Primeiro vamos no terminal e executar o comando

**dotnet new mvc**

**Ele é bastante semelhante a criação de API**

**Tem pasta controller, json de configuração**

Para executar o projeto só digitar o comando

dotnet new mvc



namespace MVC.Controllers;

public class HomeController : Controller

{

    private readonly ILogger<HomeController> \_logger;

    public HomeController(ILogger<HomeController> logger)

    {

        \_logger = logger;

    }

    public IActionResult Index()

    {

        return View();

    }

    public IActionResult Privacy()

    {

        return View();

    }

    [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

    public IActionResult Error()

    {

        return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

    }

}

É bastante parecido com API com a diferença que herda somente de Controller

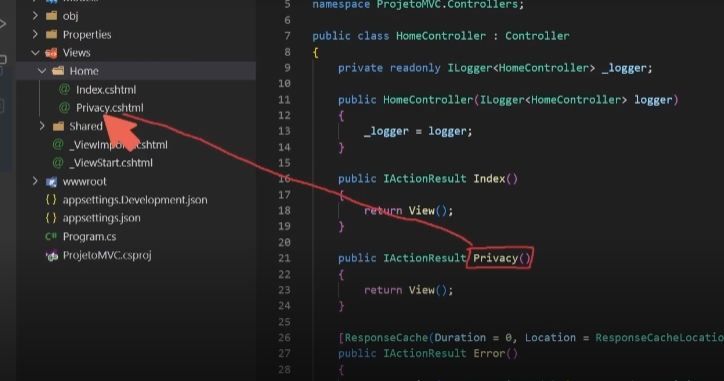
public class HomeController : Controller

Possui inúmeros métodos.

Ao analisar os métodos perceba que ele sempre retorna uma view return View(

Se for analisar a pasta view pode ser encontrada as duas pasta no caso Home, se eu tenho uma HomeCOntroller eu tb vou ter uma pasta Home na minha view.

Ou seja pode ser associado, toda a logica, todas as paginas da minha home controller ele vão ser redirecionados para pasta view da minha home



Se eu tenho um método Privacy o sistema tenta localizar o mesmo nome na views

Exemplo se eu navegar no endereço web

<https://localhost:7043/Home/Privacy>

ele vai fazer a busca da rota e carregar a pagina

/home vai procurar uma controller chamada Home

Localizou , é uma rota valida, então ele vai abrir o arquivo **Homecontroller.cs**

Ai tem /privacy, ele entrou na homecontroller, então ele vai procurar se tem um método chamado Privacy(), então ele vai entrar dentro do método , e o que método faz, ele retorna uma view, então eu vou procurar uma view , e como ele retorna uma view, exatamente com o mesmo nome do método Privacy,

Return view – o sistema vai procurar qual a controller que estou **HomeController ,** então ele vai procurar na pasta View uma pasta chamada Home, ele vai retornar uma view, com mesmo nome do método.

**Configurando o Entity Framework**

**A difença do API, é que agora temos uma tela mvc para fazer a interação**

**dotnet add package Microsoft.EntityFrameworkCore.sqlServer**

**dotnet add package Microsoft.EntityFrameworkCore.Design**

**em program.cs**

**importar**

using Microsoft.EntityFrameworkCore;

using MVC.Context;

**criar migrations**

**dotnet ef migrations add AdicionaTabelaContato**

para aplicar migrations no servidor sql/ criar banco de dados

dotnet ef database update

Criando models

Após criar o arquivo index.cshtml

Digitar

@model IEnumerable

INUMERABLE É UMA LISTA

Quando se utiliza @ misturando código c# com códigos html , @ só funciona no mvc

Configurando o método controller

As vezes o watch pode dar erro e não carregar a pagina corretamente, se o seu código não apresenta erros, pode parar o watch e executar novamente.

Aperta ctrl + c para ele parar

E roda novamente dotnet watch run

Todo imput precisa fechar com / exemplo

<input type = "submit" value="Criar" class="btn btr-primary"/>

Explicação do código index.cshtml

Começando sobre contatocontroller

 public IActionResult Index()

        {

            var contatos = \_context.Contatos.ToList(); // to list para ele transformar numa lista

            return View(contatos);

        }

Tudo que estiver faz referencia ao index, quando carrega o index, o entity framework está pegando todos os contatos e transformando em uma lista, fazendo um tolist()

Pegou todos os contatos e jogou na variável contatos, feito isso ele esta retornando a view porem está passando para view o argumento para vire o contatos

View(contatos);

Retorna essa view e retorna como parâmetro uma lista de contatos

Indo no arquivo index.cshtml, por sua vez estamos trabalhando com uma model

@model IEnumerable<MVC.Models.Contato>

Aquela lista de contatos que foi buscada do entity framework, ela está sendo carregada no index.

Com essa lista de contato que está sendo recebida ela pode ser feita um tratamento com código html

ViewData["Title"] = "Listagem de contatos";

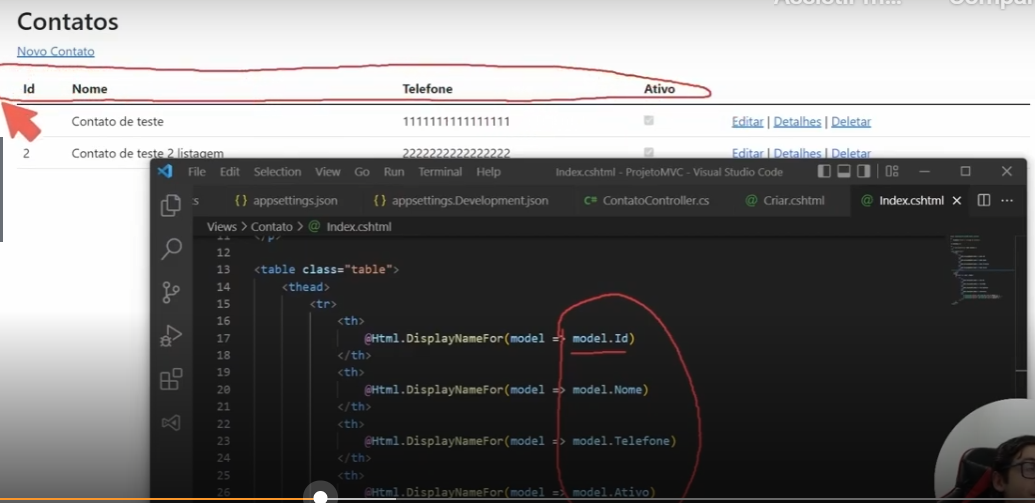
é o nome da aba

 <a asp-action="Criar"> Novo Contato</a>

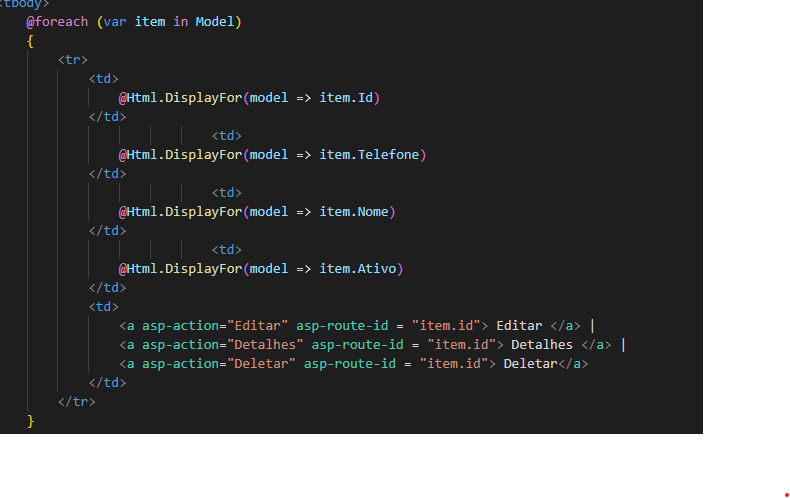
Ele é um hiper link que vai ser jogado para outra pagina

<table class="table">

Está sendo montado uma tabela, o print abaixo faz referência ao cabeçalho



Da onde vem id,telefone,etc, ele vem da models contato



O comando foreach, ou seja para cada item que foi recebido da model ( uma lista de contatos, que está sendo retornando da controller)

Para cada elemento que foi recebido da model repita o código

Se eu clicar em novo contato, ele vai chamar em asp-action criar

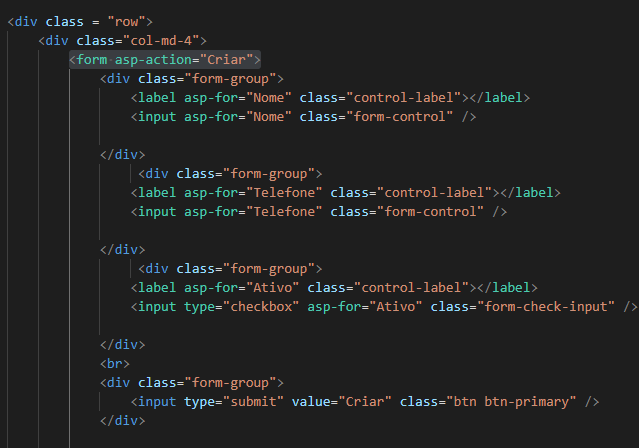
<a asp-action="Criar"> Novo Contato</a>

Vai locar no ContatoController o método Criar() que vai retornar uma view vazia pq é a primeira vez que está sendo carrega a pagina

Criar.cshtml

<form asp-action="Criar">

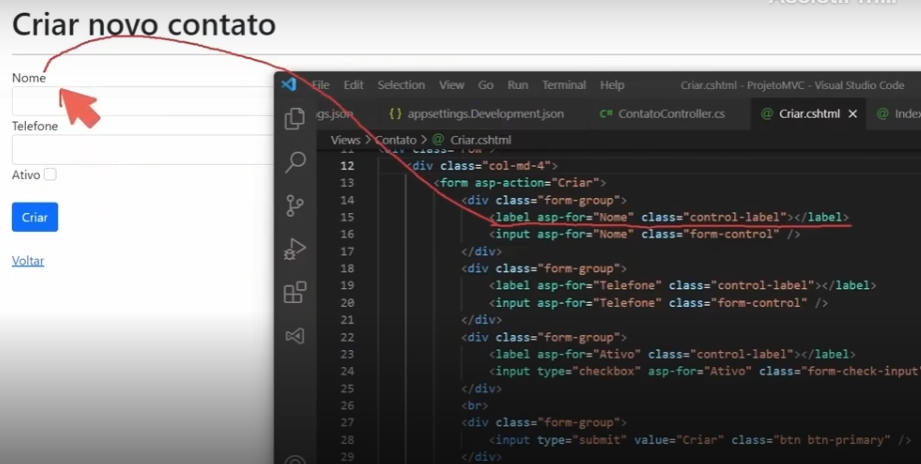
O form vai jogar no criar [httppost] porque ele vai fazer um post(salvar informação)



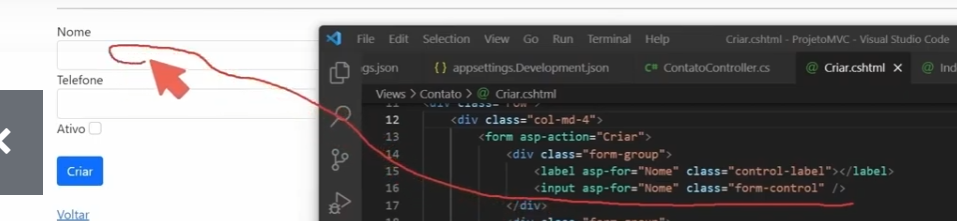
Tem varias div para poder separar os campos e fazer um layout mais bonito

É um boottrap que é um estilizador transformar o html em responsivel ,etc

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



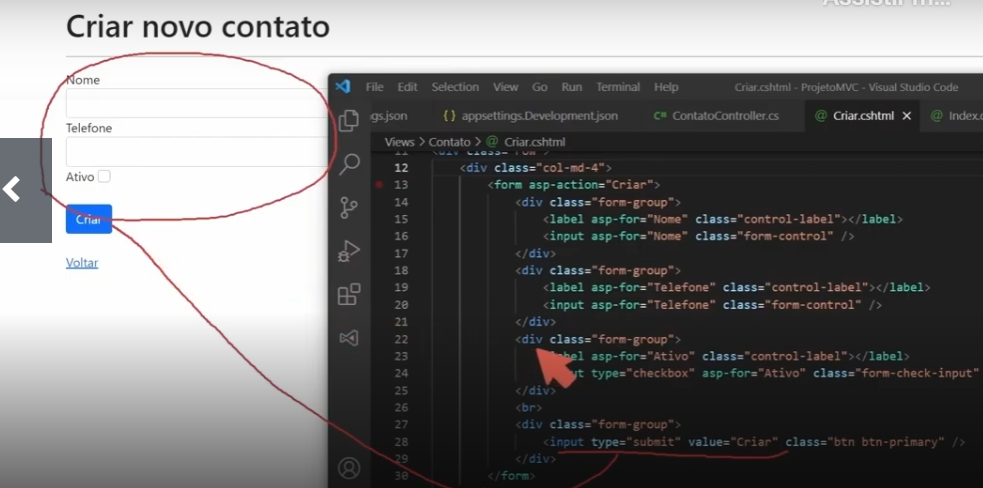
O label serve para identificar os campos



Imput é um controller é o próprio quadrado

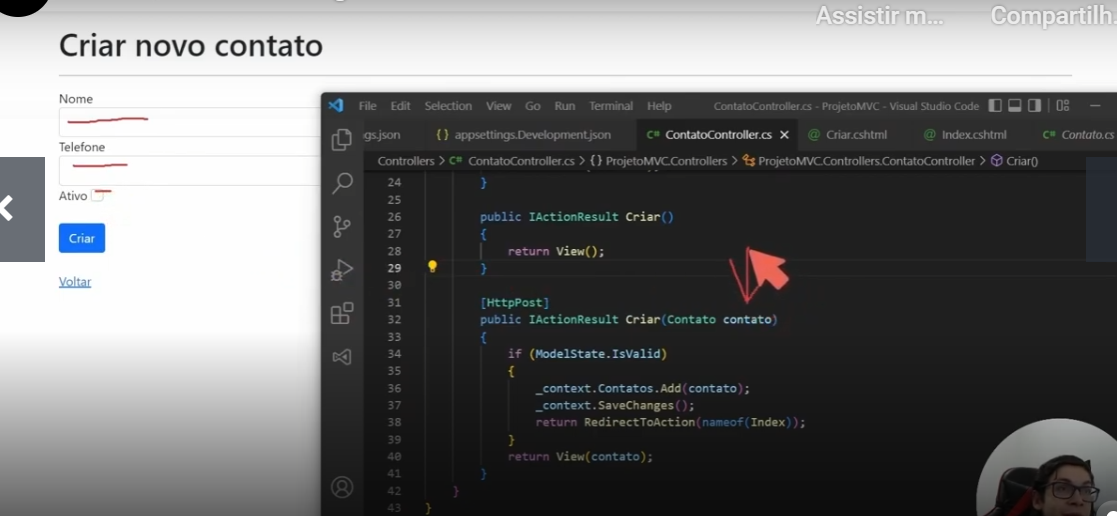
Imput checkbox é a caixa que pode deixar marcado ou desmarcado

Por sua vez tem o imput submit



Que ela vai pegar as informações e jogar no criar do post

Tudo que foi digitado nos campos nome e telefone, o Criar(Contato contato) vai receber como parâmetro



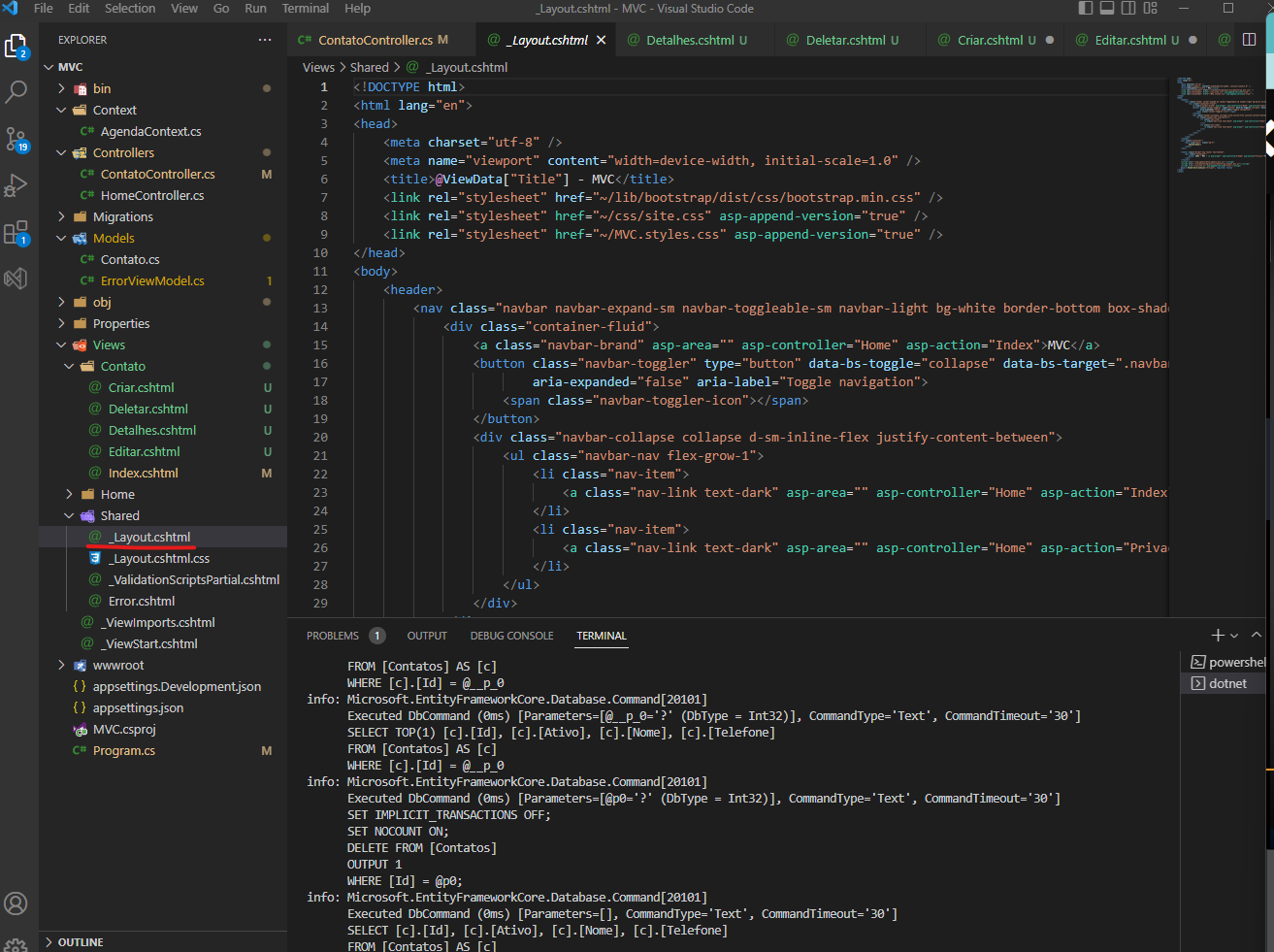
E ai faz toda logica do entityframework

E depois de salvar volta para o Index.

DisplayFor

Serve para exibir o próprio valor do campo

**Alterando menu**

****

**Ele é o layout da pagina,**

****

**Renderbody(); é tudo em comum que vai ser renderizado em na pagina.**