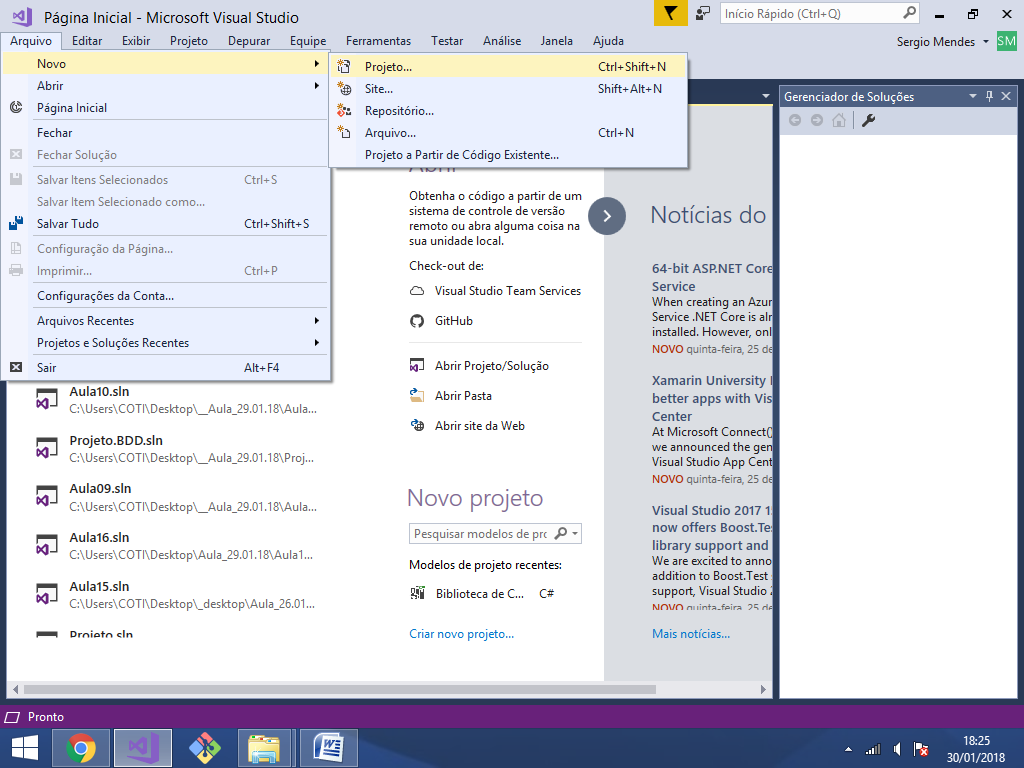
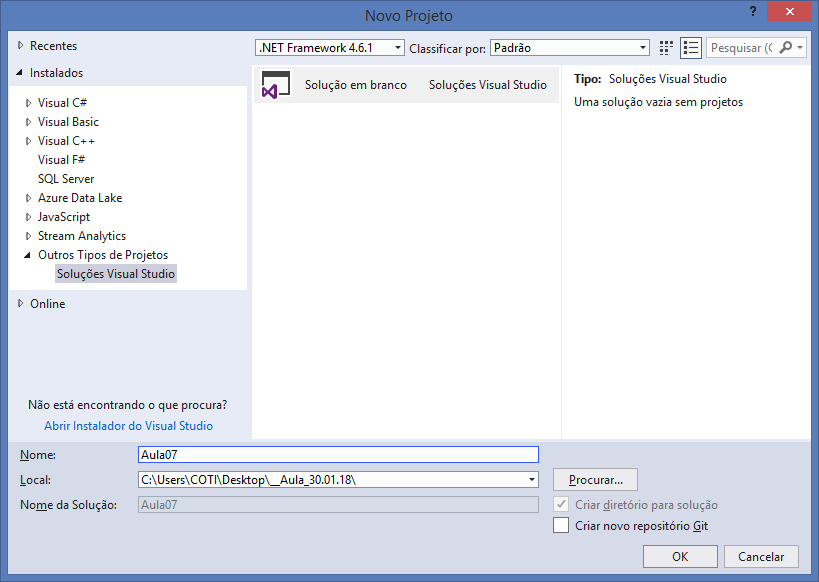
**Criando uma nova solution em branco:**

Pasta onde armazenamos os projetos



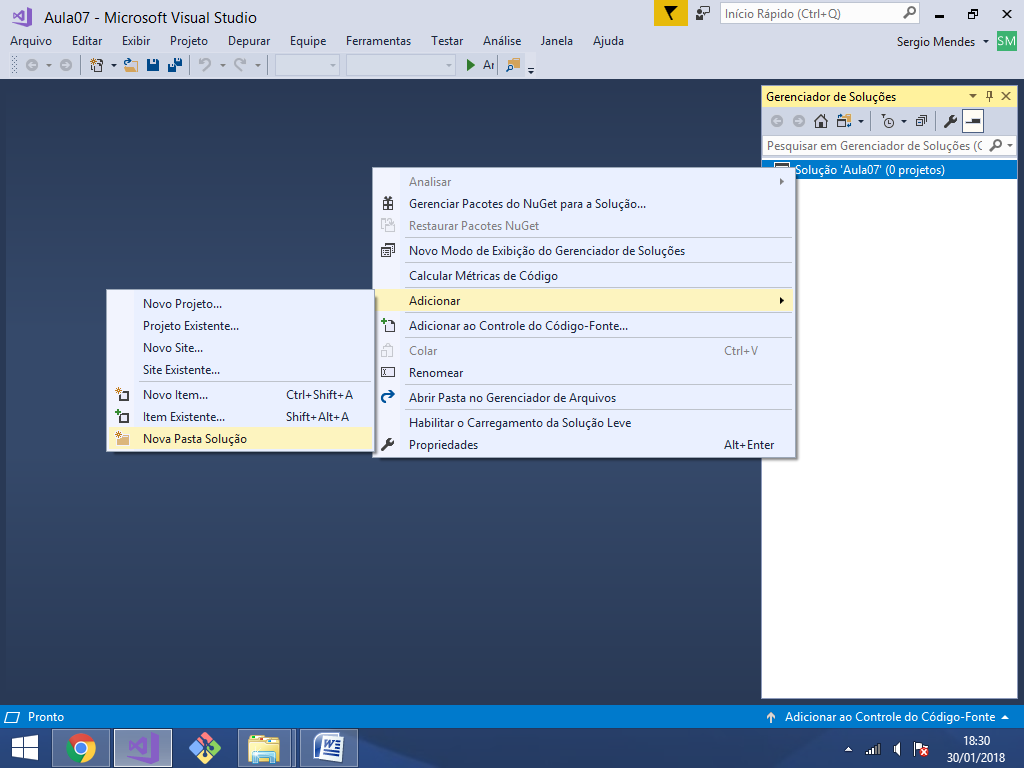


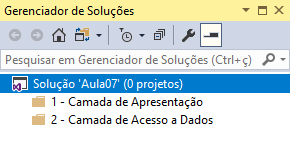
Arquitetura de projetos baseado em camadas

Tem como objetivo permitir ao desenvolvedor de sistemas criar aplicações que possuem uma estrutura de modulos e subdivisões.

Um sistema baseado em camadas irá conter varios projetos, cada um deles sendo responsavel por uma determinada "area" da aplicação (acesso a banco de dados, web, regras de negocio, etc...)

\*\* Organizando a solution em pastas:



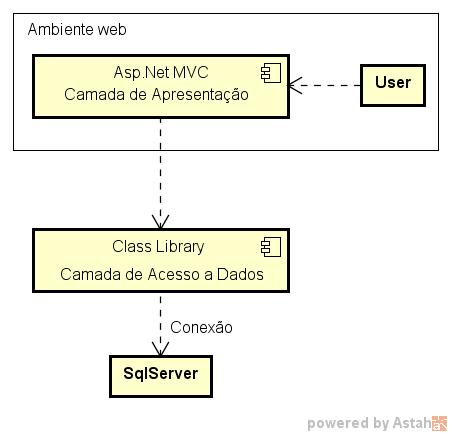


**1 - Camada de Apresentação**

Local onde estara contido o projeto **Asp.Net MVC** (web)

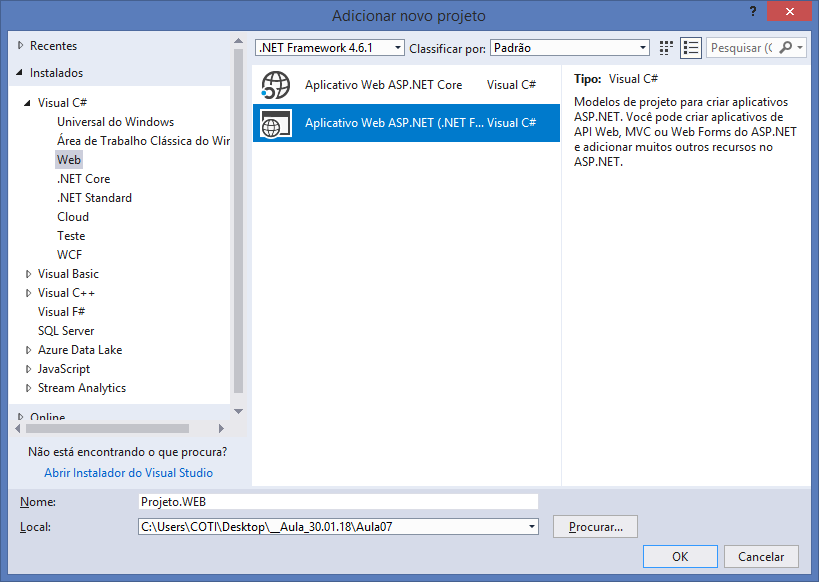
**2 - Camada de Acesso a Dados**

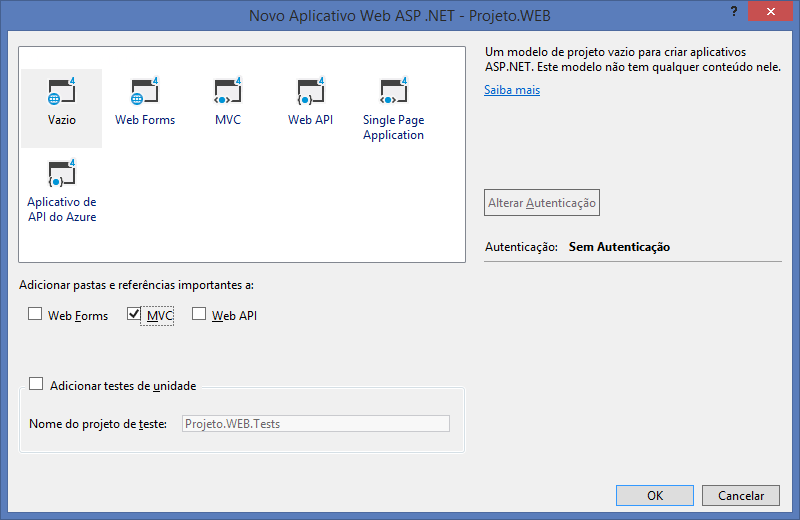
Local onde estara contido o projeto que ira realizar conexão e fazer as transações na base de dados. (Neste caso usando o **SqlClient**)



1 - Camada de Apresentação

Projeto Asp.Net MVC





Asp.Net

O Asp.Net é o conjunto de tecnologias da plataforma .NET voltadas para desenvolvimento de aplicações web. É composto de 3 principais tecnologias:

Asp.Net WebForms

Tecnologia que esta sendo cada vez mais descontinuada pela Microsoft. Trabalha com páginas de extensão .aspx, foi uma evolução do WindowsForms voltado para web.

Asp.Net MVC

Tecnologia mais alinhada com a web atual, que permite o desenvolvimento de aplicações web baseado em 3 camadas: **Model, View e Controller**

Asp.Net WebApi

Tecnologia que permite o desenvolvimento de aplicações web baseadas em serviços **RESTful**, ou seja aplicações que compartilham suas funcionalidades com outros sistemas web, mobile, etc..

MVC - Model, View e Controller

O MVC é um padrão para desenvolvimento de projetos web que tem como objetivo prover uma organização nas seguintes camadas:

View

Camada do MVC que representa as páginas web do projeto, ou seja, conteudo **HTML, CSS e Javascript**

* **HTML** - Linguagem para criação de páginas web (TAGs)
* **CSS** - Linguagem para criação de folhas de estilo que irão formatar e alterar a aparencia da página HTML.
* **JavaScript** - Linguagem de scripts utilizada para criar eventos e comportamentos diretamente no navegador.

Model

Camada responsavel pelos dados de entrada e saida do projeto, formularios de cadastro, grids de consultas, etc... É a camada model que coleta os dados de uma view e os leva ate o controle e vice-versa

Controller

Camada que representa o "backend" do projeto MVC, pois recebe todas as requisições enviadas pelas views bem como os dados das Models, faz o processamento e retorna os resultados.

--------------------------------------------

Asp.Net MVC

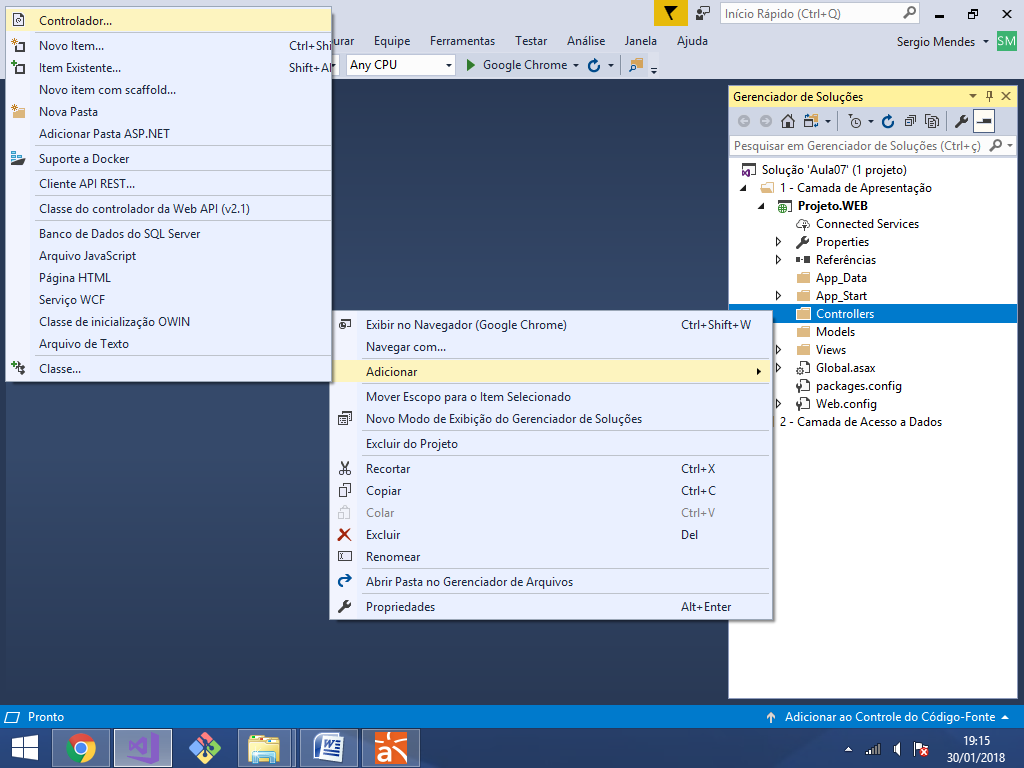
Tecnologia para desenvolvimento de aplicações web em .NET   
que utiliza o padrão MVC (Model, View e Controller).

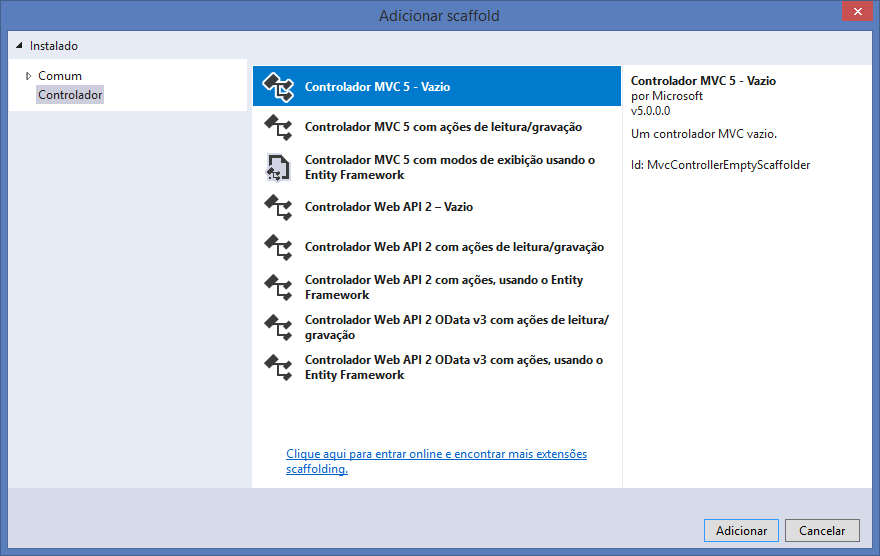
**Por onde começar?**

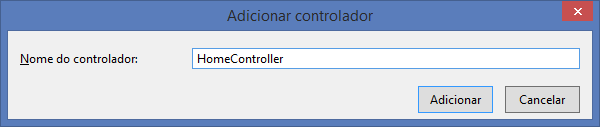
Criando a(s) classe(s) de controle

Um controller em Asp.Net MVC é uma classe que é capaz de gerar "rotas" no projeto web. Essas rotas servem para abrir páginas, executar ações, etc...

Por padrão, todo projeto Asp.Net MVC terá uma classe de controle default denominada: HomeController







**ActionResult**

É utilizado nas classes de controle para criar metodos que irao gerar algum tipo de rota (URL) no projeto, seja esta URL **GET** ou **POST**

HTTP GET

É um tipo de endereço web executado somente atraves de uma URL, por exemplo: www.meuprojeto.com.br/consultarclientes

Caso uma requisição HTTP GET precise de dados, estes serão enviados tambem pela propria URL.

Exemplo:

www.meuprojeto.com.br/consultarclientes?estado=RJ

HTTP POST

É um tipo de endereço web executado atraves de um formulario, pois alem da URL é necessario enviar um corpo de dados

Em requisições POST os dados enviados para o servidor não ficam expostos na URL do navegador.

Exemplo:

www.meuprojeto.com.br/cadastrarcliente

*Body Request*

Nome=Joao Pedro

Email=joao@gmail.com

---------------------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace Projeto.WEB.Controllers

{

public class HomeController : Controller

{

// GET: Home/Index

public ActionResult Index()

{

return View();

}

}

}

Em MVC, as rotas ou URLs de páginas e requisições GET ou POST   
sempre obedecem ao seguinte padrão:

/Home/Index

[Controller] [Action]

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace Projeto.WEB.Controllers

{

public class HomeController : Controller

{

// GET: Home/Index

public ActionResult Index()

{

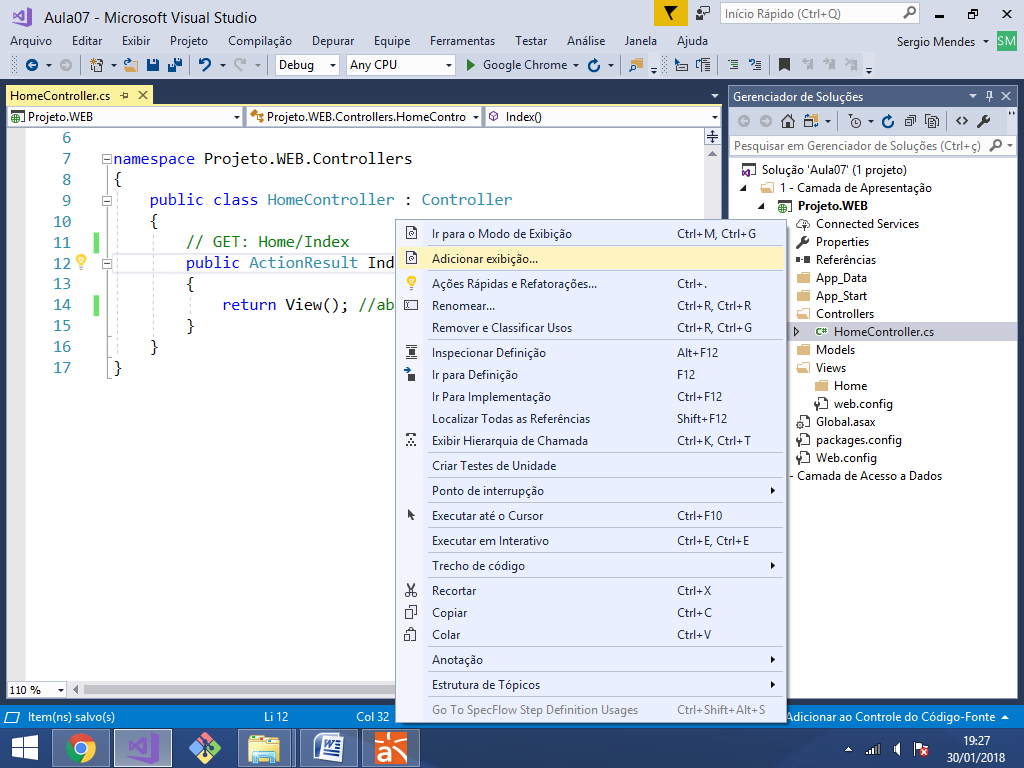
return View(); //abrir página..

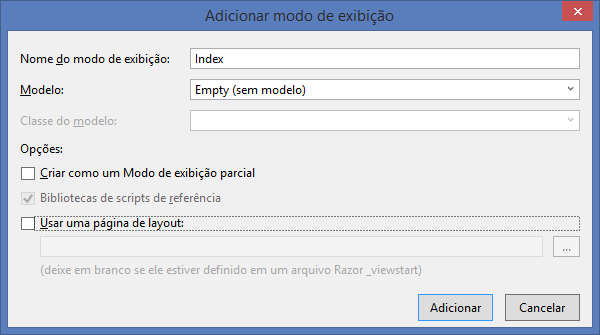
}

}

}

Criando a página:





<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>COTI Informatica</title>

</head>

<body>

<div>

<h1>Projeto Controle de Clientes</h1>

Turma de C# WebDeveloper Noite - COTI Informatica

<hr/>

Selecione a ação desejada:

<ul>

<li> <a href="/Cliente/Cadastro">Cadastrar Clientes</a> </li>

<li> <a href="/Cliente/Consulta">Consultar Clientes</a> </li>

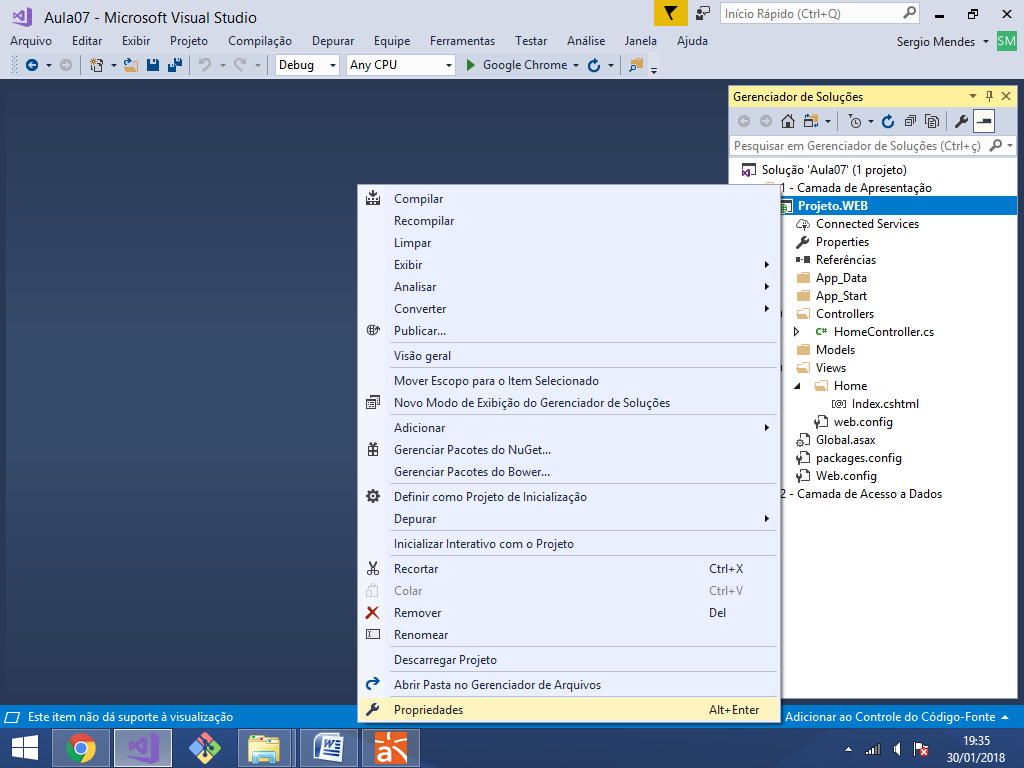
</ul>

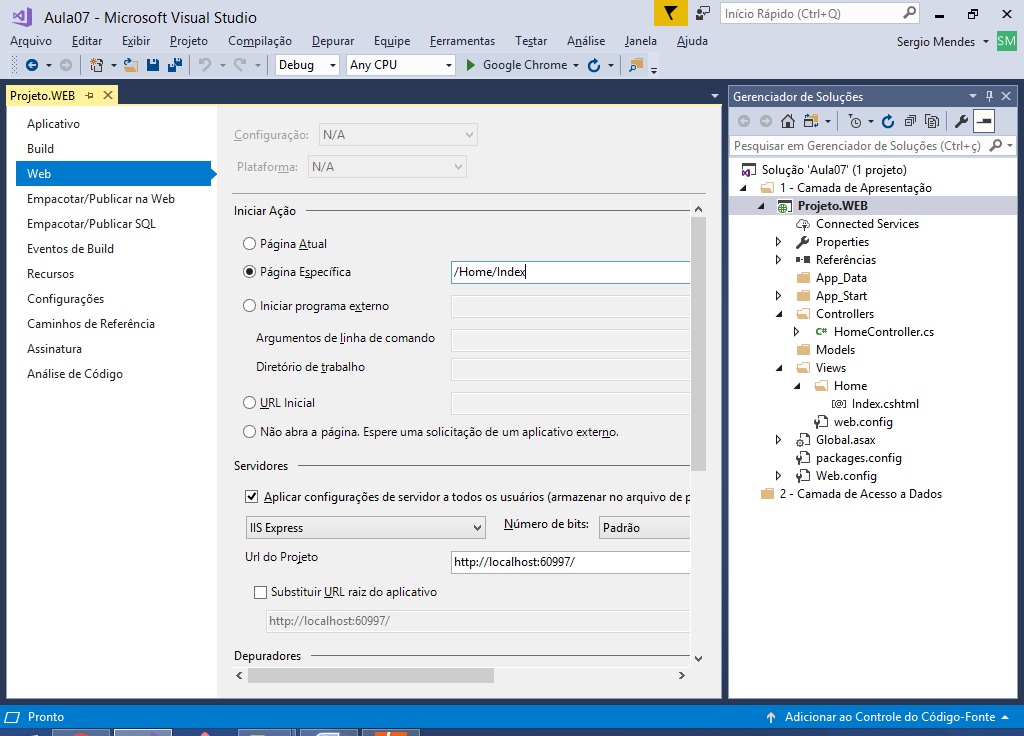
</div>

</body>

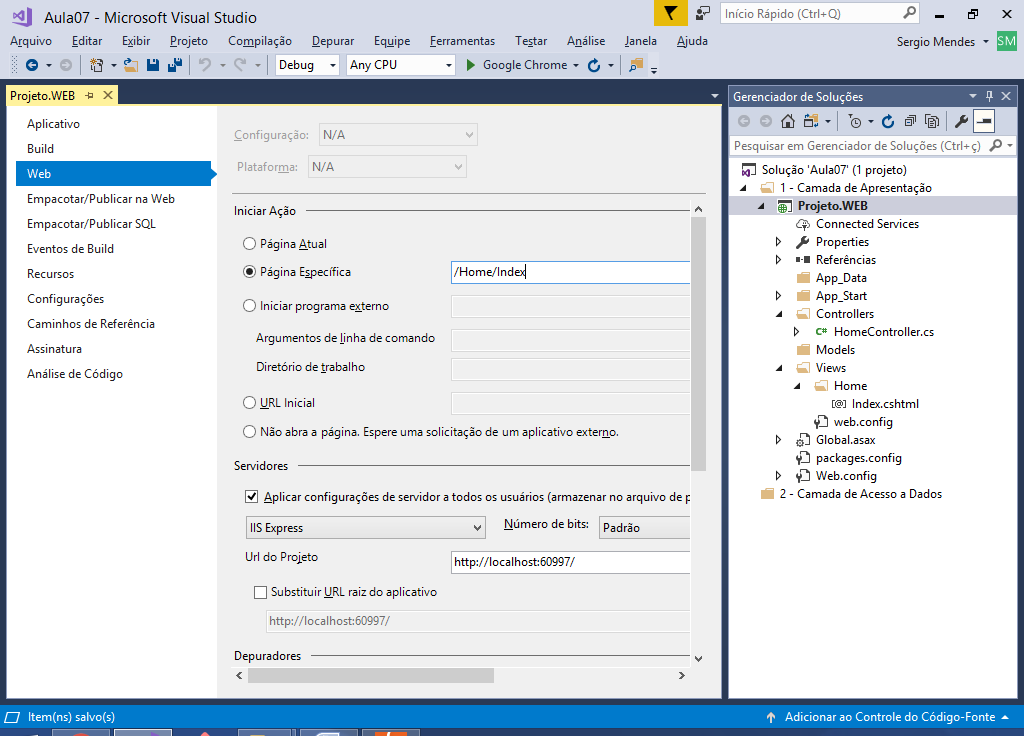
</html>

**Configurando a URL /Home/Index como   
sendo a rota inicial do projeto:**





**Executando: (F5)**

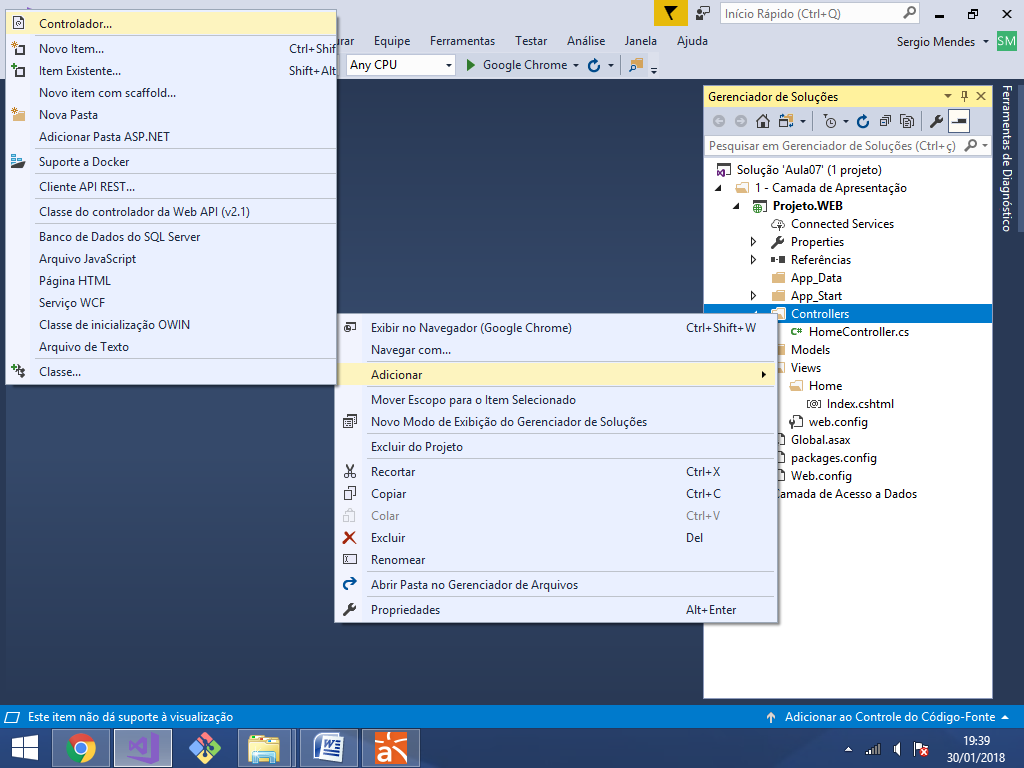


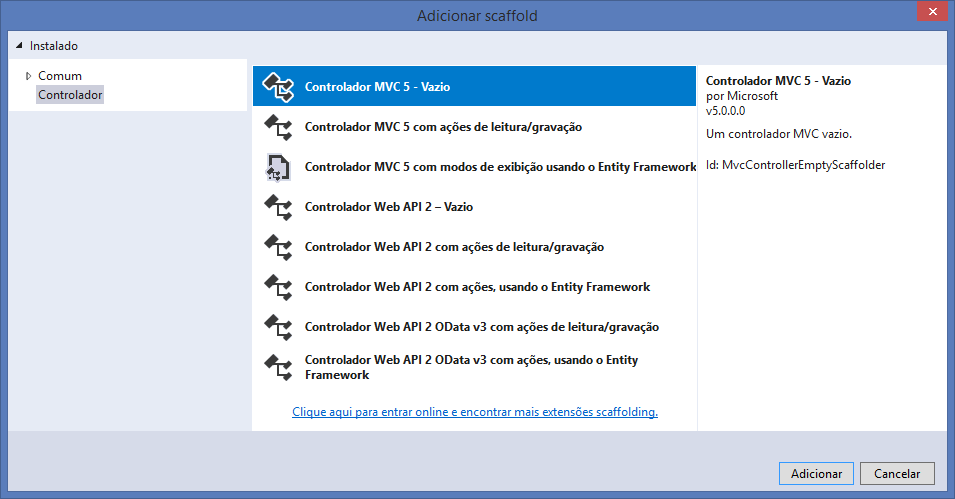
http://localhost:60997/Home/Index

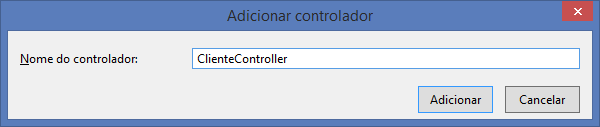


Criando uma classe de controle para Clientes

/ClienteController.cs







using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace Projeto.WEB.Controllers

{

public class ClienteController : Controller

{

// GET: Cliente/Cadastro

public ActionResult Cadastro()

{

return View();

}

// GET: Cliente/Consulta

public ActionResult Consulta()

{

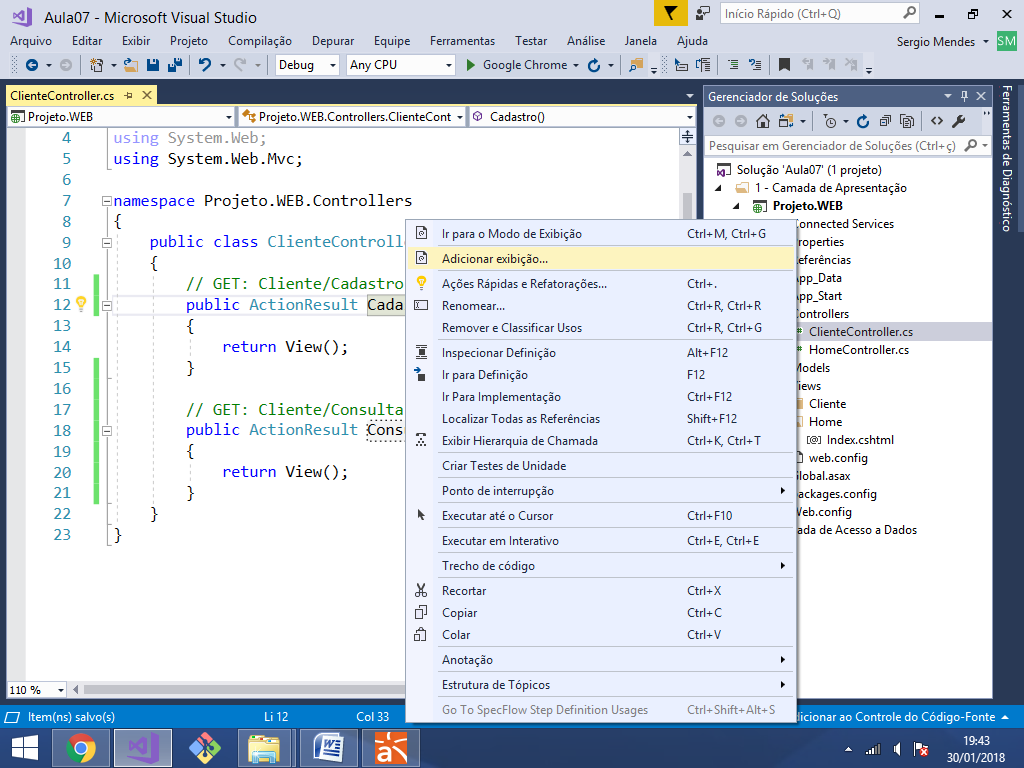
return View();

}

}

}

Criando as páginas:



<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>COTI Informática</title>

</head>

<body>

<div>

<h1>Cadastro de Clientes</h1>

<a href="/Home/Index">Página inicial</a>

<hr/>

<!-- Formulário -->

<form>

<!-- Campo nome -->

<label>Nome do Cliente:</label> <br/>

<input type="text" name="Nome" placeholder="Digite aqui"/>

<br/><br/>

<!-- Campo nome -->

<label>Email do Cliente:</label> <br />

<input type="text" name="Email" placeholder="Digite aqui" />

<br /><br />

<input type="submit" value="Cadastrar Cliente"/>

</form>

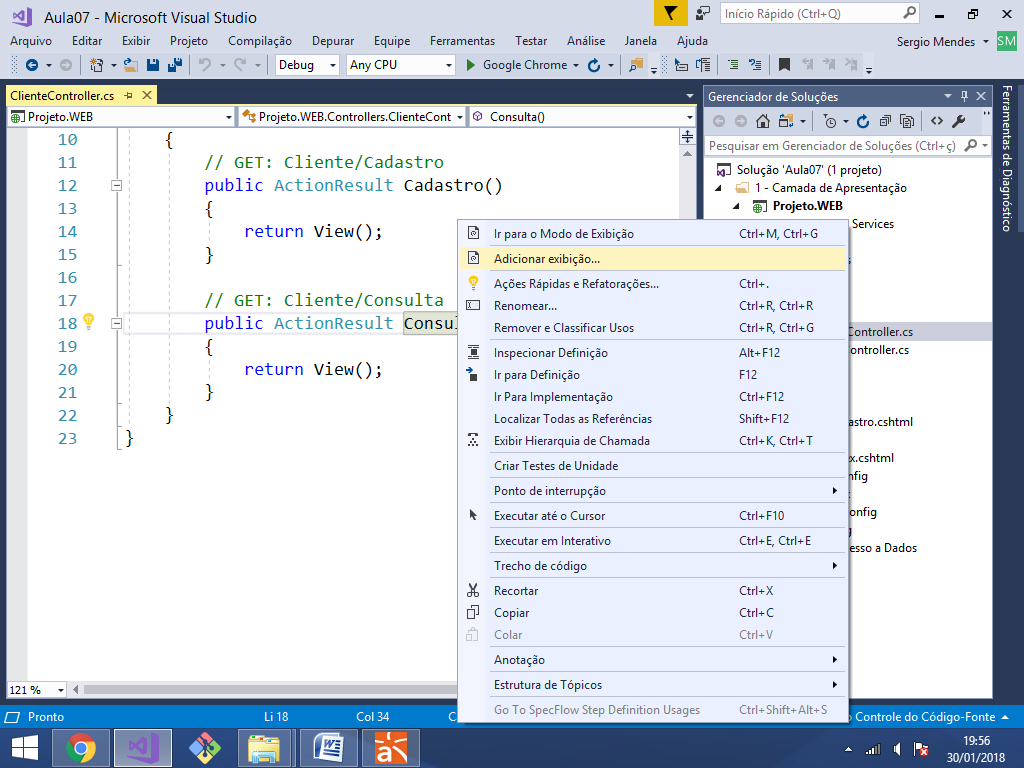
</div>

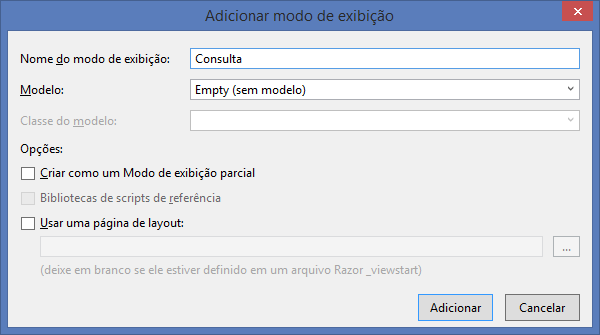
</body>

</html>



Criando a página de consulta:





<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>COTI Informática</title>

</head>

<body>

<div>

<h1>Consulta de Clientes</h1>

<a href="/Home/Index">Página inicial</a>

<hr />

</div>

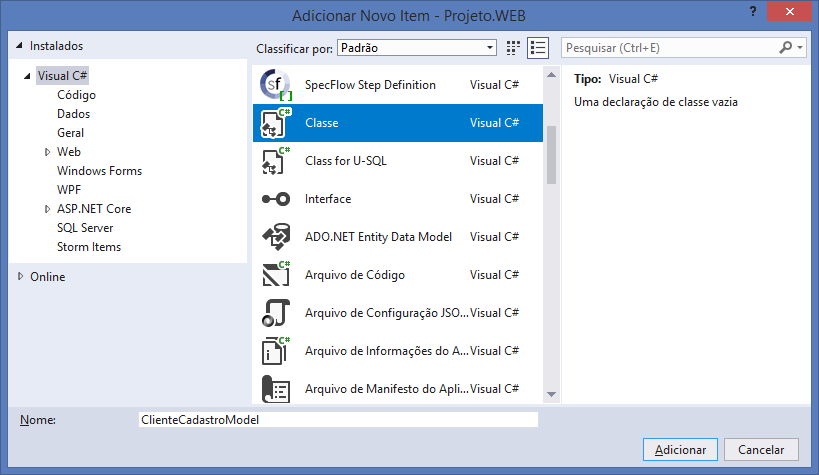
</body>

</html>

Classes de modelo (Models)

São classes em MVC responsaveis por representar os dados   
de entrada ou de saida entre as views e os controllers.

**Exemplo**: Criar uma classe de modelo para os   
dados do formulario de cadastro de clientes.



using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Projeto.WEB.Models

{

public class ClienteCadastroModel

{

public string Nome { get; set; }

public string Email { get; set; }

}

}

Validando os dados da model:

**System.ComponentModel.DataAnnotations**

Namespace do Asp.Net que contem anotações para validação e tratamento de campos da classe de modelo.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.ComponentModel.DataAnnotations; //mapeamentos..

namespace Projeto.WEB.Models

{

public class ClienteCadastroModel

{

[MinLength(6, ErrorMessage = "Por favor,

informe no mínimo {1} caracteres.")]

[MaxLength(50, ErrorMessage = "Por favor,

informe no máximo {1} caracteres.")]

[Required(ErrorMessage = "Por favor, informe o nome do cliente.")]

public string Nome { get; set; }

[EmailAddress(ErrorMessage = "Por favor, informe

um endereço de email válido.")]

[Required(ErrorMessage = "Por favor, informe o email do cliente.")]

public string Email { get; set; }

}

}

-------------------------------------------

@Razor

Sintaxe baseada em C# e utilizado em projetos MVC. Atraves do Razor podemos incluir pequenos trechos de programação nas páginas (views) com o intuito de torna-las mais dinaicas ou de facilitar a sua comunicação com as classes Model e os Controllers

<!-- Classe de modelo desta página -->

@model Projeto.WEB.Models.ClienteCadastroModel

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>COTI Informática</title>

</head>

<body>

<div>

<h1>Cadastro de Clientes</h1>

<a href="/Home/Index">Página inicial</a>

<hr/>

<!-- Formulário -->

@using (Html.BeginForm())

{

<!-- Campo nome -->

<label>Nome do Cliente:</label> <br/>

@Html.TextBoxFor(model => model.Nome,

new { @placeholder = "Digite aqui" })

<br/><br/>

<!-- Campo nome -->

<label>Email do Cliente:</label> <br />

@Html.TextBoxFor(model => model.Email,

new { @placeholder = "Digite aqui" })

<br /><br />

<input type="submit" value="Cadastrar Cliente"/>

}

</div>

</body>

</html>



Criando uma requisição POST para queo formulario de cadastro   
de cliente envie os dados da model para o controller:

**Html.BeginForm("Cadastro", "Cliente", FormMethod.Post)**

[Método ActionResult] [Controller] [Tipo da Requisição]

<!-- Classe de modelo desta página -->

@model Projeto.WEB.Models.ClienteCadastroModel

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>COTI Informática</title>

</head>

<body>

<div>

<h1>Cadastro de Clientes</h1>

<a href="/Home/Index">Página inicial</a>

<hr/>

<!-- Formulário -->

**@using (Html.BeginForm("Cadastro", "Cliente", FormMethod.Post))**

{

<!-- Campo nome -->

<label>Nome do Cliente:</label> <br/>

@Html.TextBoxFor(model => model.Nome,

new { @placeholder = "Digite aqui" })

<br/><br/>

<!-- Campo nome -->

<label>Email do Cliente:</label> <br />

@Html.TextBoxFor(model => model.Email,

new { @placeholder = "Digite aqui" })

<br /><br />

<input type="submit" value="Cadastrar Cliente"/>

}

</div>

</body>

</html>

**No controller:**

Criando o método para receber   
a requisição HTTP POST do formulario:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Projeto.WEB.Models; //classes de modelo..

namespace Projeto.WEB.Controllers

{

public class ClienteController : Controller

{

// GET: Cliente/Cadastro

public ActionResult Cadastro()

{

return View();

}

**// POST: Cliente/Cadastro**

**[HttpPost]**

**public ActionResult Cadastro(ClienteCadastroModel model)**

**{**

**return View();**

**}**

// GET: Cliente/Consulta

public ActionResult Consulta()

{

return View();

}

}

}

**Exibir as mensagens de erro de validação:**

<!-- Classe de modelo desta página -->

@model Projeto.WEB.Models.ClienteCadastroModel

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>COTI Informática</title>

</head>

<body>

<div>

<h1>Cadastro de Clientes</h1>

<a href="/Home/Index">Página inicial</a>

<hr/>

<!-- Formulário -->

@using (Html.BeginForm("Cadastro", "Cliente", FormMethod.Post))

{

<!-- Campo nome -->

<label>Nome do Cliente:</label> <br/>

@Html.TextBoxFor(model => model.Nome,

new { @placeholder = "Digite aqui" })

**@Html.ValidationMessageFor(model => model.Nome)**

<br/><br/>

<!-- Campo nome -->

<label>Email do Cliente:</label> <br />

@Html.TextBoxFor(model => model.Email,

new { @placeholder = "Digite aqui" })

**@Html.ValidationMessageFor(model => model.Email)**

<br /><br />

<input type="submit" value="Cadastrar Cliente"/>

}

</div>

</body>

</html>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Projeto.WEB.Models; //classes de modelo..

namespace Projeto.WEB.Controllers

{

public class ClienteController : Controller

{

// GET: Cliente/Cadastro

public ActionResult Cadastro()

{

return View();

}

// POST: Cliente/Cadastro

[HttpPost]

public ActionResult Cadastro(ClienteCadastroModel model)

{

//verificar se os dados obtidos pela classe model

//estão corretos (passaram nas validações?)

**if(ModelState.IsValid)**

{

}

return View();

}

// GET: Cliente/Consulta

public ActionResult Consulta()

{

return View();

}

}

}

**Testando:**



Exibindo mensagem na página:

**ViewBag**

Componente do Asp.Net MVC que pode enviar dados do Controller para   
a View, como mensagens (string) e ate mesmo objetos.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Projeto.WEB.Models; //classes de modelo..

namespace Projeto.WEB.Controllers

{

public class ClienteController : Controller

{

// GET: Cliente/Cadastro

public ActionResult Cadastro()

{

return View();

}

// POST: Cliente/Cadastro

[HttpPost]

public ActionResult Cadastro(ClienteCadastroModel model)

{

//verificar se os dados obtidos pela classe model

//estão corretos (passaram nas validações?)

if(ModelState.IsValid)

{

//criando uma mensagem que será exibida na página..

**ViewBag.Mensagem = "Cliente cadastrado com sucesso.";**

//limpar os campos do formulário..

ModelState.Clear();

}

return View();

}

// GET: Cliente/Consulta

public ActionResult Consulta()

{

return View();

}

}

}

**Na view:**

Exibindo a mensagem

<!-- Classe de modelo desta página -->

@model Projeto.WEB.Models.ClienteCadastroModel

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>COTI Informática</title>

</head>

<body>

<div>

<h1>Cadastro de Clientes</h1>

<a href="/Home/Index">Página inicial</a>

<hr/>

<!-- Formulário -->

@using (Html.BeginForm("Cadastro", "Cliente", FormMethod.Post))

{

<!-- Campo nome -->

<label>Nome do Cliente:</label> <br/>

@Html.TextBoxFor(model => model.Nome,

new { @placeholder = "Digite aqui" })

@Html.ValidationMessageFor(model => model.Nome)

<br/><br/>

<!-- Campo nome -->

<label>Email do Cliente:</label> <br />

@Html.TextBoxFor(model => model.Email,

new { @placeholder = "Digite aqui" })

@Html.ValidationMessageFor(model => model.Email)

<br /><br />

<input type="submit" value="Cadastrar Cliente"/>

<br/>

<br/>

**<strong>@ViewBag.Mensagem</strong>**

}

</div>

</body>

</html>

Executando:

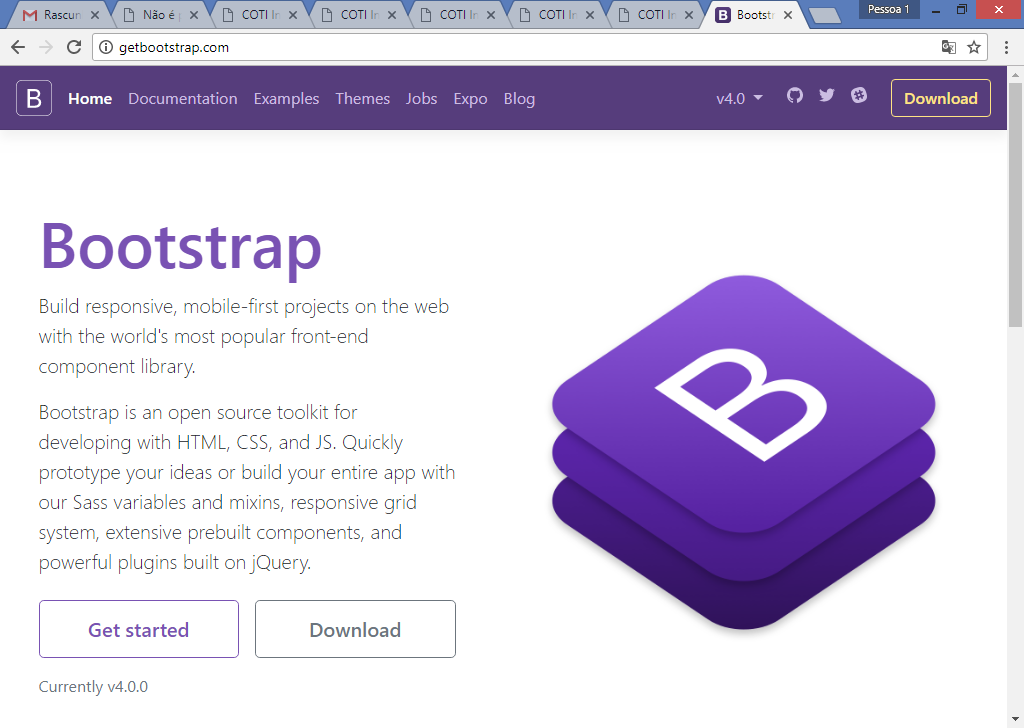


Folhas de estilo CSS - Cascading Style Sheet

Linguagem utilizada para formtação de páginas HTML

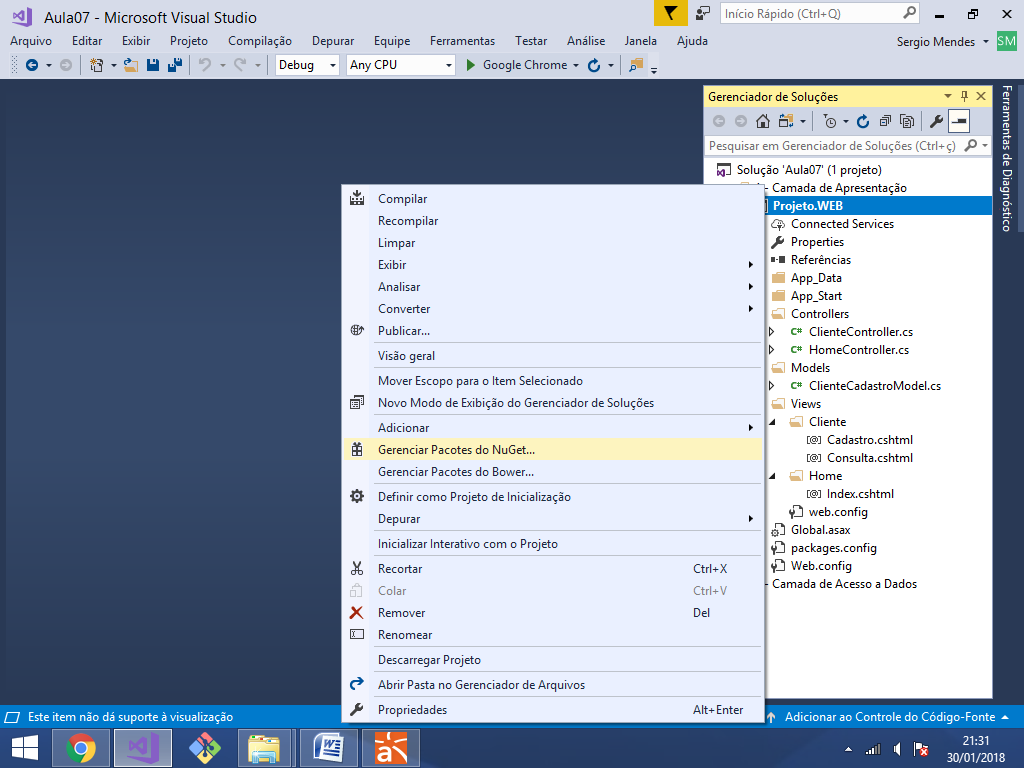
Existem varias bibliotecas ja prontas com diversos modelos de folhas de estilo CSS, dentre estas a mais conhecida e utilizada é o bootstrap

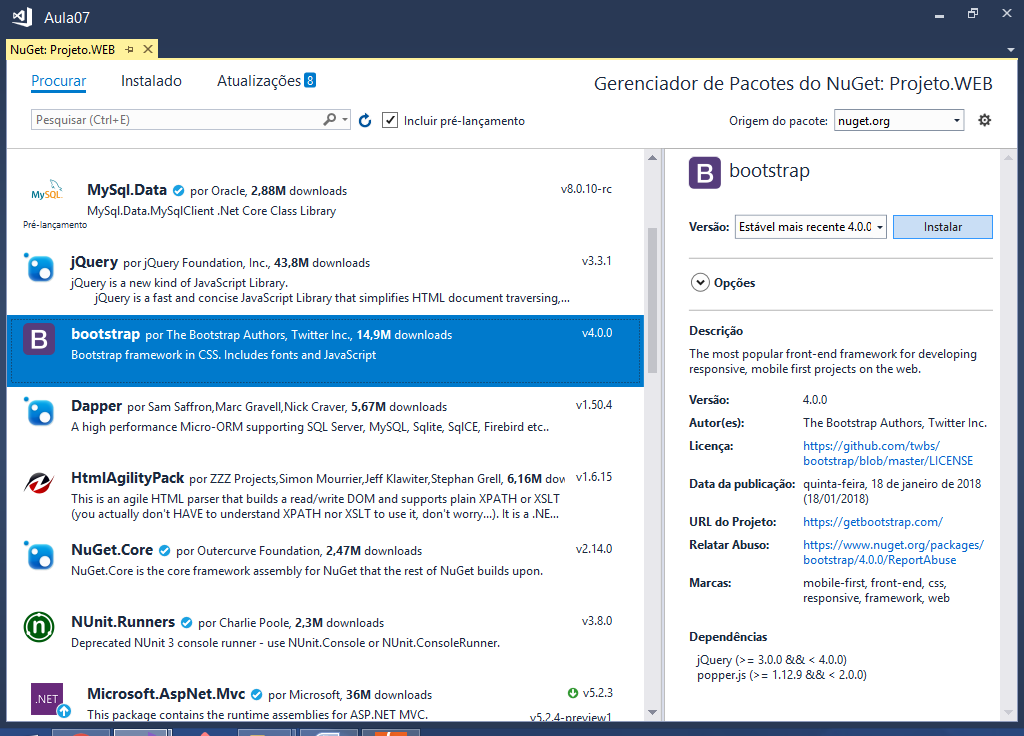
http://getbootstrap.com/

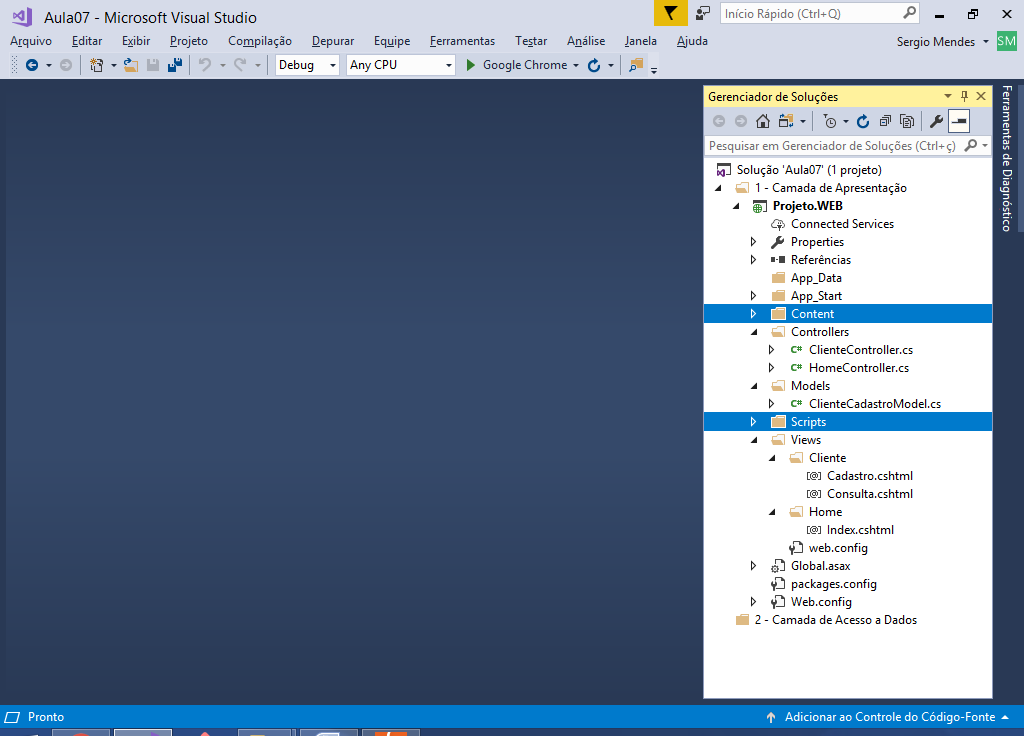


Instalando os arquivos do   
bootstrap atraves do VisualStudio

Gerenciador de pacotes do NuGet







**Utilizando o bootstrap nas páginas:**

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>COTI Informatica</title>

**<link href="~/Content/bootstrap.min.css" rel="stylesheet" />**

</head>

<body class="container">

<div>

<h1>Projeto Controle de Clientes</h1>

Turma de C# WebDeveloper Noite - COTI Informatica

<hr/>

Selecione a ação desejada:

<ul>

<li> <a href="/Cliente/Cadastro">Cadastrar Clientes</a> </li>

<li> <a href="/Cliente/Consulta">Consultar Clientes</a> </li>

</ul>

</div>

</body>

</html>





Continua...