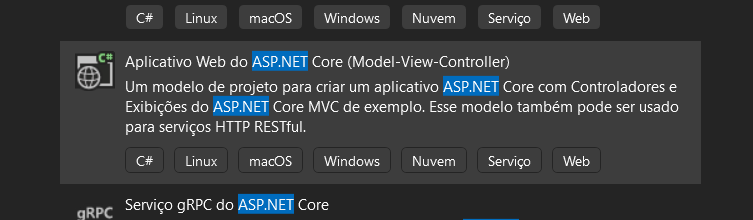
**ASP.NET Core MVC**

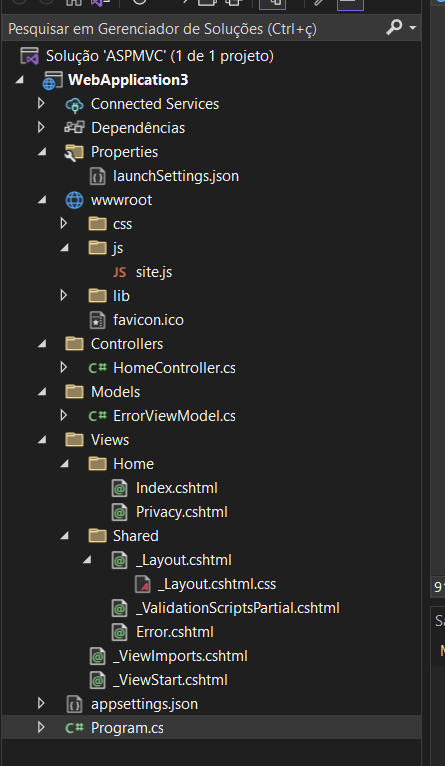
O ASP.NET Core é uma versão recente do ASP.NET, e o projeto ASP.NET Core MVC é um projeto ASP.NET básico que utiliza o padrão MVC.

Agora, descrevendo um pouco mais tecnicamente, o ASP.NET Core MVC é um novo framework de código aberto para a construção de aplicações conectadas pela internet. Ele permite o desenvolvimento e a execução de aplicações em Windows, Mac e Linux, e estas podem ser executadas no .NET Core ou no .NET Framework (versão anterior do .NET).

**Como criar o projeto:**

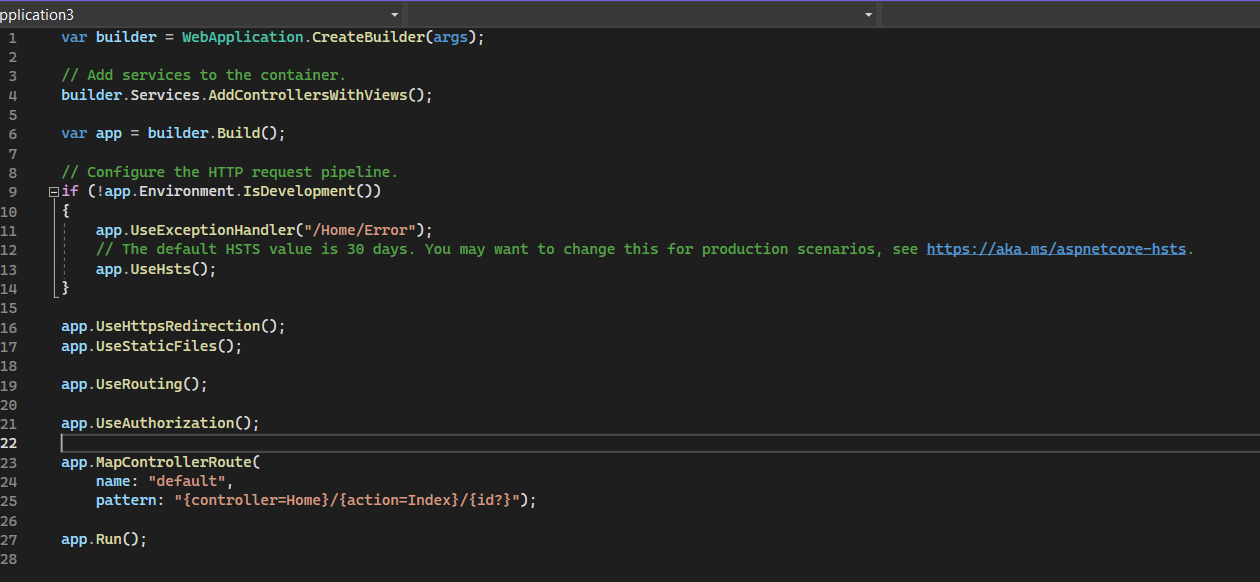
Após termos criado a solução em branco criamos um projeto ASP.NET Core MVC:





**Entendendo a estrutura do projeto:**

Analisando as pastas percebemos logo de cara a pasta dependências onde terá as dependências do projeto, em seguida temos a pasta **Properties** essa pasta ira conter as propriedades do projeto, basicamente ira conter as configurações Relacionadas ao IIS Express, em seguida temos a pasta **wwwroot** essa pasta contém os arquivos secundários ao View, como imagens, css, js e as bibliotecas do FrontEnd, em seguida temos as pastas principais da aplicação os MVC, em seguida temos o arquivo **appsetting.json** onde ira conter as configurações relacionadas ao funcionamento do projeto como a configuração do Entity Framework por exemplo, em seguida temos o arquivo **Program.cs**:



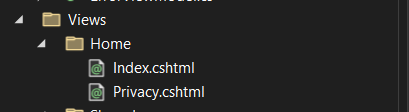
A classe Program.cs possui as configurações para rodar a aplicação, inicialmente o importante para analisarmos é o método **MapControllerRoute** que é responsável por mapear as rotas dos nossos controllers.

Basicamente como está agora ele está definindo que se não for especificado nenhum controller ele será o Home e caso não seja fornecido a action ela será Index.

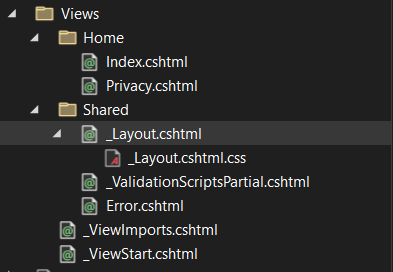
**Analisando os controllers:**

****

Analisando a classe HomeController percebemos que todo Controller deve ter o seu nome seguido da palavra Controller e deve extender a classe Controller.

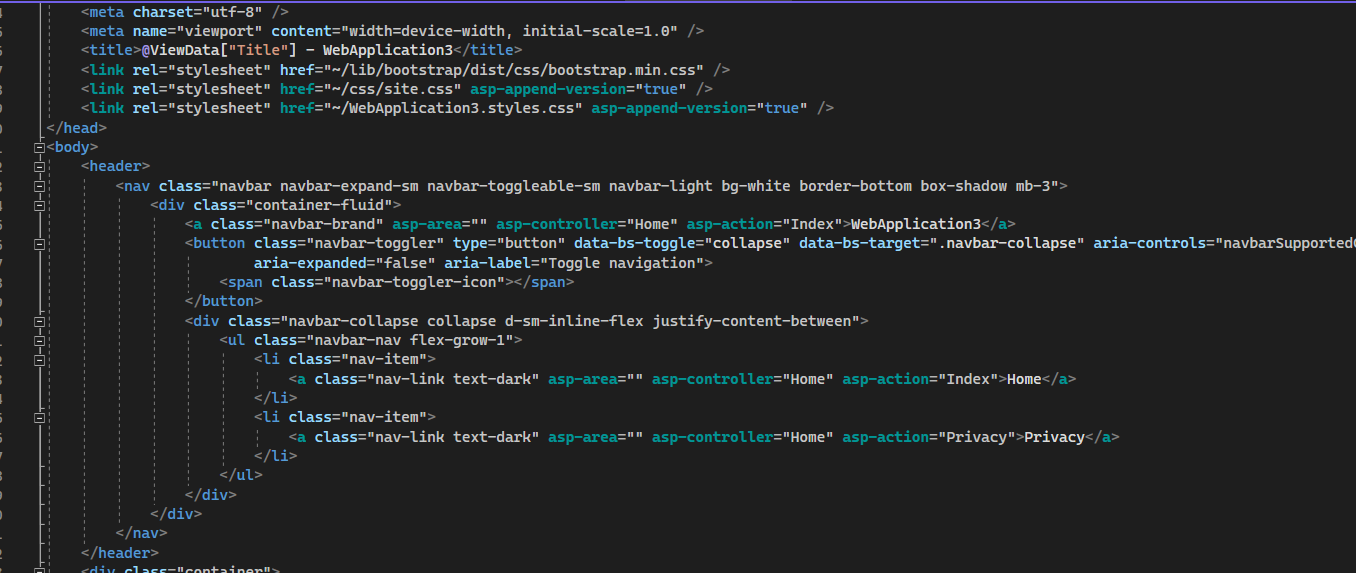
Analisando os métodos da classe percebemos que todos retornam um **IActionResult** e através disso retornam uma view, no caso dos métodos em Home podemos ver essas views na pasta view: 

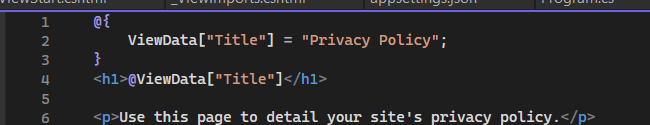
**Entendendo a view:**

****

Podemos analisar que todo controller terá a sua pasta em View com as suas páginas correspondentes, analisando percebemos que os arquivos com “\_” serão compartilhados entre as Views, dessa forma temos o \_Layout.cshtml que irá conter a estrutura html padrão, dessa forma não precisamos escrever toda a estrutura html em cada View, pois ela já é declarada em \_Layout.cshtml e será compartilhada entre as Views.

**Analisando a estrutura cshtml:**

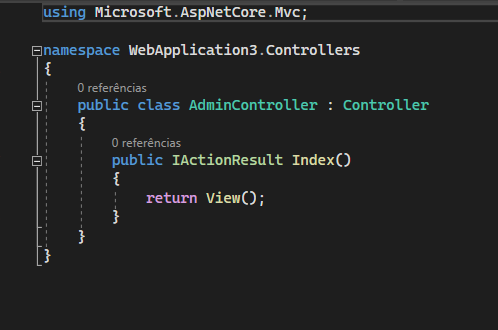
****

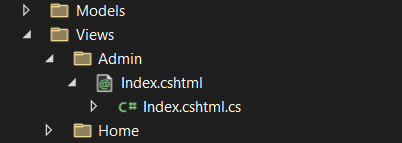
****

Podemos perceber que a estrutura cshtml é semelhante ao que é o jsp para o java, temos código html com código c#, podemos analisar isso através da anotação “@” e alguns métodos como “asp-area”, “asp-controller” e “asp-action”.

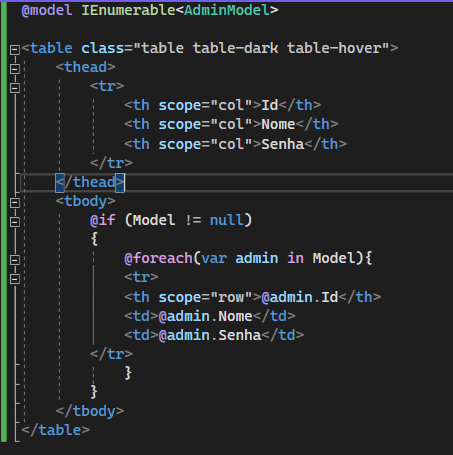
**Aplicando o contexto no controller:**

Vamos aplicar a entidade que criamos em uma view e pegar os dados, para isso vamos primeiro criar o controller de admin:

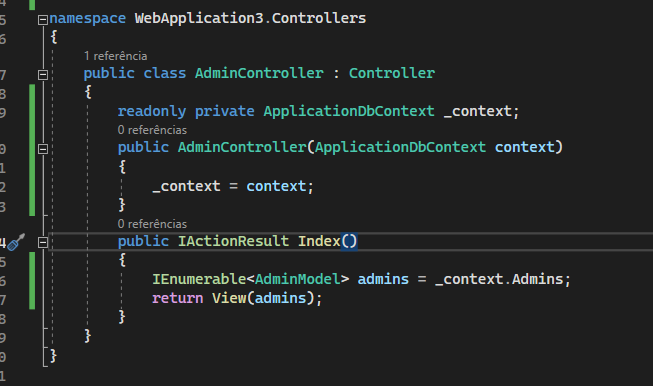




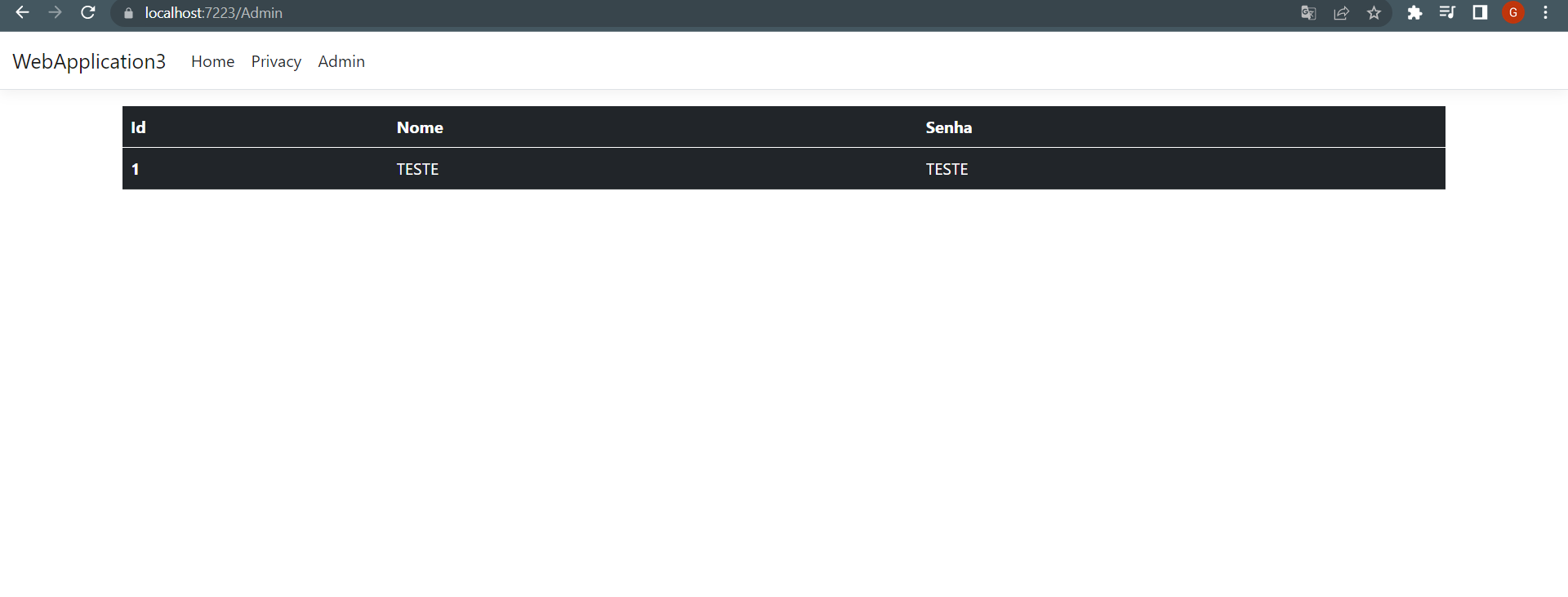
Agora vamos criar um layout de tabela para os dados:



Perceba que usamos um parâmetro IEnumerable<AdminModel> que deve ser passado pela classe controler:



Perceba usamos o constructor para pegar o contexto e armazenar em uma variável, e através dessa variável pegamos os “Admins” e passamos para a view.



**Entendendo a interação entre a view e o controller:**

É importante entender que todos os métodos adicionados no controller serão rotas do sistema, sendo assim para mandarmos dados do view para o controller devemos apenas acessar essas rotas:

public IActionResult Create(string nome, string senha)

{

// código para criar um novo administrador

}

Pelo cshtml:

<form asp-controller="Admin" asp-action="Create">

<div class="form-group">

<label for="Nome">Nome</label>

<input type="text" class="form-control" id="Nome" name="Nome">

</div>

<div class="form-group">

<label for="Senha">Senha</label>

<input type="password" class="form-control" id="Senha" name="Senha">

</div>

<button type="submit" class="btn btn-primary">Enviar</button>

</form>

Pelo JS:

Ajax:

$(document).ready(function() {

$("#sendButton").click(function() {

$.ajax({

url: "/Admin/Create",

type: "POST",

data: { nome: "João", senha: "123456" },

success: function(result) {

// código para atualizar a página com os dados retornados pelo servidor

}

});

});

});

Fetch:

fetch("/Admin/Create", {

method: "POST",

headers: {

"Content-Type": "application/json"

},

body: JSON.stringify({ nome: "João", senha: "123456" })

})

.then(response => response.json())

.then(data => {

// código para atualizar a página com os dados retornados pelo servidor

});