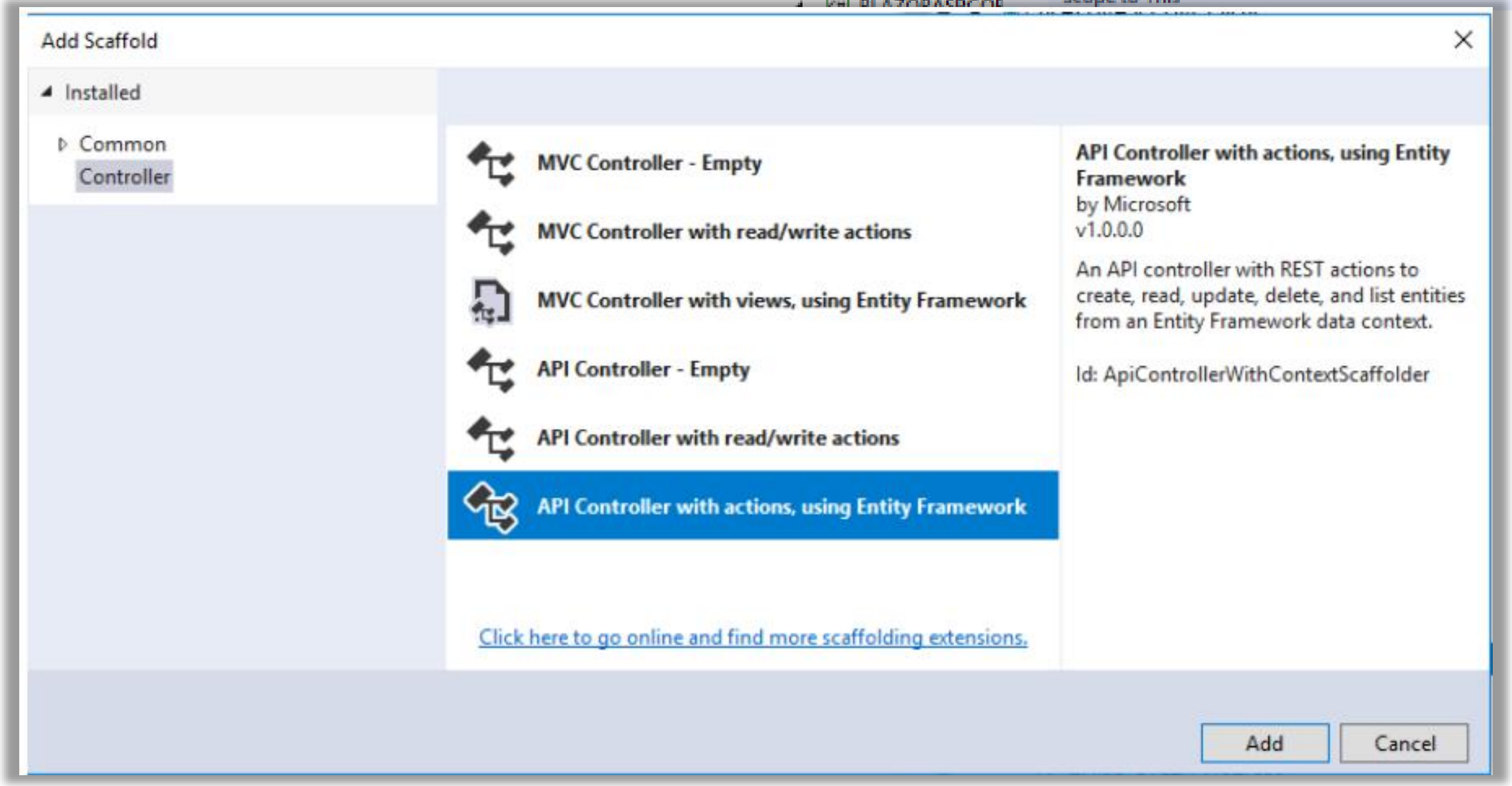
```
◢ CRUDMasterDetailBlazor.Shared
  ◢ Dependencies
    ◢ NuGet
      ▷ Microsoft.EntityFrameworkCore.SqlServer (2.0.3)
      ▷ Microsoft.EntityFrameworkCore.Tools (2.0.3)
    ▷ SDK
  ◢ Models
    ▷ OrderDetails.cs
    ▷ OrderManagementContext.cs
    ▷ OrderMasters.cs
  ▷ WeatherForecast.cs
```

Model

Exemplo

- Cria Web API Controller



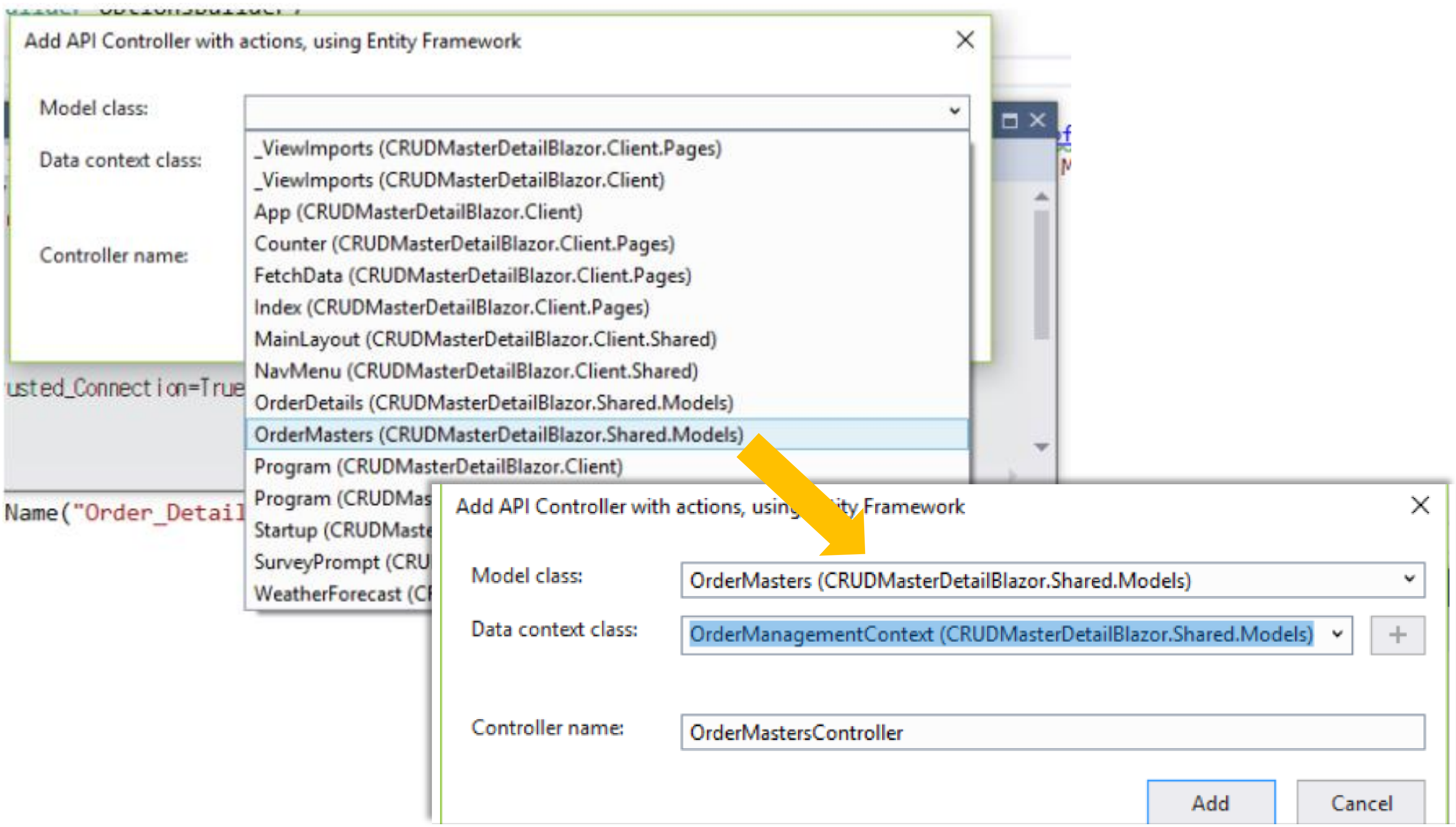

Add Scaffold
Installed
Common
Controller
MVC Controller - Empty
MVC Controller with read/write actions
MVC Controller with views, using Entity Framework
API Controller - Empty
API Controller with read/write actions
API Controller with actions, using Entity Framework
Click here to go online and find more scaffolding extensions.
Add
Cancel

API Controller with actions, using Entity Framework
by Microsoft
v1.0.0.0
An API controller with REST actions to create, read, update, delete, and list entities from an Entity Framework data context.

Id: ApiControllerWithContextScaffolder

Exemplo

- **Seleciona OrderMasters para CRUD**


Add API Controller with actions, using Entity Framework

Model class:

Data context class:

Controller name:

_ViewImports (CRUDMasterDetailBlazor.Client.Pages)
_ViewImports (CRUDMasterDetailBlazor.Client)
App (CRUDMasterDetailBlazor.Client)
Counter (CRUDMasterDetailBlazor.Client.Pages)
FetchData (CRUDMasterDetailBlazor.Client.Pages)
Index (CRUDMasterDetailBlazor.Client.Pages)
MainLayout (CRUDMasterDetailBlazor.Client.Shared)
NavMenu (CRUDMasterDetailBlazor.Client.Shared)
OrderDetails (CRUDMasterDetailBlazor.Shared.Models)
OrderMasters (CRUDMasterDetailBlazor.Shared.Models)
Program (CRUDMasterDetailBlazor.Client)
Program (CRUDMas
Startup (CRUDMaste
SurveyPrompt (CRU
WeatherForecast (CF

```
usted_Connection=True
Name("Order_Detail
```

Add API Controller with actions, using Entity Framework

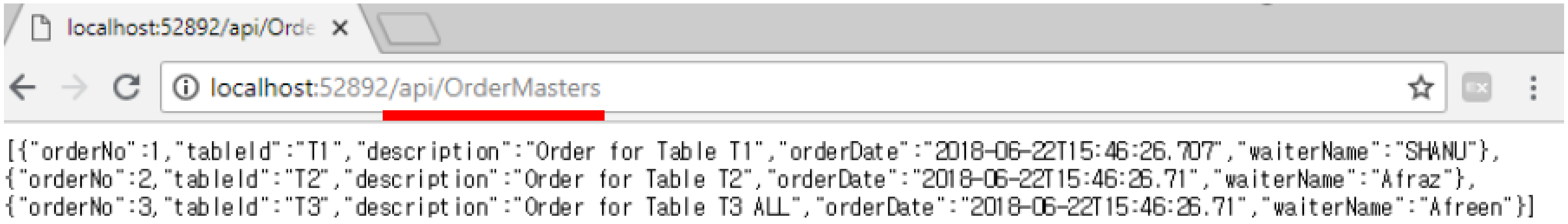
Model class: OrderMasters (CRUDMasterDetailBlazor.Shared.Models)

Data context class: OrderManagementContext (CRUDMasterDetailBlazor.Shared.Models)

Controller name: OrderMastersController

Exemplo

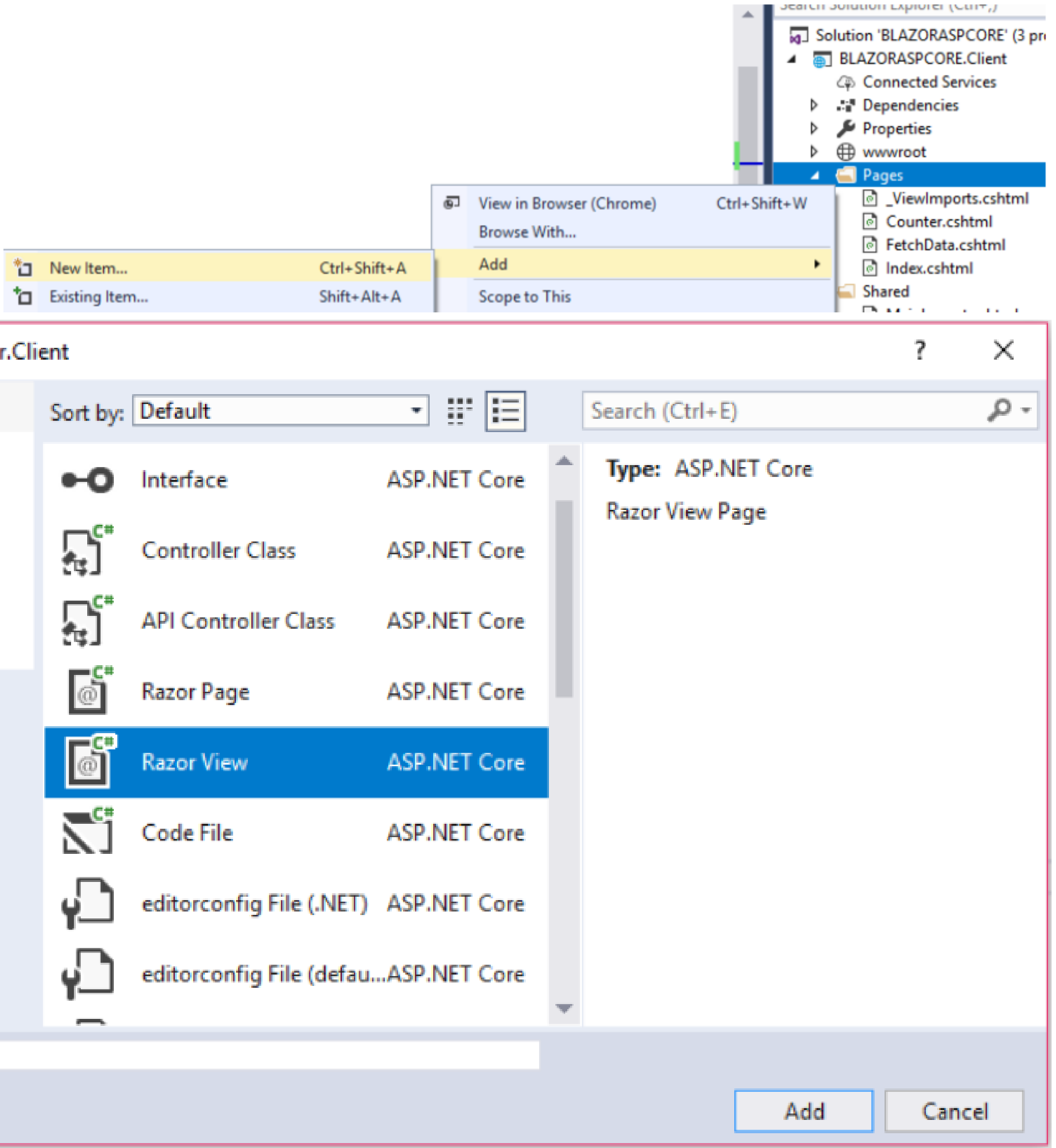
- **Testando GET**



```
[{"orderNo":1,"tableId":"T1","description":"Order for Table T1","orderDate":"2018-06-22T15:46:26.707","waiterName":"SHANU"},
{"orderNo":2,"tableId":"T2","description":"Order for Table T2","orderDate":"2018-06-22T15:46:26.71","waiterName":"Afraz"},
{"orderNo":3,"tableId":"T3","description":"Order for Table T3 ALL","orderDate":"2018-06-22T15:46:26.71","waiterName":"Afreen"}]
```

Exemplo

- *Client Project*
- *Adicionando View*



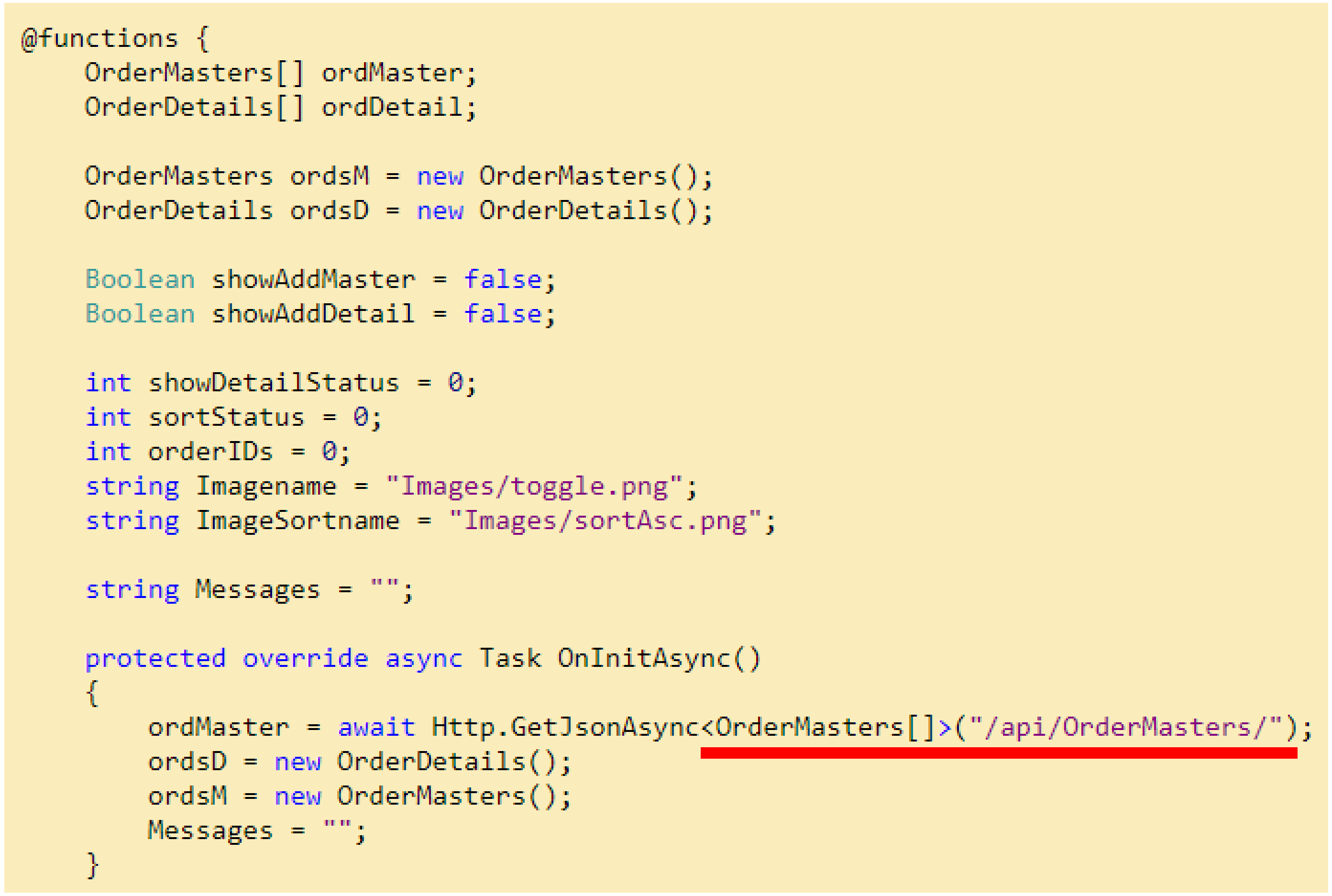
Type: ASP.NET Core

Razor View Page

Name: Orders.cshtml

Exemplo

- Client Project
- Inicialização



```
@functions {
    OrderMasters[] ordMaster;
    OrderDetails[] ordDetail;

    OrderMasters ordsM = new OrderMasters();
    OrderDetails ordsD = new OrderDetails();

    Boolean showAddMaster = false;
    Boolean showAddDetail = false;

    int showDetailStatus = 0;
    int sortStatus = 0;
    int orderIDs = 0;
    string Imagename = "Images/toggle.png";
    string ImageSortname = "Images/sortAsc.png";

    string Messages = "";

    protected override async Task OnInitAsync()
    {
        ordMaster = await Http.GetJsonAsync<OrderMasters[]>("/api/OrderMasters/");
        ordsD = new OrderDetails();
        ordsM = new OrderMasters();
        Messages = "";
    }
```

Exemplo

- **Views**

```
@using MasterDetailCRUD.Shared
@using MasterDetailCRUD.Shared.Models
@page "/Orders"
@using Microsoft.AspNetCore.Blazor.Browser.Interop

@using System.Collections.Generic
@using Microsoft.AspNetCore.Blazor
```

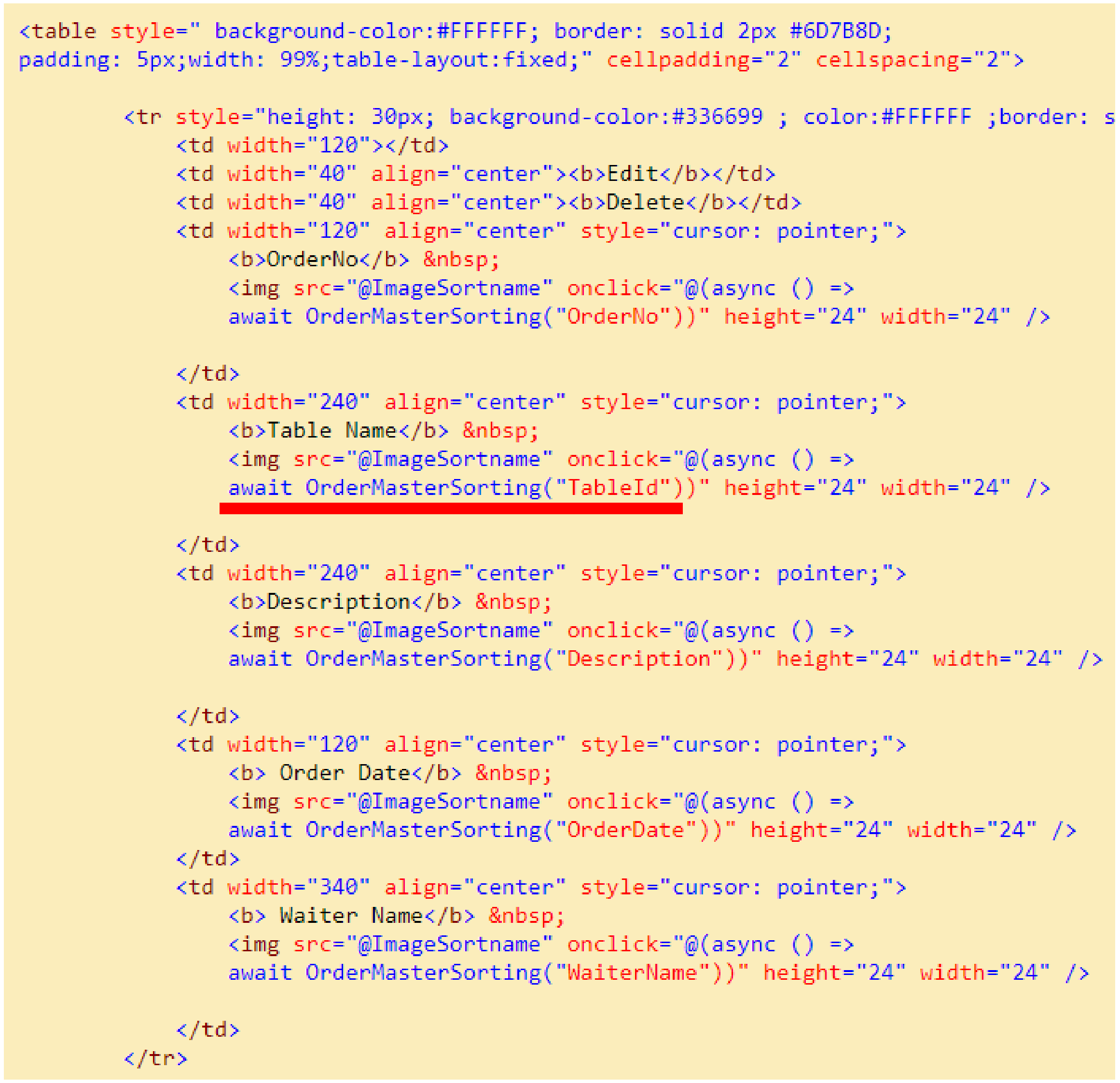
```
@foreach (var OrderMasterobj in ordMaster)
        {
            <tr style="border-style:dashed;border-width:2px;border-color:
            @(OrderMasterobj.OrderNo == orderIDs ? "#ff6a00": "#a2aabe")">
                <td align="center" style="border:
                solid 1px #659EC7; padding: 5px;table-layout:fixed;">

                    @if (@OrderMasterobj.OrderNo == orderIDs)
                    {
                        <img src="@Imagename" onclick="@(async () =>
                        await getOrderDetails(@OrderMasterobj.OrderNo))" />
                    }
                    else
                    {
                        <img src="Images/toggle.png" onclick="@(asy
                        await getOrderDetails(@OrderMasterobj.OrderN

                    }

                </td>
                <td align="center" style="border: solid 1px #659EC7;
                padding: 5px;table-layout:fixed;">
                    <span style="color:#9F000F">
                        <img src="Images/edit.gif" alt="Edit"

                        width="24px" height="24px" onclick="@(async () =>
                        await EditOrderMaster(@OrderMasterobj.OrderNo))" />
                    </span>
                </td>
                <td align="center" style="border: solid 1px #659EC7;
                padding: 5px;table-layout:fixed;">
                    <span style="color:#9F000F">
                        <img src="Images/delete.gif" alt="Delete"

                        width="24px" height="24px" onclick="@(async () =>
                        await DeleteOrderMaster(@OrderMasterobj.OrderNo))" />
                    </span>
                </td>
                <td align="left" style="border: solid 1px #659EC7;
                padding: 5px;table-layout:fixed;">
                    <span style="color:#9F000F">
                        @OrderMasterobj.OrderNo
                    </span>
                </td>
                <td align="left" style="border: solid 1px #659EC7;
                padding: 5px;table-layout:fixed;">
                    <span style="color:#9F000F">
                        @OrderMasterobj.TableId
                    </span>
                </td>
                <td align="left" style="border: solid 1px #659EC7;
                padding: 5px;table-layout:fixed;">
                    <span style="color:#9F000F">
                        @OrderMasterobj.Description
                    </span>
                </td>
                <td align="left" style="border: solid 1px #659EC7;
                padding: 5px;table-layout:fixed;">
                    <span style="color:#9F000F">
                        @OrderMasterobj.OrderDate
                    </span>
                </td>
                <td align="left" style="border: solid 1px #659EC7;
                padding: 5px;table-layout:fixed;">
                    <span style="color:#9F000F">
                        @OrderMasterobj.WaiterName
                    </span>
                </td>
            </tr>
```

```
@foreach (var OrderMasterobj in ordMaster)
        {
            <tr style="border-style:dashed;border-width:2px;border-color:
            @(OrderMasterobj.OrderNo == orderIDs ? "#ff6a00": "#a2aabe")">
                <td align="center" style="border:
                solid 1px #659EC7; padding: 5px;table-layout:fixed;">

                    @if (@OrderMasterobj.OrderNo == orderIDs)
                    {
                        <img src="@Imagename" onclick="@(async () =>
                        await getOrderDetails(@OrderMasterobj.OrderNo))" />
                    }
                    else
                    {
                        <img src="Images/toggle.png" onclick="@(async () =>
                        await getOrderDetails(@OrderMasterobj.OrderNo))" />

                    }

                </td>
```

```
                <td align="center" style="border: solid 1px #659EC7;
                padding: 5px;table-layout:fixed;">
                    <span style="color:#9F000F">
                        <img src="Images/delete.gif" alt="Delete"

                        width="24px" height="24px" onclick="@(async () =>
                        await DeleteOrderMaster(@OrderMasterobj.OrderNo))" />
                    </span>
                </td>
                <td align="left" style="border: solid 1px #659EC7;
                padding: 5px;table-layout:fixed;">
                    <span style="color:#9F000F">
                        @OrderMasterobj.OrderNo
                    </span>
                </td>
```

Exemplo

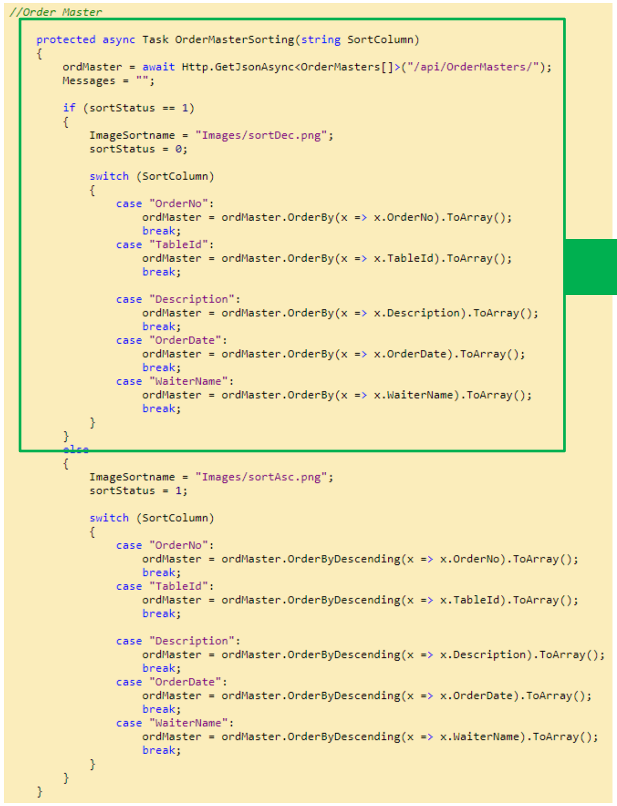
- *Client Project*
- *Cabeçalho Tabela*
- *Ordenação*



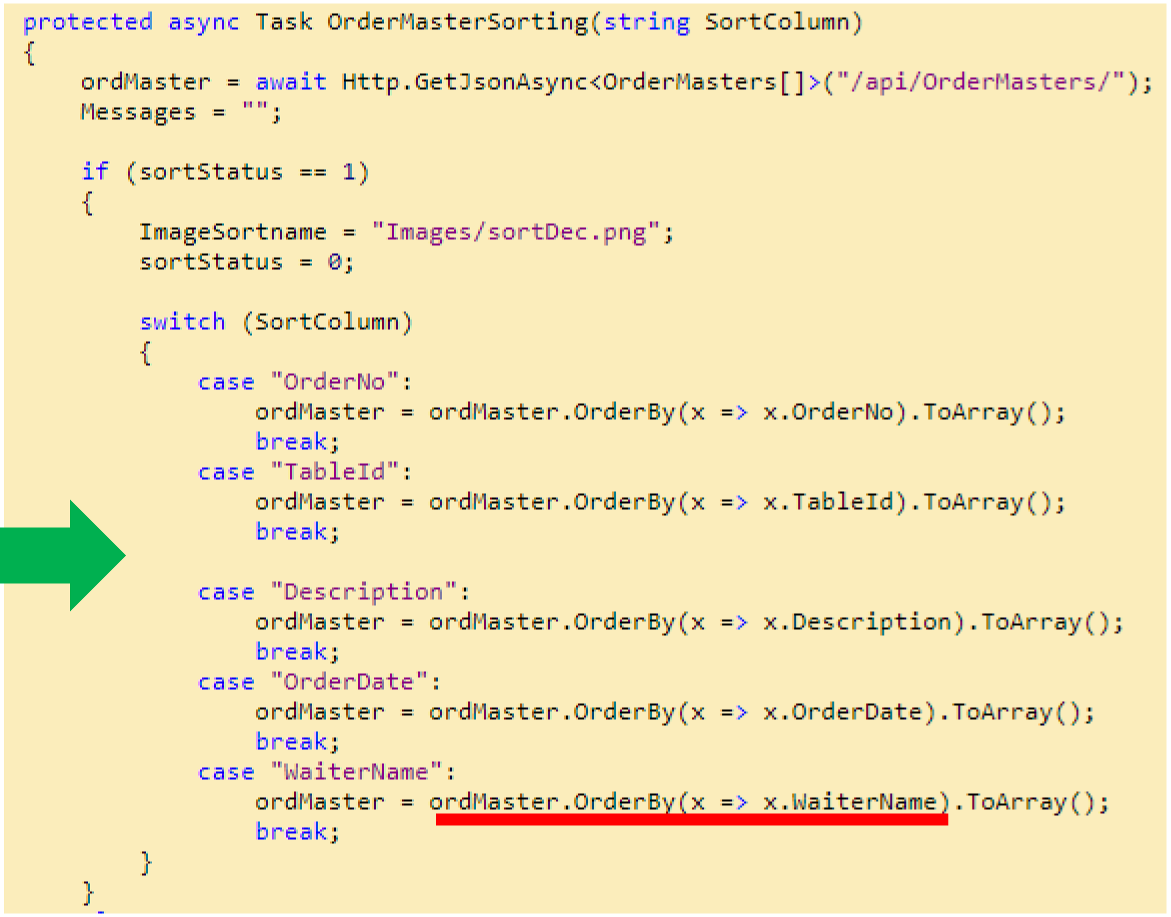
```
<table style=" background-color:#FFFFFF; border: solid 2px #6D7B8D;
padding: 5px;width: 99%;table-layout:fixed;" cellpadding="2" cellspacing="2">

        <tr style="height: 30px; background-color:#336699 ; color:#FFFFFF ;border: s
            <td width="120"></td>
            <td width="40" align="center"><b>Edit</b></td>
            <td width="40" align="center"><b>Delete</b></td>
            <td width="120" align="center" style="cursor: pointer;">
                <b>OrderNo</b> &nbsp;
                <img src="@ImageSortname" onclick="@(async () =>
                await OrderMasterSorting("OrderNo"))" height="24" width="24" />

            </td>
            <td width="240" align="center" style="cursor: pointer;">
                <b>Table Name</b> &nbsp;
                <img src="@ImageSortname" onclick="@(async () =>
                await OrderMasterSorting("TableId"))" height="24" width="24" />

            </td>
            <td width="240" align="center" style="cursor: pointer;">
                <b>Description</b> &nbsp;
                <img src="@ImageSortname" onclick="@(async () =>
                await OrderMasterSorting("Description"))" height="24" width="24" />

            </td>
            <td width="120" align="center" style="cursor: pointer;">
                <b> Order Date</b> &nbsp;
                <img src="@ImageSortname" onclick="@(async () =>
                await OrderMasterSorting("OrderDate"))" height="24" width="24" />
            </td>
            <td width="340" align="center" style="cursor: pointer;">
                <b> Waiter Name</b> &nbsp;
                <img src="@ImageSortname" onclick="@(async () =>
                await OrderMasterSorting("WaiterName"))" height="24" width="24" />

            </td>
        </tr>
```

Exemplo

- Client Project
- Cabeçalho Tabela
- Ordenação



```
//Order Master
    protected async Task OrderMasterSorting(string SortColumn)
    {
        ordMaster = await Http.GetJsonAsync<OrderMasters[]>("/api/OrderMasters/");
        Messages = "";

        if (sortStatus == 1)
        {
            ImageSortname = "Images/sortDec.png";
            sortStatus = 0;

            switch (SortColumn)
            {
                case "OrderNo":
                    ordMaster = ordMaster.OrderBy(x => x.OrderNo).ToArray();
                    break;
                case "TableId":
                    ordMaster = ordMaster.OrderBy(x => x.TableId).ToArray();
                    break;

                case "Description":
                    ordMaster = ordMaster.OrderBy(x => x.Description).ToArray();
                    break;
                case "OrderDate":
                    ordMaster = ordMaster.OrderBy(x => x.OrderDate).ToArray();
                    break;
                case "WaiterName":
                    ordMaster = ordMaster.OrderBy(x => x.WaiterName).ToArray();
                    break;
            }
        }
        else
        {
            ImageSortname = "Images/sortAsc.png";
            sortStatus = 1;

            switch (SortColumn)
            {
                case "OrderNo":
                    ordMaster = ordMaster.OrderByDescending(x => x.OrderNo).ToArray();
                    break;
                case "TableId":
                    ordMaster = ordMaster.OrderByDescending(x => x.TableId).ToArray();
                    break;

                case "Description":
                    ordMaster = ordMaster.OrderByDescending(x => x.Description).ToArray();
                    break;
                case "OrderDate":
                    ordMaster = ordMaster.OrderByDescending(x => x.OrderDate).ToArray();
                    break;
                case "WaiterName":
                    ordMaster = ordMaster.OrderByDescending(x => x.WaiterName).ToArray();
                    break;
            }
        }
    }
```



```
protected async Task OrderMasterSorting(string SortColumn)
{
    ordMaster = await Http.GetJsonAsync<OrderMasters[]>("/api/OrderMasters/");
    Messages = "";

    if (sortStatus == 1)
    {
        ImageSortname = "Images/sortDec.png";
        sortStatus = 0;

        switch (SortColumn)
        {
            case "OrderNo":
                ordMaster = ordMaster.OrderBy(x => x.OrderNo).ToArray();
                break;
            case "TableId":
                ordMaster = ordMaster.OrderBy(x => x.TableId).ToArray();
                break;

            case "Description":
                ordMaster = ordMaster.OrderBy(x => x.Description).ToArray();
                break;
            case "OrderDate":
                ordMaster = ordMaster.OrderBy(x => x.OrderDate).ToArray();
                break;
            case "WaiterName":
                ordMaster = ordMaster.OrderBy(x => x.WaiterName).ToArray();
                break;
        }
    }
```