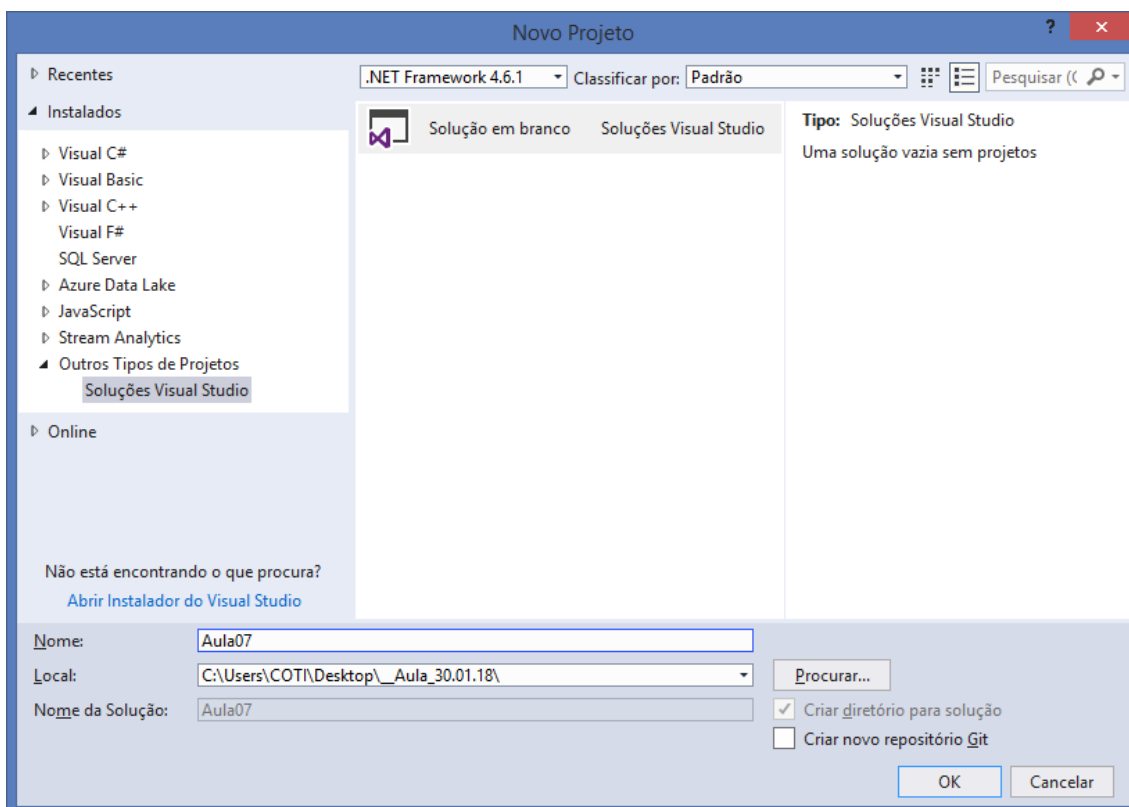
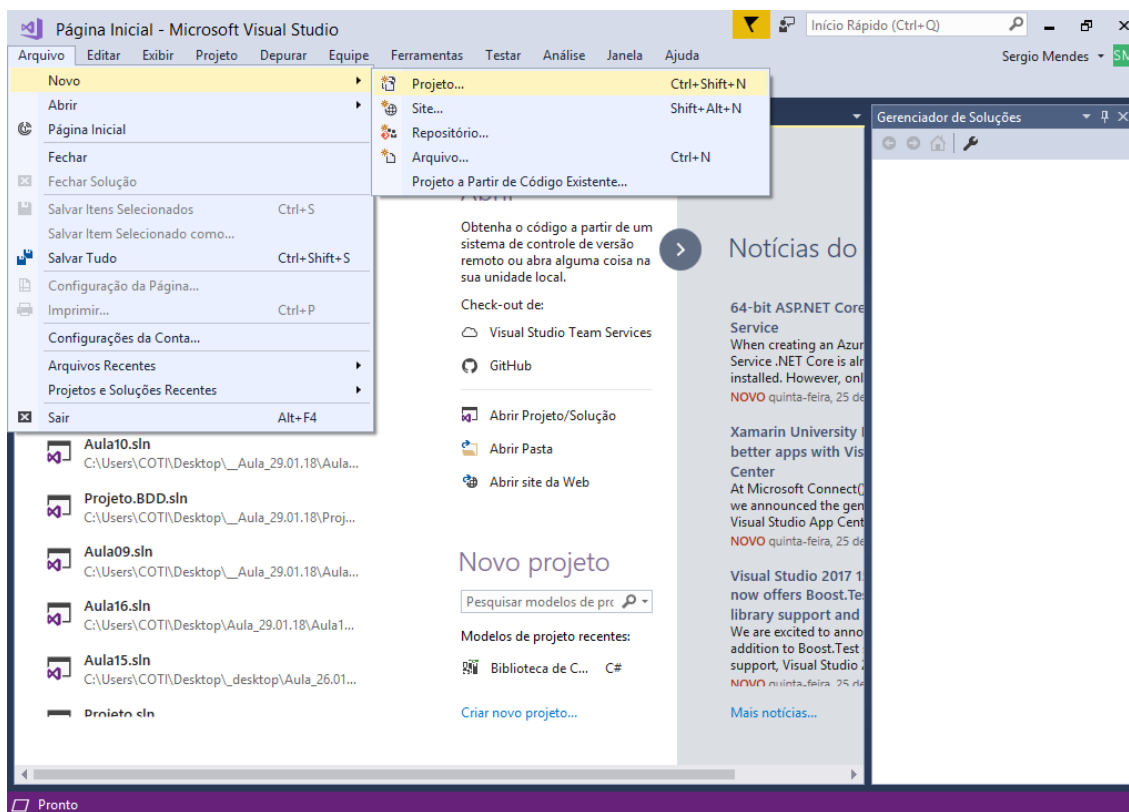


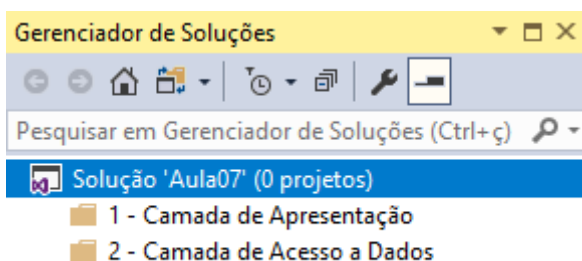
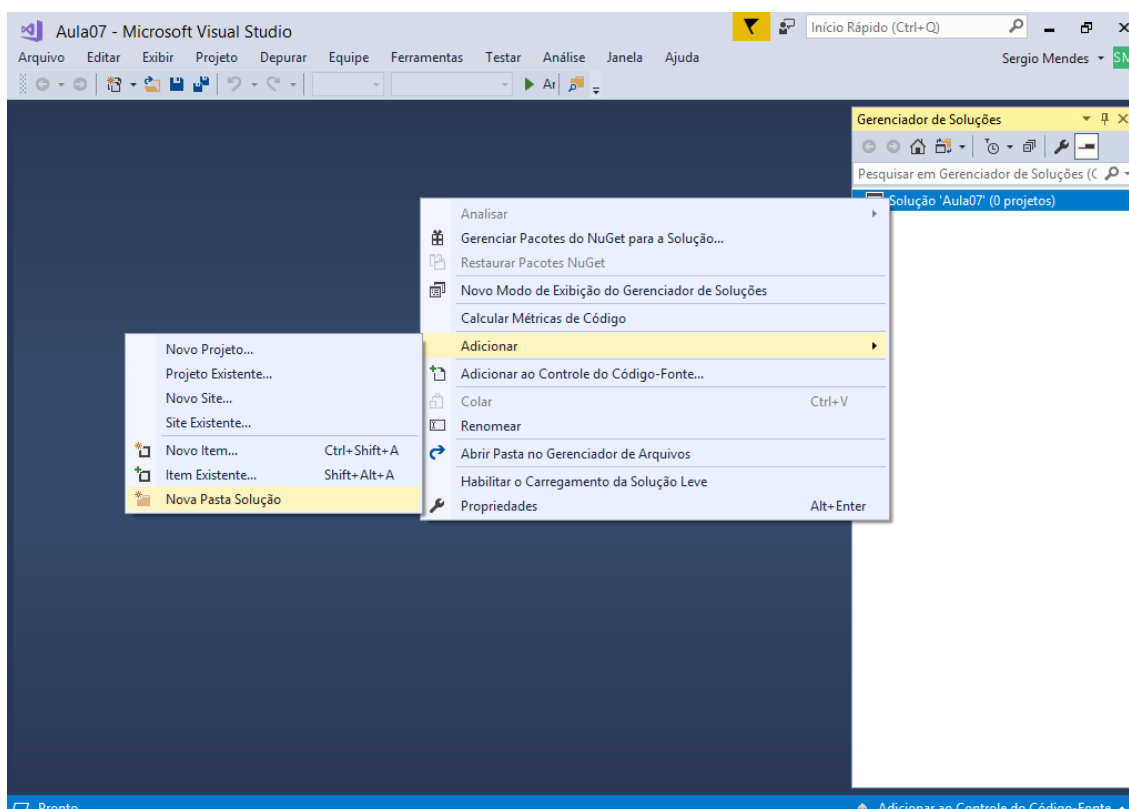
Criando uma nova solution em branco: Pasta onde armazenamos os projetos



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:

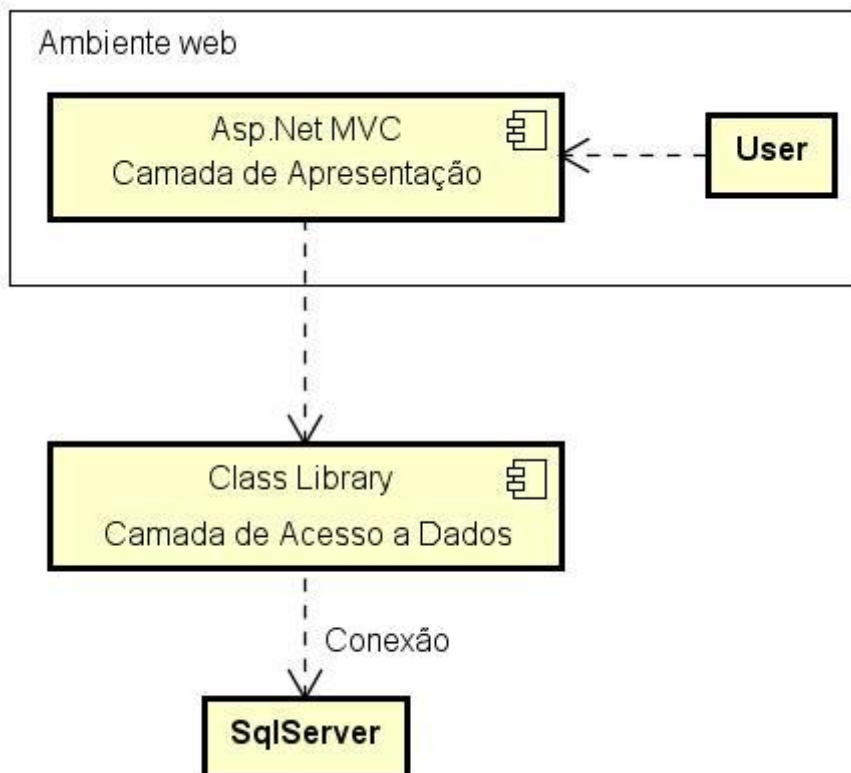


1 - Camada de Apresentação

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

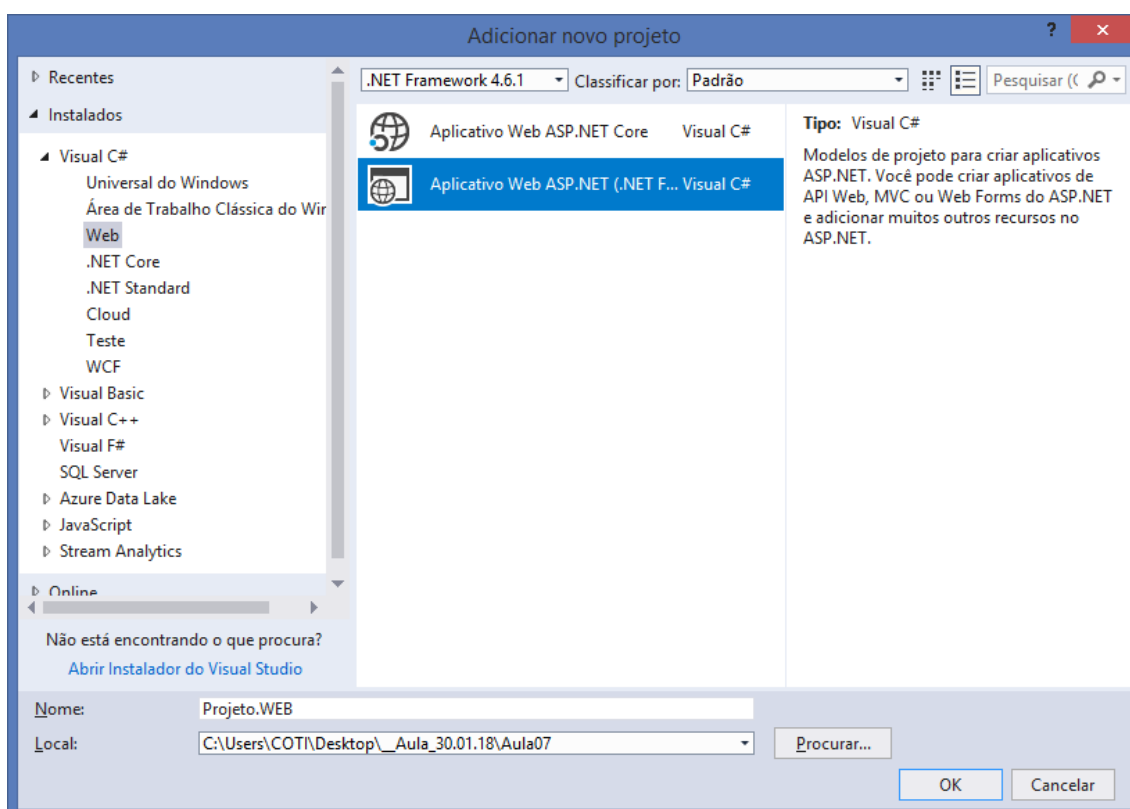
2 - Camada de Acesso a Dados

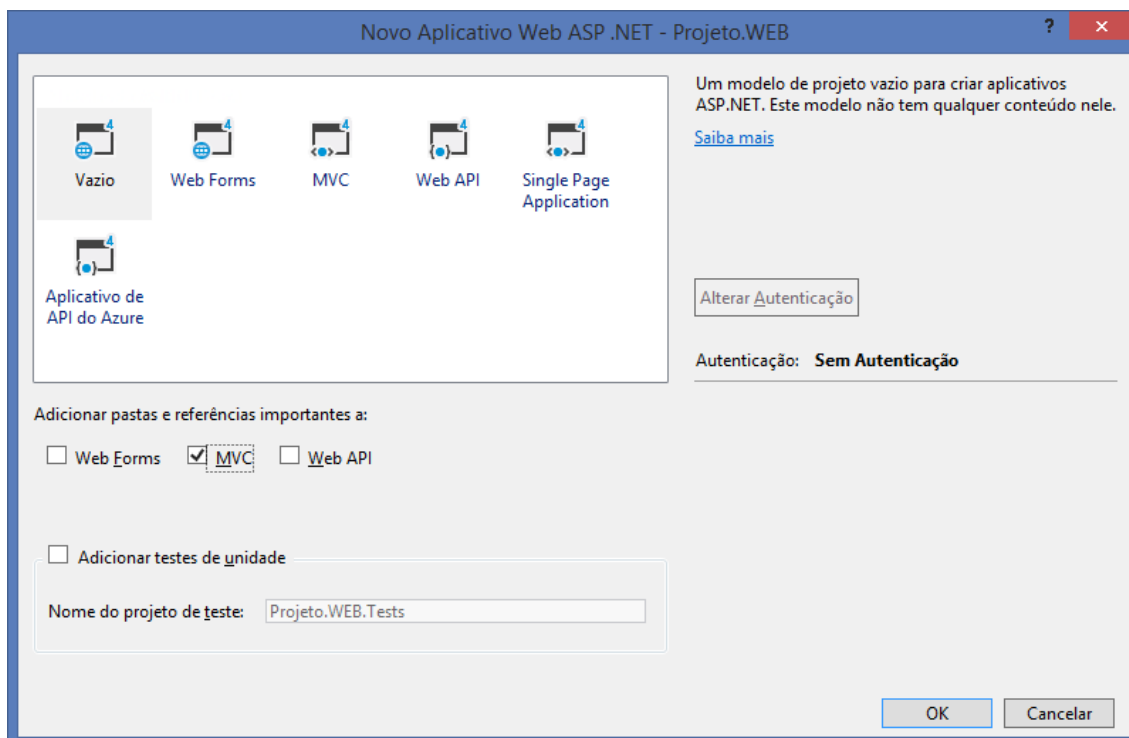
Local onde estara contido o projeto que ira realizar conexão e fazer as transações na base de dados. (Neste caso usando o **SqlClient**)



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 está 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, conteúdo **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 aparência da página HTML.
- **JavaScript** - Linguagem de scripts utilizada para criar eventos e comportamentos diretamente no navegador.

Model

Camada responsável pelos dados de entrada e saída do projeto, formulários de cadastro, grids de consultas, etc... É a camada model que coleta os dados de uma view e os leva até 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

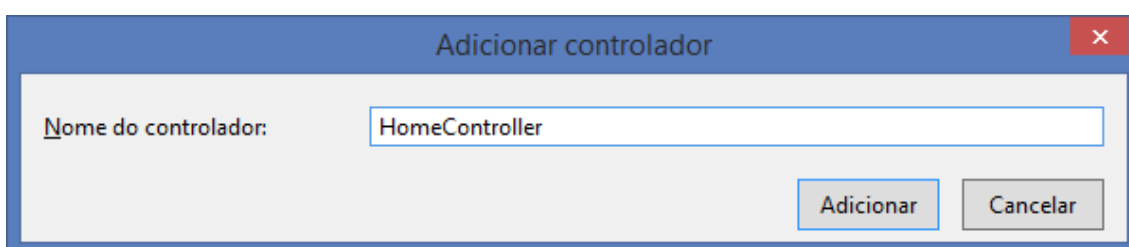
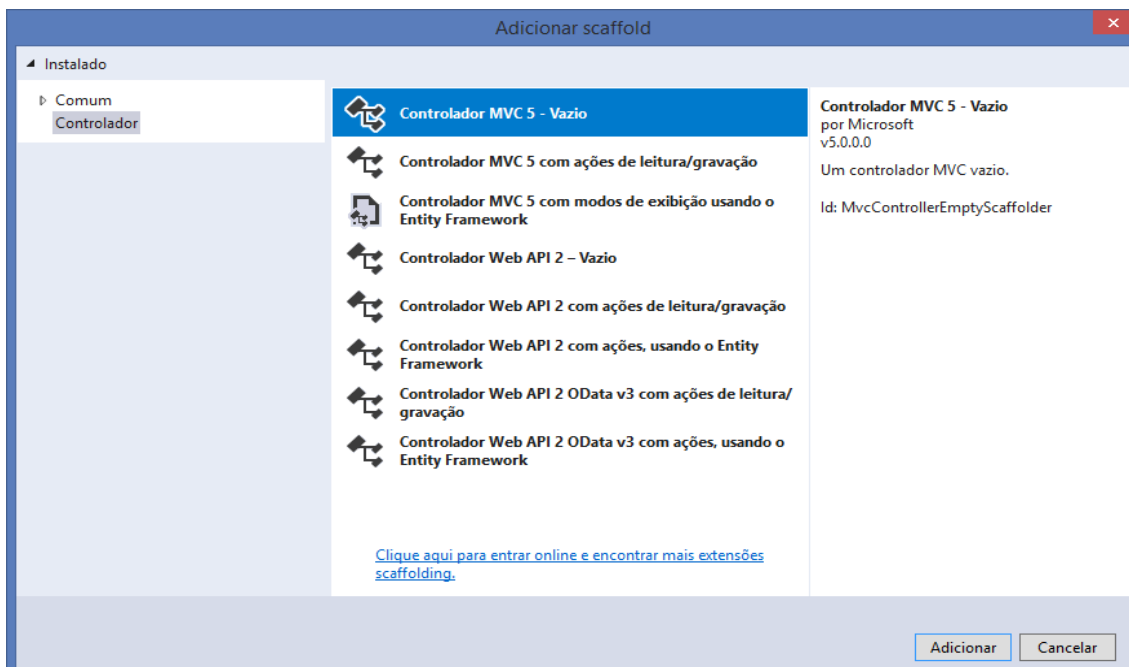
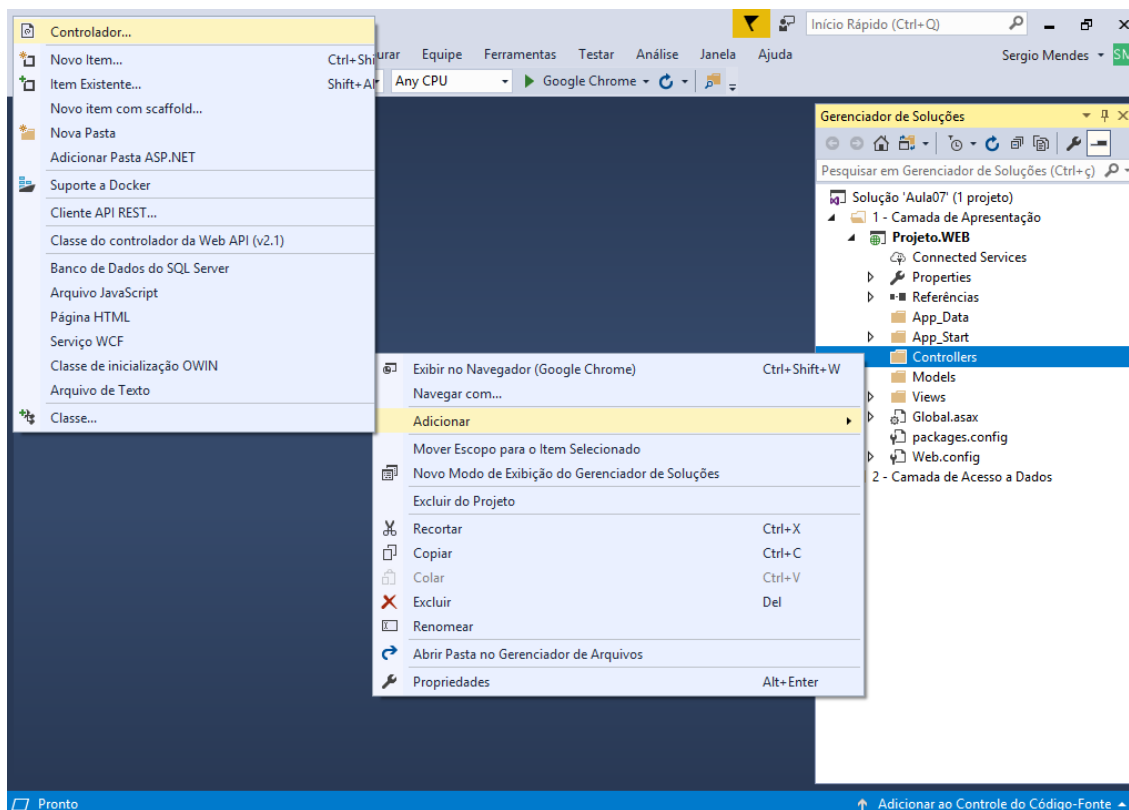
Tecnologia para desenvolvimento de aplicações web em .NET que utiliza o padrão MVC (Model, View e Controller).

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:

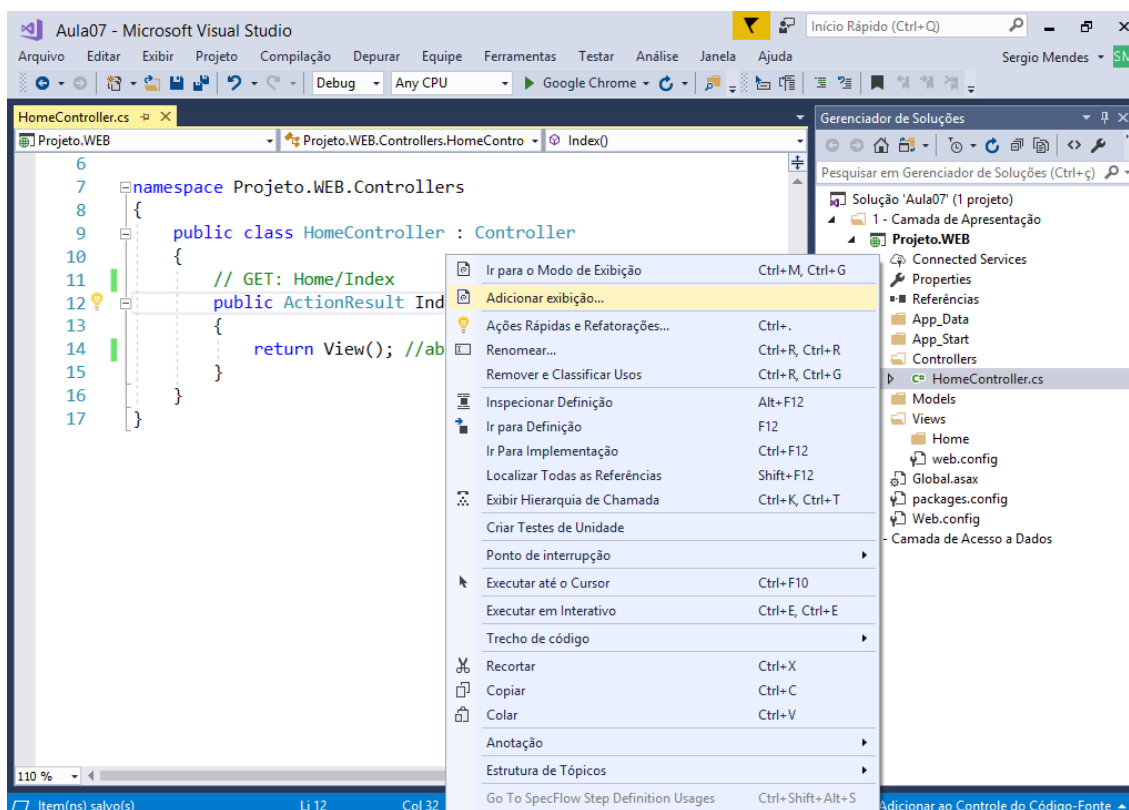
/Home/Index

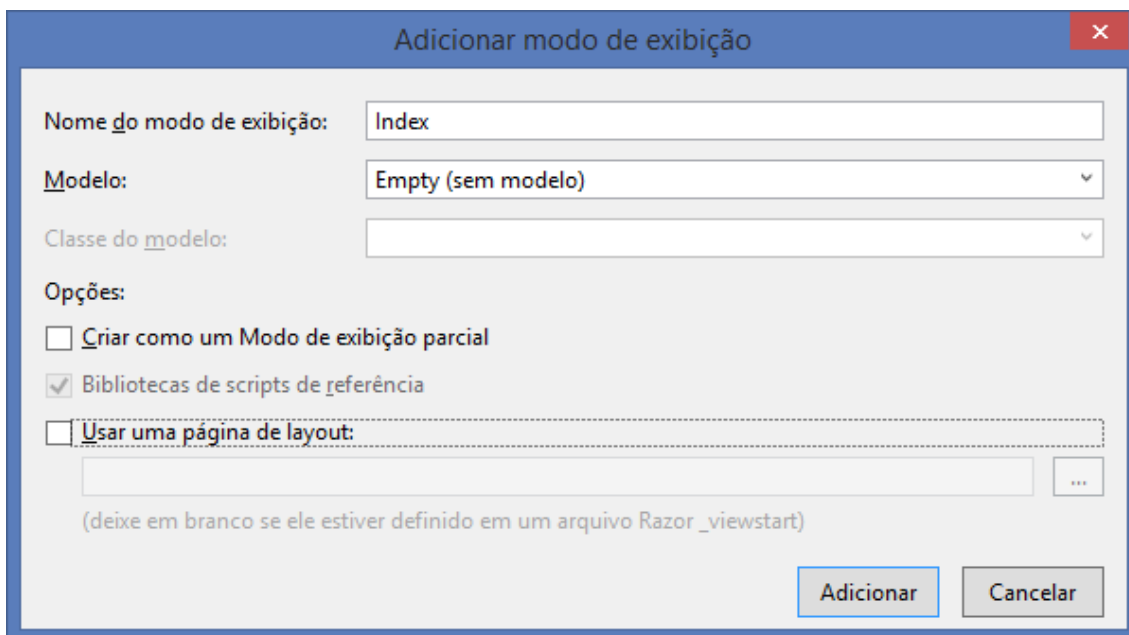
[Controller] [Action]

```
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>
```

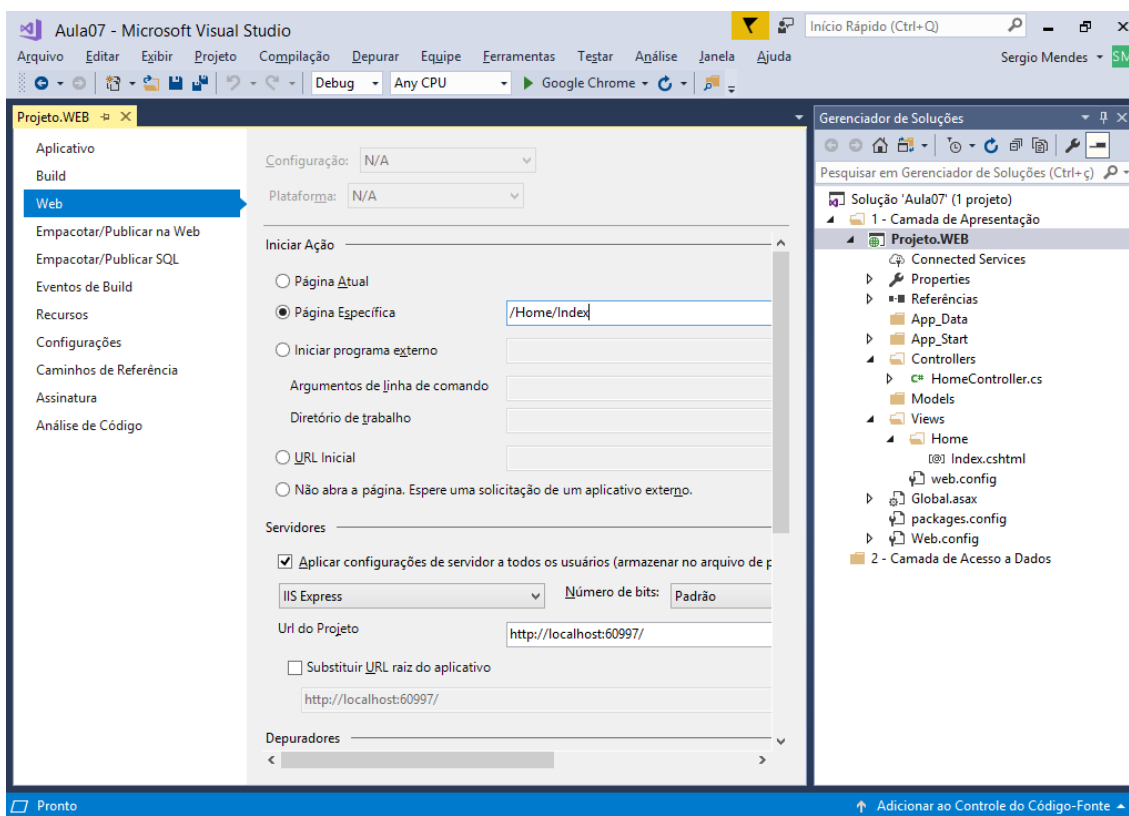
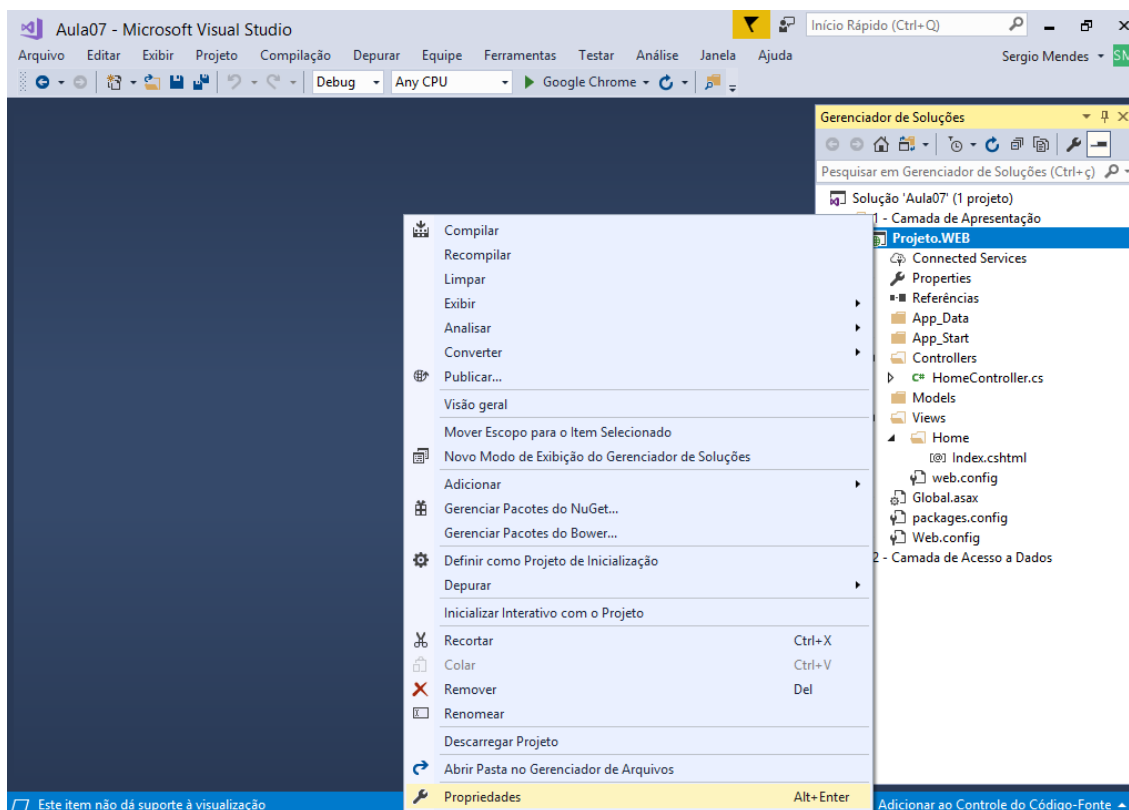
```
        </ul>
```

```
    </div>
```

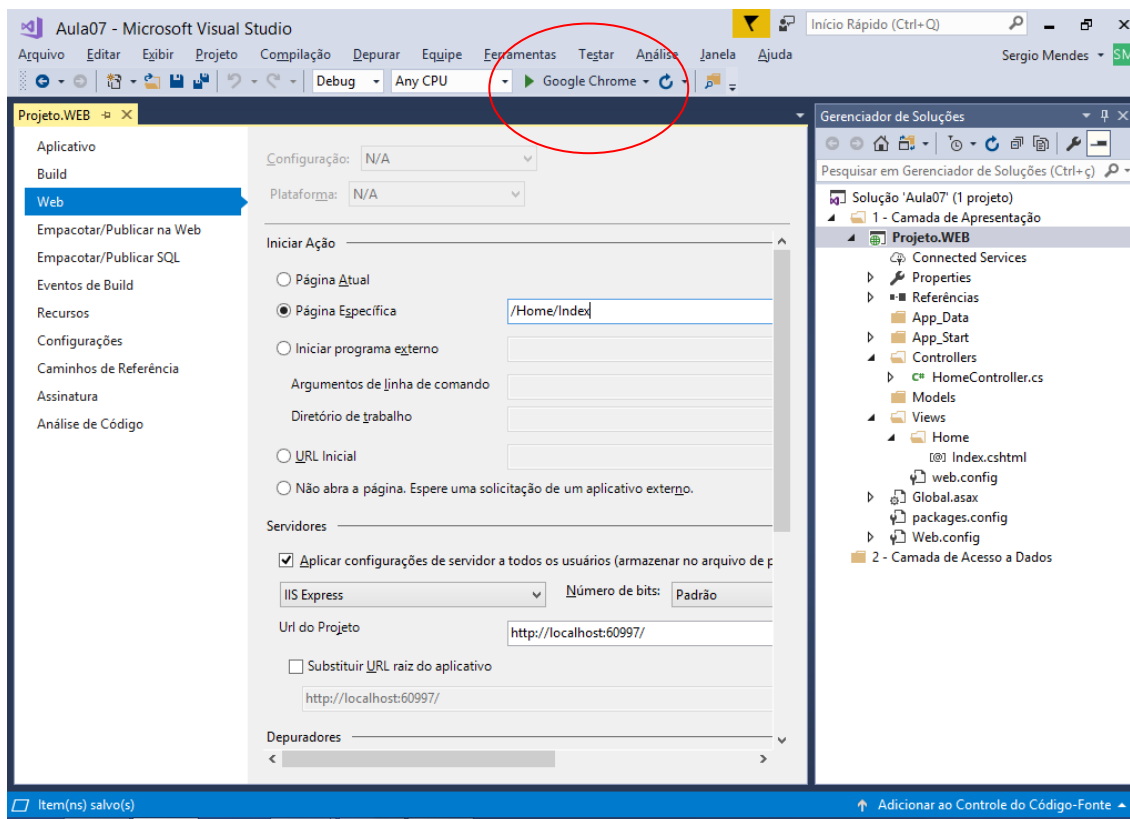
```
</body>
```

```
</html>
```

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



Executando: (F5)



<http://localhost:60997/Home/Index>



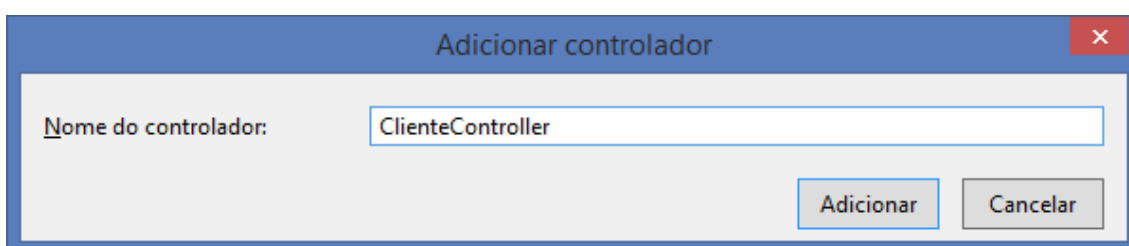
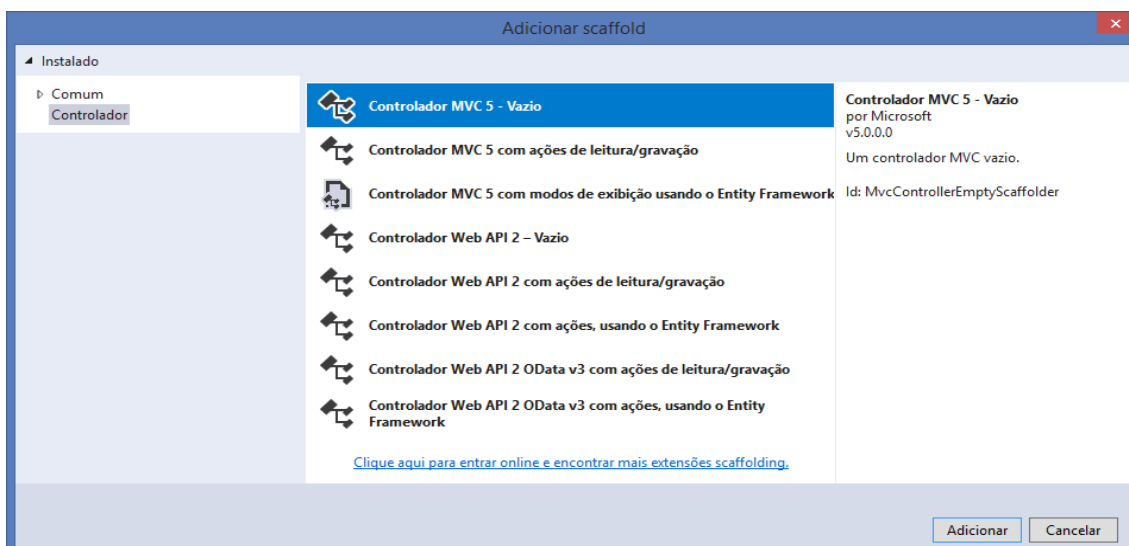
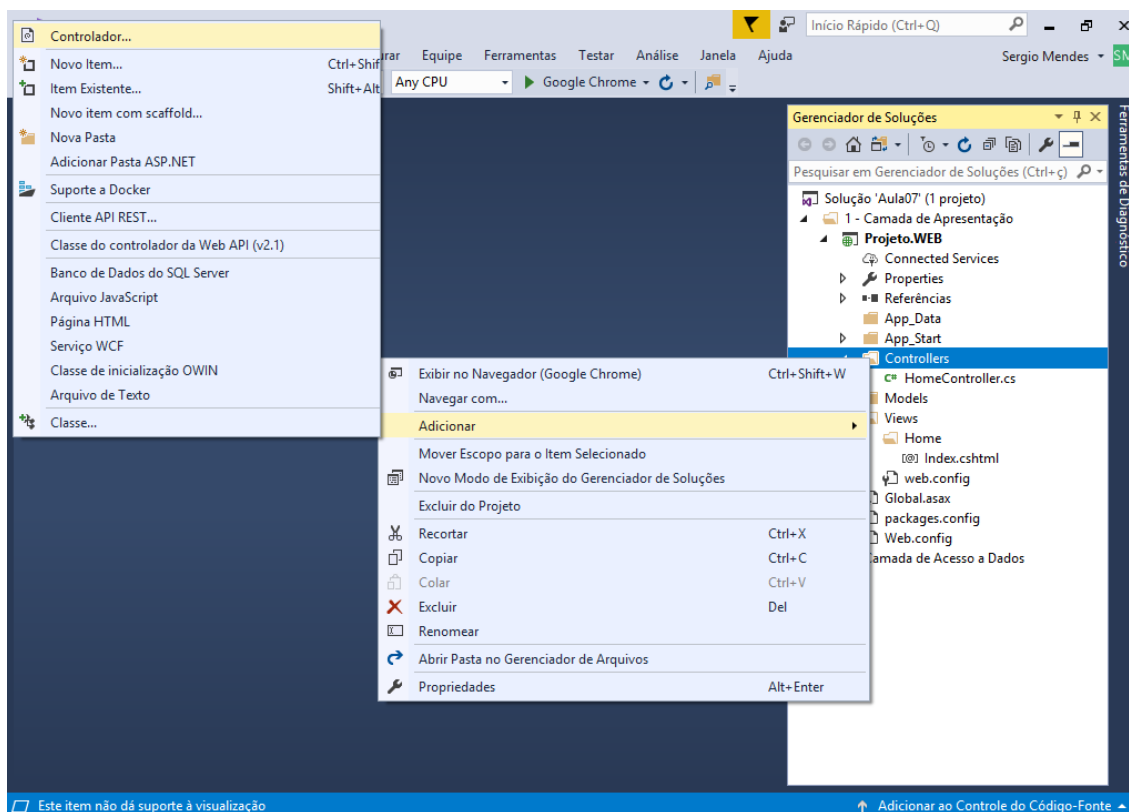
Projeto Controle de Clientes

Turma de C# WebDeveloper Noite - COTI Informatica

Selecione a ação desejada:

- [Cadastrar Clientes](#)
- [Consultar Clientes](#)

Criando uma classe de controle para Clientes /ClienteController.cs

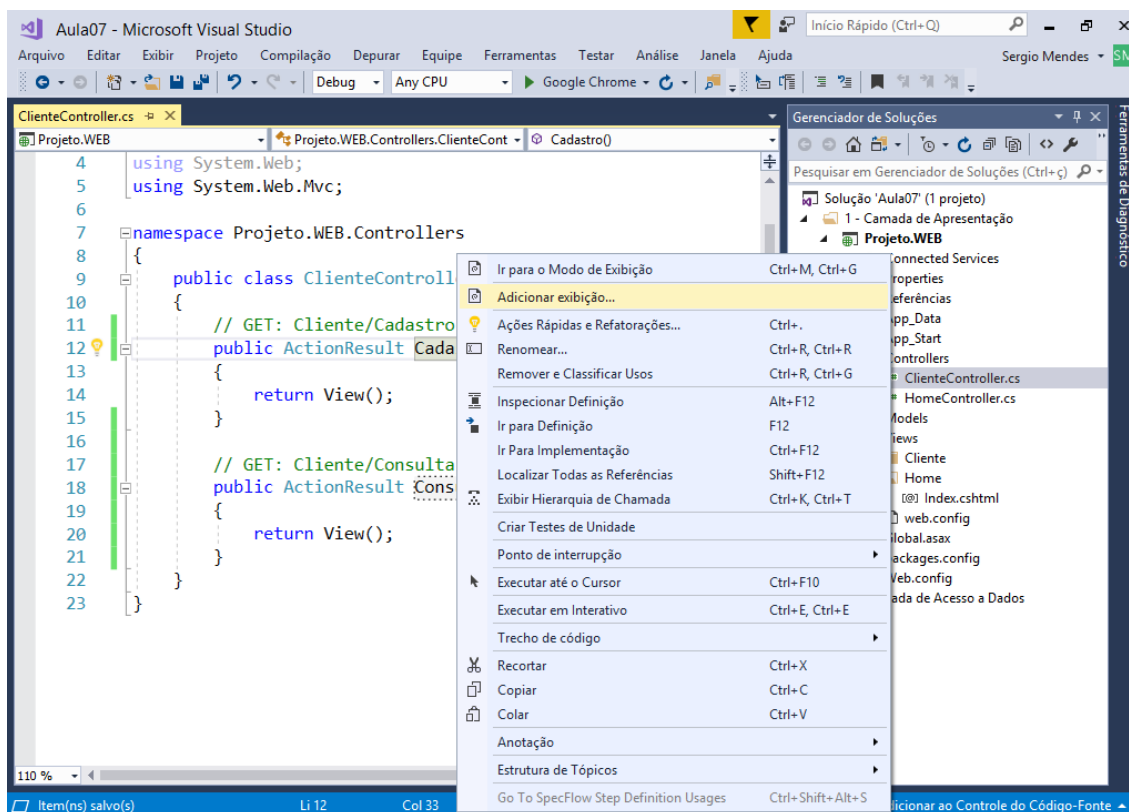


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

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/>

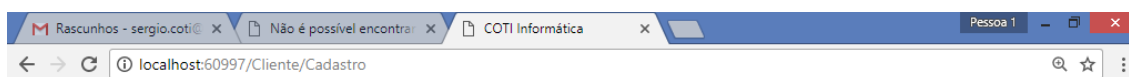
      <!-- Campo nome -->
      <label>Email do Cliente:</label> <br />
      <input type="text" name="Email" placeholder="Digite aqui" />
      <br /><br />

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

    </form>

  </div>

</body>
</html>
```



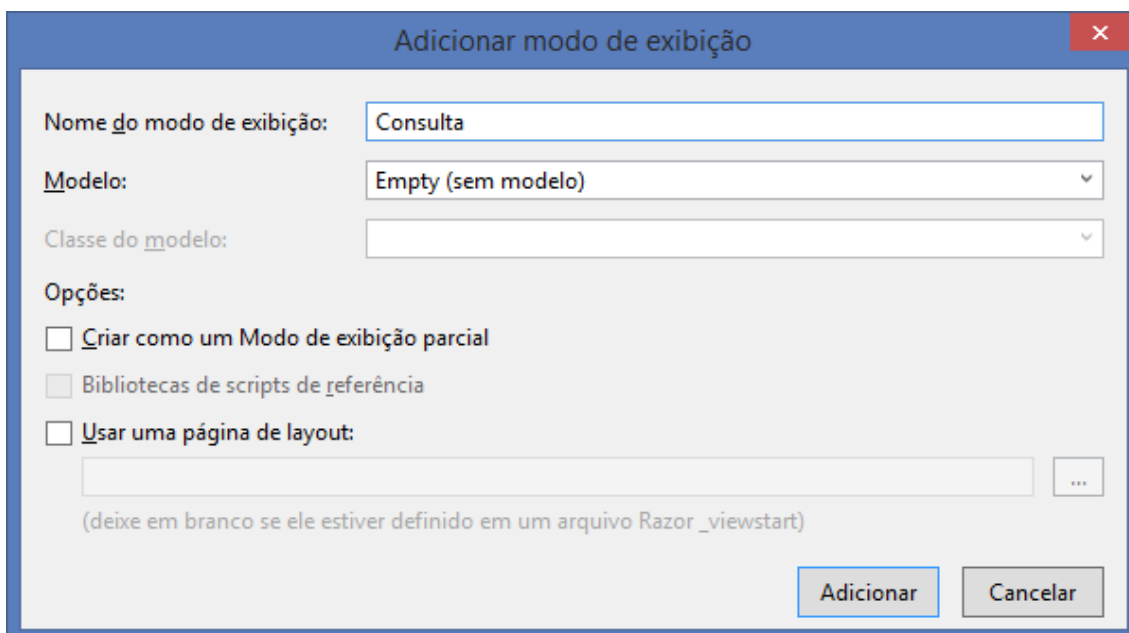
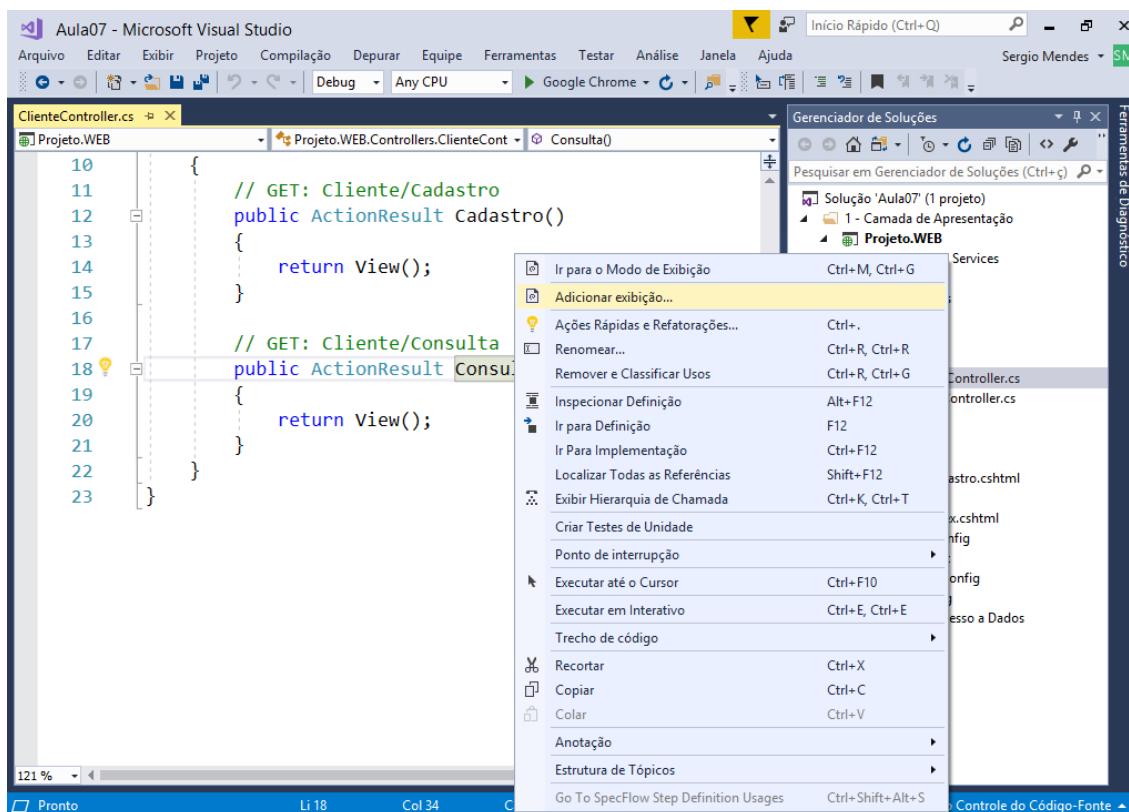
Cadastro de Clientes

[Página inicial](#)

Nome do Cliente:

Email do Cliente:

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 />

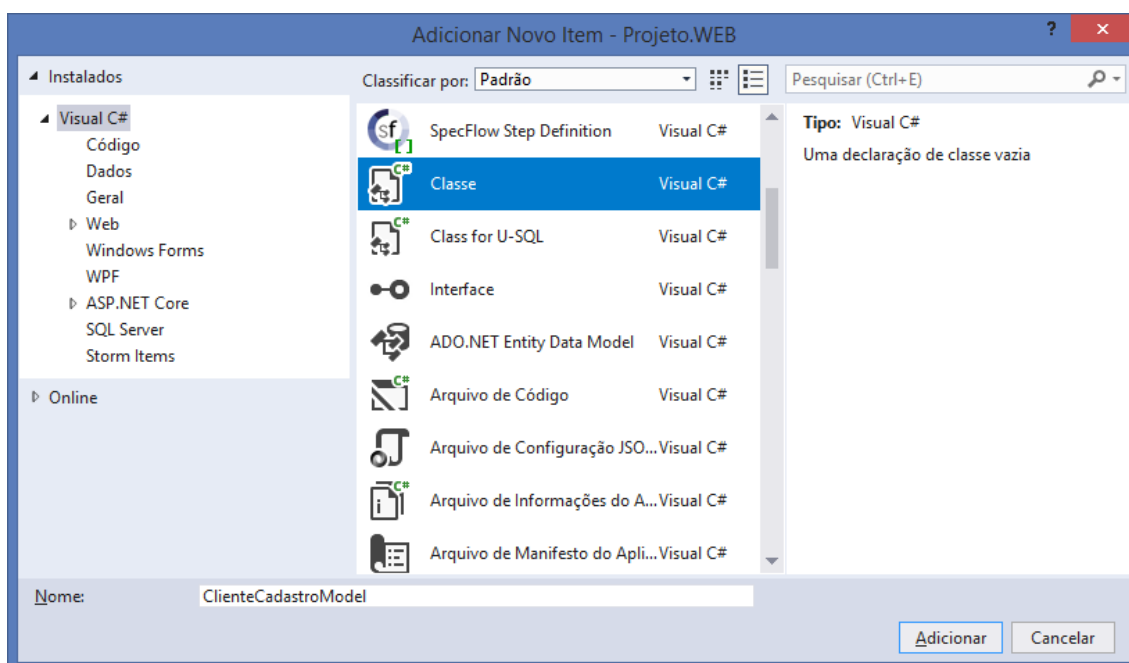
    </div>

</body>
</html>
```

Classes de modelo (Models)

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

Exemplo: Criar uma classe de modelo para os dados do formulário 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 dinamicas 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/>

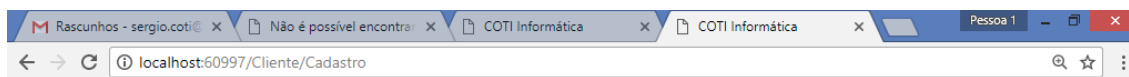
        <!-- Formulário -->
        @using (Html.BeginForm())
        {
            <!-- Campo nome -->
            <label>Nome do Cliente:</label> <br/>
            @Html.TextBoxFor(model => model.Nome,
                new { @placeholder = "Digite aqui" })
            <br/><br/>

            <!-- 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>
```



Cadastro de Clientes

[Página inicial](#)

Nome do Cliente:

Email do Cliente:

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();
        }

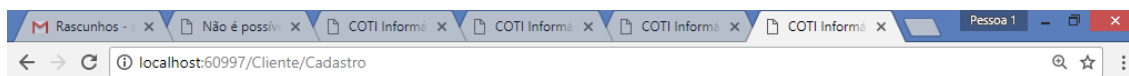
        // POST: Cliente/Cadastro
        [HttpPost]
        public ActionResult Cadastro(ClienteCadastroModel model)
        {
            //verificar se os dados obtidos pela classe model
            //estão corretos (passaram nas validações?)
            if(ModelState.IsValid)
            {
            }

            return View();
        }

        // GET: Cliente/Consulta
        public ActionResult Consulta()
        {
            return View();
        }
    }
}

```

Testando:



Cadastro de Clientes

[Página inicial](#)

Nome do Cliente:

Digite aqui Por favor, informe o nome do cliente.

Email do Cliente:

Digite aqui Por favor, informe o email do cliente.

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 />
        }
    
```

```


<br/>
<br/>

<strong>@ViewBag.Mensagem</strong>
    }
</div>
</body>
</html>

```

Executando:

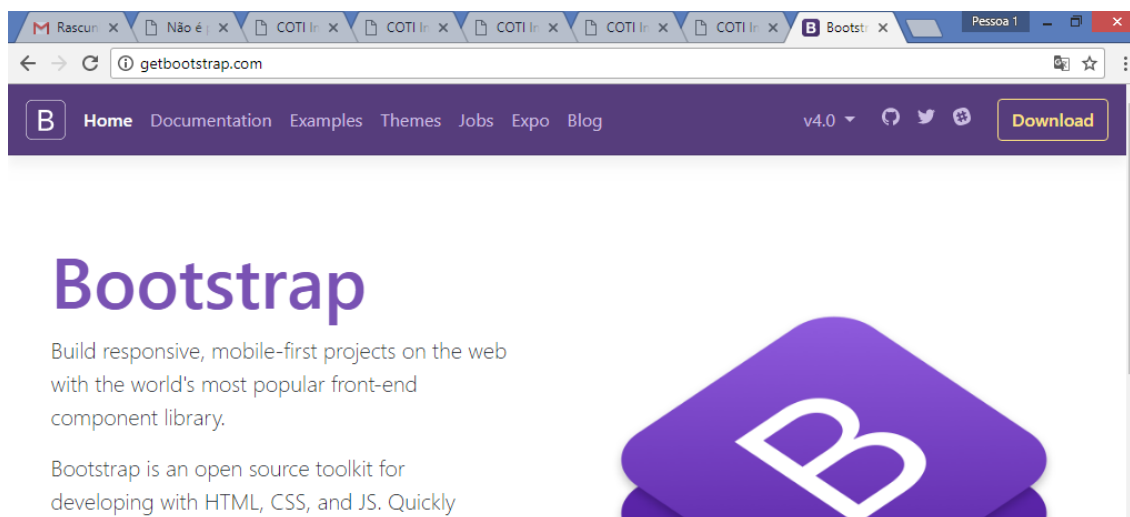


Folhas de estilo CSS - Cascading Style Sheet

Linguagem utilizada para formatação de páginas HTML

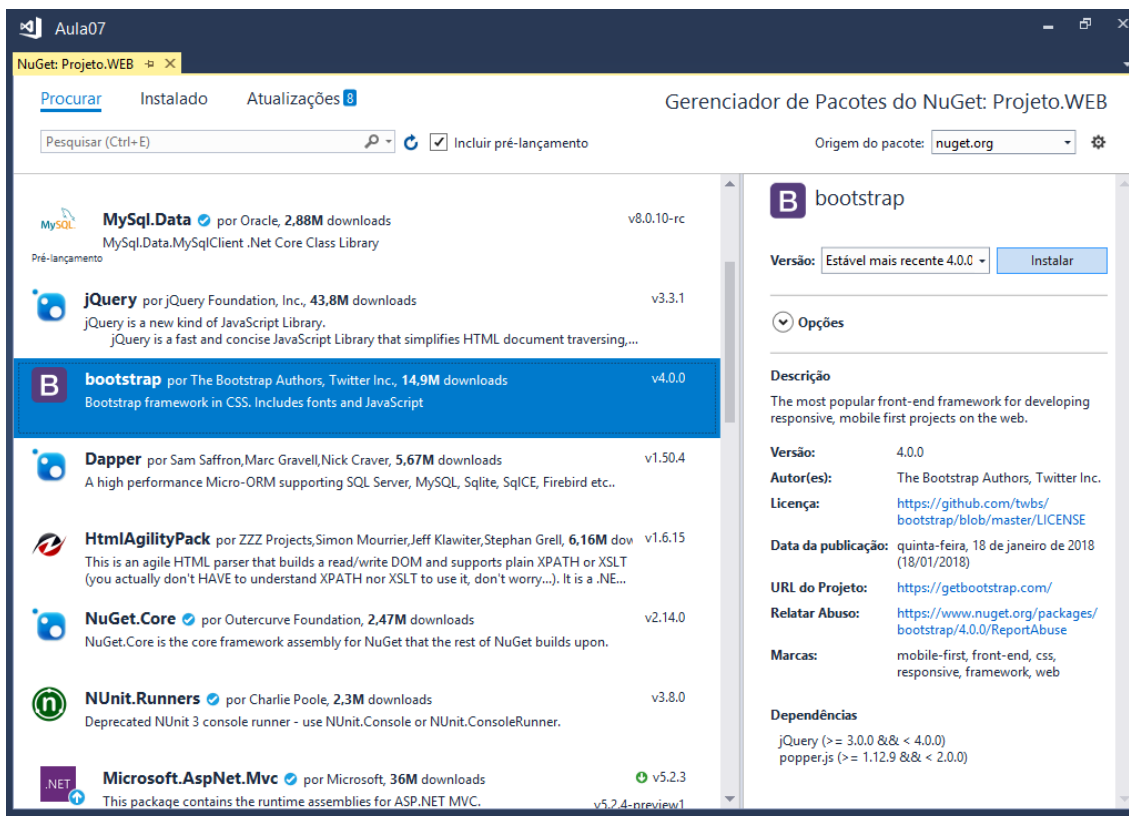
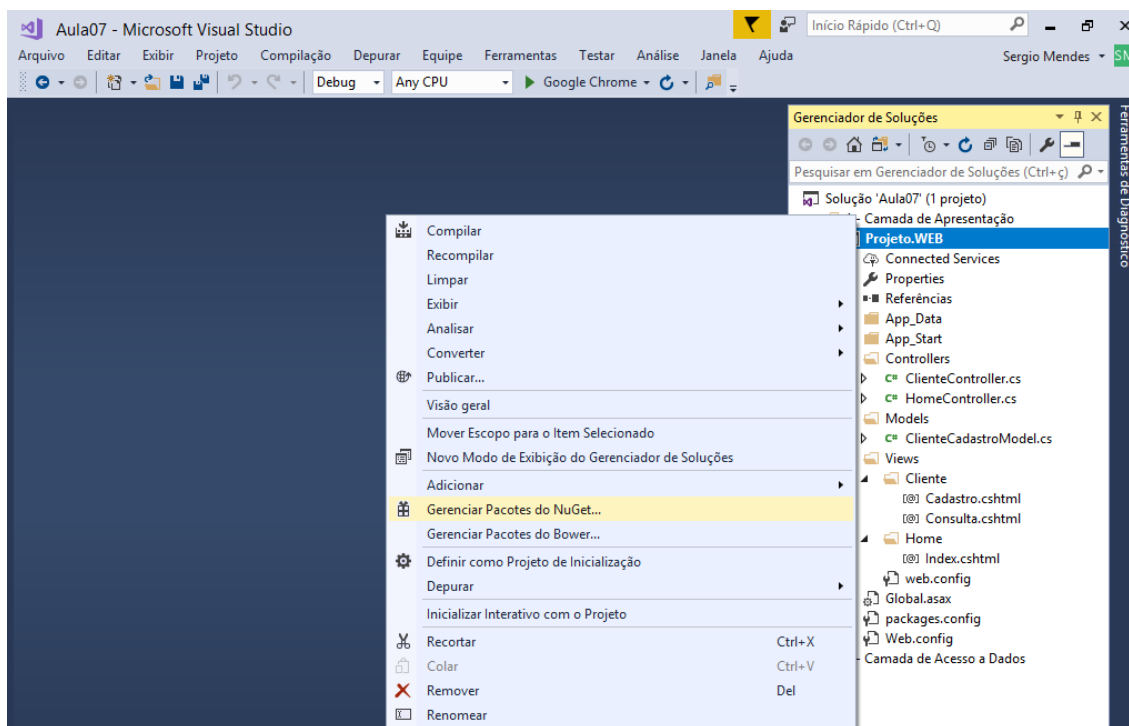
Existem varias bibliotecas ja prontas com diversos modelos de folhas de estilo CSS, dentre estas a mais conhecida e utilizada é o **bootstrap**

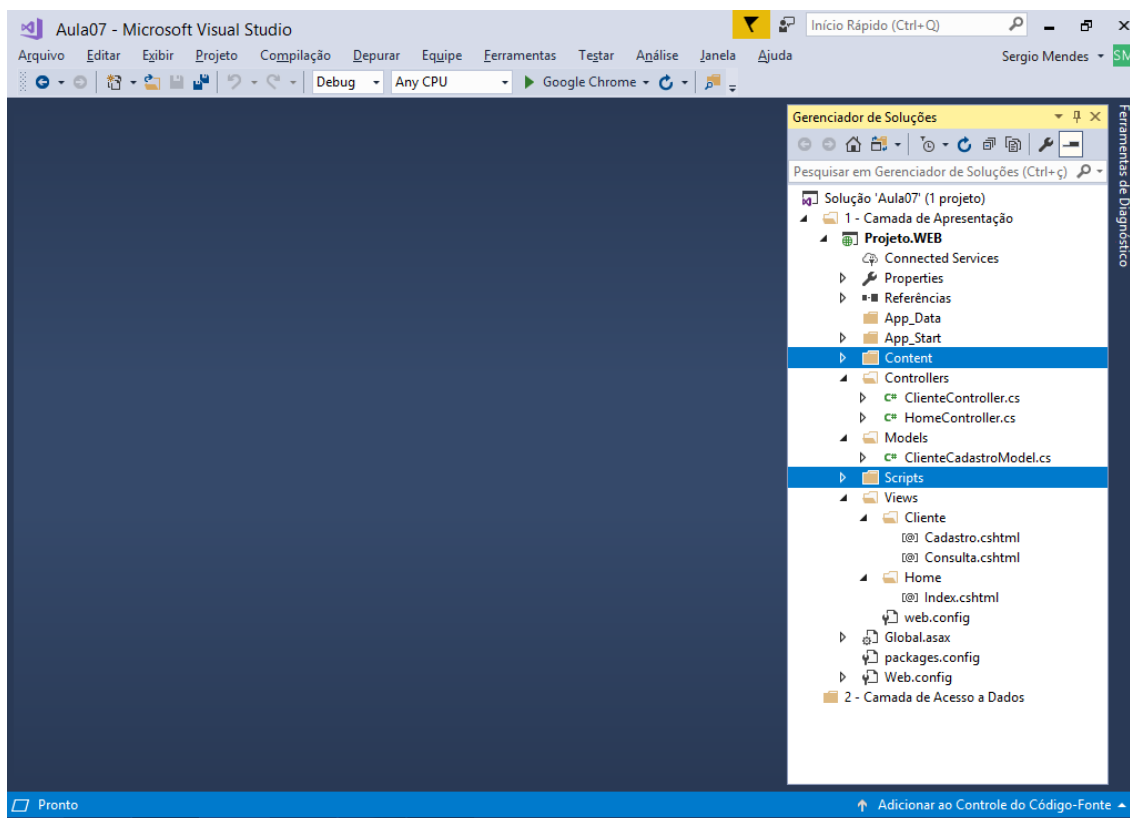
<http://getbootstrap.com/>



Instalando os arquivos do bootstrap através 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>

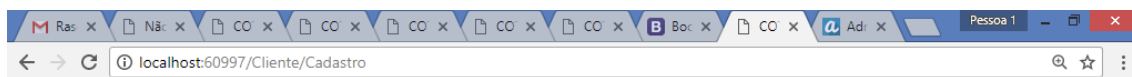
  <link href="/Content/bootstrap.min.css" rel="stylesheet" />
</head>
<body class="container">
  <div>

    <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>
```



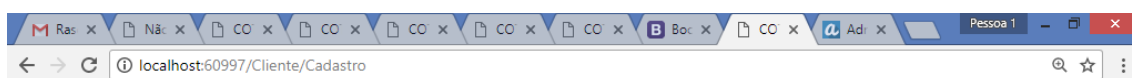
Cadastro de Clientes

[Página inicial](#)

Nome do Cliente:

Email do Cliente:

Cadastrar Cliente



Cadastro de Clientes

[Página inicial](#)

Nome do Cliente:

Por favor, informe o nome do cliente.

Email do Cliente:

Por favor, informe o email do cliente.

Cadastrar Cliente

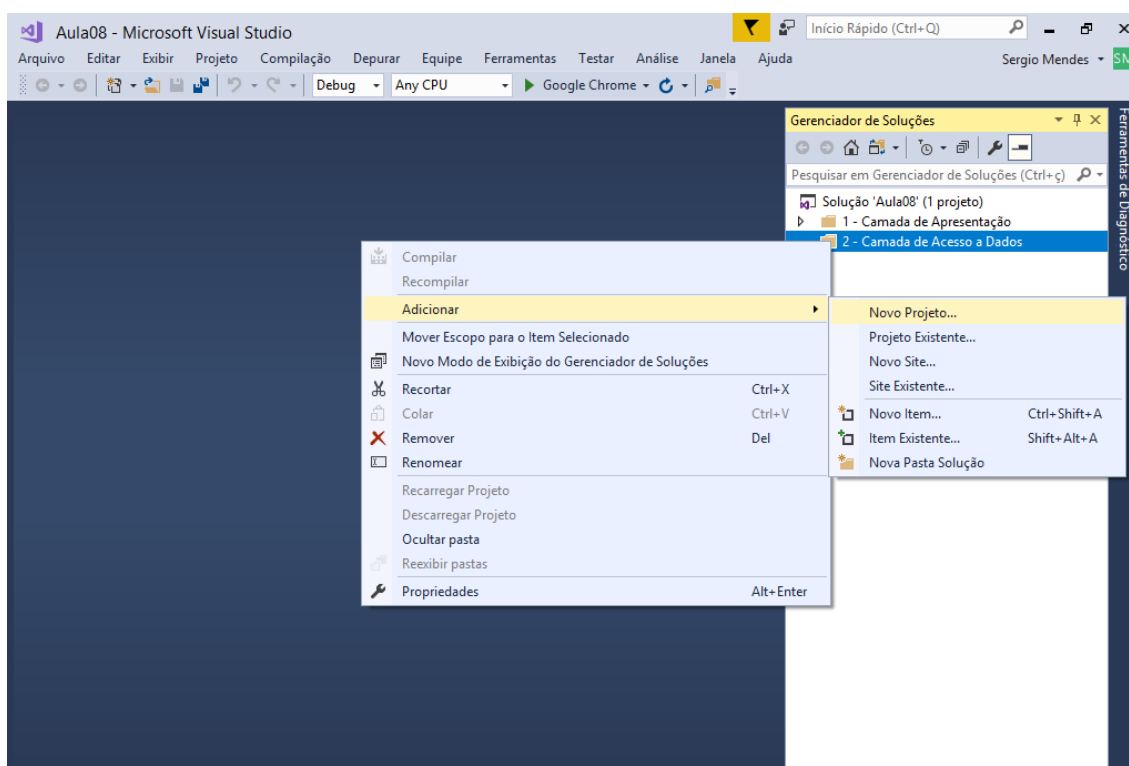
2 - Camada de Acesso a Dados

Projeto para conexão, transações e consultas a tabelas do banco de dados.

DAL - Data Access Layer

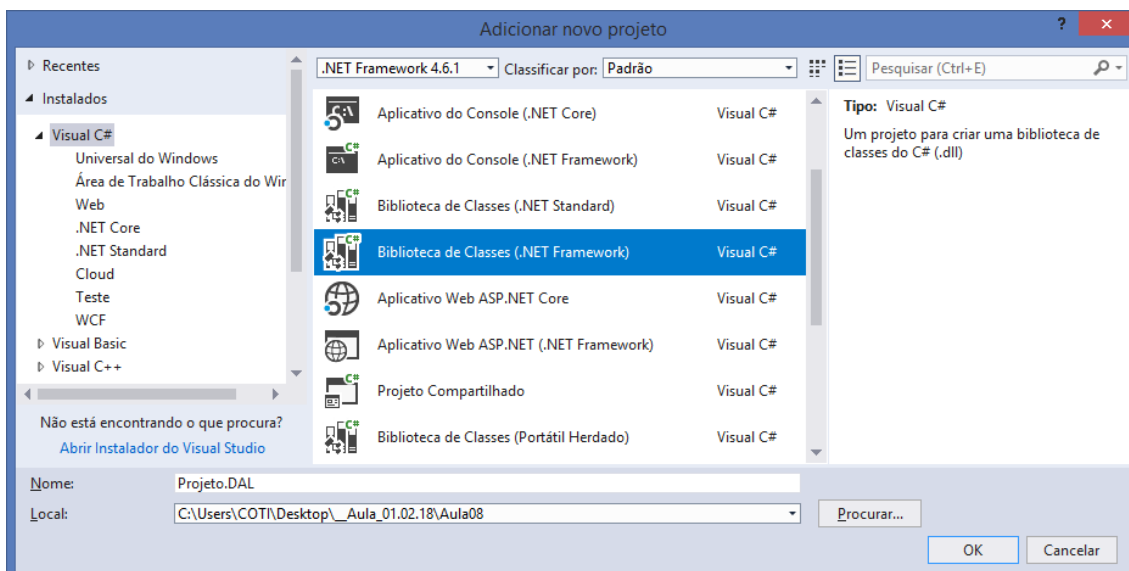
Camada de acesso a dados

Nome dado a camada de um projeto responsável por realizar a comunicação com uma base de dados (repositorio de dados)



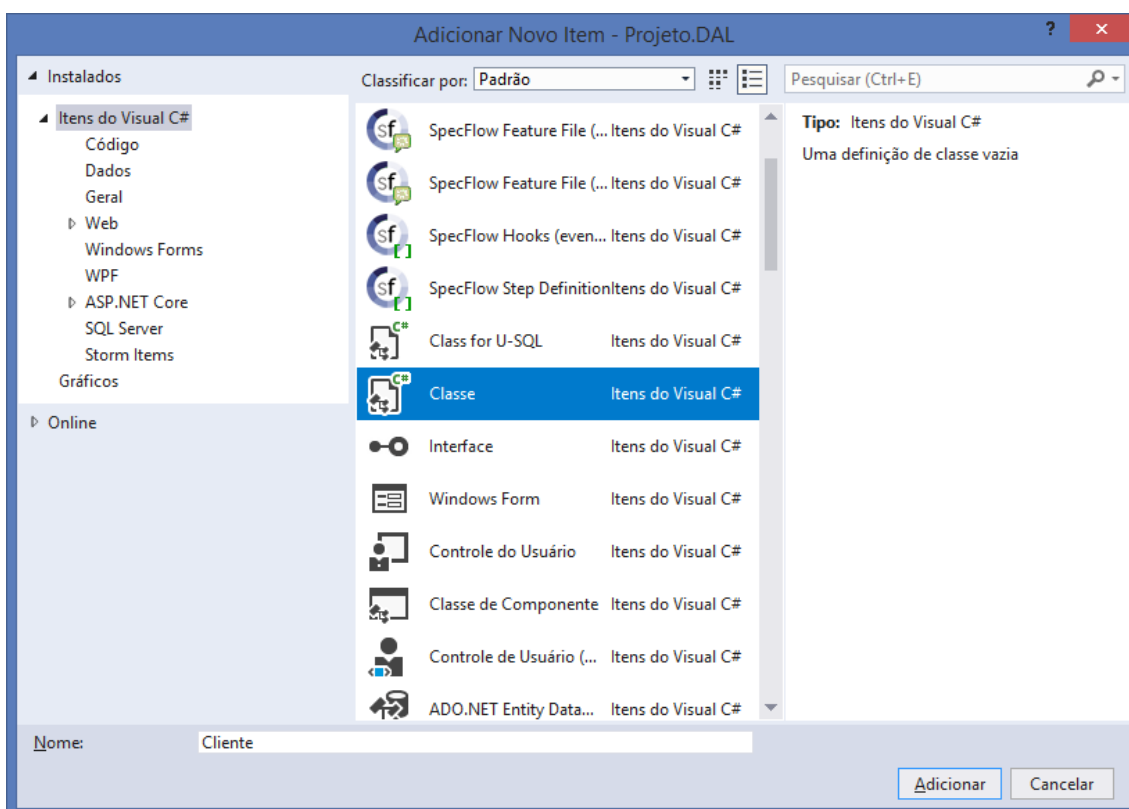
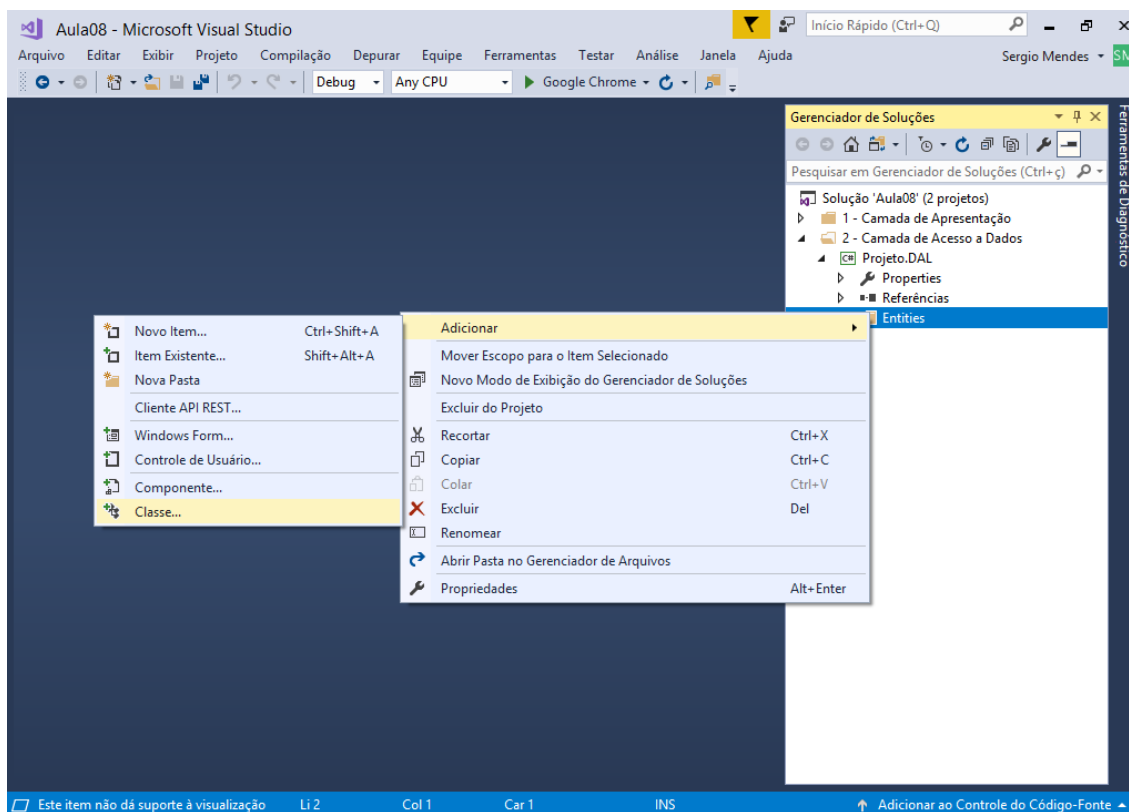
Biblioteca de Classes (Class Library)

Projeto simples que tem como objetivo armazenar classes



Modelagem de entidades

Classes para representar o modelo de dados do sistema, similar as tabelas do banco de dados



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projeto.DAL.Entities
{
    public class Cliente
    {
        //propriedades set e get
        //[prop] + 2x[tab]
        public int IdCliente { get; set; }
        public string Nome { get; set; }
        public string Email { get; set; }
        public DateTime DataCadastro { get; set; }

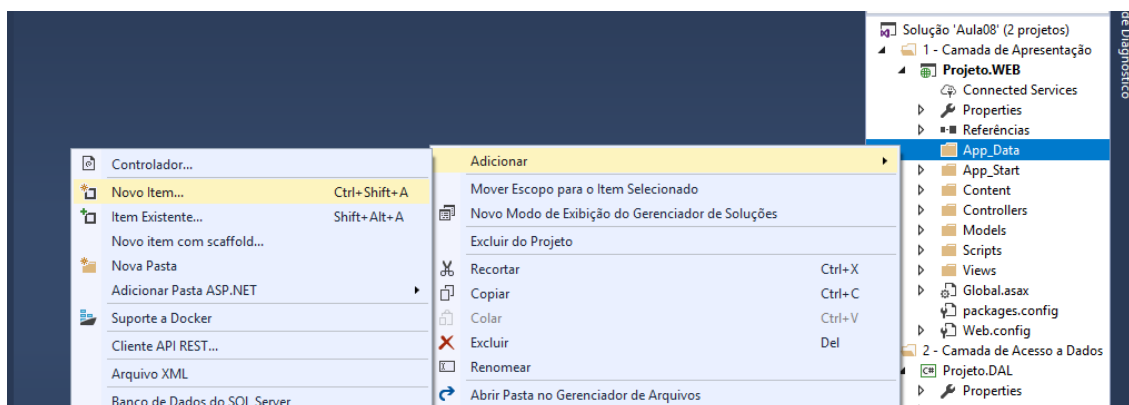
        //construtor default
        //[ctor] + 2x[tab]
        public Cliente()
        {
            //vazio..
        }

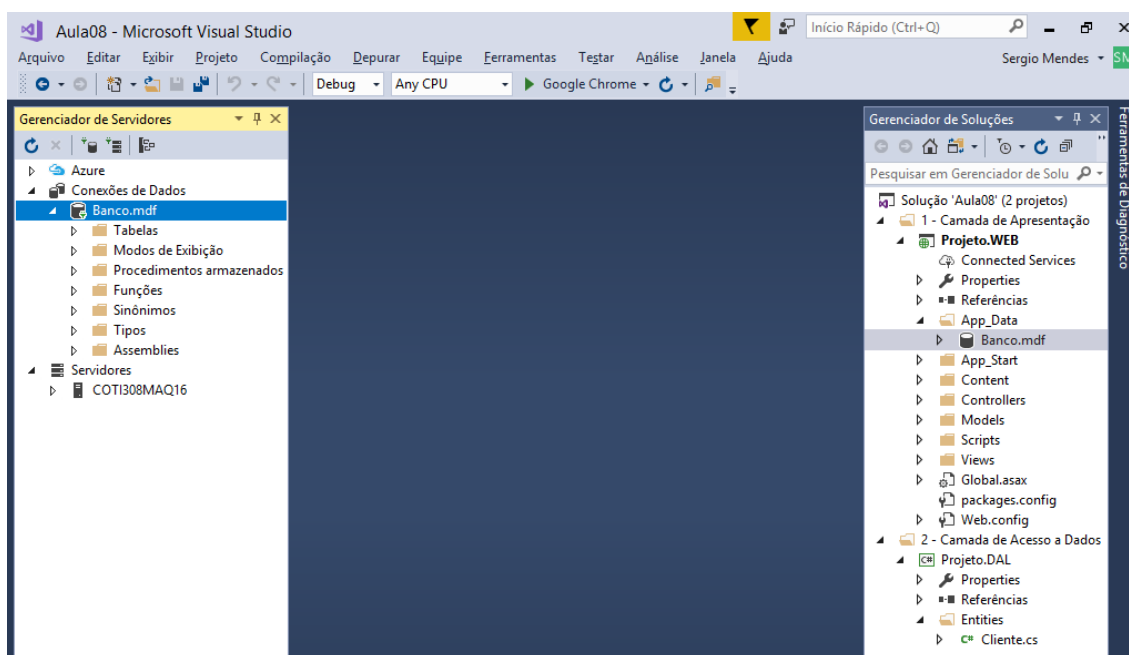
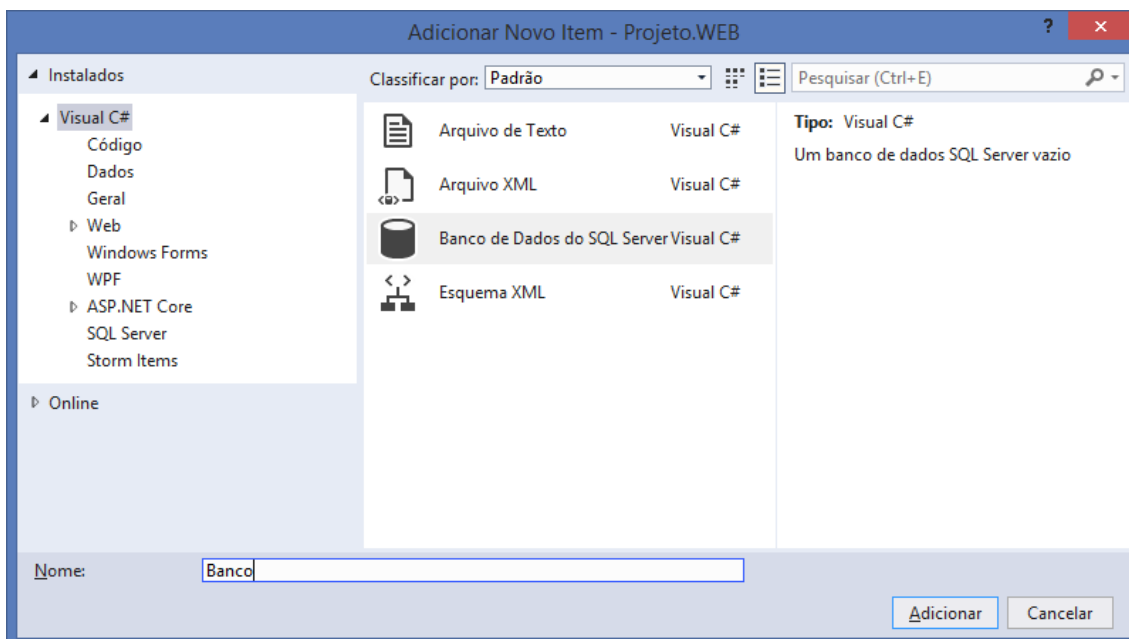
        //sobrecarga (overloading) de construtores..
        public Cliente(int idCliente, string nome, string email,
            DateTime dataCadastro)
        {
            IdCliente = idCliente;
            Nome = nome;
            Email = email;
            DataCadastro = dataCadastro;
        }

        //sobrescrita (override) do método
        public override string ToString()
        {
            return $"Id do Cliente: {IdCliente}, Nome: {Nome},
                Email: {Email}, Data de Cadastro: {DataCadastro}";
        }
    }
}
```

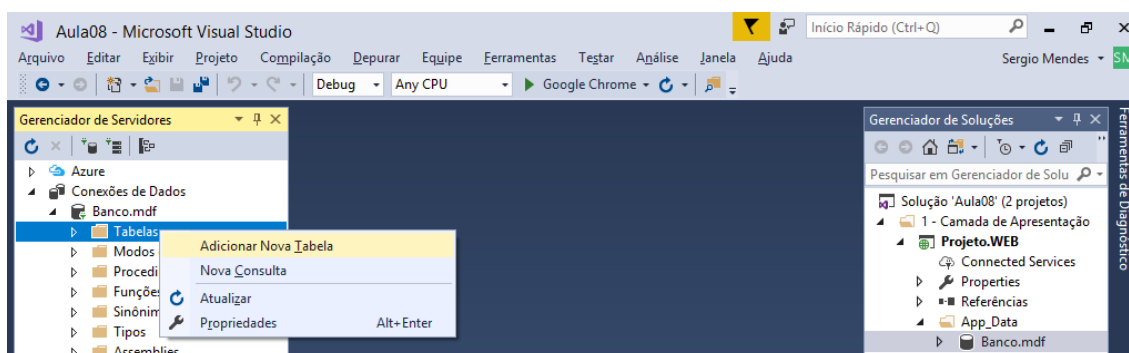
Criando um banco de dados:

MDF - Master Database File (arquivo de banco de dados do SqlServer)





Criando uma tabela de Clientes:



```
create table Cliente(  
    IdCliente      integer      identity,  
    Nome           nvarchar(50) not null,  
    Email          nvarchar(50) not null unique,  
    DataCadastro   datetime     not null,  
    primary key(IdCliente))
```

Executando e criando a tabela de cliente:

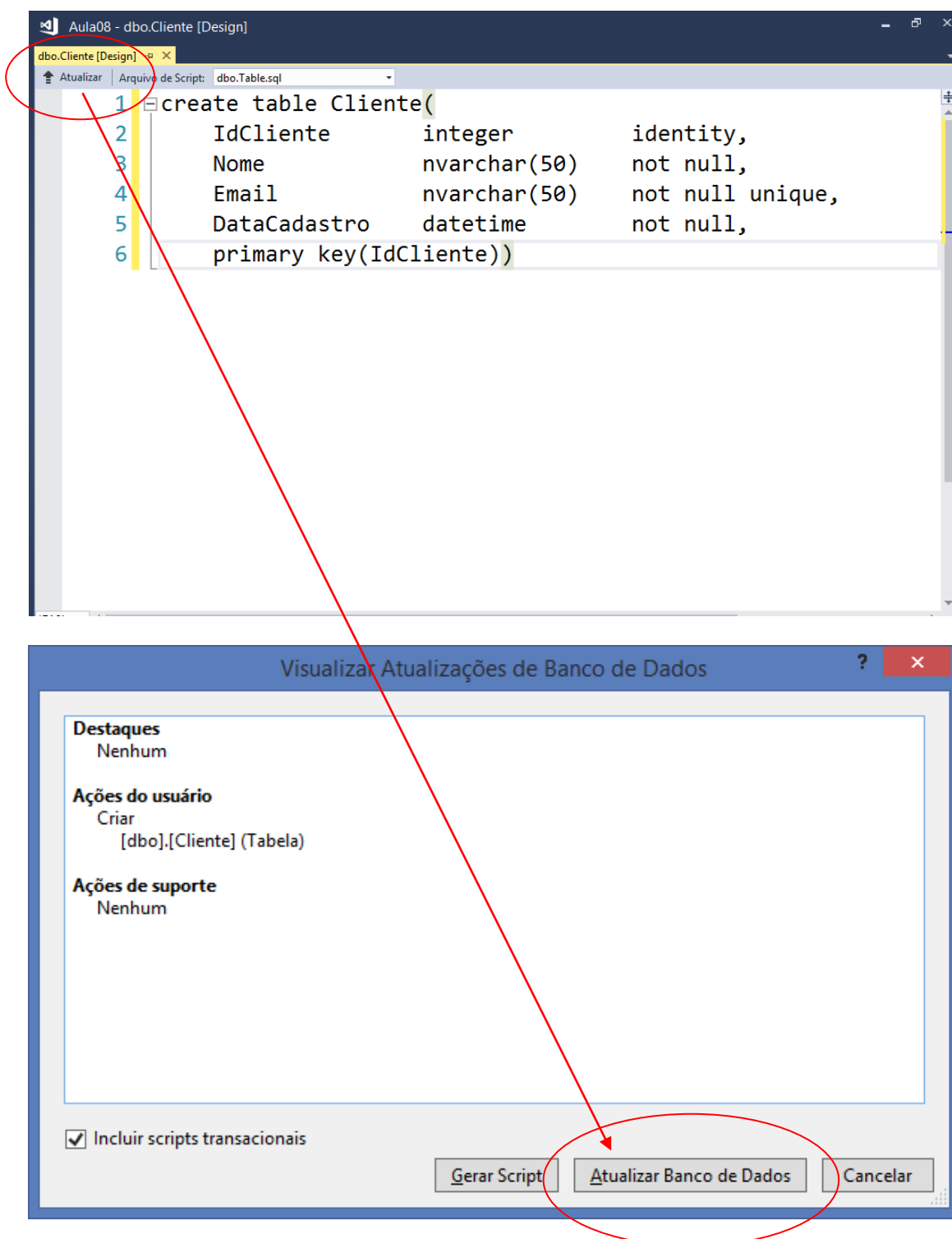
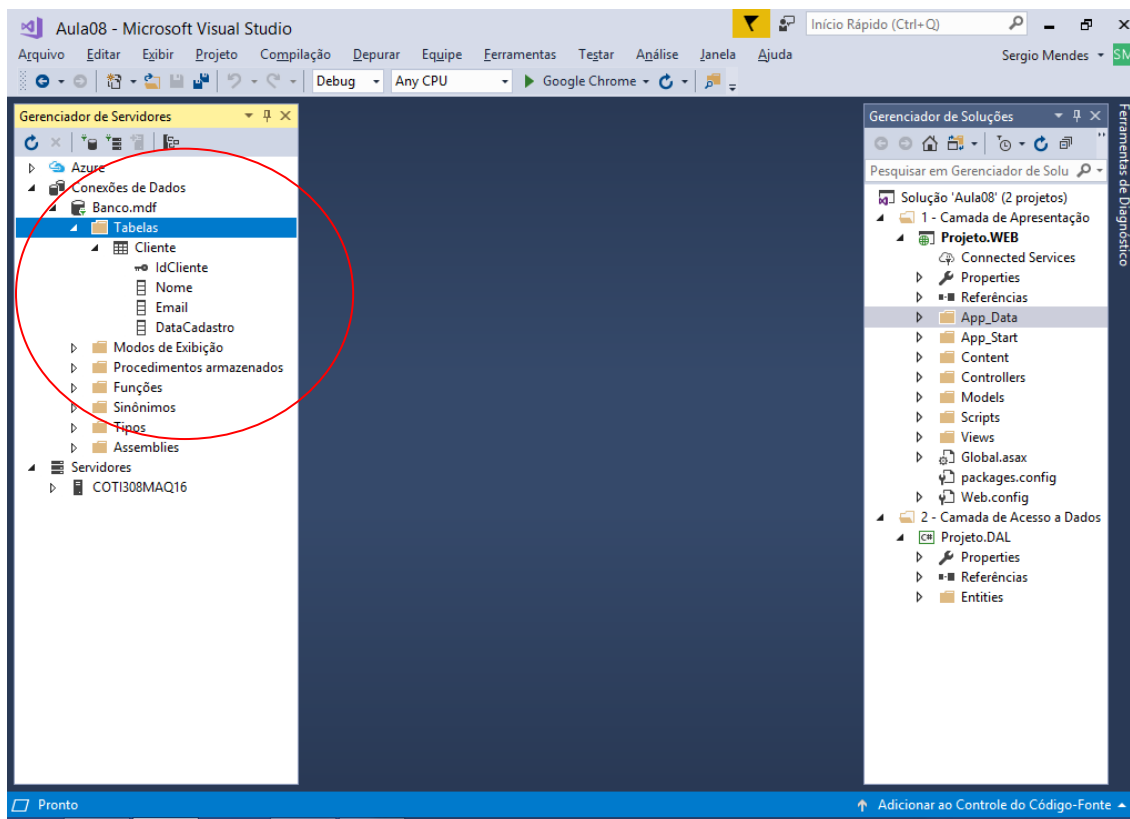
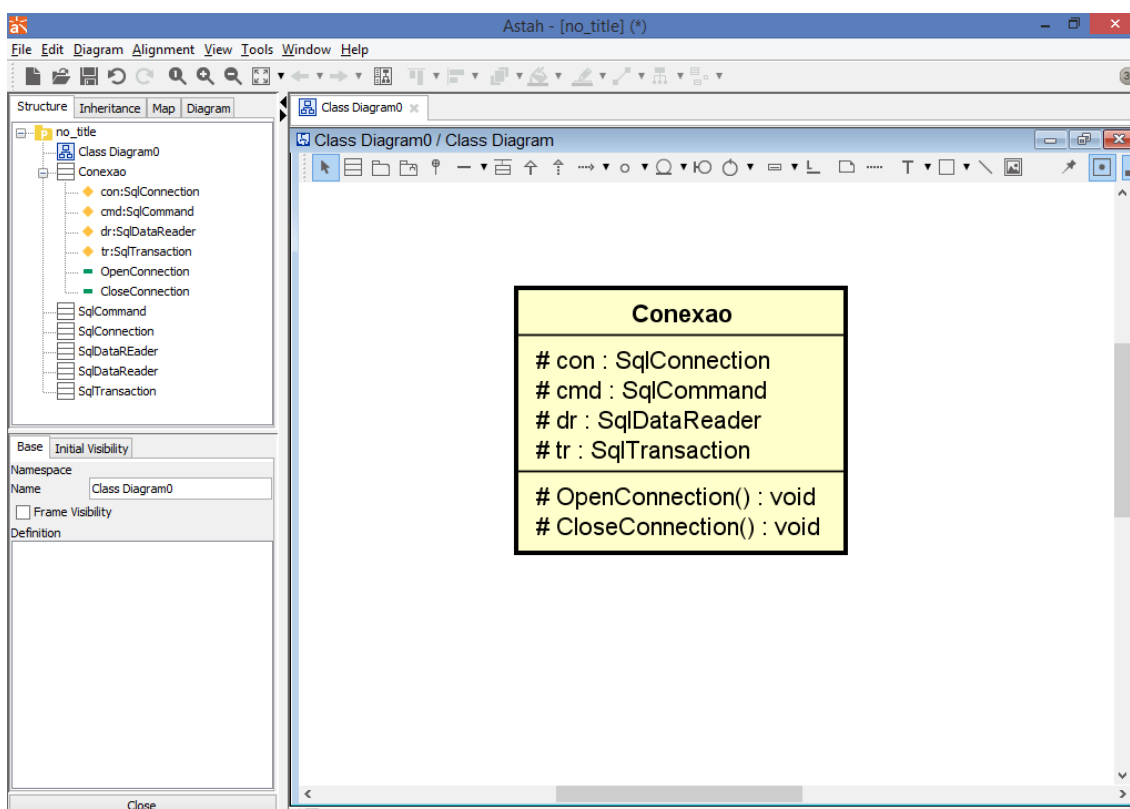


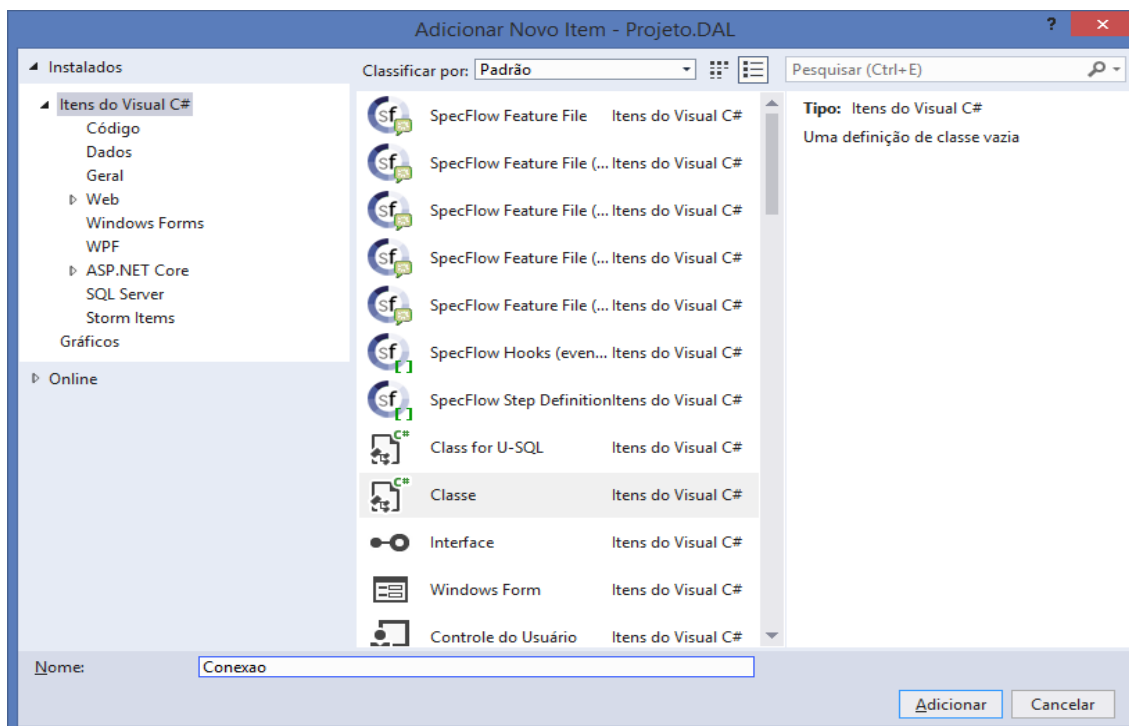
Tabela criada:



Criando uma classe de conexão com o banco de dados:



Repositorio de dados



System.Data.SqlClient

Namespace do .NET onde estão a maioria das classes voltadas para acesso a banco de dados do SqlServer.

SqlConnection

Classe para conexão com o banco de dados, através dela podemos criar programas que irão conectar e desconectar do banco de dados. Lembrando que, para que o SqlConnection possa estabelecer uma conexão com o BD, é necessário que tenhamos aConnectionString do banco

SqlCommand

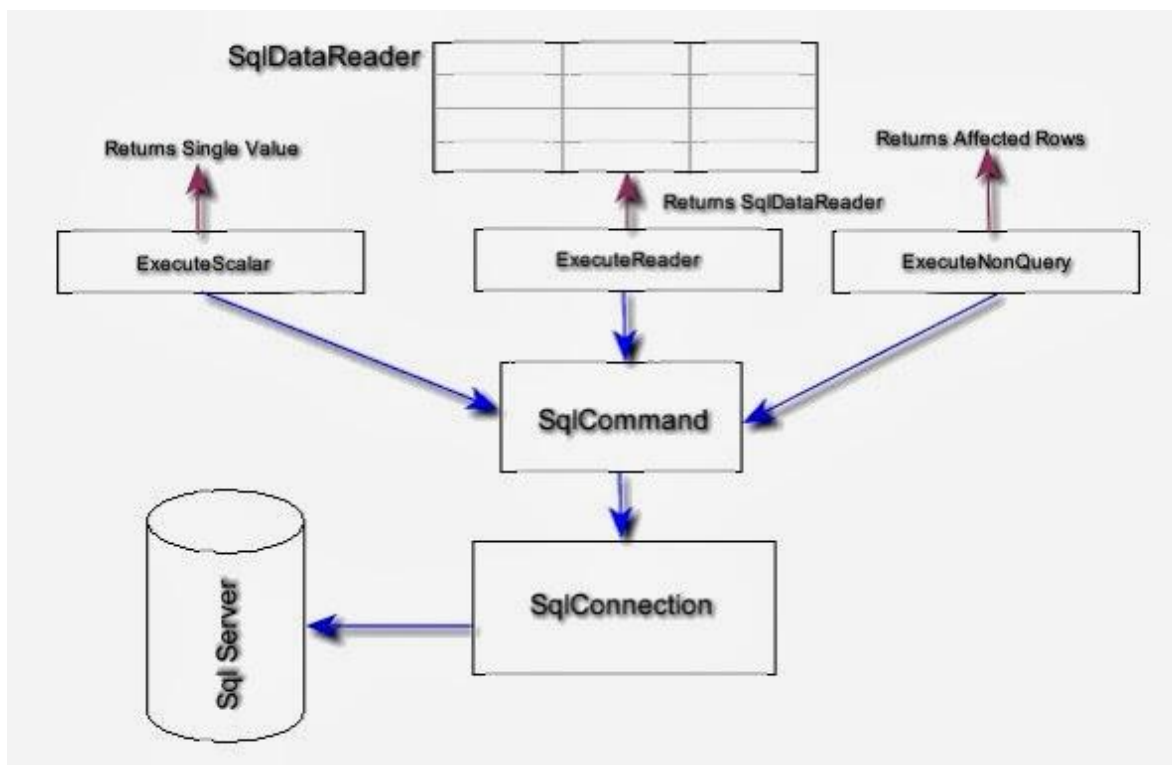
Classe para executar comandos SQL no banco de dados (INSERT, DELETE, UPDATE, SELECT etc. E até mesmo StoredProcedures.)

SqlDataReader

Classe utilizada para ler registros e resultados obtidos de consultas ao banco de dados. Todo comando do tipo SELECT irá precisar do SqlDataReader para ler o resultado obtido da consulta.

SqlTransaction

Classe utilizada para trabalhar com transações na base de dados, ou seja, comandos COMMIT ou ROLLBACK.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;

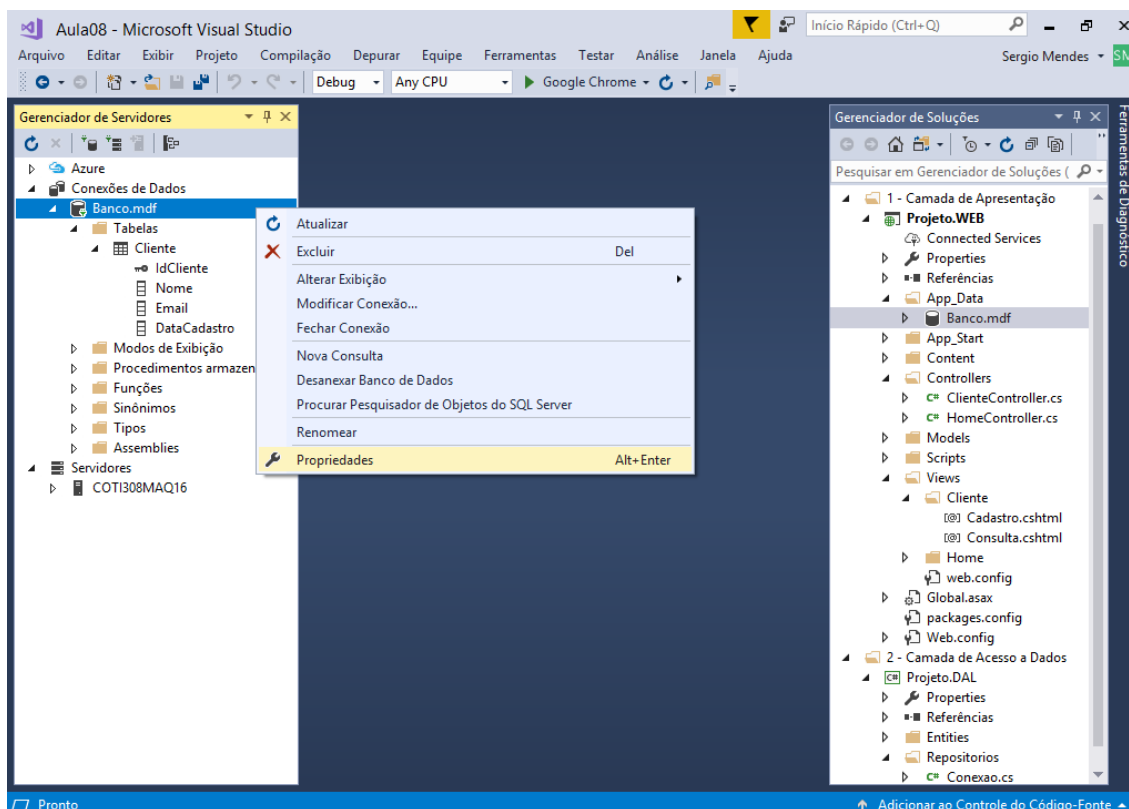
namespace Projeto.DAL.Repositorios
{
    public class Conexao
    {
        //atributos..
        //protected -> permite acesso por meio de herança
        protected SqlConnection con;
        protected SqlCommand cmd;
        protected SqlDataReader dr;
        protected SqlTransaction tr;

        //método para abrir conexão com o banco de dados..
        protected void OpenConnection()
        {
            con = new SqlConnection();
            con.Open(); //conectado..
        }

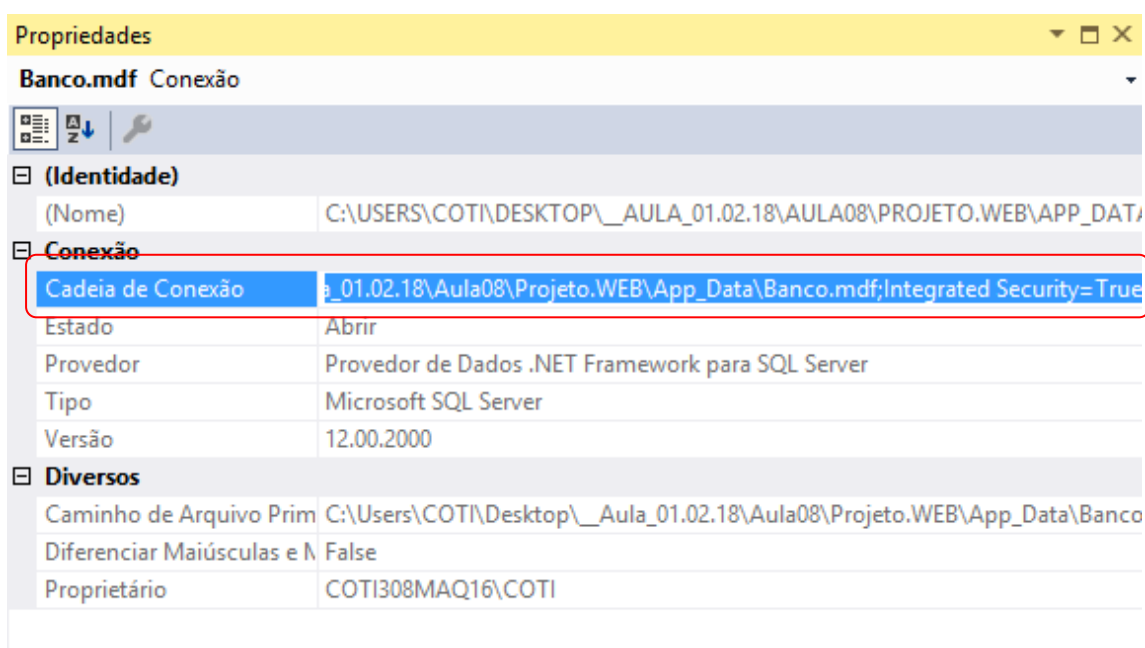
        //método para fechar a conexão com o banco de dados..
        protected void CloseConnection()
        {
            con.Close(); //desconectado..
        }
    }
}
```

ConnectionString

É uma linha de texto que contém todas as informações necessárias para que o .NET possa conectar-se a uma base de dados.



Copie a connectionstring:



Web.config.xml

Principal arquivo de configuração do projeto Asp.Net, podemos utiliza-lo para armazenar o endereço da connectionstring de um banco de dados de modo que a classe de conexão localizada no projeto DAL possa obter deste XML o endereço necessario para conectar-se no banco de dados.

```
<!-- Mapeamento da connectonstring -->
<connectionStrings>
  <add
    name="aula08"
    connectionString="Data Source=(LocalDB)\MSSQLLocalDB;
      AttachDbFilename=C:\Users\COTI\Desktop\__Aula_01.02.18\
      Aula08\Projeto.WEB\App_Data\Banco.mdf;Integrated Security=True"
  />
</connectionStrings>

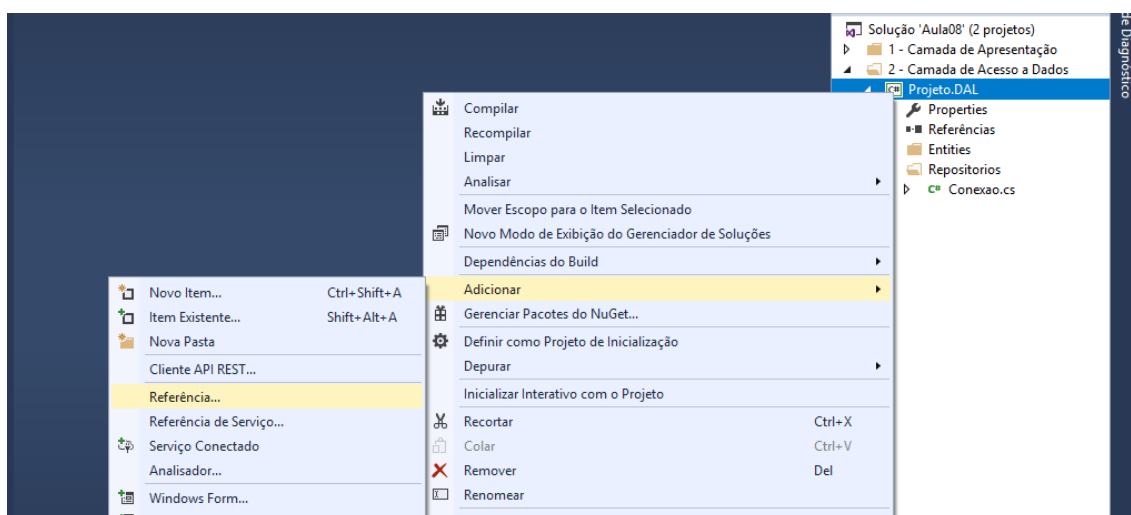
-----

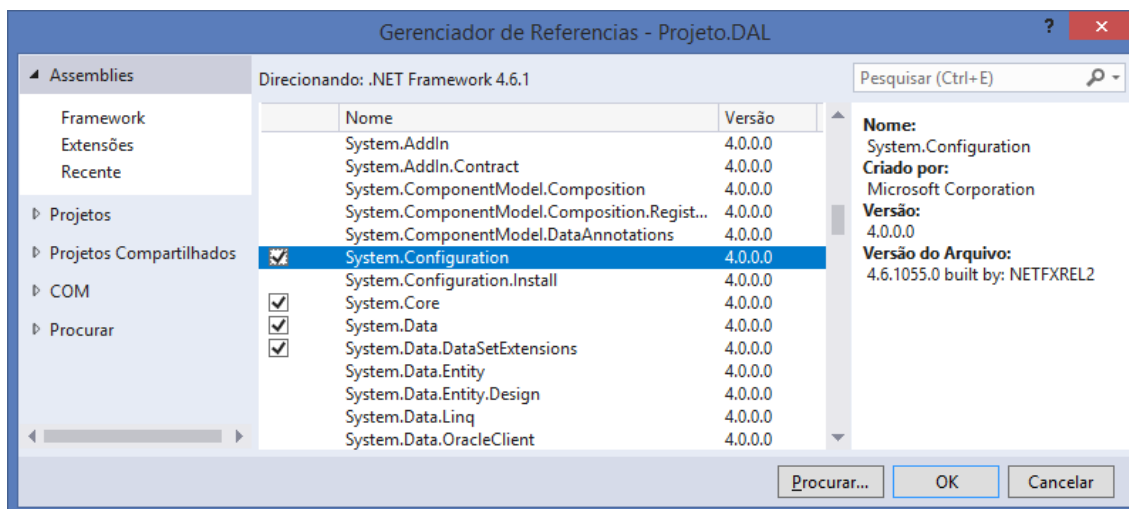
<?xml version="1.0" encoding="utf-8"?>
<!--
  Para obter mais informações sobre como configurar seu aplicativo ASP.NET,
  visite
  https://go.microsoft.com/fwlink/?LinkId=301880
-->
<configuration>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  </appSettings>
  <system.web>
    <compilation debug="true" targetFramework="4.6.1" />
    <httpRuntime targetFramework="4.6.1" />
  </system.web>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Helpers"
          publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.WebPages"
          publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="0.0.0.0-1.0.0.0" newVersion="1.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Mvc" publicKeyToken="31bf3856ad364e35"
          />
        <bindingRedirect oldVersion="1.0.0.0-5.2.3.0" newVersion="5.2.3.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Razor"
          publicKeyToken="31bf3856ad364e35" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-1.0.0.0" newVersion="1.0.0.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

```
<dependentAssembly>
  <assemblyIdentity name="System.Web.WebPages.Razor"
    publicKeyToken="31bf3856ad364e35" culture="neutral" />
  <bindingRedirect oldVersion="0.0.0.0-1.0.0.0" newVersion="1.0.0.0" />
</dependentAssembly>
</assemblyBinding>
</runtime>
<system.codedom>
  <compilers>
    <compiler language="c#;cs;csharp" extension=".cs"
      type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider,
      Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=1.0.7.0,
      Culture=neutral, PublicKeyToken=31bf3856ad364e35" warningLevel="4"
      compilerOptions="/langversion:default /nowarn:1659;1699;1701" />
    <compiler language="vb;vbs;visualbasic;vbscript" extension=".vb"
      type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.VBCodeProvider,
      Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=1.0.7.0,
      Culture=neutral, PublicKeyToken=31bf3856ad364e35" warningLevel="4"
      compilerOptions="/langversion:default /nowarn:41008
      /define:_MYTYPE=&quot;Web&quot; /optionInfer+" />
  </compilers>
</system.codedom>

<!-- Mapeamento da connectonstring -->
<connectionStrings>
  <add
    name="aula08"
    connectionString="Data Source=(LocalDB)\MSSQLLocalDB;
    AttachDbFilename=C:\Users\COTI\Desktop\__Aula_01.02.18\
    Aula08\Projeto.WEB\App_Data\Banco.mdf;Integrated Security=True"
  />
</connectionStrings>
</configuration>
```

Para que o projeto DAL possa obter a connectionstring mapeada no arquivo Web.config.xml é necessário que adicionemos uma biblioteca no projeto DAL denominada **System.Configuration**





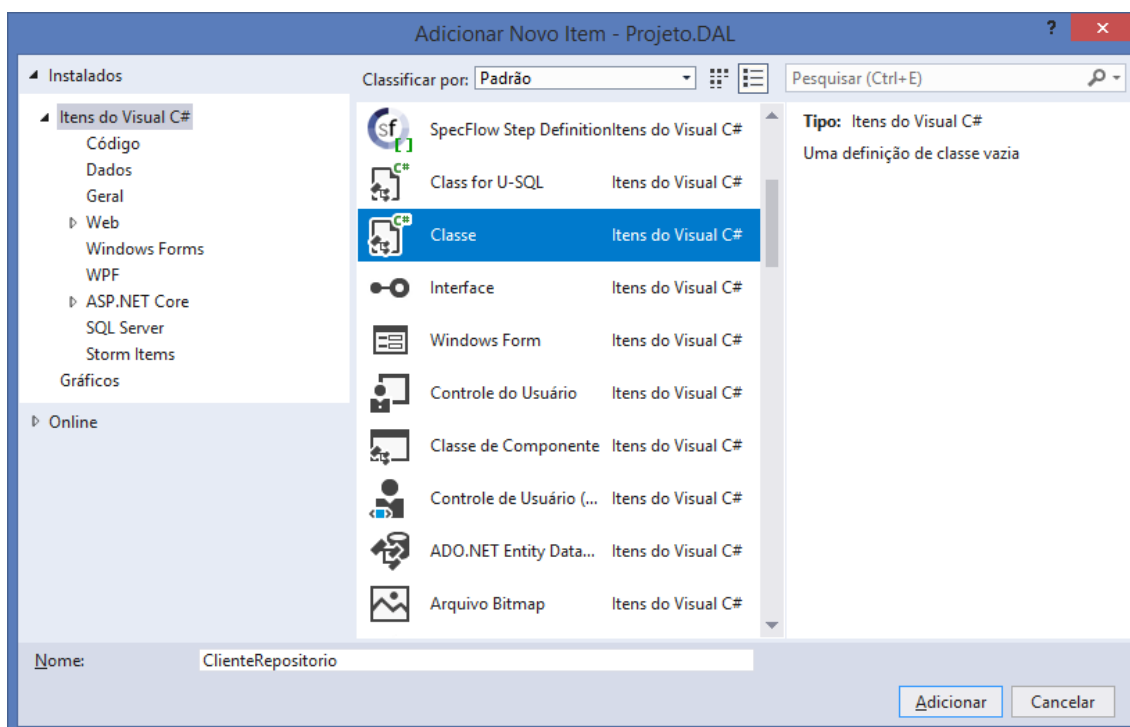
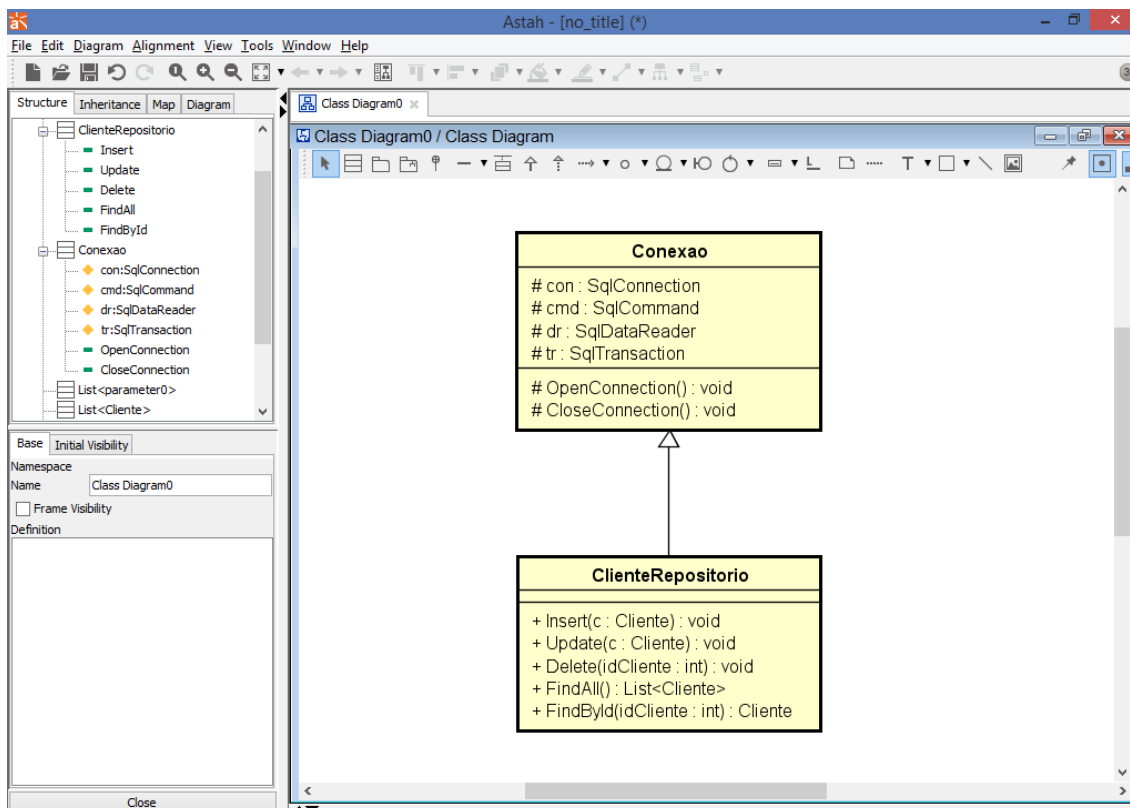
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;
using System.Configuration;

namespace Projeto.DAL.Repositorios
{
    public class Conexao
    {
        //atributos..
        //protected -> permite acesso pormeio de herança
        protected SqlConnection con;
        protected SqlCommand cmd;
        protected SqlDataReader dr;
        protected SqlTransaction tr;

        //método para abrir conexão com o banco de dados..
        protected void OpenConnection()
        {
            con = new SqlConnection(ConfigurationManager.ConnectionStrings
                ["aula08"].ConnectionString);
            con.Open(); //conectado..
        }

        //método para fechar a conexão com o banco de dados..
        protected void CloseConnection()
        {
            con.Close(); //desconectado..
        }
    }
}
```

Criando a classe de repositório para cliente:



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
  
```



```
using System.Threading.Tasks;
using System.Data.SqlClient; //acesso ao banco de dados..
using Projeto.DAL.Entities; //classes de entidade..

namespace Projeto.DAL.Repositorios
{
    //classe para repositório de dados (CRUD)
    //com a tabela/entidade de Cliente
    public class ClienteRepositorio : Conexao
    {
        //método para inserir um cliente na base de dados..
        public void Insert(Cliente c)
        {
            OpenConnection(); //abrindo conexão com o banco de dados..

            //escrevendo a query SQL..
            string query = "insert into Cliente(Nome, Email, DataCadastro) "
                + "values(@Nome, @Email, GetDate())";

            //criando e executando o comando SQL..
            cmd = new SqlCommand(query, con);
            cmd.Parameters.AddWithValue("@Nome", c.Nome);
            cmd.Parameters.AddWithValue("@Email", c.Email);
            cmd.ExecuteNonQuery(); //executando..

            CloseConnection(); //fechando conexão..
        }
    }
}
```

Implementando um método para verificar se um determinado Email já está cadastrado na tabela de Cliente;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient; //acesso ao banco de dados..
using Projeto.DAL.Entities; //classes de entidade..

namespace Projeto.DAL.Repositorios
{
    //classe para repositório de dados (CRUD)
    //com a tabela/entidade de Cliente
    public class ClienteRepositorio : Conexao
    {
        //método para inserir um cliente na base de dados..
        public void Insert(Cliente c)
        {
            OpenConnection(); //abrindo conexão com o banco de dados..

            //escrevendo a query SQL..
            string query = "insert into Cliente(Nome, Email, DataCadastro) "
                + "values(@Nome, @Email, GetDate())";
```

```
//criando e executando o comando SQL..
cmd = new SqlCommand(query, con);
cmd.Parameters.AddWithValue("@Nome", c.Nome);
cmd.Parameters.AddWithValue("@Email", c.Email);
cmd.ExecuteNonQuery(); //executando..

CloseConnection(); //fechando conexão..
}

//método para verificar se um email ja
//esta cadastrado na tabela de cliente..
public bool HasEmail(string email)
{
    OpenConnection(); //abrir conexão com o banco de dados..

    string query = "select count(*) from Cliente "
        + "where Email = @Email";

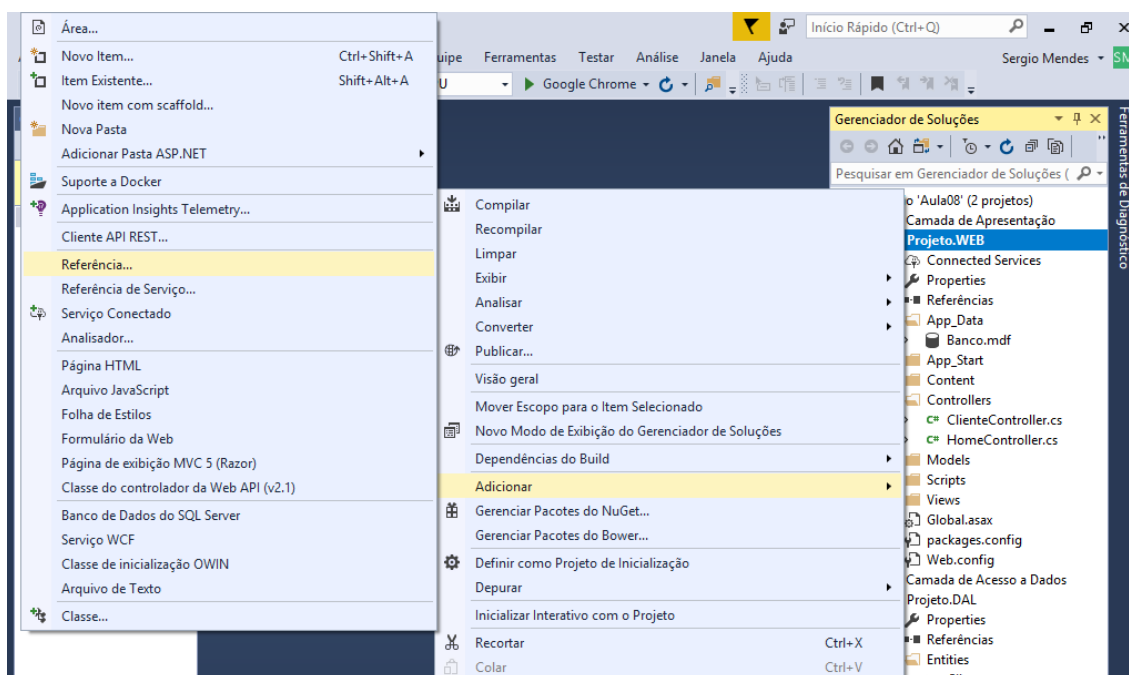
    cmd = new SqlCommand(query, con);
    cmd.Parameters.AddWithValue("@Email", email);
    int count = Convert.ToInt32(cmd.ExecuteScalar());

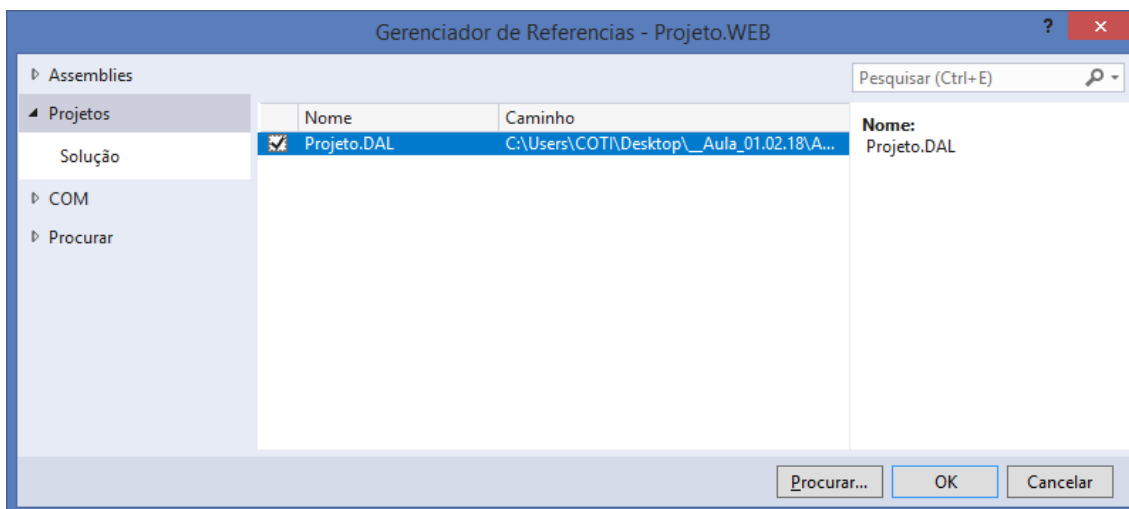
    CloseConnection(); //fechar conexão com o banco de dados..

    //retornando true / false
    return count > 0;
}
}
```

Voltando ao projeto Asp.Net MVC

Adicionando referencia no projeto MVC para o projeto DAL





Voltando ao ClienteController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Projeto.WEB.Models; //classes de modelo..
using Projeto.DAL.Entities; //classes de entidade..
using Projeto.DAL.Repositorios; //classes de acesso ao banco de dados..

namespace Projeto.WEB.Controllers
{
    public class ClienteController : Controller
    {
        // GET: Cliente/Cadastro
        public ActionResult Cadastro()
        {
            return View();
        }

        // POST: Cliente/Cadastro
        [HttpPost] //recebe requisições HTTP POST (formulários)
        public ActionResult Cadastro(ClienteCadastroModel model)
        {
            //verificar se os dados obtidos pela classe model
            //estão corretos (passaram nas validações?)
            if (ModelState.IsValid)
            {
                try
                {
                    Cliente c = new Cliente(); //instanciando..
                    c.Nome = model.Nome;
                    c.Email = model.Email;

                    ClienteRepositorio rep = new ClienteRepositorio();
                    rep.Insert(c);

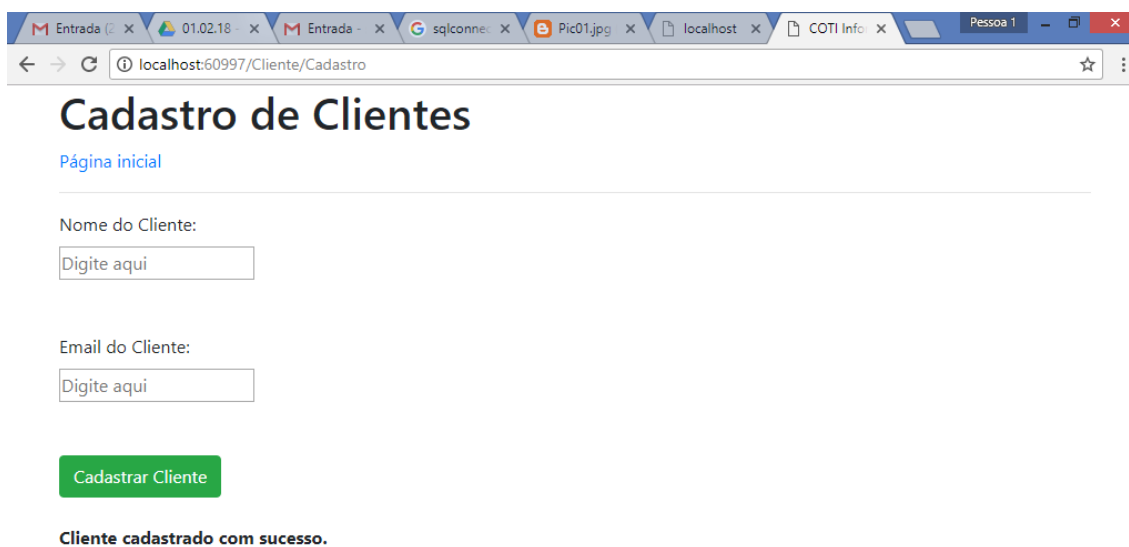
                    //criando uma mensagem que será exibida na página..
                    ViewBag.Mensagem = "Cliente cadastrado com sucesso.";
                }
            }
        }
    }
}
```

```
//limpar os campos do formulário..
ModelState.Clear();
}
catch(Exception e)
{
    //gerando uma mensagem de erro..
    ViewBag.Mensagem = e.Message;
}
}

return View();
}

// GET: Cliente/Consulta
public ActionResult Consulta()
{
    return View();
}
}
```

Executando:



Cadastro de Clientes

[Página inicial](#)

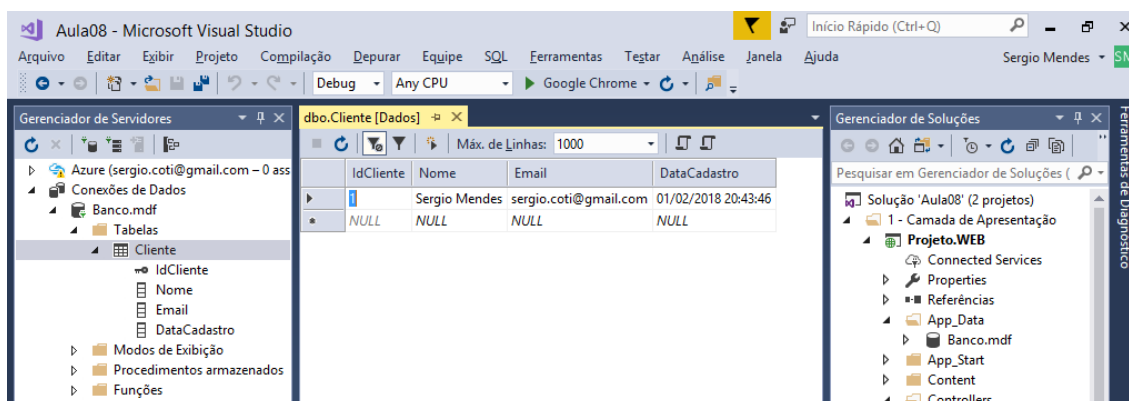
Nome do Cliente:

Email do Cliente:

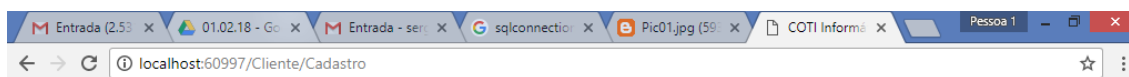
Cadastrar Cliente

Cliente cadastrado com sucesso.

No banco de dados:



IdCliente	Nome	Email	DataCadastro
1	Sergio Mendes	sergio.coti@gmail.com	01/02/2018 20:43:46



Cadastro de Clientes

[Página inicial](#)

Nome do Cliente:

Sergio Mendes

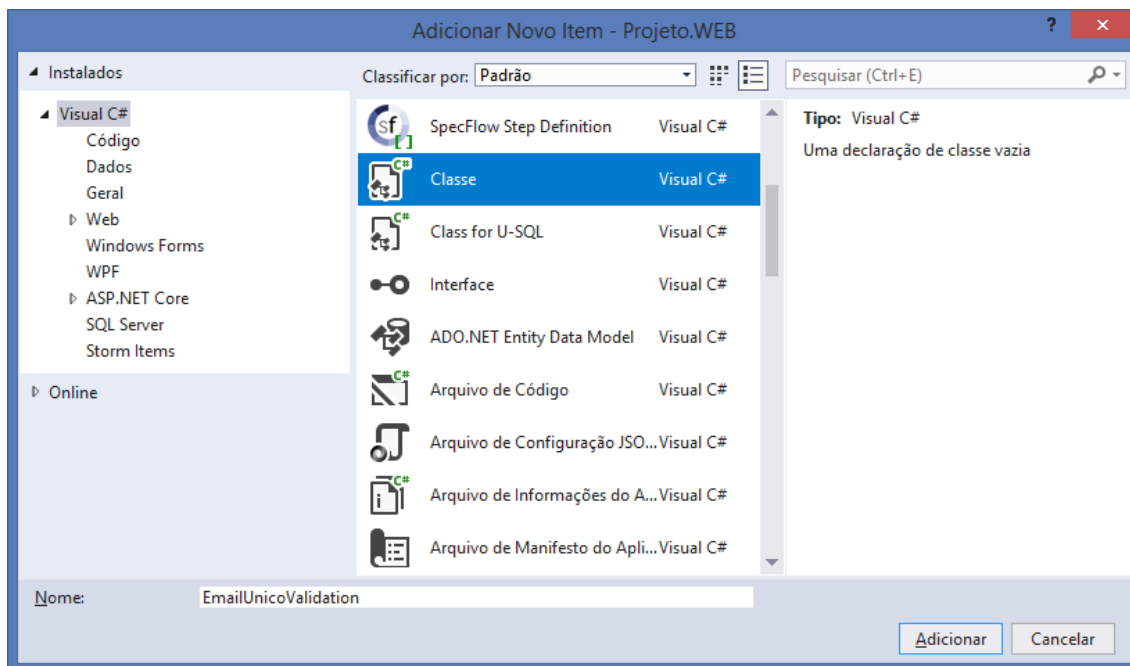
Email do Cliente:

sergio.coti@gmail.com

Cadastrar Cliente

Violation of UNIQUE KEY constraint 'UQ_Cliente_A9D105340481EC9B'. Cannot insert duplicate key in object 'dbo.Cliente'. The duplicate key value is (sergio.coti@gmail.com). The statement has been terminated.

Criando uma classe para validação do email unico do cliente:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;
using Projeto.DAL.Repositorios;

namespace Projeto.WEB.Validations
{
```

```
//Regra 1) Herdar a classe ValidationAttribute
public class EmailUnicoValidation : ValidationAttribute
{
    //Regra 2) Sobrescrever o método IsValid
    //pertencente a classe ValidationAttribute
    public override bool IsValid(object value)
    {
        //verificando se o objeto não está vazio e se é do tipo string
        if(value != null && value is string)
        {
            //converter para string..
            string email = (string) value;

            //verificar no banco de dados se o email não existe..
            ClienteRepositorio rep = new ClienteRepositorio();
            //retornando se o email não existe no banco..
            return ! rep.HasEmail(email);
        }
        return false;
    }
}
```

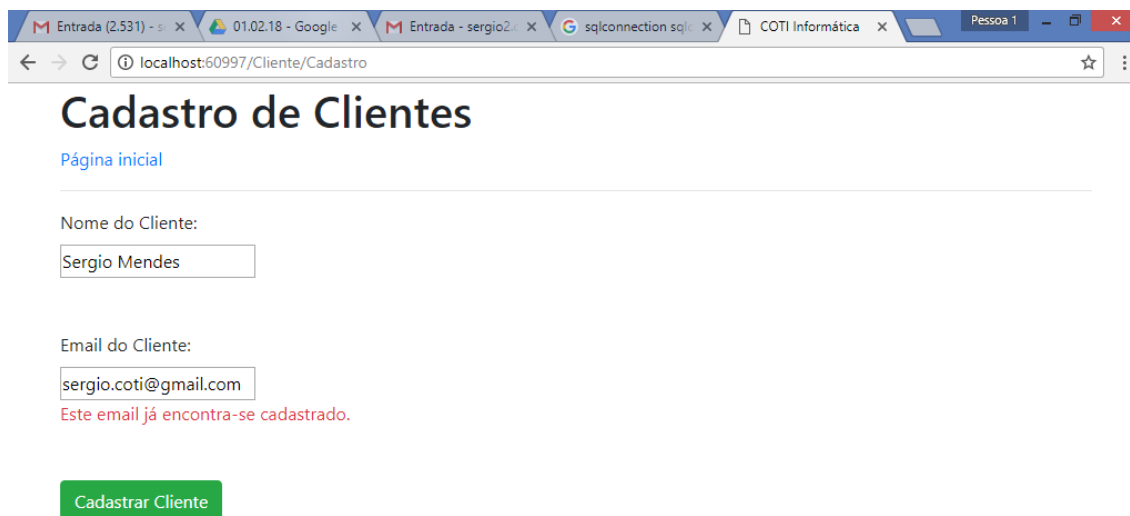
Aplicando o validador na classe de modelo:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations; //mapeamentos..
using Projeto.WEB.Validations; //classes de validação..

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; }

        [EmailUnicoValidation(ErrorMessage = "Este email já  
encontra-se cadastrado.")]
        [EmailAddress(ErrorMessage = "Por favor, informe um  
endereço de email válido.")]
        [Required(ErrorMessage = "Por favor, informe o email do cliente.")]
        public string Email { get; set; }
    }
}
```

Executando:



Voltando na classe ClienteRepositorio

Programando os demais métodos de CRUD

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient; //acesso ao banco de dados..
using Projeto.DAL.Entities; //classes de entidade..

namespace Projeto.DAL.Repositorios
{
    //classe para repositório de dados (CRUD)
    //com a tabela/entidade de Cliente
    public class ClienteRepositorio : Conexao
    {
        //método para inserir um cliente na base de dados..
        public void Insert(Cliente c)
        {
            OpenConnection(); //abrindo conexão com o banco de dados..

            //escrevendo a query SQL..
            string query = "insert into Cliente(Nome, Email, DataCadastro) "
                + "values(@Nome, @Email, GetDate())";

            //criando e executando o comando SQL..
            cmd = new SqlCommand(query, con);
            cmd.Parameters.AddWithValue("@Nome", c.Nome);
            cmd.Parameters.AddWithValue("@Email", c.Email);
            cmd.ExecuteNonQuery(); //executando..

            CloseConnection(); //fechando conexão..
        }
    }
}
```

```
//método para verificar se um email ja esta
//cadastrado na tabela de cliente..
public bool HasEmail(string email)
{
    OpenConnection(); //abrir conexão com o banco de dados..

    string query = "select count(*) from Cliente "
        + "where Email = @Email";

    cmd = new SqlCommand(query, con);
    cmd.Parameters.AddWithValue("@Email", email);
    int count = Convert.ToInt32(cmd.ExecuteScalar());

    CloseConnection(); //fechar conexão com o banco de dados..

    //retornando true / false
    return count > 0;
}

//método para atualizar os dados do cliente..
public void Update(Cliente c)
{
    OpenConnection();

    string query = "update Cliente set Nome = @Nome, Email = @Email "
        + "where IdCliente = @IdCliente";

    cmd = new SqlCommand(query, con);
    cmd.Parameters.AddWithValue("@Nome", c.Nome);
    cmd.Parameters.AddWithValue("@Email", c.Email);
    cmd.Parameters.AddWithValue("@IdCliente", c.IdCliente);
    cmd.ExecuteNonQuery();

    CloseConnection();
}

//método para excluir um cliente pelo id..
public void Delete(int idCliente)
{
    OpenConnection();

    string query = "delete from Cliente where IdCliente = @IdCliente";

    cmd = new SqlCommand(query, con);
    cmd.Parameters.AddWithValue("@IdCliente", idCliente);
    cmd.ExecuteNonQuery();

    CloseConnection();
}

//método para listar todos os cliente..
public List<Cliente> FindAll()
{
    OpenConnection();

    string query = "select * from Cliente";

    cmd = new SqlCommand(query, con);
    dr = cmd.ExecuteReader();
}
```



```
//declarando uma lista de cliente..
List<Cliente> lista = new List<Cliente>();

while(dr.Read()) //percorrendo cada registro do SqlDataReader..
{
    Cliente c = new Cliente();
    c.IdCliente = Convert.ToInt32(dr["IdCliente"]);
    c.Nome = Convert.ToString(dr["Nome"]);
    c.Email = Convert.ToString(dr["Email"]);
    c.DataCadastro = Convert.ToDateTime(dr["DataCadastro"]);

    lista.Add(c); //adicionando o cliente na lista.
}

CloseConnection();
//retornando a lista..
return lista;
}
}
```

Continua...