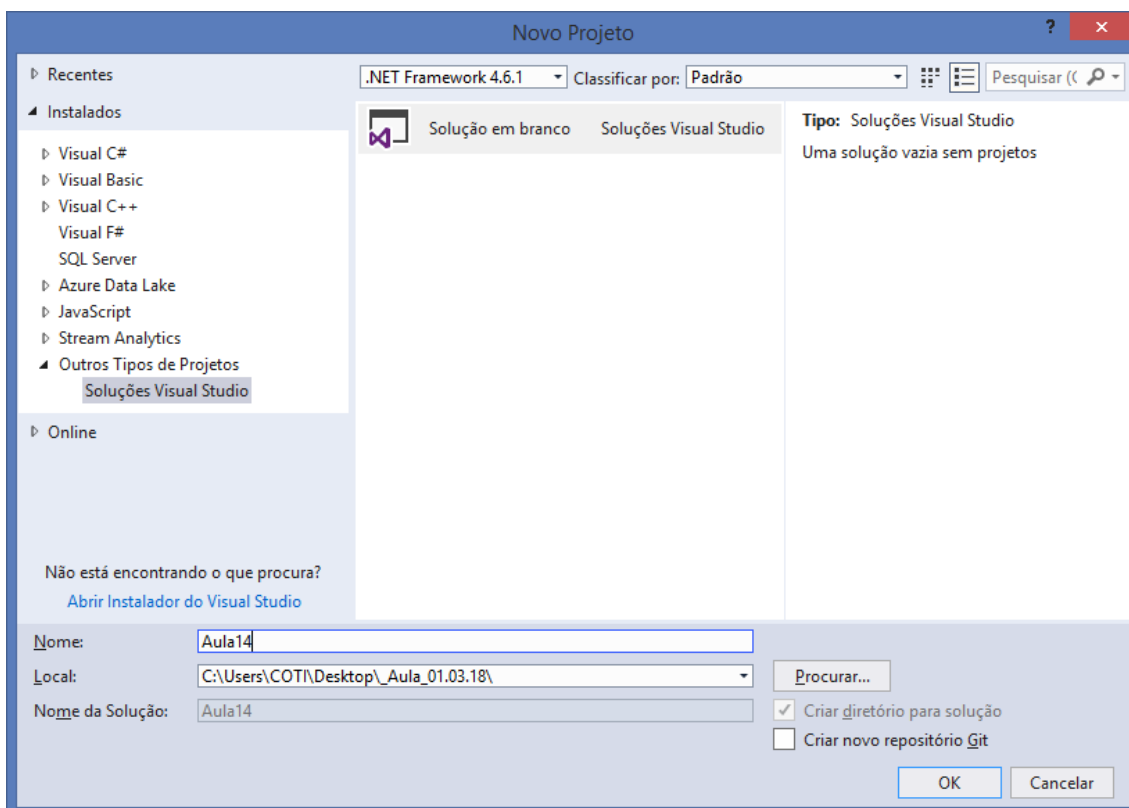
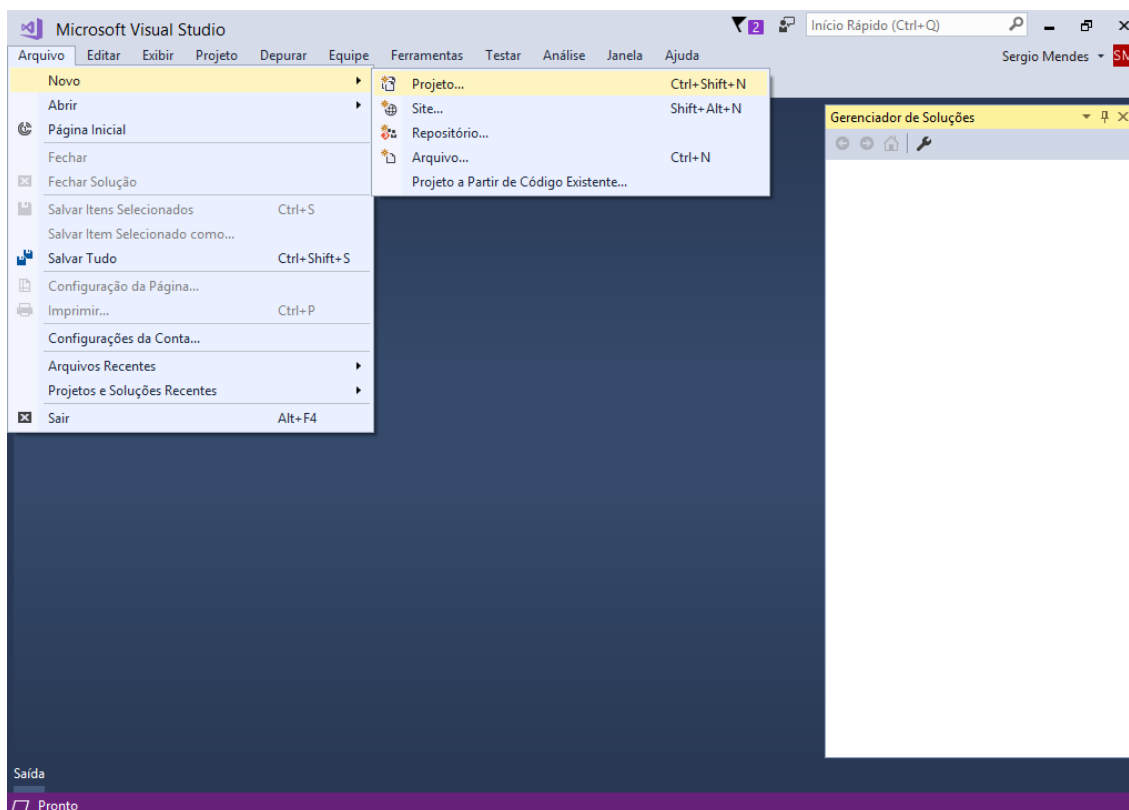
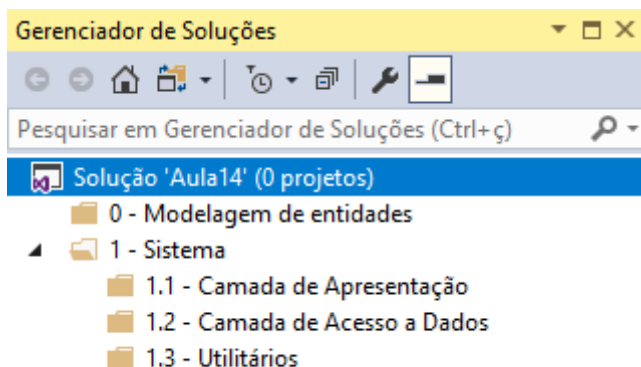


## Nova solução em branco:

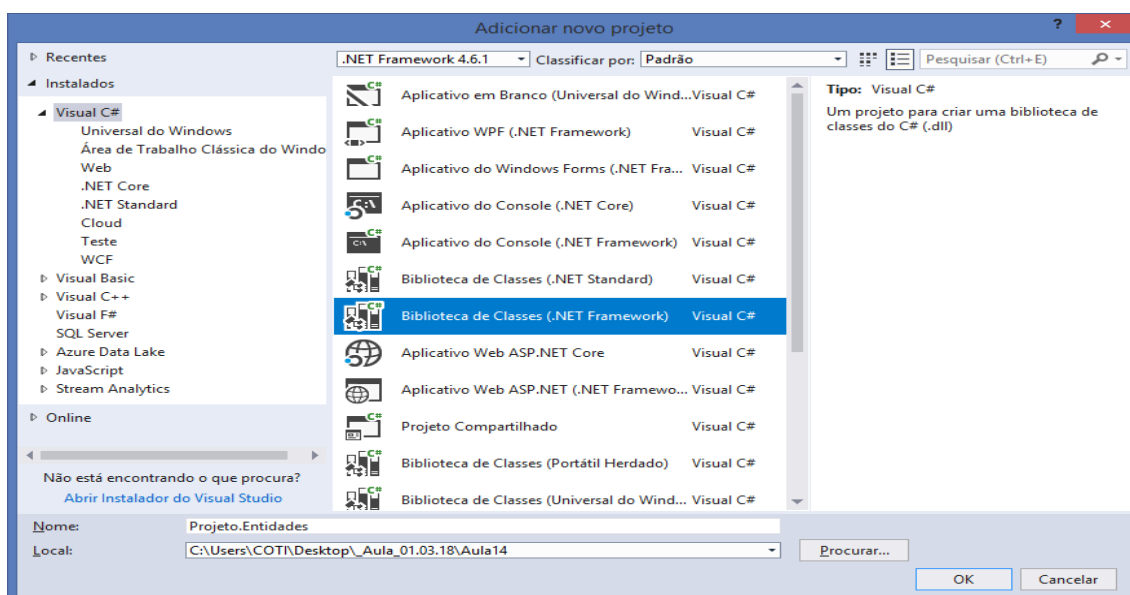


## Organização da Solution



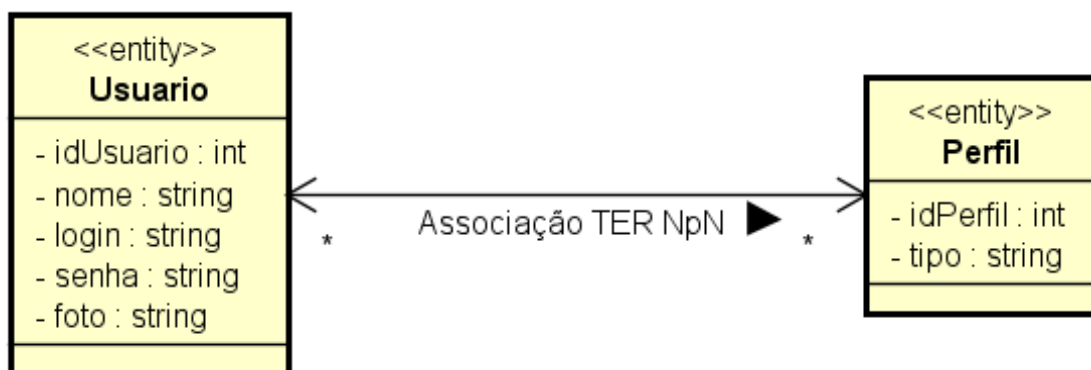
## 0 - Modelagem de entidades

Biblioteca de Classes (.NET Framework)



## Diagrama de Classes

Modelagem de Associação muitos para muitos



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

namespace Projeto.Entidades
{
    public class Usuario
    {
        public int IdUsuario { get; set; }
        public string Nome { get; set; }
        public string Login { get; set; }
        public string Senha { get; set; }
        public string Foto { get; set; }
        public DateTime DataCadastro { get; set; }

        //Associação (TER-MUITOS)
        public List<Perfil> Perfis { get; set; }

        public Usuario()
        {
        }

        public Usuario(int idUsuario, string nome, string login,
            string senha, string foto, DateTime dataCadastro)
        {
            IdUsuario = idUsuario;
            Nome = nome;
            Login = login;
            Senha = senha;
            Foto = foto;
            DataCadastro = dataCadastro;
        }

        public override string ToString()
        {
            return $"Id do Usuario: {IdUsuario}, Nome: {Nome},
                Login: {Login}, Data de cadastr: {DataCadastro}";
        }
    }
}
```

-----

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

namespace Projeto.Entidades
{
    public class Perfil
    {
        //propriedades
        public int IdPerfil { get; set; }
        public string Tipo { get; set; }
    }
}
```

```
//Relacionamento de Associação (TER-MUITOS)
public List<Usuario> Usuario { get; set; }

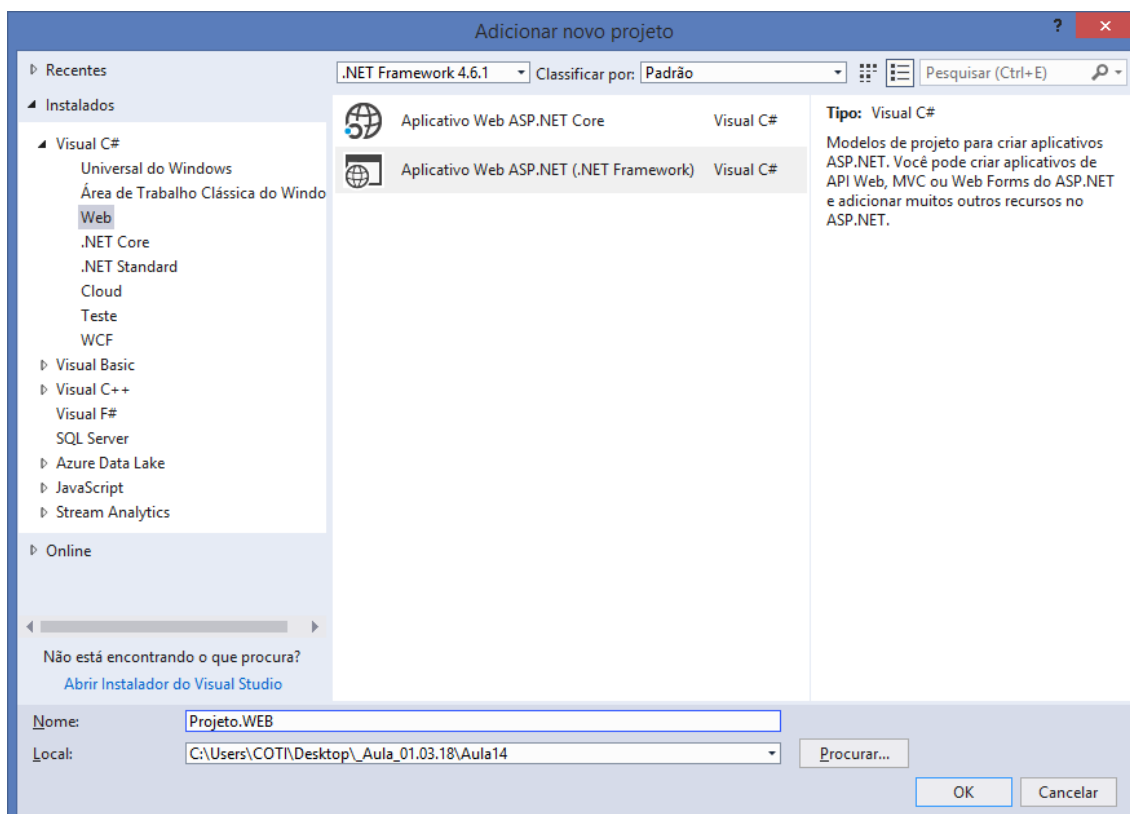
//construtor default..
public Perfil()
{
}

//sobrecarga de construtores
public Perfil(int idPerfil, string tipo)
{
    IdPerfil = idPerfil;
    Tipo = tipo;
}

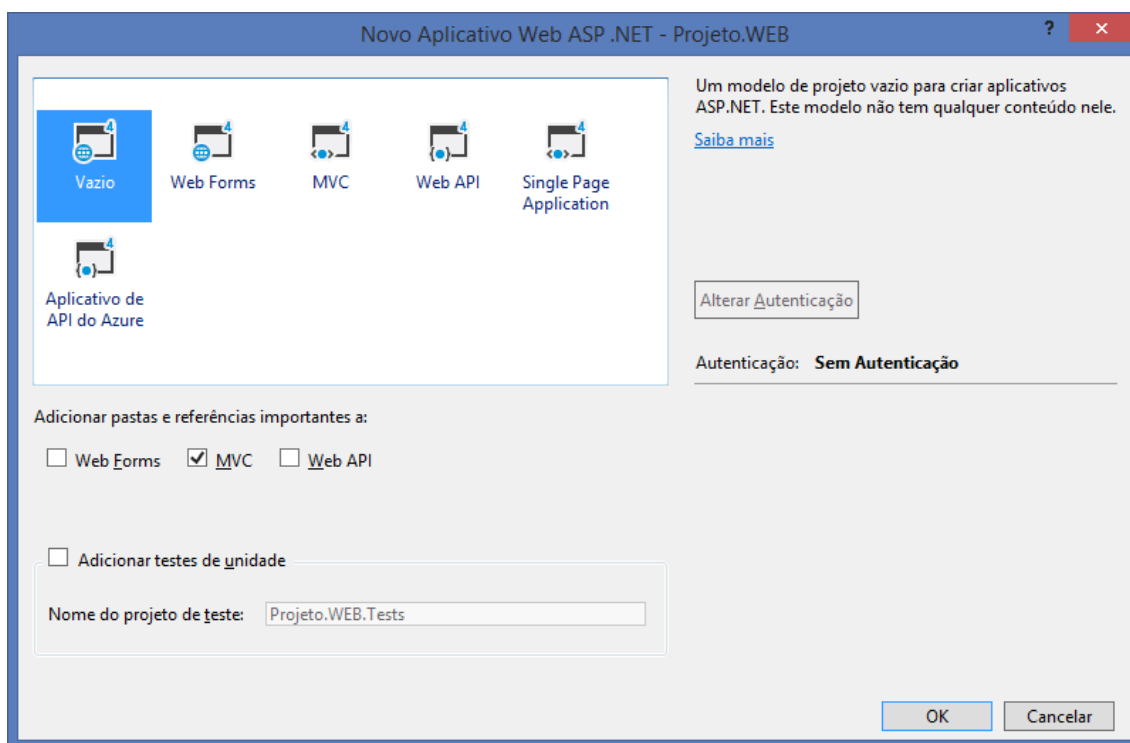
//sobrescrita do método toString..
public override string ToString()
{
    return $"Id do Perfil: {IdPerfil}, Tipo: {Tipo}";
}
}
}
```

## 1.1 - Camada de Apresentação

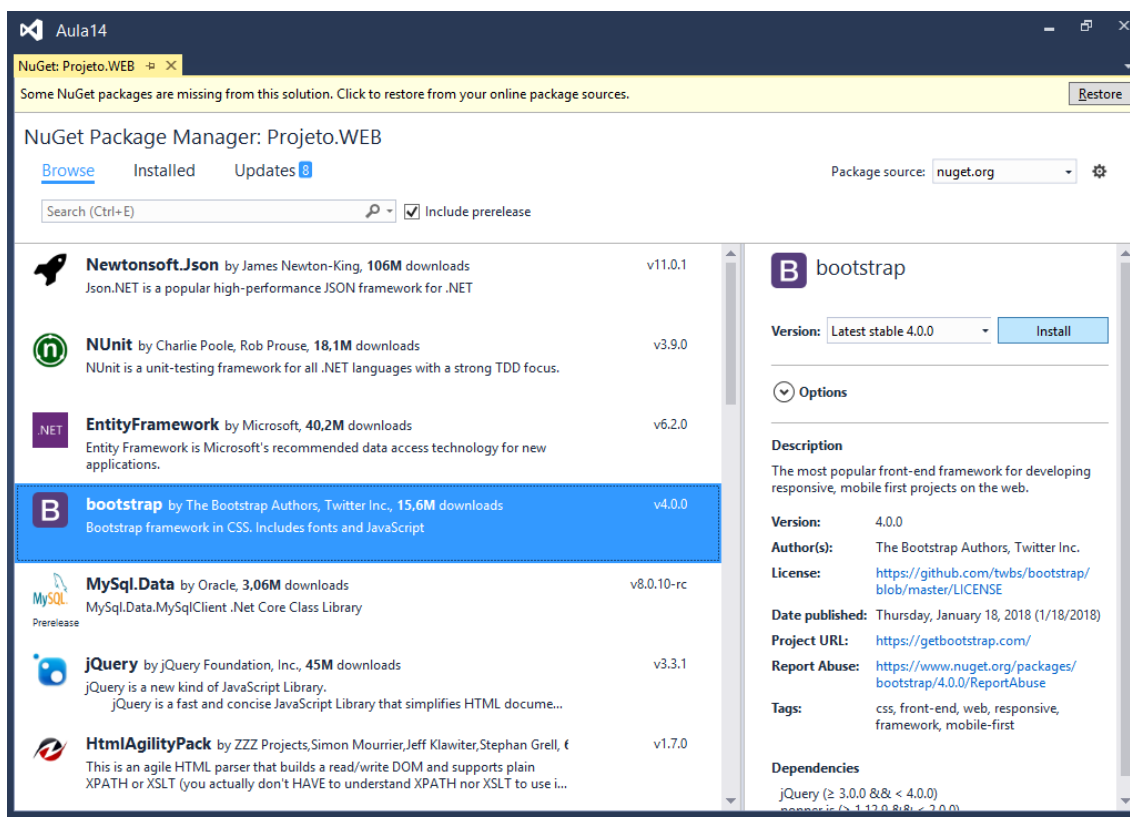
### Asp.Net MVC WebApplication



## Selecione empty / mvc

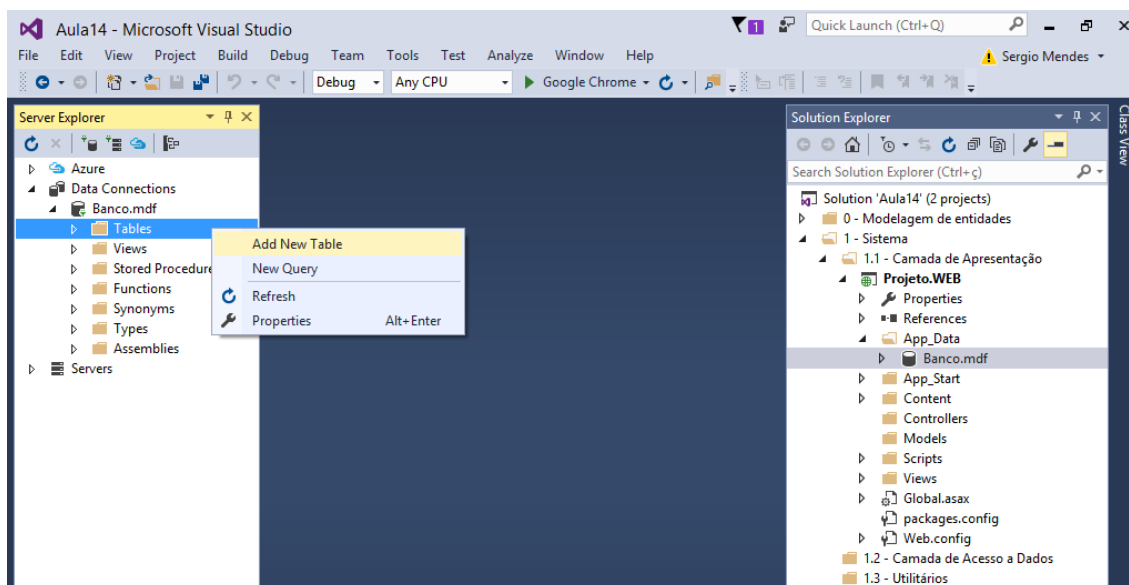
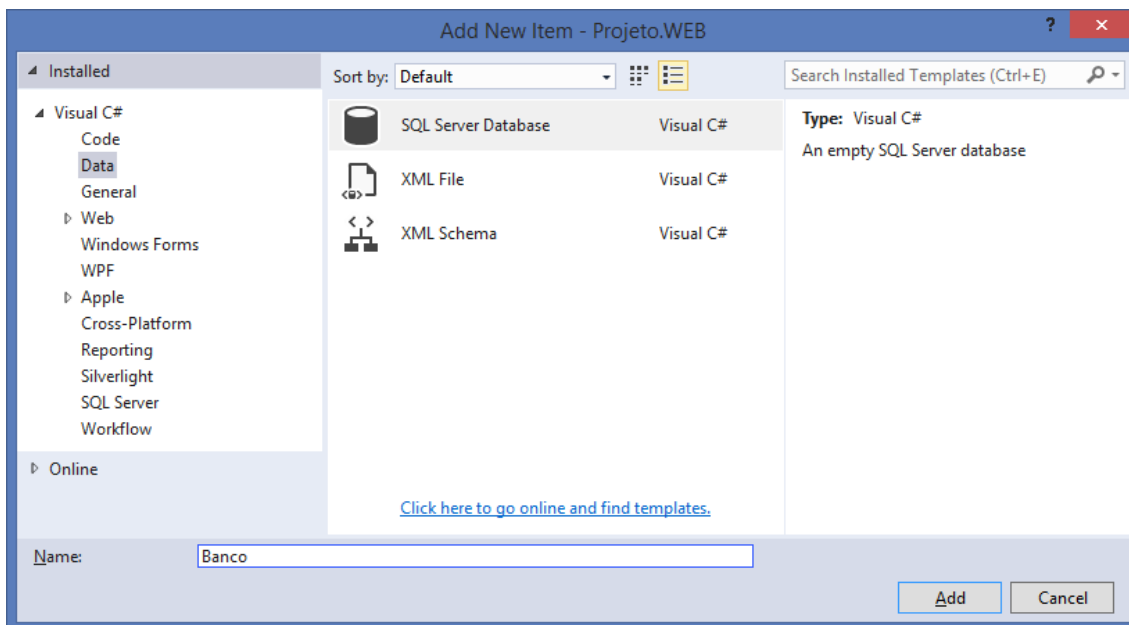


## Instalando o bootstrap: Gerenciador de pacotes do nuget



## Criando a base de dados

### MDF - Master Database File

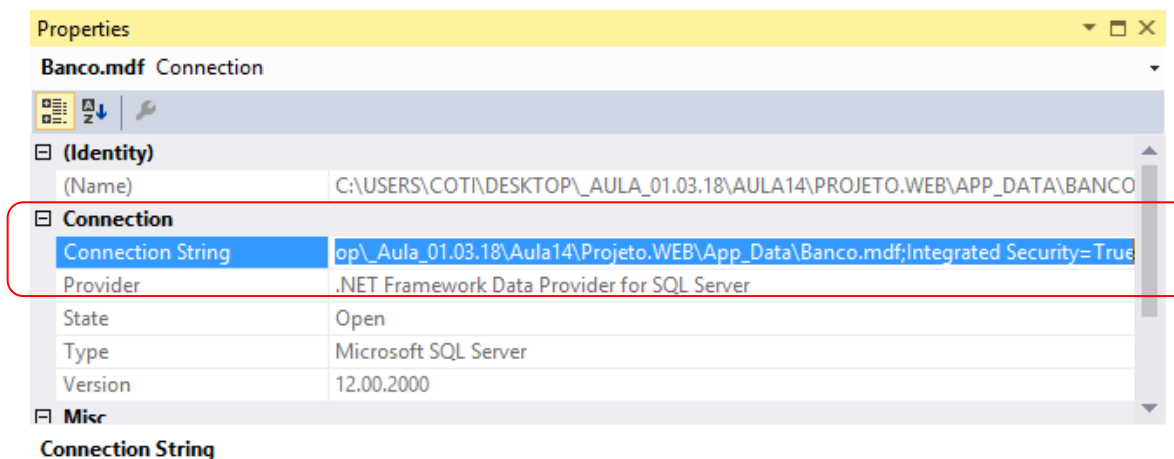
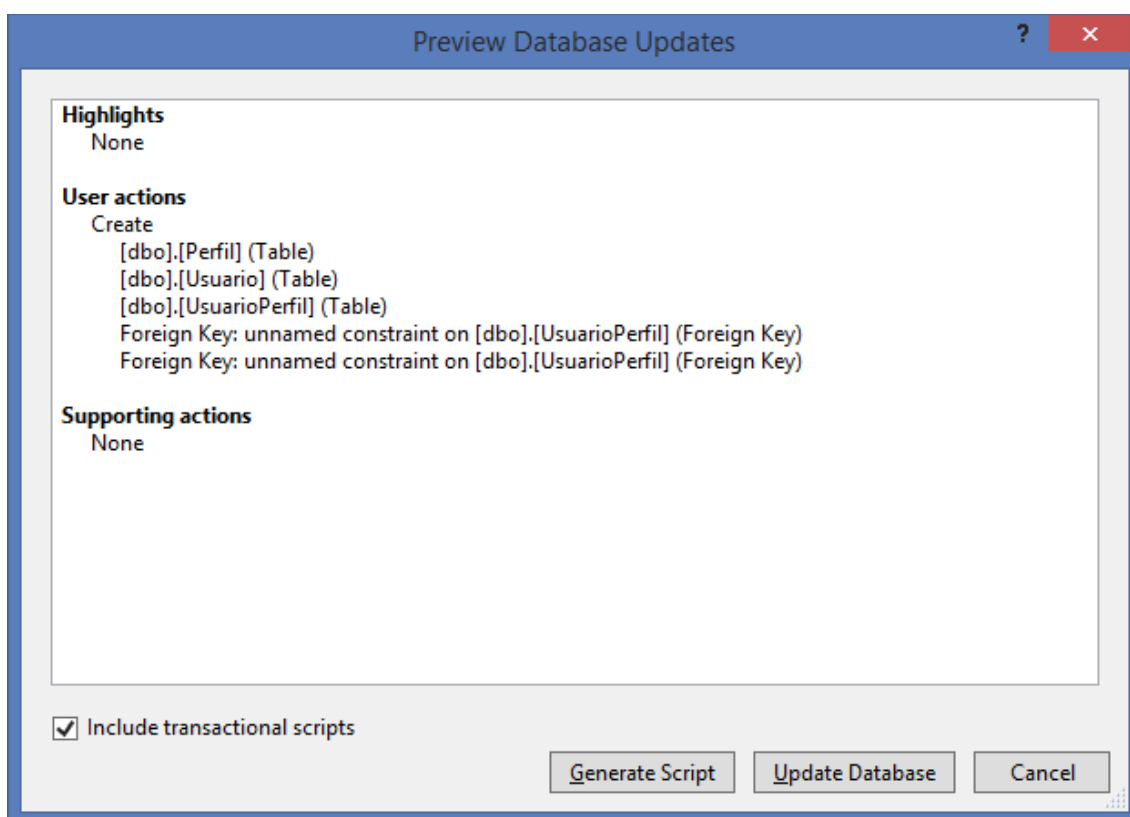


```
create table Perfil(
    IdPerfil      integer                identity(1,1),
    Tipo          nvarchar(50) not null,
    primary key(IdPerfil))
go

create table Usuario(
    IdUsuario     integer                identity(1,1),
    Nome          nvarchar(50) not null,
    [Login]       nvarchar(25) not null unique,
    Senha         nvarchar(50) not null,
    Foto          nvarchar(255) not null,
    primary key (IdUsuario));
go
```

```
--entidade associativa
create table UsuarioPerfil(
    IdUsuario    integer                not null,
    IdPerfil     integer                not null,
    primary key (IdUsuario, IdPerfil),
    foreign key (IdUsuario) references Usuario(IdUsuario),
    foreign key (IdPerfil) references Perfil(IdPerfil))
go
```

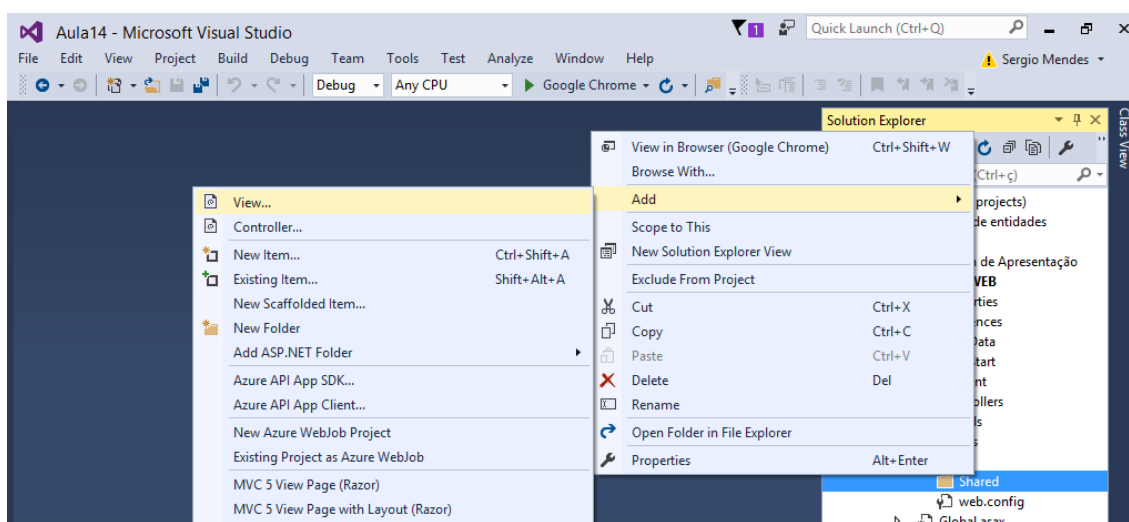
## Executando:



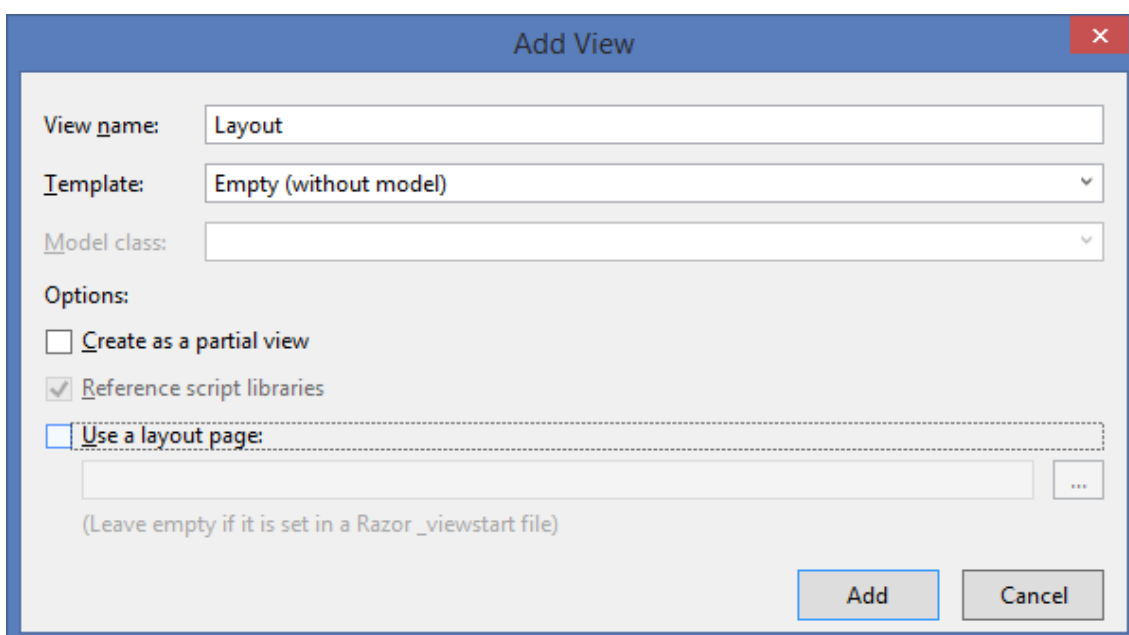
## \Web.config.xml

```
<connectionStrings>
  <add
    name="aula"
    connectionString="Data Source=(LocalDB)\MSSQLLocalDB;
    AttachDbFilename=C:\Users\COTI\Desktop\_Aula_01.03.18\
    Aula14\Projeto.WEB\App_Data\Banco.mdf;Integrated Security=True"
  />
</connectionStrings>
```

Criando uma página de layout (/Views/)



Criando uma página de layout:





```
<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>COTI Informatica</title>

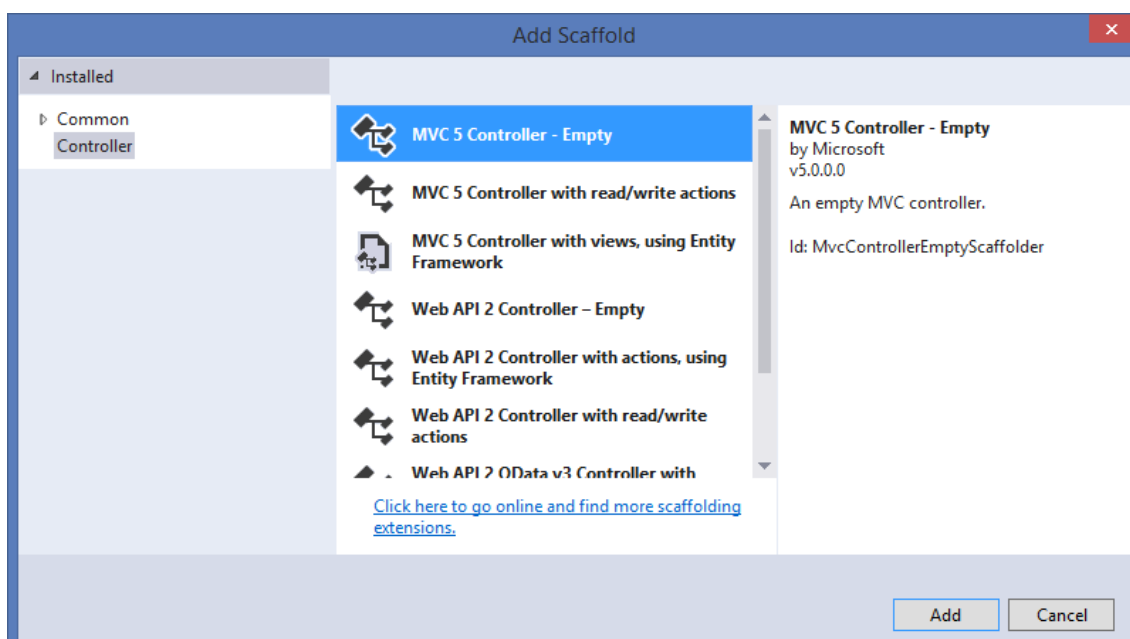
    <!-- folhas de estilo CSS do bootstrap -->
    <link href="~/Content/bootstrap.min.css" rel="stylesheet" />

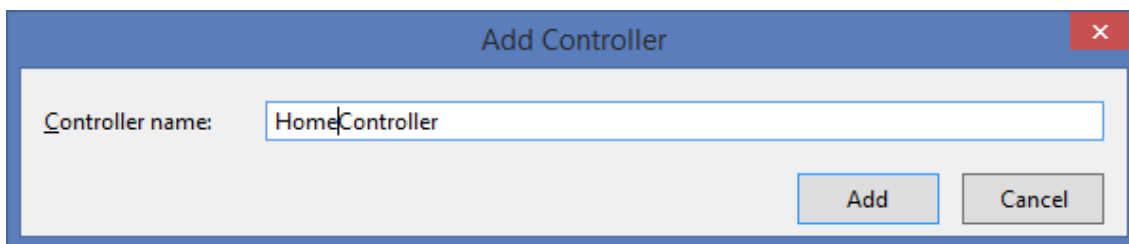
</head>
<body class="container">

    <div class="row">
        <div class="col-md-12">
            <h2>Sistema de Controle de Clientes</h2>
            REALIZE SEU LOGIN OU CADASTRE-SE
        </div>
    </div>

    <div class="row">
        <div class="col-md-12">
            <!-- Local para entrada do conteudo das demais páginas -->
            @RenderBody()
        </div>
    </div>

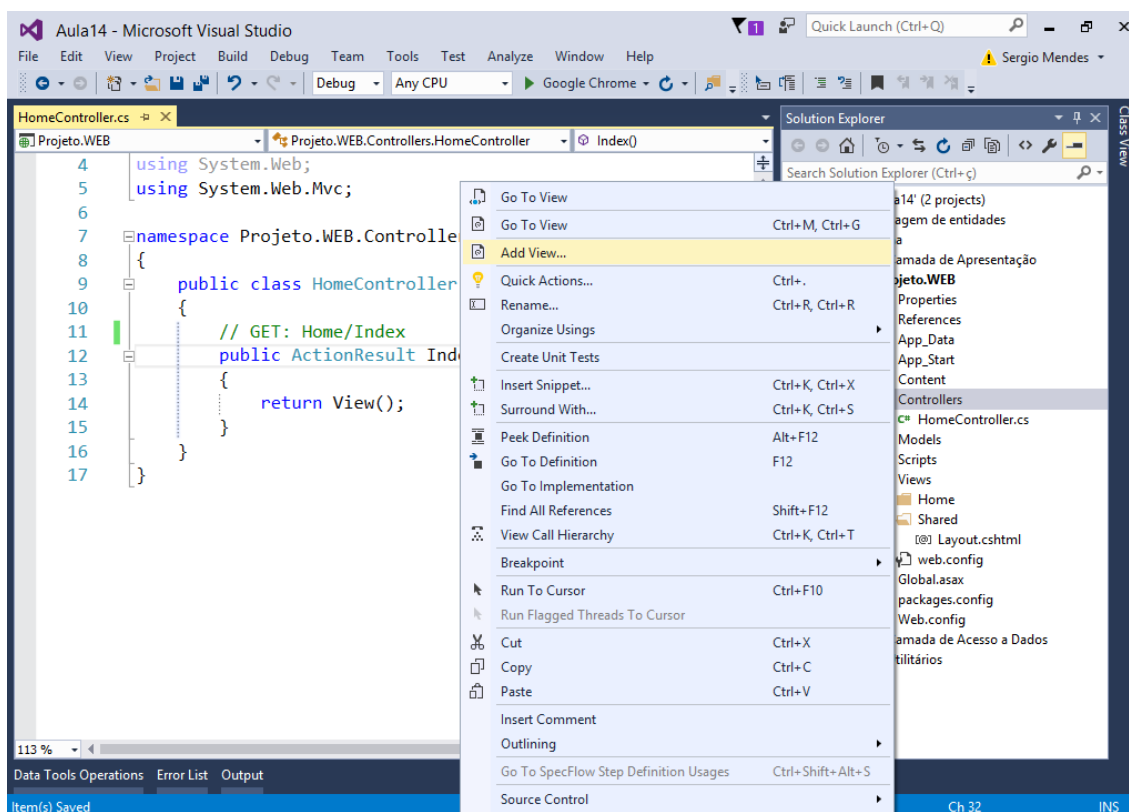
</body>
</html>
```

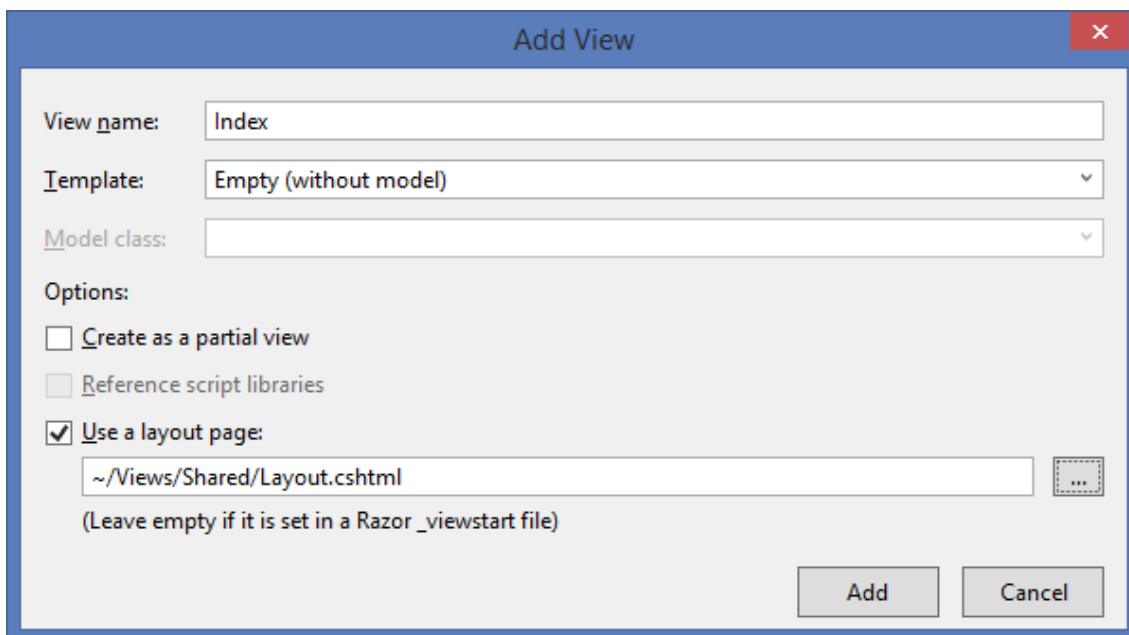




```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Projeto.WEB.Controllers
{
    public class HomeController : Controller
    {
        // GET: Home/Index
        public ActionResult Index()
        {
            return View();
        }
    }
}
```





**Add View**

View name:

Template:

Model class:

Options:

☐ Create as a partial view

☐ Reference script libraries

☒ Use a layout page:

(Leave empty if it is set in a Razor \_viewstart file)

@{

```
ViewBag.Title = "Index";
Layout = "~/Views/Shared/Layout.cshtml";
```

}

<h4>Bem vindo ao projeto</h4>

<hr/>

Selecione:

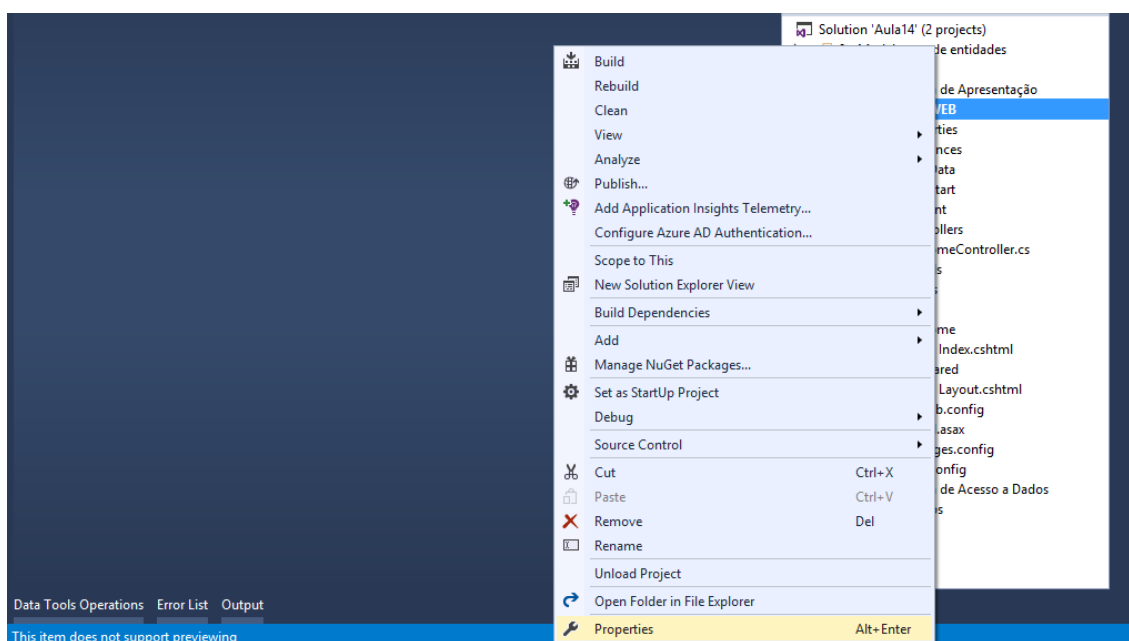
<hr />

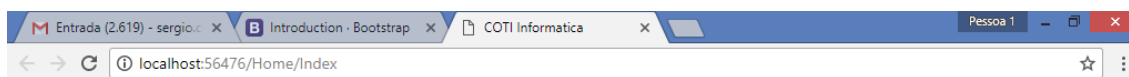
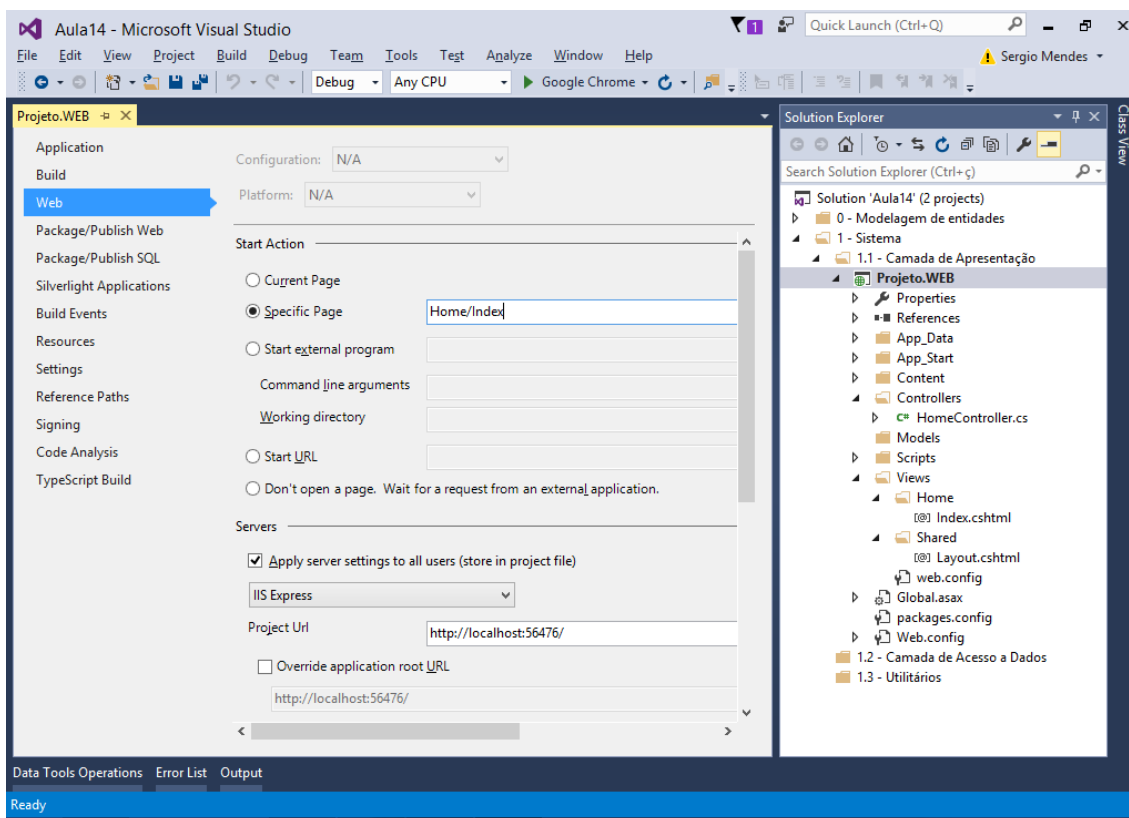
<ul>

<li> <a href="/Usuario/Login">Realizar Autenticação</a> </li>

<li> <a href="/Usuario/Cadastro">Cadastrar conta de Usuário</a> </li>

</ul>





## Sistema de Controle de Clientes

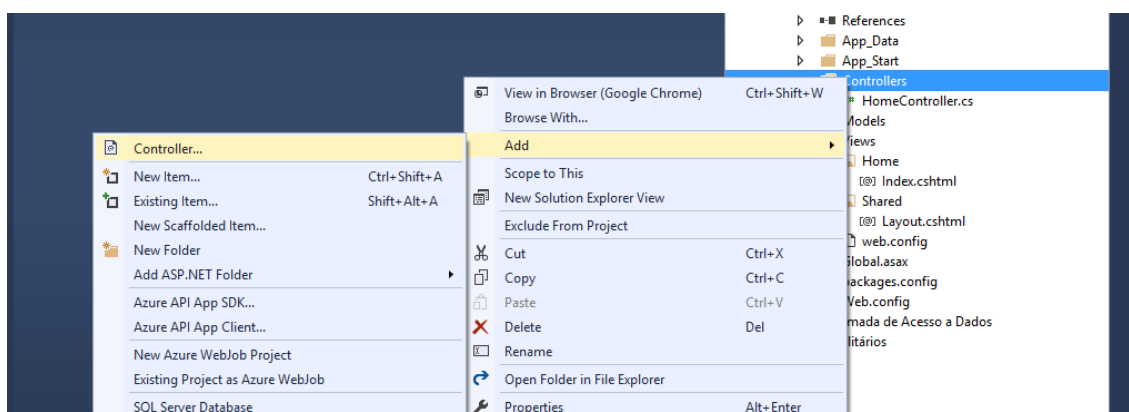
REALIZE SEU LOGIN OU CADASTRE-SE

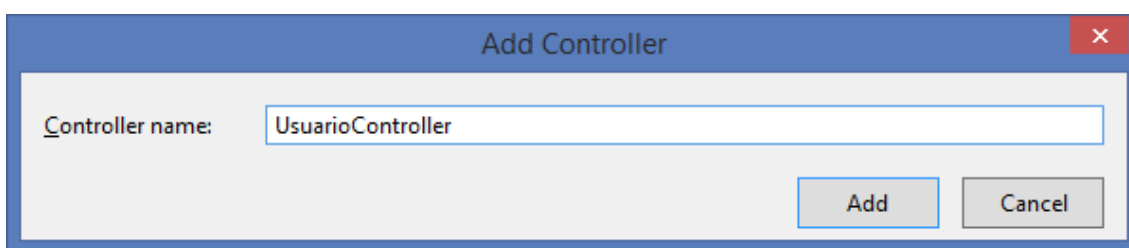
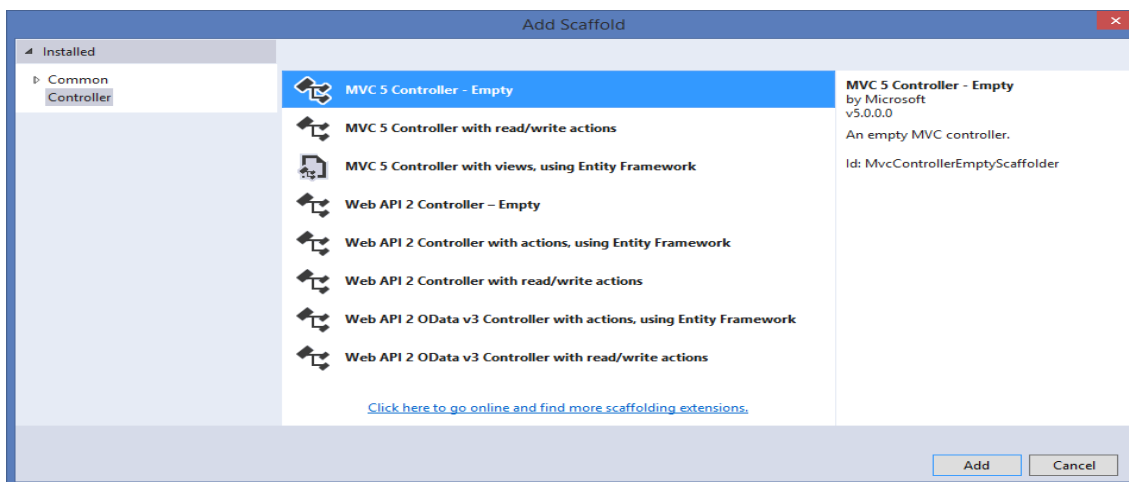
Bem vindo ao projeto

Selecione:

- [Realizar Autenticação](#)
- [Cadastrar conta de Usuário](#)

## Criaremos mais 2 classes de controle além da HomeController



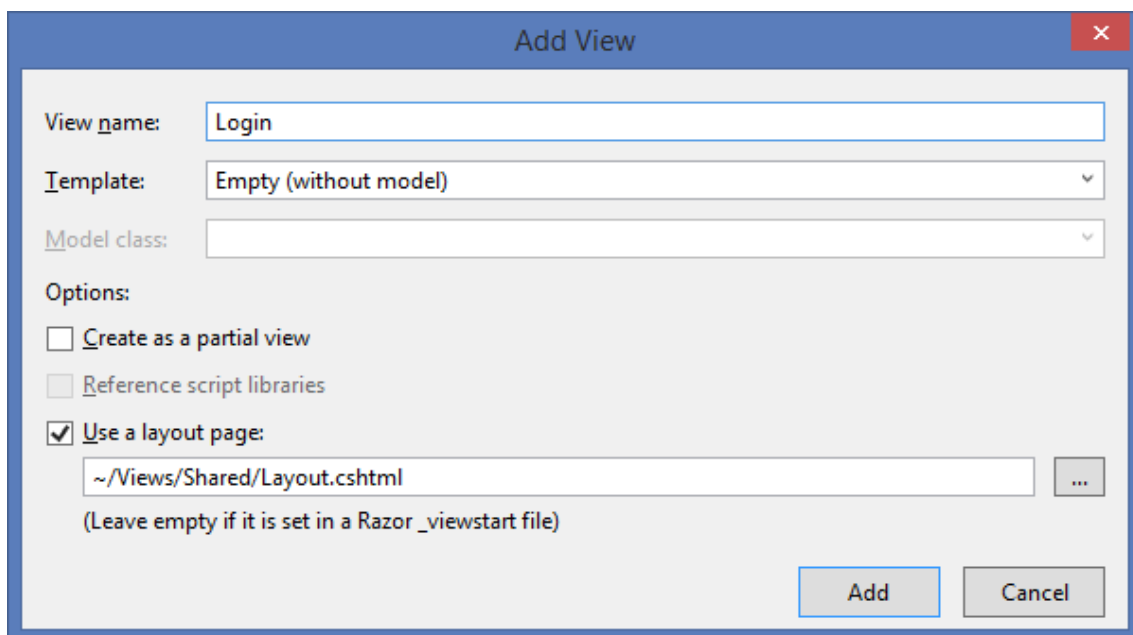
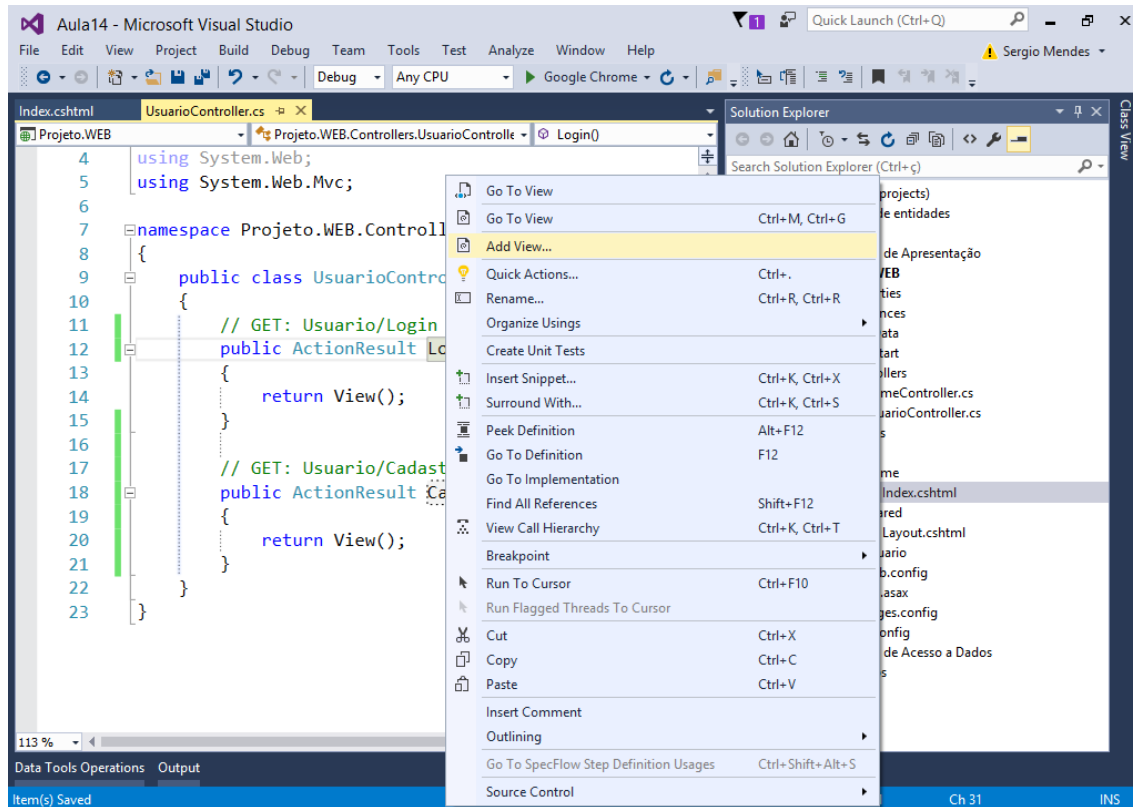


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Projeto.WEB.Controllers
{
    public class UsuarioController : Controller
    {
        // GET: Usuario/Login
        public ActionResult Login()
        {
            return View();
        }

        // GET: Usuario/Cadastro
        public ActionResult Cadastro()
        {
            return View();
        }
    }
}
```

## Criando a página de login:



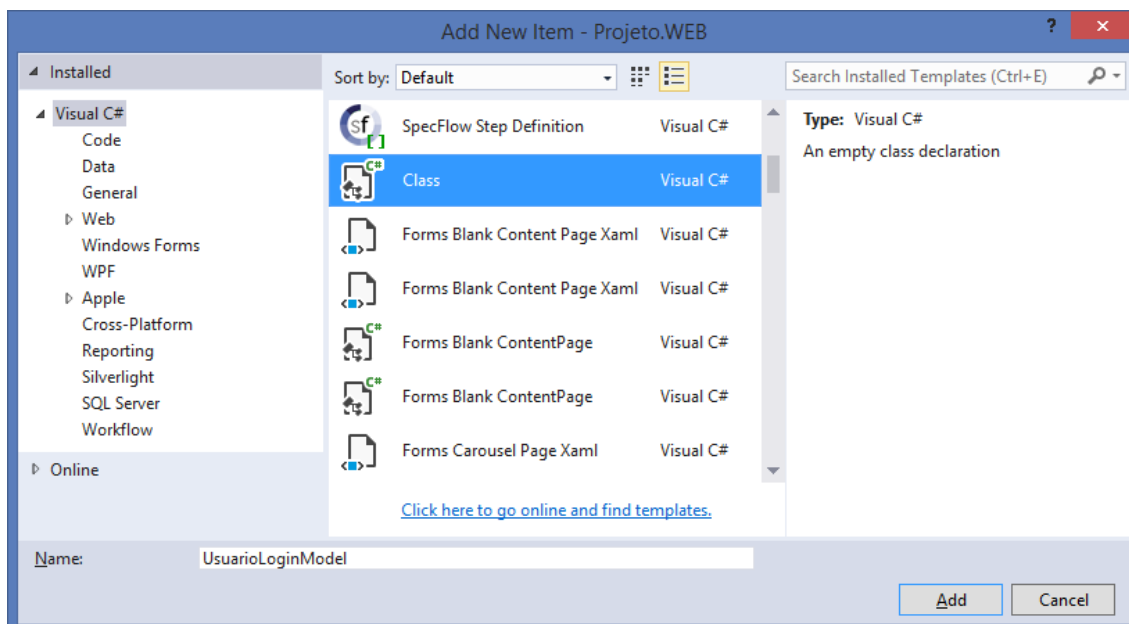
```
@{
    ViewBag.Title = "Login";
    Layout = "~/Views/Shared/Layout.cshtml";
}
```

```
<h4>Login de Usuários</h4>
<a href="/Home/Index">Voltar</a>
<hr/>

<div class="row">
    <div class="col-md-3">

        </div>
    </div>
</div>
```

**Criando a classe de modelo para construirmos o formulario de login de usuario:**



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations; //mapeamentos..

namespace Projeto.WEB.Models
{
    public class UsuarioLoginModel
    {
        [Required(ErrorMessage = "Informe seu login.")]
        public string Login { get; set; }

        [Required(ErrorMessage = "Informe sua senha.")]
        public string Senha { get; set; }
    }
}
```

```
@{
    ViewBag.Title = "Login";
    Layout = "~/Views/Shared/Layout.cshtml";
}

<h4>Login de Usuários</h4>
<a href="/Home/Index">Voltar</a>
<hr/>

<div class="row">
    <div class="col-md-3">

        <!-- Declarando a classe de modelo.. -->
        @model Projeto.WEB.Models.UsuarioLoginModel

        <!-- Abrindo a área do formulário -->
        @using (Html.BeginForm())
        {
            <label>Login de Acesso:</label>
            @Html.TextBoxFor(model => model.Login)
            <br/>

            <label>Senha de Acesso:</label>
            @Html.PasswordFor(model => model.Senha)
            <br />

            <input type="submit" value="Acessar Sistema"/>
        }
    </div>
</div>
```

## Executando:

```
@{
    ViewBag.Title = "Login";
    Layout = "~/Views/Shared/Layout.cshtml";
}

<h4>Login de Usuários</h4>
<a href="/Home/Index">Voltar</a>
<hr />

<div class="row">
    <div class="col-md-3">

        <!-- Declarando a classe de modelo.. -->
        @model Projeto.WEB.Models.UsuarioLoginModel

        <!-- Abrindo a área do formulário -->
        @using (Html.BeginForm())
        {
            <label>Login de Acesso:</label>
            @Html.TextBoxFor(model => model.Login,
                new
                {
                    @class = "form-control",
                    @placeholder = "Login de Acesso"
                })
            <br />
        }
    </div>
</div>
```



```
<label>Senha de Acesso:</label>
@Html.PasswordFor(model => model.Senha,
    new
    {
        @class = "form-control",
        @placeholder = "Sena de Acesso"
    });
<br />

<input type="submit" value="Acessar Sistema"
    class="btn btn-success" />
}

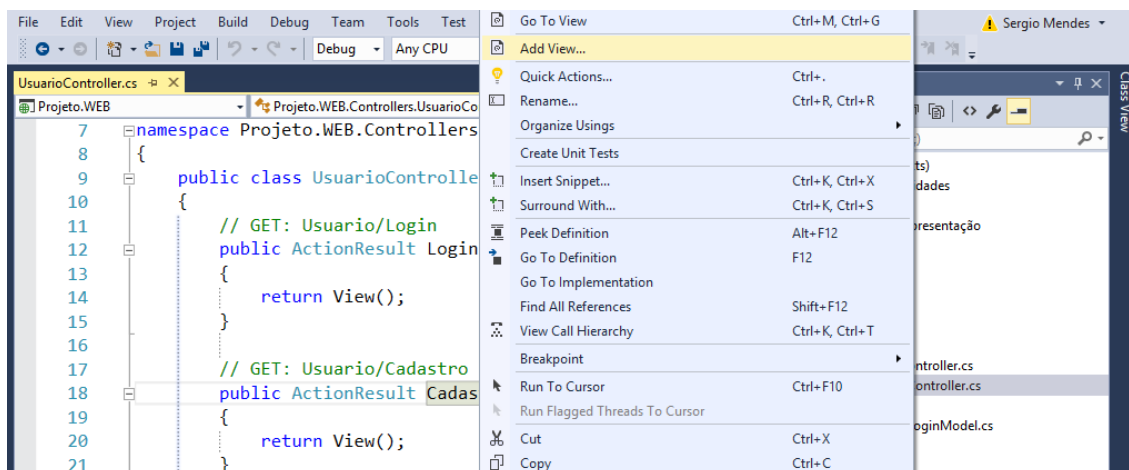
</div>
</div>
```

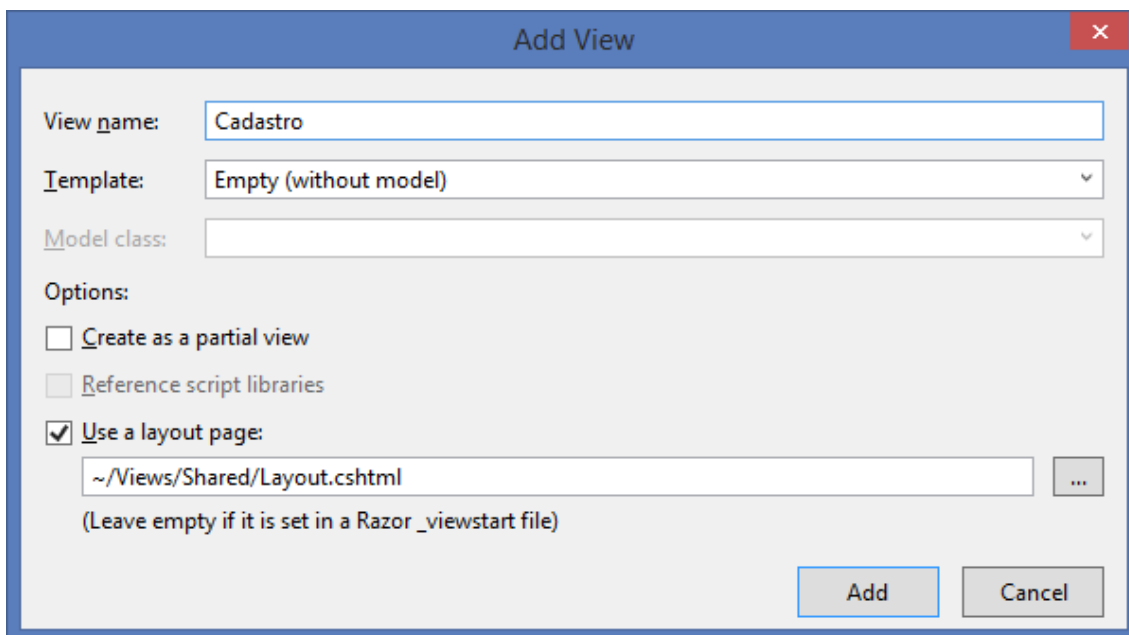
## Executando:



The screenshot shows a web browser window with the URL `localhost:56476/Usuario/Login`. The page title is "Sistema de Controle de Clientes". Below the title, it says "REALIZE SEU LOGIN OU CADASTRE-SE" and "Login de Usuários". There is a link "Voltar". The login form consists of two input fields: "Login de Acesso:" and "Senha de Acesso:". Below these fields is a green button labeled "Acessar Sistema".

## Criando a página de cadastro do usuario:





**Add View**

View name:

Template:

Model class:

Options:

☐ Create as a partial view

☐ Reference script libraries

☒ Use a layout page:

...

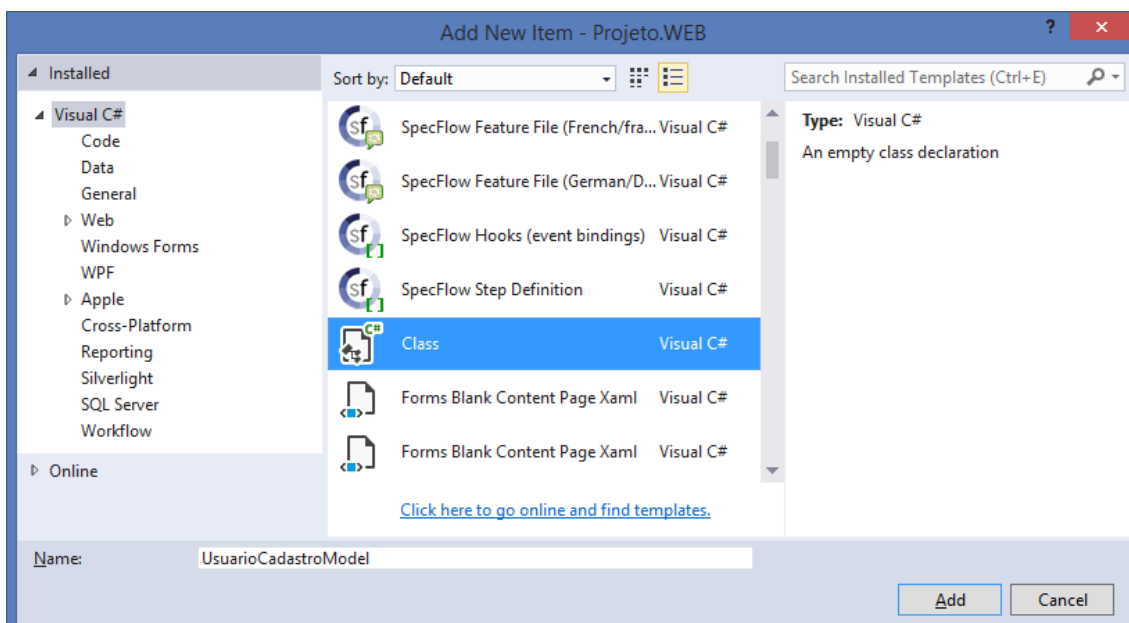
(Leave empty if it is set in a Razor \_viewstart file)

```
@{
    ViewBag.Title = "Cadastro";
    Layout = "~/Views/Shared/Layout.cshtml";
}
```

```
<h4>Criar conta de Usuário</h4>
<a href="/Home/Index">Voltar</a>
<hr/>
```

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

```
    </div>
</div>
```



**Add New Item - Projeto.WEB**

Sort by: Default

Search Installed Templates (Ctrl+E)

Visual C#

- Code
- Data
- General
- Web
  - Windows Forms
  - WPF
- Apple
  - Cross-Platform
  - Reporting
  - Silverlight
  - SQL Server
  - Workflow
- Online

SpecFlow Feature File (French/fra... Visual C#

SpecFlow Feature File (German/D... Visual C#

SpecFlow Hooks (event bindings) Visual C#

SpecFlow Step Definition Visual C#

**Class Visual C#**

Forms Blank Content Page Xaml Visual C#

Forms Blank Content Page Xaml Visual C#

Click here to go online and find templates.

Type: Visual C#

An empty class declaration

Name:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;

namespace Projeto.WEB.Models
{
    public class UsuarioCadastroModel
    {
        [Required(ErrorMessage = "Informe o nome.")]
        public string Nome { get; set; }

        [Required(ErrorMessage = "Informe o login.")]
        public string Login { get; set; }

        [Required(ErrorMessage = "Informe a senha.")]
        public string Senha { get; set; }

        [Compare("Senha", ErrorMessage = "Senhas não conferem.")]
        [Required(ErrorMessage = "Confirme a senha.")]
        public string SenhaConfirm { get; set; }

        [Required(ErrorMessage = "Envie a foto.")]
        public HttpPostedFileBase Foto { get; set; }
    }
}
```

## Criando o formulário de cadastro de usuario:

```
@{
    ViewBag.Title = "Cadastro";
    Layout = "~/Views/Shared/Layout.cshtml";
}

<h4>Criar conta de Usuário</h4>
<a href="/Home/Index">Voltar</a>
<hr/>

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

        <!-- classe de modelo da página -->
        @model Projeto.WEB.Models.UsuarioCadastroModel

        <!-- criando o formulário.. -->
        @using (Html.BeginForm())
        {
            <label>Nome do Usuário:</label>
            @Html.TextBoxFor(model => model.Nome,
                new
                {
                    @class = "form-control",
                    @placeholder = "Nome do Usuário"
                })
            <br />
        }
```

```
<label>Login de Acesso:</label>
@Html.TextBoxFor(model => model.Login,
    new
    {
        @class = "form-control",
        @placeholder = "Login de Acesso"
    })
<br />
<label>Senha de Acesso:</label>
@Html.PasswordFor(model => model.Senha,
    new
    {
        @class = "form-control",
        @placeholder = "Senha de Acesso"
    })
<br />
<label>Confirme sua Senha:</label>
@Html.PasswordFor(model => model.SenhaConfirm,
    new
    {
        @class = "form-control",
        @placeholder = "Confirme sua Senha"
    })
<br />
<label>Envie sua Foto:</label>
<input type="file" name="Foto" class="form-control"/>
<br/>
<input type="submit" value="Cadastrar Usuário"
    class="btn btn-success"/>
}
</div>
</div>
```

## Executando:



**Sistema de Controle de Clientes**

REALIZE SEU LOGIN OU CADASTRE-SE

**Criar conta de Usuário**

[Voltar](#)

Nome do Usuário:

Login de Acesso:

Senha de Acesso:

Confirme sua Senha:

Envie sua Foto:

 Nenhum...onado

## Criando as requisições POST para enviar os dados de cada formulario para o controller:

```
@{
    ViewBag.Title = "Login";
    Layout = "~/Views/Shared/Layout.cshtml";
}

<h4>Login de Usuários</h4>
<a href="/Home/Index">Voltar</a>
<hr />

<div class="row">
    <div class="col-md-3">

        <!-- Declarando a classe de modelo.. -->
        @model Projeto.WEB.Models.UsuarioLoginModel

        <!-- Abrindo a área do formulário -->
        @using (Html.BeginForm("Login", "Usuario", FormMethod.Post))
        {
            <label>Login de Acesso:</label>
            @Html.TextBoxFor(model => model.Login,
                new
                {
                    @class = "form-control",
                    @placeholder = "Login de Acesso"
                })
            <span class="text-danger">
                @Html.ValidationMessageFor(model => model.Login)
            </span>
            <br />

            <label>Senha de Acesso:</label>
            @Html.PasswordFor(model => model.Senha,
                new
                {
                    @class = "form-control",
                    @placeholder = "Senha de Acesso"
                });
            <span class="text-danger">
                @Html.ValidationMessageFor(model => model.Senha)
            </span>
            <br />

            <input type="submit" value="Acessar Sistema"
                class="btn btn-success" />
        }

    </div>

</div>

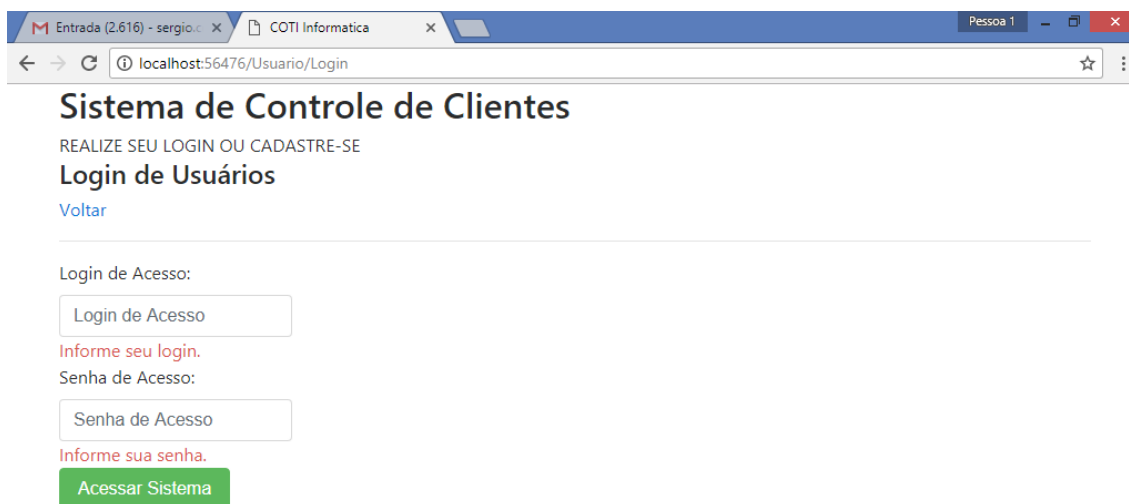
</div>
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Projeto.WEB.Models;

namespace Projeto.WEB.Controllers
{
    public class UsuarioController : Controller
    {
        // GET: Usuario/Login
        public ActionResult Login()
        {
            return View();
        }

        // POST: Usuario/Login
        [HttpPost]
        public ActionResult Login(UsuarioLoginModel model)
        {
            return View();
        }

        // GET: Usuario/Cadastro
        public ActionResult Cadastro()
        {
            return View();
        }
    }
}
```



## Formulário de cadastro:

```
@{
    ViewBag.Title = "Cadastro";
    Layout = "~/Views/Shared/Layout.cshtml";
}

<h4>Criar conta de Usuário</h4>
<a href="/Home/Index">Voltar</a>
```

```
<hr/>

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

        <!-- classe de modelo da página -->
        @model Projeto.WEB.Models.UsuarioCadastroModel

        <!-- criando o formulário.. -->
        @using (Html.BeginForm("Cadastro", "Usuario", FormMethod.Post,
                                new { @enctype = "multipart/form-data" }))
        {
            <label>Nome do Usuário:</label>
            @Html.TextBoxFor(model => model.Nome,
                new
                {
                    @class = "form-control",
                    @placeholder = "Nome do Usuário"
                })
            <span class="text-danger">
                @Html.ValidationMessageFor(model => model.Nome)
            </span>
            <br />

            <label>Login de Acesso:</label>
            @Html.TextBoxFor(model => model.Login,
                new
                {
                    @class = "form-control",
                    @placeholder = "Login de Acesso"
                })
            <span class="text-danger">
                @Html.ValidationMessageFor(model => model.Login)
            </span>
            <br />

            <label>Senha de Acesso:</label>
            @Html.PasswordFor(model => model.Senha,
                new
                {
                    @class = "form-control",
                    @placeholder = "Senha de Acesso"
                })
            <span class="text-danger">
                @Html.ValidationMessageFor(model => model.Senha)
            </span>
            <br />

            <label>Confirme sua Senha:</label>
            @Html.PasswordFor(model => model.SenhaConfirm,
                new
                {
                    @class = "form-control",
                    @placeholder = "Confirme sua Senha"
                })
            <span class="text-danger">
                @Html.ValidationMessageFor(model => model.SenhaConfirm)
            </span>
            <br />
        }
    }
}
```

```
<label>Envie sua Foto:</label>
<input type="file" name="Foto" class="form-control"/>
<span class="text-danger">
    @Html.ValidationMessageFor(model => model.Foto)
</span>
<br/>

<input type="submit" value="Cadastrar Usuário"
        class="btn btn-success"/>
    }

</div>
</div>
```

## No controller:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Projeto.WEB.Models;

namespace Projeto.WEB.Controllers
{
    public class UsuarioController : Controller
    {
        // GET: Usuario/Login
        public ActionResult Login()
        {
            return View();
        }

        // POST: Usuario/Login
        [HttpPost]
        public ActionResult Login(UsuarioLoginModel model)
        {
            return View();
        }

        // GET: Usuario/Cadastro
        public ActionResult Cadastro()
        {
            return View();
        }

        // POST: Usuario/Cadastro
        [HttpPost]
        public ActionResult Cadastro(UsuarioCadastroModel model)
        {
            return View();
        }
    }
}
```



## Executando:



Entrada (2,616) - sergio.c... x COTI Informatica x COTI Informatica x Pessoa 1

localhost:56476/Usuario/Cadastro

### Sistema de Controle de Clientes

REALIZE SEU LOGIN OU CADASTRE-SE

#### Criar conta de Usuário

[Voltar](#)

Nome do Usuário:

Informe o nome.

Login de Acesso:

Informe o login.

Senha de Acesso:

Informe a senha.

Confirme sua Senha:

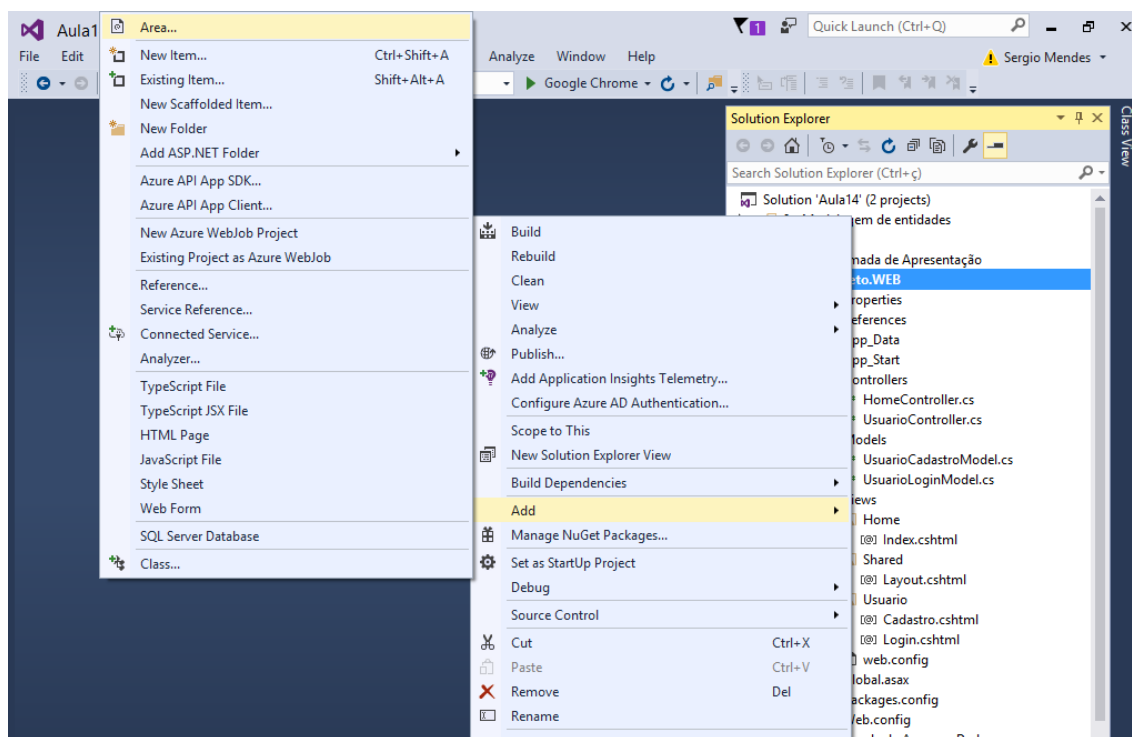
Confirme a senha.

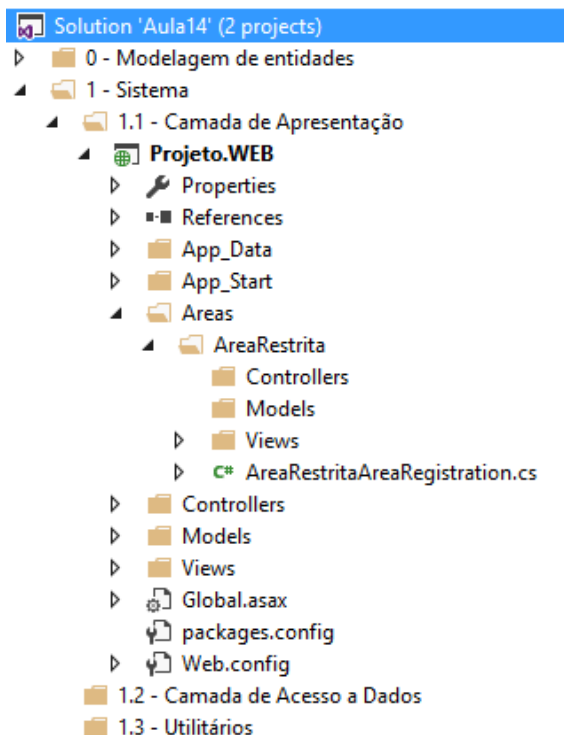
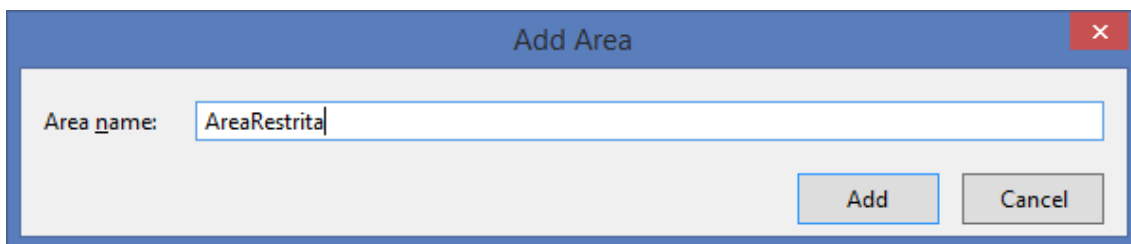
Envie sua Foto:

 Nenhum... onado

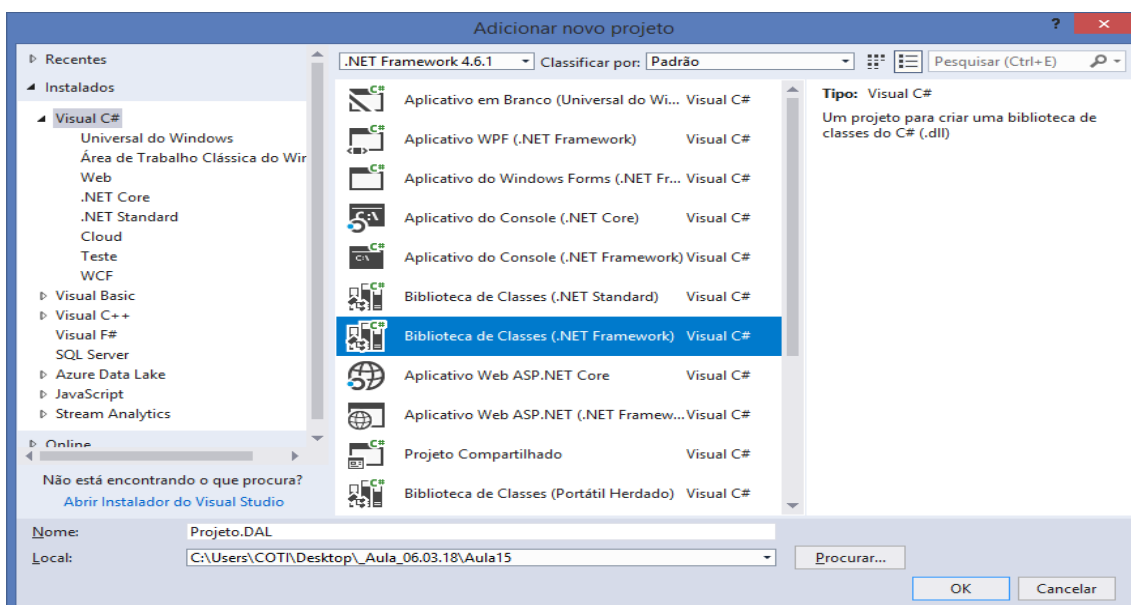
Envie a foto.

## Criando a area restrita do projeto MVC: (SubProjeto)

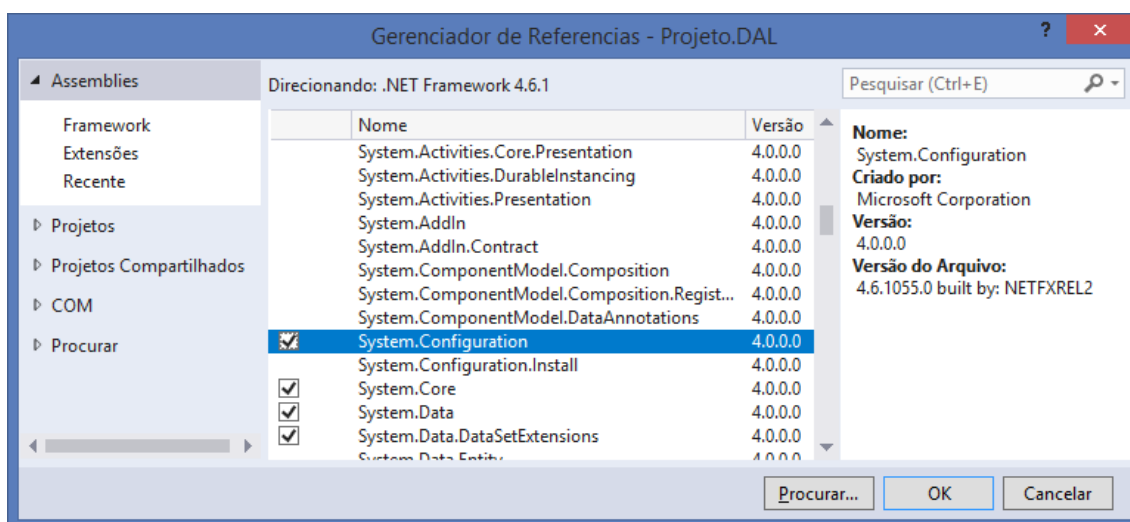
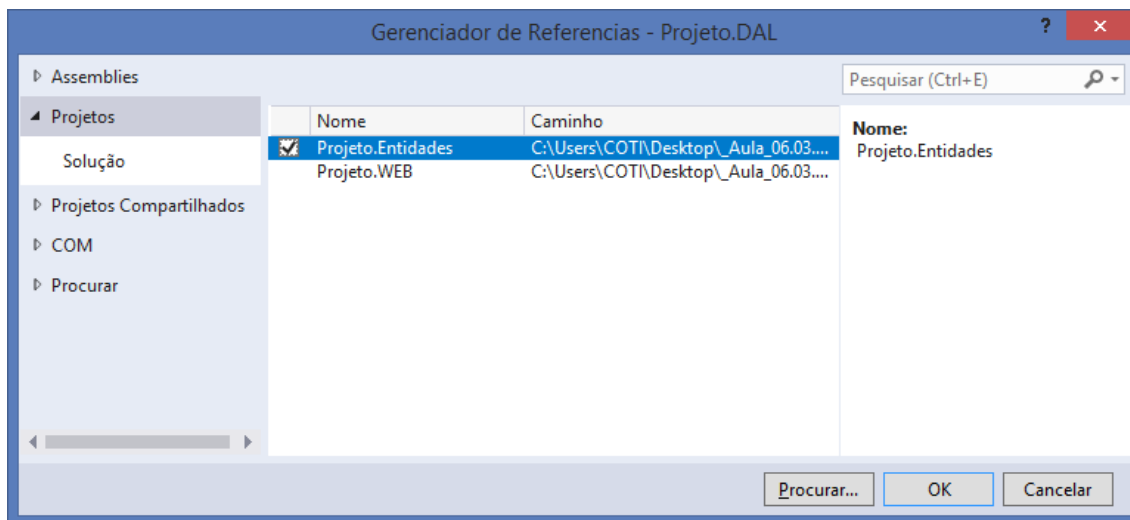




## 1.2 Camada de Acesso a dados Data Access Layer (DAL)



## Adicionando referencias:



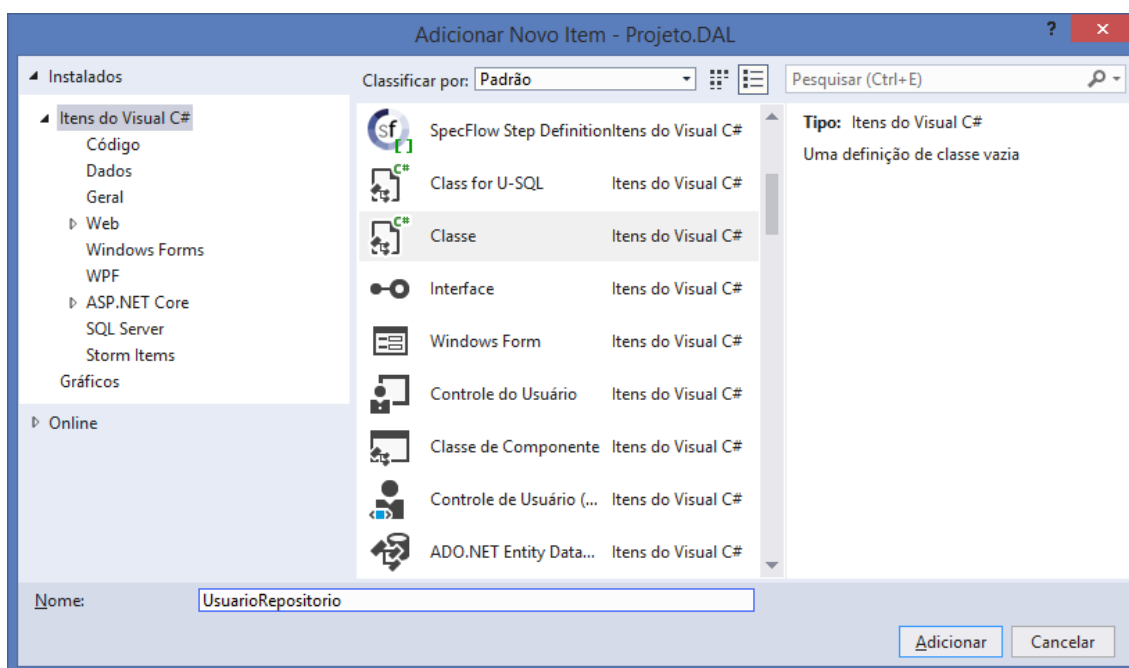
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient; //acesso ao sqlserver..
using System.Configuration; //connectionstring..
```

```
namespace Projeto.DAL
{
    public class Conexao
    {
        //atributos..
        protected SqlConnection con;
        protected SqlCommand cmd;
        protected SqlDataReader dr;
        protected SqlTransaction tr;
```

```
//métodos..
protected void OpenConnection()
{
    con = new SqlConnection(ConfigurationManager.ConnectionStrings
        ["aula"].ConnectionString);
    con.Open(); //conectado na base..
}

//métodos..
protected void CloseConnection()
{
    con.Close(); //desconectado da base..
}
}
```

## Classe de repositório de dados para a entidade Usuario:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient; //sqlserver..
using Projeto.Entidades; //classes de entidade..

namespace Projeto.DAL
{
    public class UsuarioRepositorio : Conexao
    {
        //método para inserir um usuario na base de dados..
        public void Insert(Usuario u)
        {

```

```

OpenConnection();

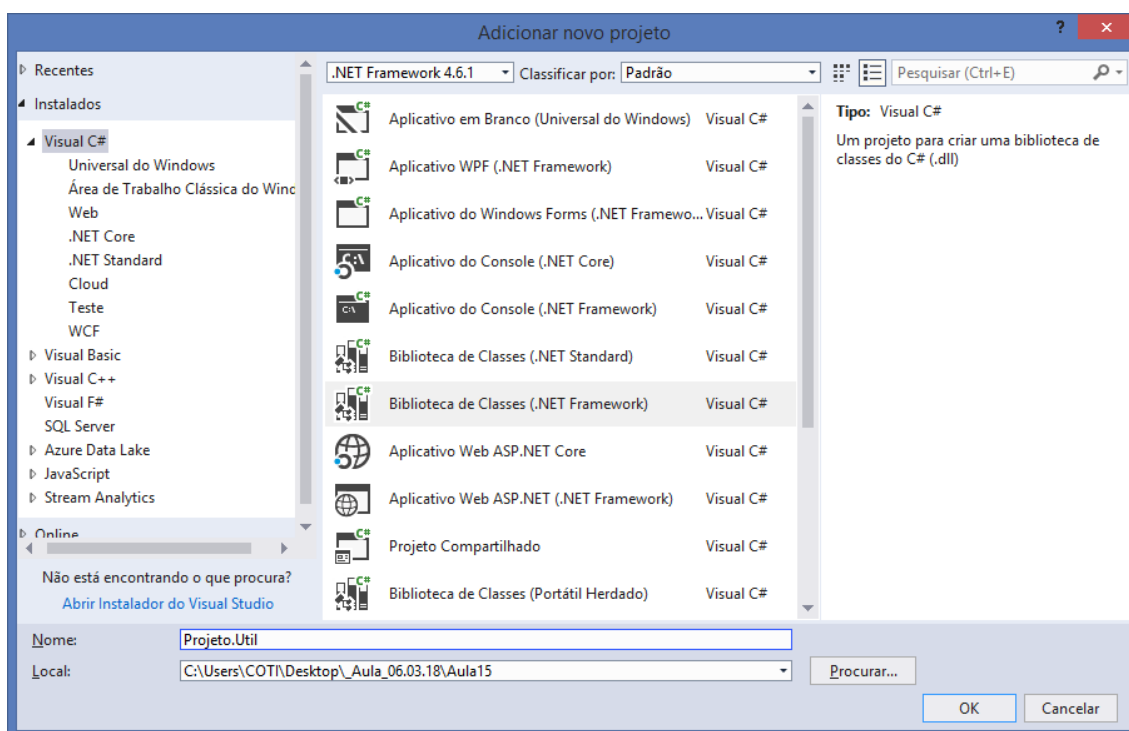
//escrever o comando SQL para o banco de dados..
string query = "insert into Usuario(Nome, Login, Senha, Foto) "
               + "values(@Nome, @Login, @Senha, @Foto)";

cmd = new SqlCommand(query, con);
cmd.Parameters.AddWithValue("@Nome", u.Nome);
cmd.Parameters.AddWithValue("@Login", u.Login);
cmd.Parameters.AddWithValue("@Senha", u.Senha);
cmd.Parameters.AddWithValue("@Foto", u.Foto);
cmd.ExecuteNonQuery();

CloseConnection();
    }
}
}

```

## Projeto para implementar programas "utilitarios":



## MD5 (Message Digest 5)

Padrão de criptografia de 128bits baseado em HASH, ou seja, o "codigo" gerado pela encriptação não pode ser descriptografado.

O MD5 gera um HASH de 128bits ou seja são 32 caracteres hexadecimais que compoem o codigo gerado pela encriptação.

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

namespace Projeto.Util
{
    public class Criptografia
    {
        //método para encriptar um valor em MD5..
        public static string EncriptarParaMD5(string valor)
        {
            MD5 md5 = new MD5CryptoServiceProvider();

            //passo 1) converter o parametro 'valor' de string para bytes..
            byte[] valorEmBytes = Encoding.UTF8.GetBytes(valor);

            //aplicar a criptografia..
            byte[] hash = md5.ComputeHash(valorEmBytes);

            //converter o hash em uma string hexadecimal..
            string resultado = string.Empty;

            foreach(byte b in hash)
            {
                resultado += b.ToString("X2"); //hexadecimal (X2)
            }

            //retornando..
            return resultado;
        }
    }
}
```

## SHA1

Padrão de criptografia de até 256bits que também utiliza o conceito de HASH hexadecimal. Seu retorno é um hash de 40 caracteres.

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

namespace Projeto.Util
{
    public class Criptografia
    {
        //método para encriptar um valor em MD5..
        public static string EncriptarParaMD5(string valor)
        {
            MD5 md5 = new MD5CryptoServiceProvider();
```

```
//passo 1) converter o parametro 'valor' de string para bytes..
byte[] valorEmBytes = Encoding.UTF8.GetBytes(valor);

//aplicar a criptografia..
byte[] hash = md5.ComputeHash(valorEmBytes);

//converter o hash em uma string hexadecimal..
string resultado = string.Empty;

foreach(byte b in hash)
{
    resultado += b.ToString("X2"); //hexadecimal (X2)
}

//retornando..
return resultado;
}

//método para encriptar um valor em SHA1..
public static string EncriptarParaSHA1(string valor)
{
    SHA1 sha1 = new SHA1CryptoServiceProvider();

    //passo 1) converter o parametro 'valor' de string para bytes..
    byte[] valorEmBytes = Encoding.UTF8.GetBytes(valor);

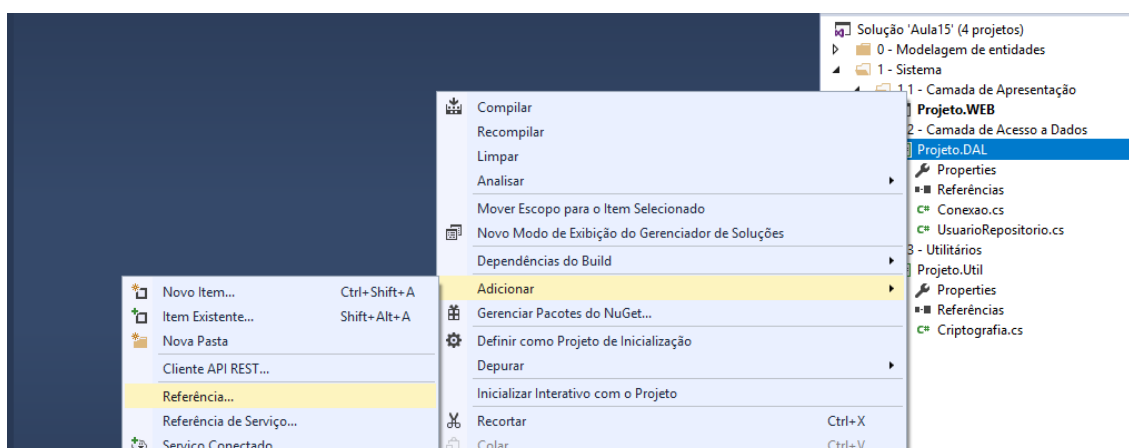
    //aplicar a criptografia..
    byte[] hash = sha1.ComputeHash(valorEmBytes);

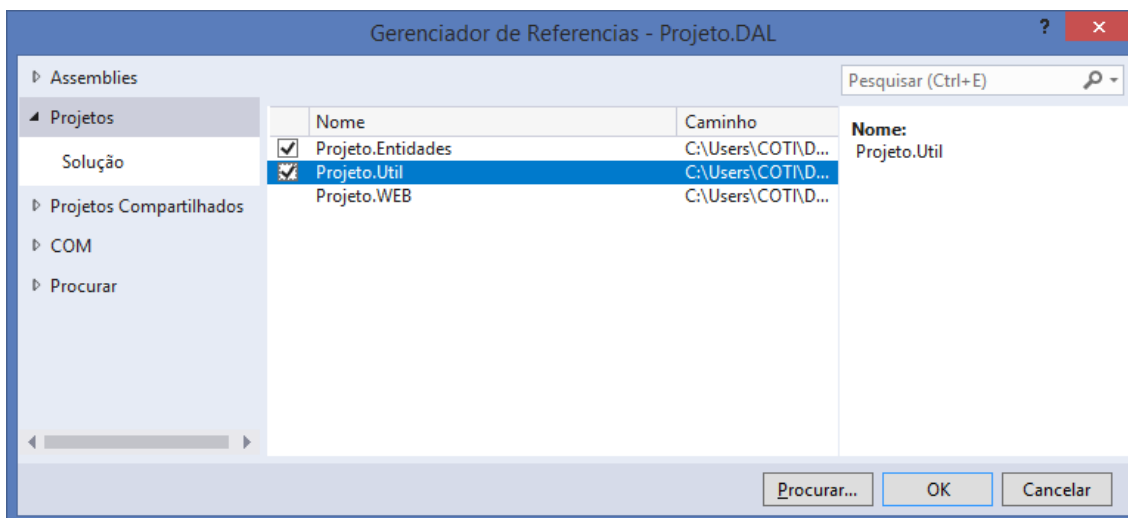
    //converter o hash em uma string hexadecimal..
    string resultado = string.Empty;

    foreach (byte b in hash)
    {
        resultado += b.ToString("X2"); //hexadecimal (X2)
    }

    //retornando..
    return resultado;
}
}
```

## Adicionando referencia no projeto DAL:





## Voltando na classe UsuarioRepositorio:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient; //sqlserver..
using Projeto.Entidades; //classes de entidade..
using Projeto.Util; //utilitários..

namespace Projeto.DAL
{
    public class UsuarioRepositorio : Conexao
    {
        //método para inserir um usuario na base de dados..
        public void Insert(Usuario u)
        {
            OpenConnection();

            //escrever o comando SQL para o banco de dados..
            string query = "insert into Usuario(Nome, Login, Senha, Foto) "
                + "values(@Nome, @Login, @Senha, @Foto)";

            cmd = new SqlCommand(query, con);
            cmd.Parameters.AddWithValue("@Nome", u.Nome);
            cmd.Parameters.AddWithValue("@Login", u.Login);
            cmd.Parameters.AddWithValue("@Senha",
                Criptografia.EncriptarParaMD5(u.Senha));
            cmd.Parameters.AddWithValue("@Foto", u.Foto);
            cmd.ExecuteNonQuery();

            CloseConnection();
        }
    }
}
```



```
//método para verificar se um login ja esta cadastrado na base..
public bool HasLogin(string login)
{
    OpenConnection();

    string query = "select count(*) from Usuario
                    where Login = @Login";

    cmd = new SqlCommand(query, con);
    cmd.Parameters.AddWithValue("@Login", login);
    int qtd = Convert.ToInt32(cmd.ExecuteScalar());

    CloseConnection();
    return qtd > 0;
}

//método para buscar 1 usuario pelo login e senha..
public Usuario Find(string login, string senha)
{
    OpenConnection();

    string query = "select * from Usuario
                    where Login = @Login and Senha = @Senha";

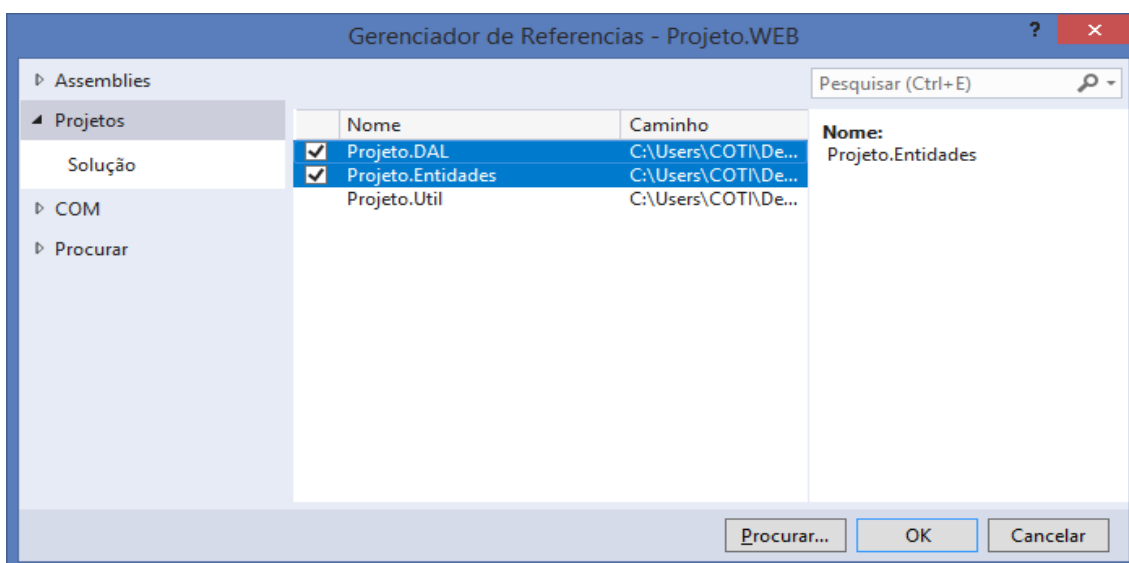
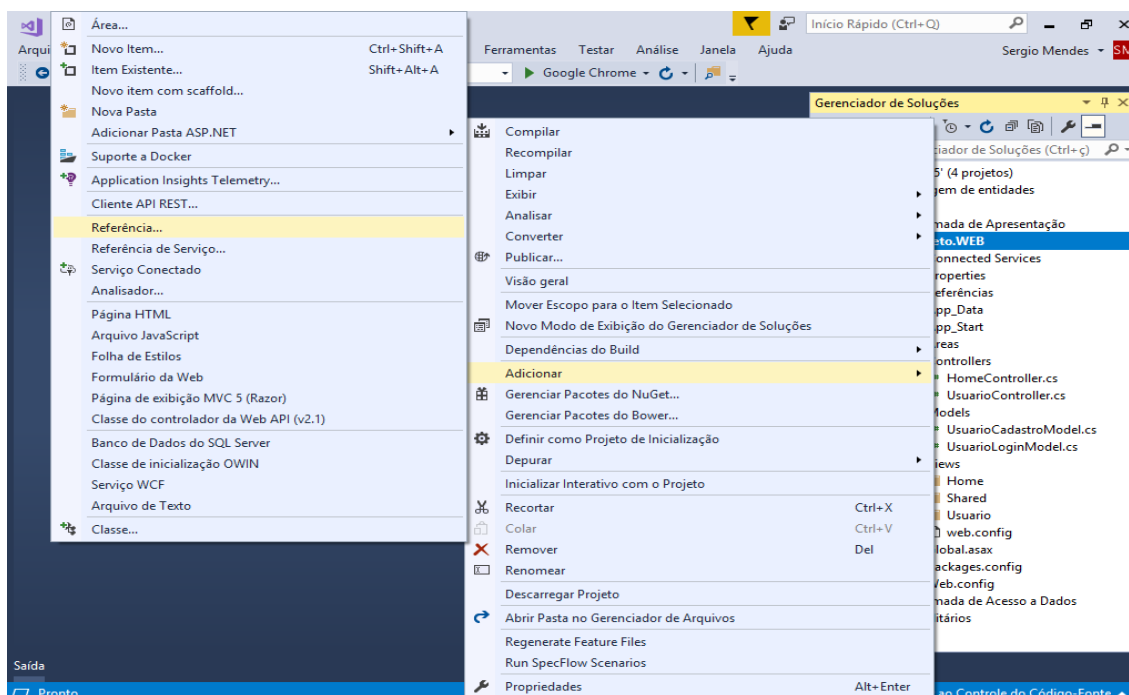
    cmd = new SqlCommand(query, con);
    cmd.Parameters.AddWithValue("@Login", login);
    cmd.Parameters.AddWithValue("@Senha",
                              Criptografia.EncriptarParaMD5(senha));
    dr = cmd.ExecuteReader();

    Usuario u = null; //sem espaço de memória..

    //verificar se o usuario foi encontrado..
    if(dr.Read())
    {
        u = new Usuario(); //instanciando..

        u.IdUsuario = Convert.ToInt32(dr["IdUsuario"]);
        u.Nome = Convert.ToString(dr["Nome"]);
        u.Login = Convert.ToString(dr["Login"]);
        u.Senha = Convert.ToString(dr["Senha"]);
        u.Foto = Convert.ToString(dr["Foto"]);
    }

    CloseConnection();
    return u; //retornando..
}
}
```



Criando uma classe de validação customizada para o campo de upload de foto:

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

namespace Projeto.WEB.Validations
{
    public class UploadFotoValidation
    {
    }
}
```

## Implementando:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations; //mapeamento..
using System.IO; //manipulação de arquivos..

namespace Projeto.WEB.Validations
{
    //Regra 1) HERDAR ValidationAttribute
    public class UploadFotoValidation : ValidationAttribute
    {
        //Regra 2) Sobrescrever (override) o método IsValid
        public override bool IsValid(object value)
        {
            //verificar o tipo do parametro 'value'
            if (value != null && value is HttpPostedFileBase)
            {
                //casting para o tipo HttpPostedFileBase
                HttpPostedFileBase arquivo = (HttpPostedFileBase) value;

                //obter a extensão do arquivo..
                string extensao = Path.GetExtension(arquivo.FileName);

                //testando..
                return extensao.Equals(".jpg") || extensao.Equals(".jpeg")
                    || extensao.Equals(".png") || extensao.Equals(".gif");
            }

            return false;
        }
    }
}
```

## Aplicando o validador:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;
using Projeto.WEB.Validations;

namespace Projeto.WEB.Models
{
    public class UsuarioCadastroModel
    {
        [Required(ErrorMessage = "Informe o nome.")]
        public string Nome { get; set; }

        [Required(ErrorMessage = "Informe o login.")]
        public string Login { get; set; }

        [Required(ErrorMessage = "Informe a senha.")]
        public string Senha { get; set; }
    }
}
```

```
[Compare("Senha", ErrorMessage = "Senhas não conferem.")]
[Required(ErrorMessage = "Confirme a senha.")]
public string SenhaConfirm { get; set; }

[UploadFotoValidation(ErrorMessage = "Envie apenas imagens.")]
[Required(ErrorMessage = "Envie a foto.")]
public HttpPostedFileBase Foto { get; set; }
}
}
```

Implementando o cadastro do usuario:

## /UsuarioController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Projeto.WEB.Models;
using Projeto.Entidades;
using Projeto.DAL;
using System.IO;

namespace Projeto.WEB.Controllers
{
    public class UsuarioController : Controller
    {
        // GET: Usuario/Login
        public ActionResult Login()
        {
            return View();
        }

        // POST: Usuario/Login
        [HttpPost]
        public ActionResult Login(UsuarioLoginModel model)
        {
            return View();
        }

        // GET: Usuario/Cadastro
        public ActionResult Cadastro()
        {
            return View();
        }

        // POST: Usuario/Cadastro
        [HttpPost]
        public ActionResult Cadastro(UsuarioCadastroModel model)
        {
            //verificar se os campos da model passaram na validação..
            if(ModelState.IsValid)
            {
                try
                {
                    UsuarioRepositorio rep = new UsuarioRepositorio();
                    //verificar se o login do usuario ja foi cadastrado..
                }
            }
        }
    }
}
```

```

if( ! rep.HasLogin(model.Login))
{
    //cadastrar o usuario..
    Usuario u = new Usuario(); //instanciando..
    u.Nome = model.Nome;
    u.Login = model.Login;
    u.Senha = model.Senha;
    u.Foto = Guid.NewGuid().ToString()
        + Path.GetExtension(model.Foto.FileName);

    //cadastrando o usuario..
    rep.Insert(u);

    //upload..
    string pasta = Server.MapPath("/Imagens/");
    model.Foto.SaveAs(pasta + u.Foto);

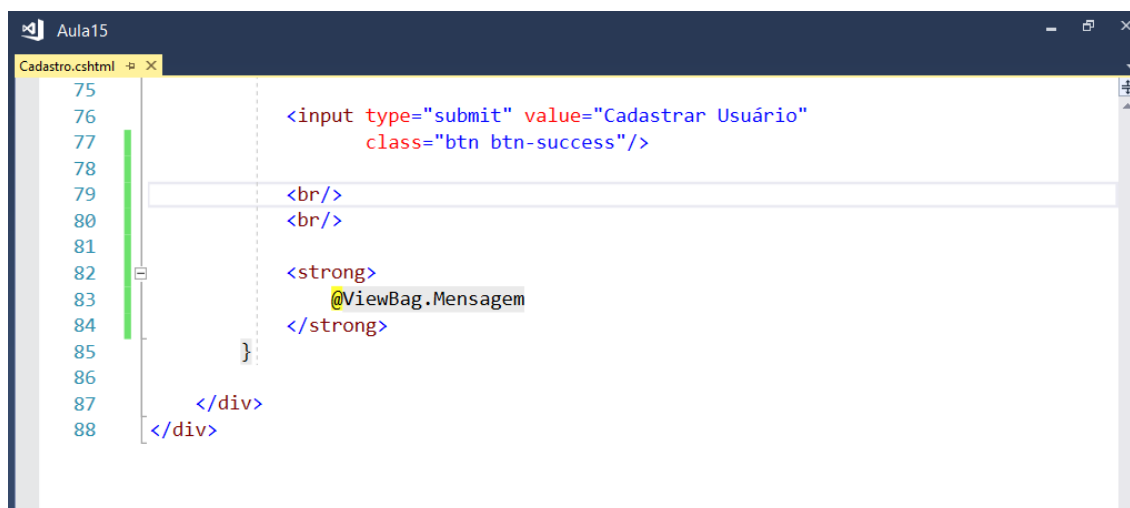
    ModelState.Clear(); //limpar os campos do formulário..
    ViewBag.Mensagem = $"Usuário {u.Nome},
        cadastrado com sucesso.";
}
else
{
    ViewBag.Mensagem = "Erro. Este login já
        encontra-se cadastrado. Tente outro.";
}
}
catch(Exception e)
{
    //enviar mensagem de erro para a página..
    ViewBag.Mensagem = e.Message;
}
}

return View();
}
}
}

```

## Executando:

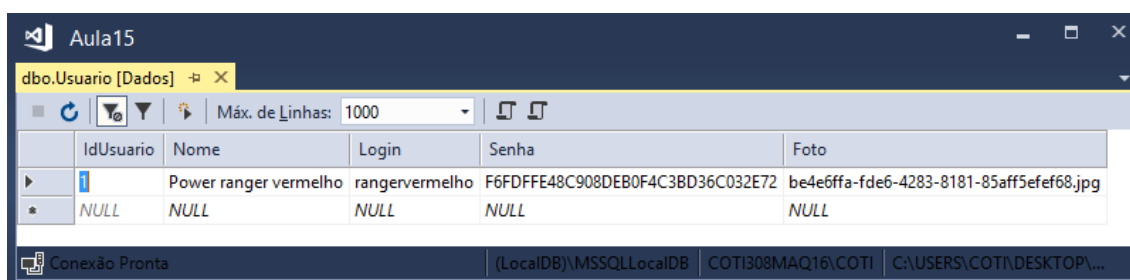
Exibindo a mensagem



## Resultado:

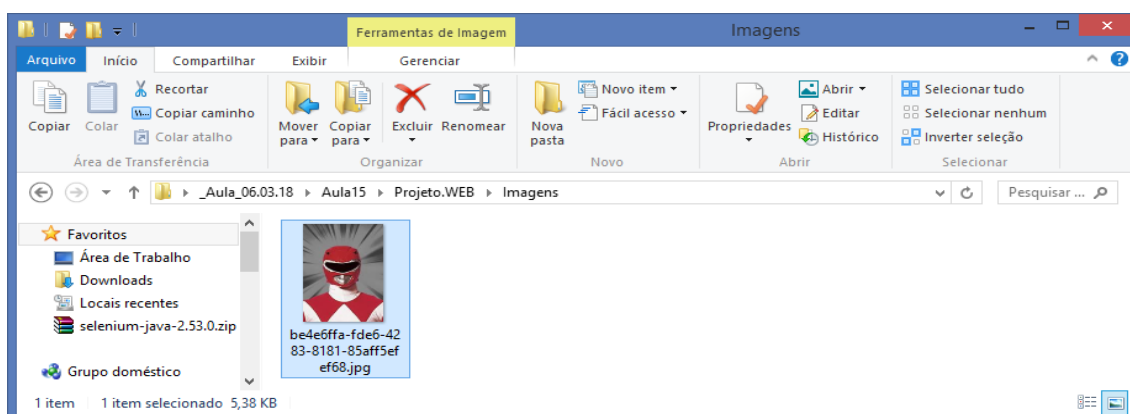


## No banco de dados:



	IdUsuario	Nome	Login	Senha	Foto
1		Power ranger vermelho	rangervermelho	F6FDFFE48C908DEB0F4C3BD36C032E72	be4e6ffa-fde6-4283-8181-85aff5efef68.jpg
*	NULL	NULL	NULL	NULL	NULL

## Foto enviada:



## Resgatar os dados enviados pelo formulário de Login de Usuario:

```
@{
    ViewBag.Title = "Login";
    Layout = "~/Views/Shared/Layout.cshtml";
}

<h4>Login de Usuários</h4>
<a href="/Home/Index">Voltar</a>
<hr />

<div class="row">
    <div class="col-md-3">

        <!-- Declarando a classe de modelo.. -->
        @model Projeto.WEB.Models.UsuarioLoginModel

        <!-- Abrindo a área do formulário -->
        @using (Html.BeginForm("Login", "Usuario", FormMethod.Post))
        {
            <label>Login de Acesso:</label>
            @Html.TextBoxFor(model => model.Login,
                new
                {
                    @class = "form-control",
                    @placeholder = "Login de Acesso"
                })
            <span class="text-danger">
                @Html.ValidationMessageFor(model => model.Login)
            </span>
            <br />

            <label>Senha de Acesso:</label>
            @Html.PasswordFor(model => model.Senha,
                new
                {
                    @class = "form-control",
                    @placeholder = "Senha de Acesso"
                });
            <span class="text-danger">
                @Html.ValidationMessageFor(model => model.Senha)
            </span>
            <br />

            <input type="submit" value="Acessar Sistema"
                class="btn btn-success" />

            <br />
            <br />

            <strong>
                @ViewBag.Mensagem
            </strong>
        }
    </div>
</div>
```

## Classe de controle:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Projeto.WEB.Models;
using Projeto.Entidades;
using Projeto.DAL;
using System.IO;

namespace Projeto.WEB.Controllers
{
    public class UsuarioController : Controller
    {
        // GET: Usuario/Login
        public ActionResult Login()
        {
            return View();
        }

        // POST: Usuario/Login
        [HttpPost]
        public ActionResult Login(UsuarioLoginModel model)
        {
            if(ModelState.IsValid)
            {
                try
                {
                    //buscar o usuario no banco de dados..
                    UsuarioRepositorio rep = new UsuarioRepositorio();
                    Usuario u = rep.Find(model.Login, model.Senha);

                    if(u != null) //se o usuario foi encontrado..
                    {
                        //TODO
                        ViewBag.Mensagem = "Usuario encontrado.";
                    }
                    else
                    {
                        ViewBag.Mensagem = "Acesso Negado.
                        Usuario não encontrado.";
                    }
                }
                catch(Exception e)
                {
                    //mensagem de erro..
                    ViewBag.Mensagem = e.Message;
                }
            }

            return View();
        }

        // GET: Usuario/Cadastro
        public ActionResult Cadastro()
        {
            return View();
        }
    }
}
```



```
// POST: Usuario/Cadastro
[HttpPost]
public ActionResult Cadastro(UsuarioCadastroModel model)
{
    //verificar se os campos da model passaram na validação..
    if(ModelState.IsValid)
    {
        try
        {
            UsuarioRepositorio rep = new UsuarioRepositorio();
            //verificar se o login do usuario ja foi cadastrado..
            if( ! rep.HasLogin(model.Login))
            {
                //cadastrar o usuario..
                Usuario u = new Usuario(); //instanciando..
                u.Nome = model.Nome;
                u.Login = model.Login;
                u.Senha = model.Senha;
                u.Foto = Guid.NewGuid().ToString()
                    + Path.GetExtension(model.Foto.FileName);

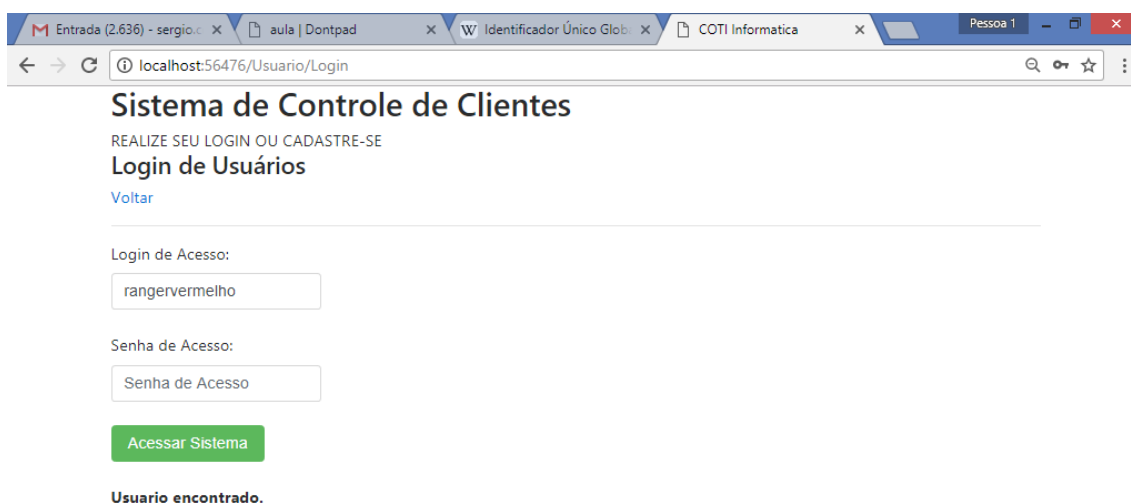
                //cadastrando o usuario..
                rep.Insert(u);

                //upload..
                string pasta = Server.MapPath("/Imagens/");
                model.Foto.SaveAs(pasta + u.Foto);

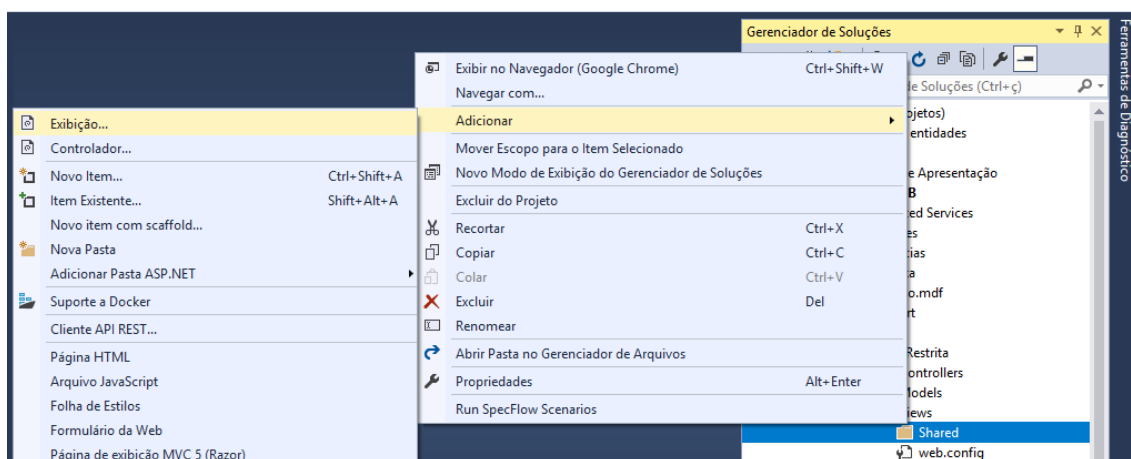
                ModelState.Clear(); //limpar os campos do formulário..
                ViewBag.Mensagem = $"Usuário {u.Nome},
                    cadastrado com sucesso.";
            }
            else
            {
                ViewBag.Mensagem = "Erro. Este login
                    já encontra-se cadastrado. Tente outro.";
            }
        }
        catch(Exception e)
        {
            //enviar mensagem de erro para a página..
            ViewBag.Mensagem = e.Message;
        }
    }

    return View();
}
}
```

## Executando:



## Criando uma página de layout:



Adicionar modo de exibição

Nome do modo de exibição:

Modelo:

Classe do modelo:

Opções:

☐ Criar como um Modo de exibição parcial

☐ Bibliotecas de scripts de referência

☐ Usar uma página de layout:

(deixe em branco se ele estiver definido em um arquivo Razor \_viewstart)

Adicionar Cancelar

```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>COTI Informática</title>

  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
alpha.6/css/bootstrap.min.css" integrity="sha384-
rwoIResjU2yc3z8GV/NPeZWAv56rSmLldC3R/AZzGRnGxQQKnKkoFVhFQhNUwEyJ"
crossorigin="anonymous">
</head>
<body class="container">

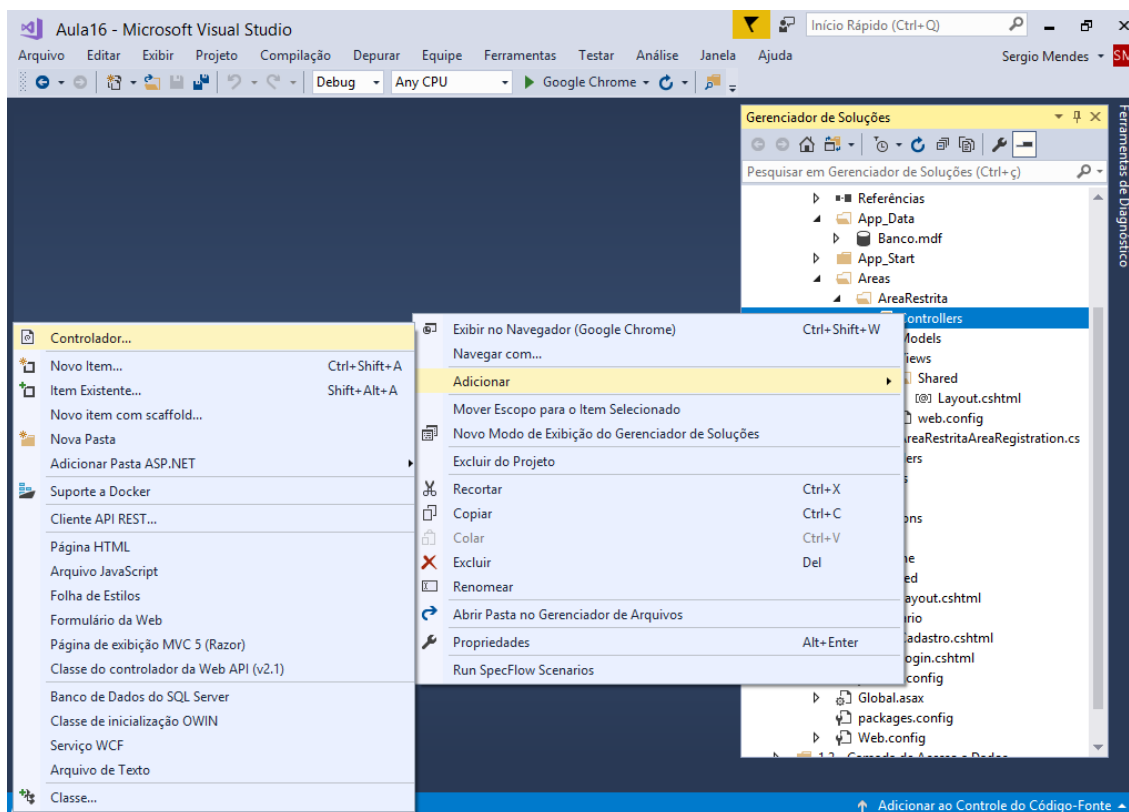
  <div>
    <h2>Área Restrita do Sistema</h2>
    Seja bem vindo: <strong></strong>
  </div>

  <hr/>

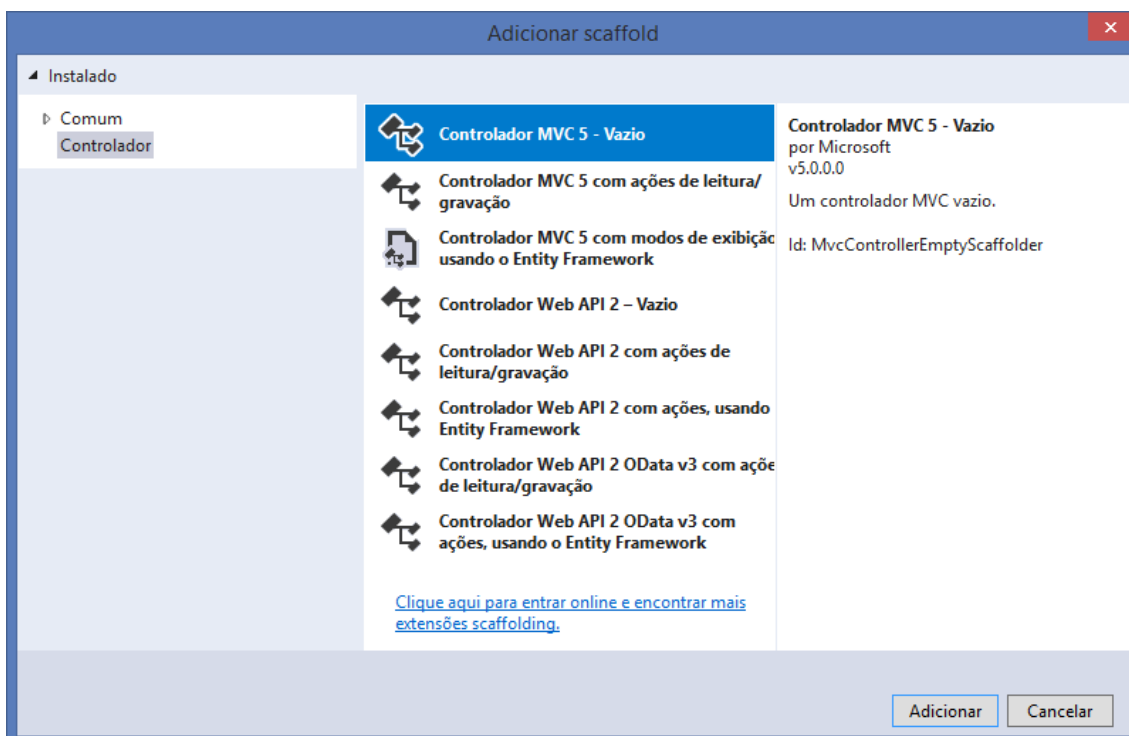
  <!-- local para entrada do conteudo das demais páginas -->
  @RenderBody()

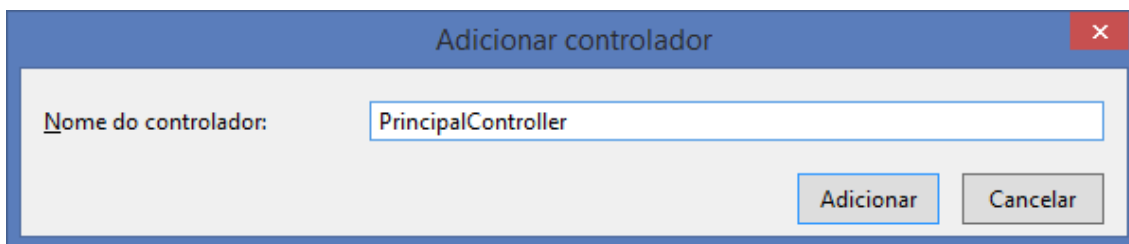
</body>
</html>
```

## Criando o controller principal da área de acesso restrito:



## Selecione: Controlador MVC5 Vazio

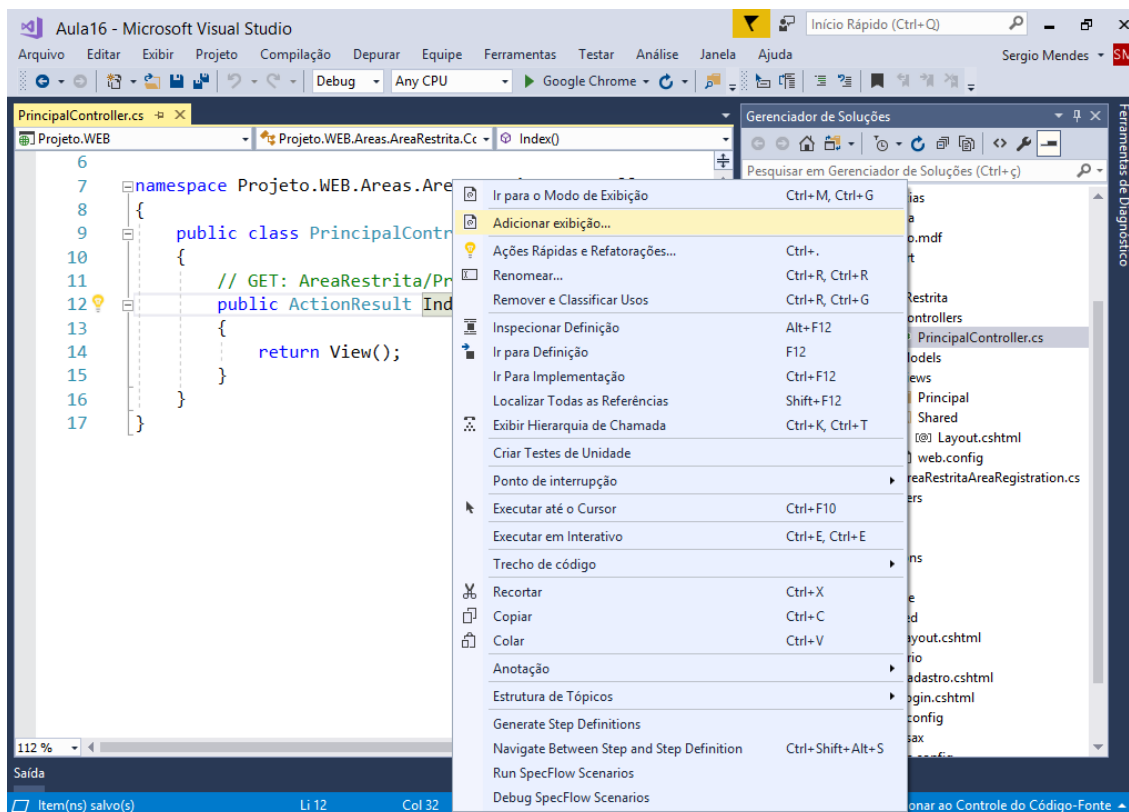


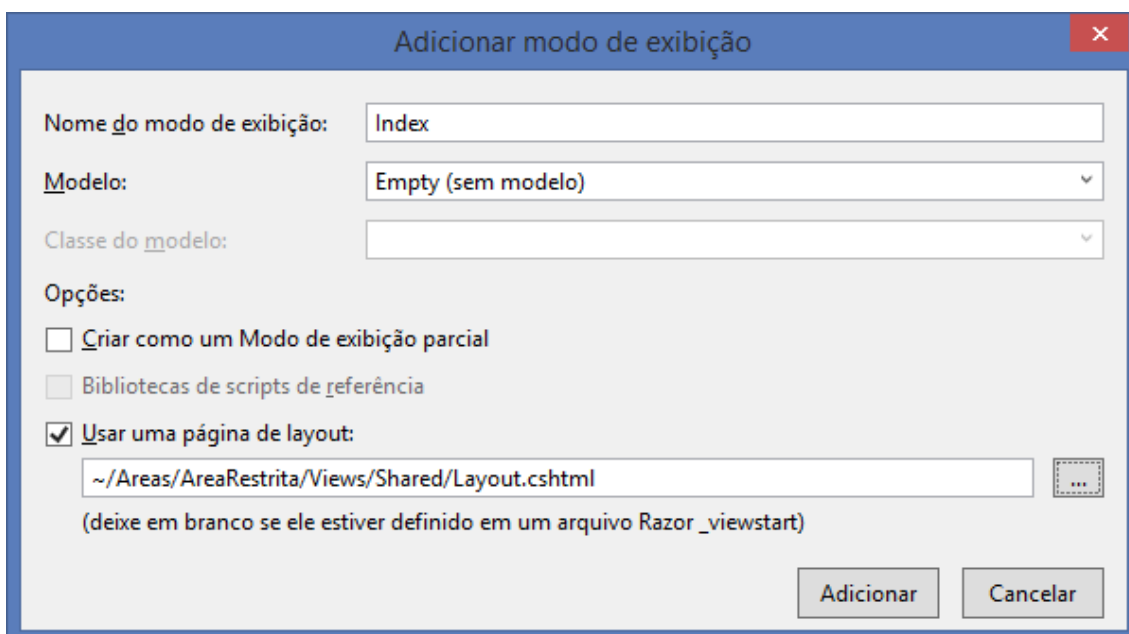
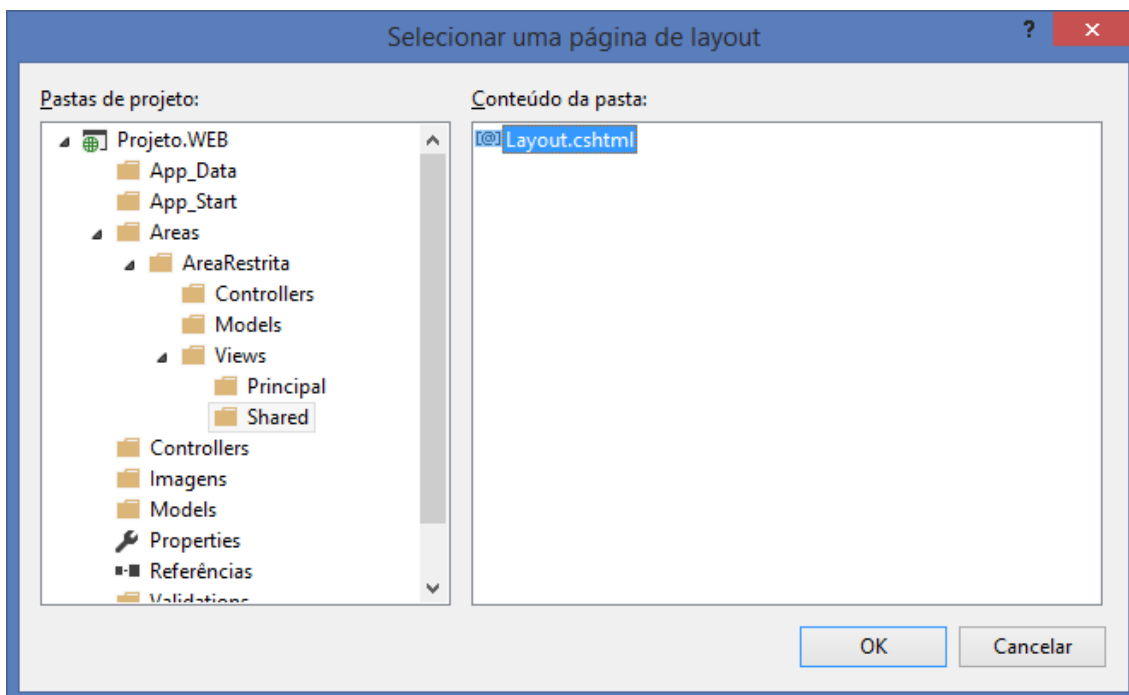


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

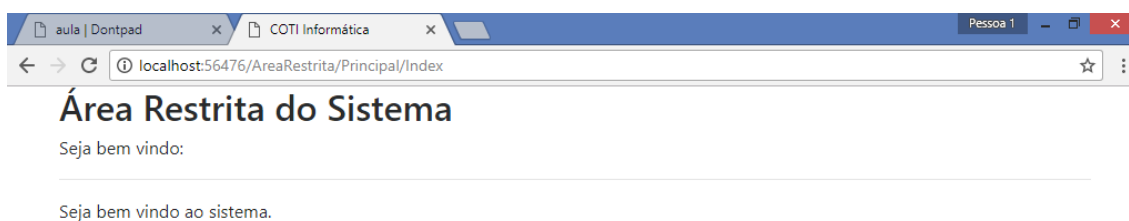
namespace Projeto.WEB.Areas.AreaRestrita.Controllers
{
    public class PrincipalController : Controller
    {
        // GET: AreaRestrita/Principal
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

## Gerando a página Index:





<http://localhost:56476/AreaRestrita/Principal/Index>

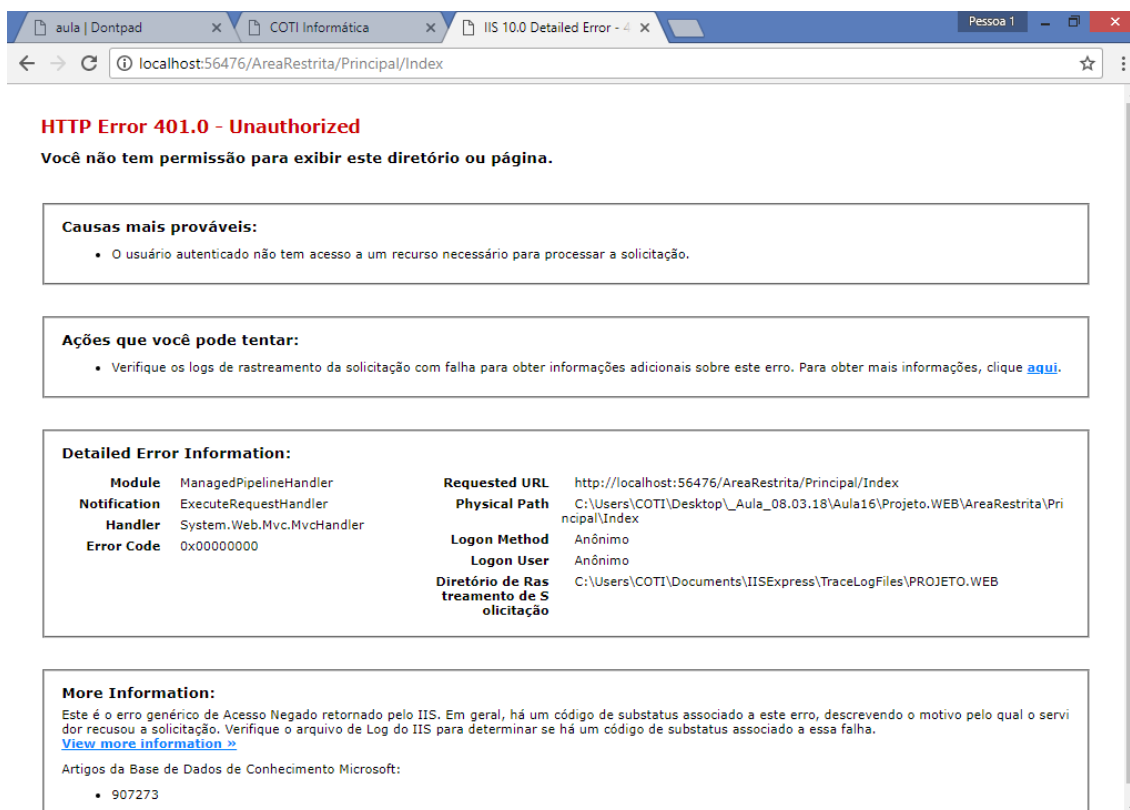


Restringindo o acesso ao **PrincipalController** para que só permita usuarios autenticados:

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

namespace Projeto.WEB.Areas.AreaRestrita.Controllers
{
    [Authorize]
    public class PrincipalController : Controller
    {
        // GET: AreaRestrita/Principal
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

## Erro 401 (Acesso não autorizado)



The screenshot shows a web browser window with the address bar displaying `localhost:56476/AreaRestrita/Principal/Index`. The page content is an IIS error message:

**HTTP Error 401.0 - Unauthorized**  
 Você não tem permissão para exibir este diretório ou página.

**Causas mais prováveis:**

- O usuário autenticado não tem acesso a um recurso necessário para processar a solicitação.

**Ações que você pode tentar:**

- Verifique os logs de rastreamento da solicitação com falha para obter informações adicionais sobre este erro. Para obter mais informações, clique [aqui](#).

**Detailed Error Information:**

<b>Module</b>	ManagedPipelineHandler	<b>Requested URL</b>	http://localhost:56476/AreaRestrita/Principal/Index
<b>Notification</b>	ExecuteRequestHandler	<b>Physical Path</b>	C:\Users\COTI\Desktop_Aula_08.03.18\Aula16\Projeto.WEB\AreaRestrita\Principal\Index
<b>Handler</b>	System.Web.Mvc.MvcHandler	<b>Logon Method</b>	Anônimo
<b>Error Code</b>	0x00000000	<b>Logon User</b>	Anônimo
		<b>Diretório de Rastreamento de Solicitação</b>	C:\Users\COTI\Documents\IISExpress\TraceLogFiles\PROJETO.WEB

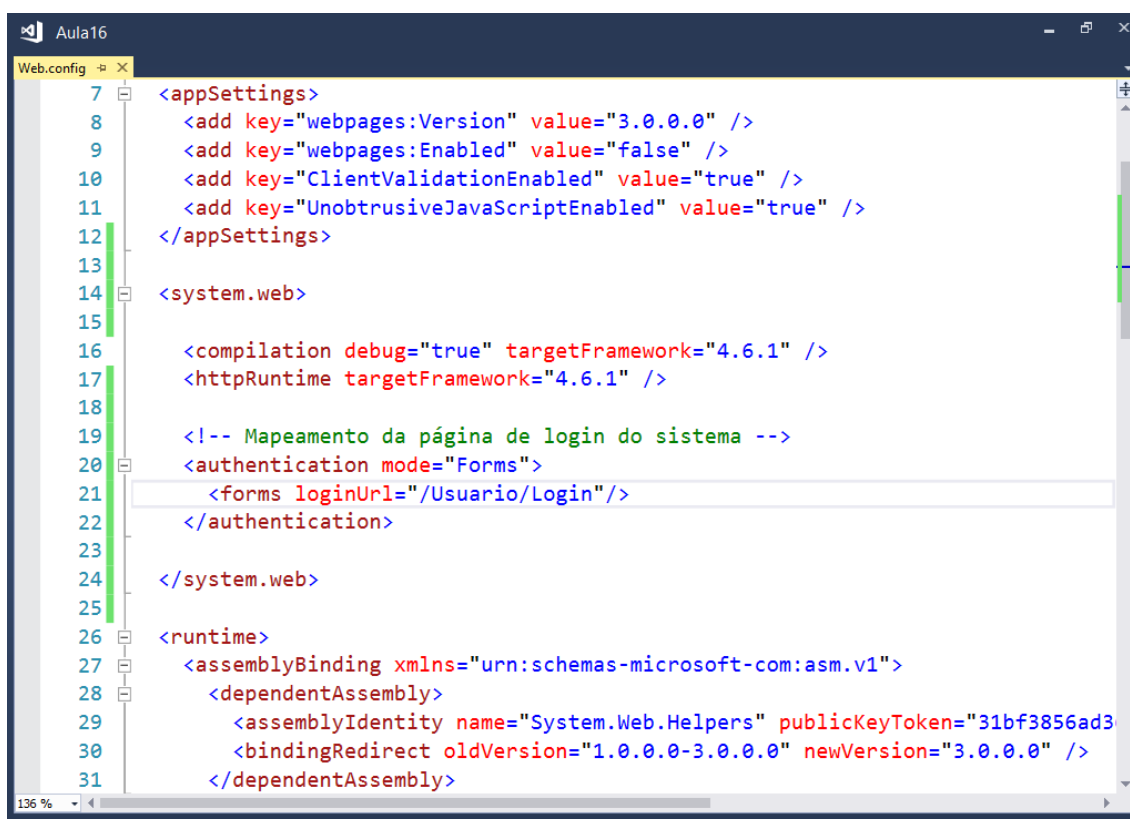
**More Information:**  
 Este é o erro genérico de Acesso Negado retornado pelo IIS. Em geral, há um código de substatus associado a este erro, descrevendo o motivo pelo qual o servidor recusou a solicitação. Verifique o arquivo de Log do IIS para determinar se há um código de substatus associado a essa falha.  
[View more information »](#)

Artigos da Base de Dados de Conhecimento Microsoft:

- 907273

Configuração para que o Asp.Net ao encontrar um erro de acesso não autorizado (401), redirecione para a página de login do site.

## \Web.config.xml



```

7 <appSettings>
8   <add key="webpages:Version" value="3.0.0.0" />
9   <add key="webpages:Enabled" value="false" />
10  <add key="ClientValidationEnabled" value="true" />
11  <add key="UnobtrusiveJavaScriptEnabled" value="true" />
12 </appSettings>
13
14 <system.web>
15
16   <compilation debug="true" targetFramework="4.6.1" />
17   <httpRuntime targetFramework="4.6.1" />
18
19   <!-- Mapeamento da página de login do sistema -->
20   <authentication mode="Forms">
21     <forms loginUrl="/Usuario/Login"/>
22   </authentication>
23
24 </system.web>
25
26 <runtime>
27   <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
28     <dependentAssembly>
29       <assemblyIdentity name="System.Web.Helpers" publicKeyToken="31bf3856ad3
30       <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
31     </dependentAssembly>

```

<system.web>

<compilation debug="true" targetFramework="4.6.1" />

<httpRuntime targetFramework="4.6.1" />

<!-- Mapeamento da página de login do sistema -->

<authentication mode="Forms">

<forms loginUrl="/Usuario/Login"/>

</authentication>

</system.web>



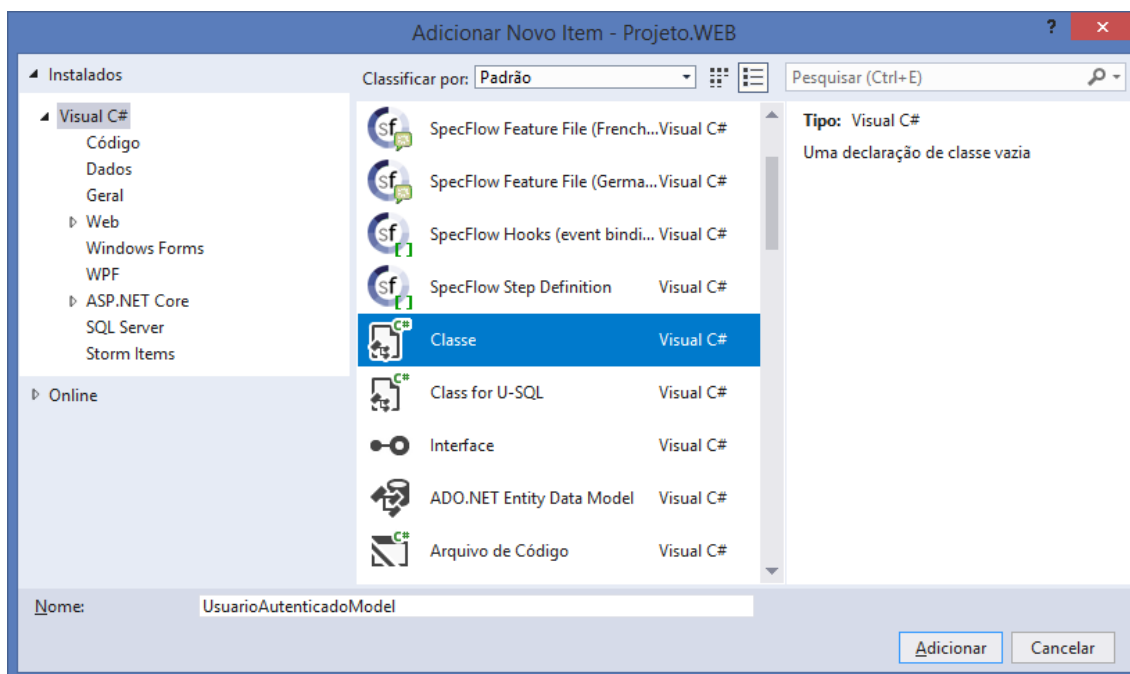
## Ticket de Acesso

Para que o Asp.Net possa autenticar um usuario, é criado um ticket de acesso contendo os dados do usuario autenticado. Para que os controles que possuem a anotação <authorize> possam permitir o acesso do usuario, este ticket deverá ser gravado em um cookie no navegador do usuario.

Tudo isso será feito com o auxilio das classes que estão no namespace:

## System.Web.Security

\*\* Para que possamos armazenar os dados do usuario no ticket, iremos criar uma classe Model (modelo) para representar estes dados.



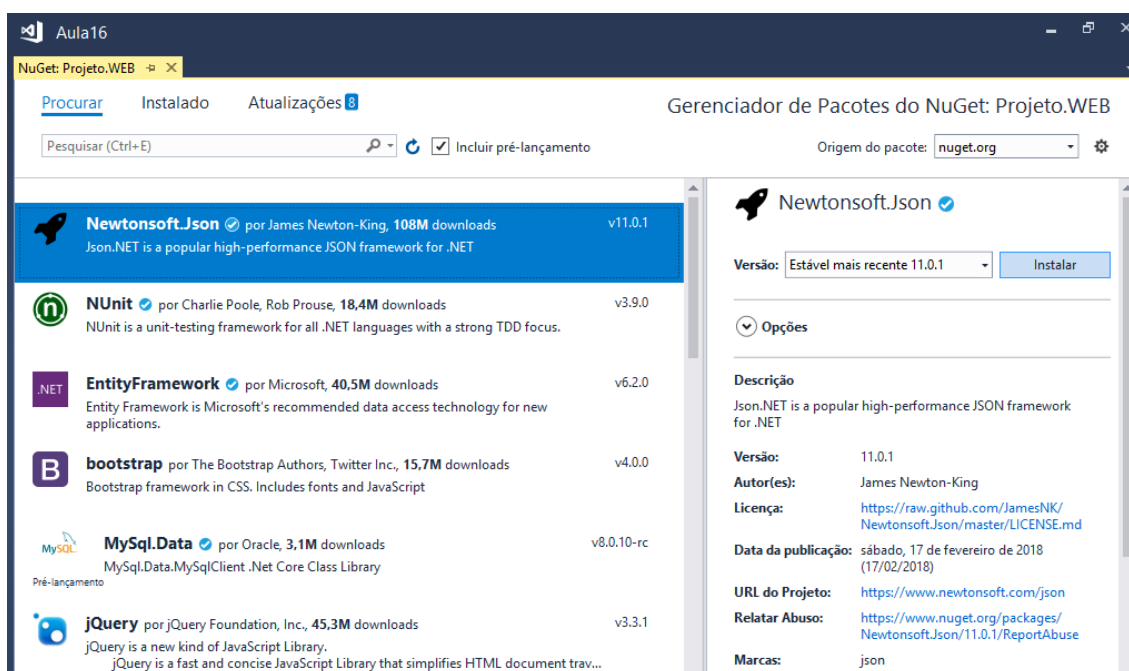
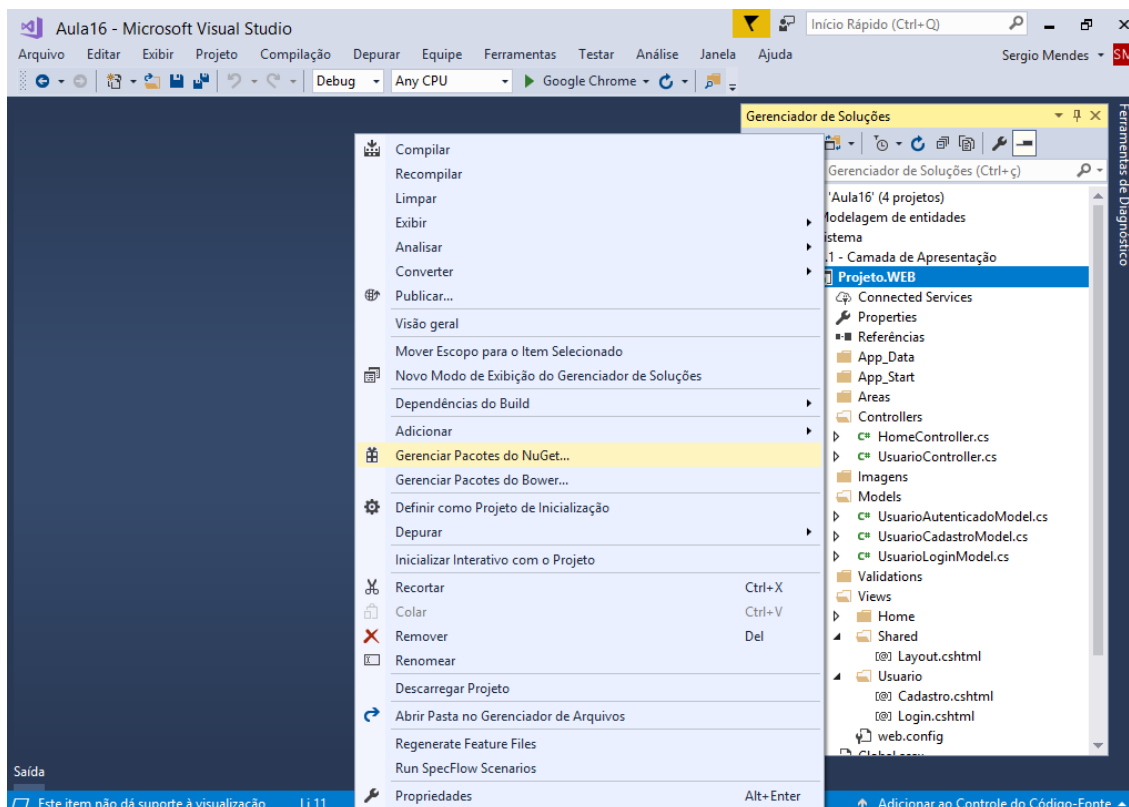
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```
namespace Projeto.WEB.Models
```

```
{
    public class UsuarioAutenticadoModel
    {
        public int IdUsuario { get; set; }
        public string Nome { get; set; }
        public string Login { get; set; }
        public DateTime DataHoraAcesso { get; set; }
        public string HostOrigem { get; set; }
    }
}
```

## JSON (JavaScript Object Notation)

Iremos utilizar uma biblioteca denominada **NewtonSoft.Json**, que nos permitira **serializar** um objeto de uma classe para JSON (em formato string) e também posteriormente **deserializar** um tipo JSON (string) para objeto (Classe)



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Projeto.WEB.Models;
using Projeto.Entidades;
using Projeto.DAL;
using System.IO;
using System.Web.Security; //autenticação..
using Newtonsoft.Json; //JSON..

namespace Projeto.WEB.Controllers
{
    public class UsuarioController : Controller
    {
        // GET: Usuario/Login
        public ActionResult Login()
        {
            return View();
        }

        // POST: Usuario/Login
        [HttpPost]
        public ActionResult Login(UsuarioLoginModel model)
        {
            if(ModelState.IsValid)
            {
                try
                {
                    //buscar o usuario no banco de dados..
                    UsuarioRepositorio rep = new UsuarioRepositorio();
                    Usuario u = rep.Find(model.Login, model.Senha);

                    if(u != null) //se o usuario foi encontrado..
                    {
                        UsuarioAutenticadoModel auth
                            = new UsuarioAutenticadoModel();
                        auth.IdUsuario = u.IdUsuario;
                        auth.Nome = u.Nome;
                        auth.Login = u.Login;
                        auth.DataHoraAcesso = DateTime.Now;
                        auth.HostOrigem = Request.UserHostAddress;

                        string json = JsonConvert.SerializeObject(auth);

                        //criando o ticket do usuario..
                        FormsAuthenticationTicket ticket
                            = new FormsAuthenticationTicket(json, false, 10);

                        //criando o cookie que irá armazenar o ticket..
                        HttpCookie cookie = new HttpCookie(FormsAuthentication
                            .FormsCookieName,
                            FormsAuthentication.Encrypt(ticket));

                        Response.Cookies.Add(cookie); //gravando no navegador..

                        //redirecionamento..
                        return RedirectToAction("Index", "Principal",
                            new { area = "AreaRestrita" });
                    }
                }
            }
        }
    }
}
```

```

        else
        {
            ViewBag.Mensagem = "Acesso Negado.
                                Usuario não encontrado.";
        }
    }
    catch(Exception e)
    {
        //mensagem de erro..
        ViewBag.Mensagem = e.Message;
    }
}

return View();
}

// GET: Usuario/Cadastro
public ActionResult Cadastro()
{
    return View();
}

// POST: Usuario/Cadastro
[HttpPost]
public ActionResult Cadastro(UsuarioCadastroModel model)
{
    //verificar se os campos da model passaram na validação..
    if(ModelState.IsValid)
    {
        try
        {
            UsuarioRepositorio rep = new UsuarioRepositorio();
            //verificar se o login do usuario ja foi cadastrado..
            if( ! rep.HasLogin(model.Login))
            {
                //cadastrar o usuario..
                Usuario u = new Usuario(); //instanciando..
                u.Nome = model.Nome;
                u.Login = model.Login;
                u.Senha = model.Senha;
                u.Foto = Guid.NewGuid().ToString()
                    + Path.GetExtension(model.Foto.FileName);

                //cadastrando o usuario..
                rep.Insert(u);

                //upload..
                string pasta = Server.MapPath("/Imagens/");
                model.Foto.SaveAs(pasta + u.Foto);

                ModelState.Clear(); //limpar os campos do formulário..
                ViewBag.Mensagem = $"Usuário {u.Nome},
                                    cadastrado com sucesso.";
            }
        }
        else
        {
            ViewBag.Mensagem = "Erro. Este login
                                já encontra-se cadastrado. Tente outro.";
        }
    }
}

```

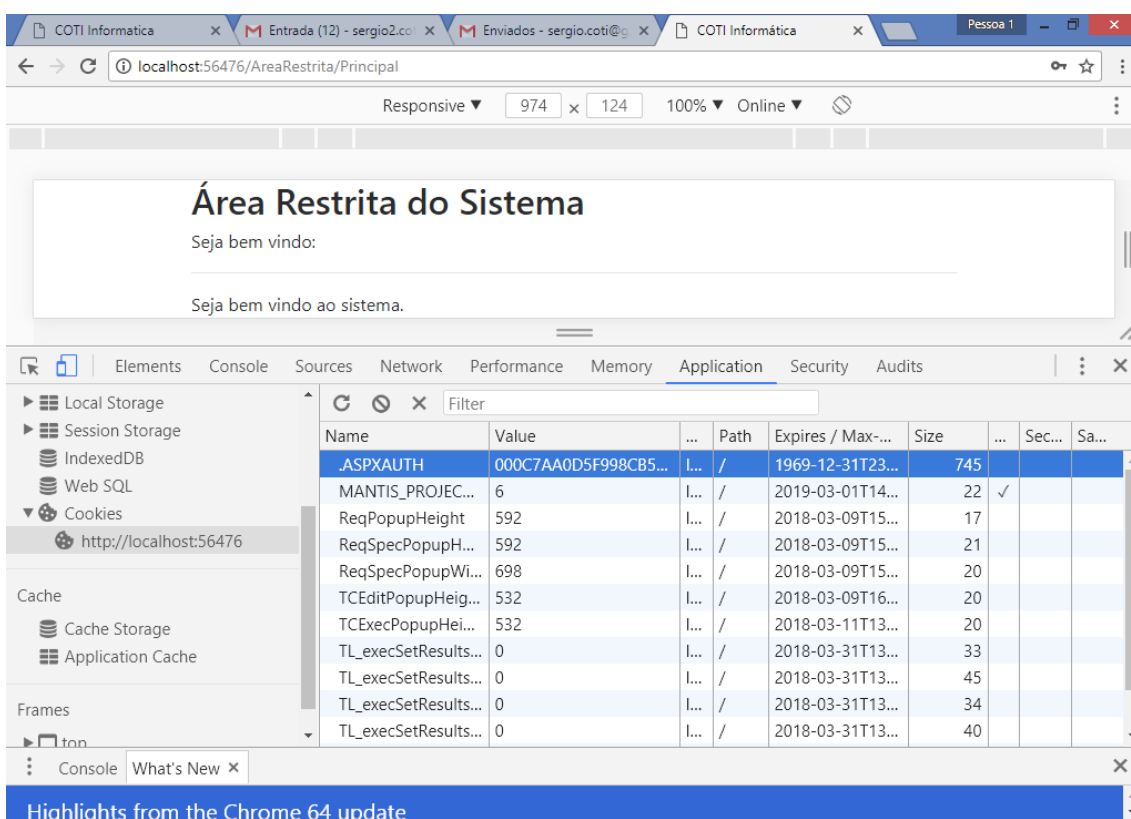
```

        catch(Exception e)
        {
            //enviar mensagem de erro para a página..
            ViewBag.Mensagem = e.Message;
        }
    }

    return View();
}
}
}
}

```

## Cookie gravado:



Name	Value	Path	Expires / Max-Age	Size
.ASPXAUTH	000C7AA0D5F998CB5...	/	1969-12-31T23...	745
MANTIS_PROJEC...	6	/	2019-03-01T14...	22
ReqPopupHeight	592	/	2018-03-09T15...	17
ReqSpecPopupH...	592	/	2018-03-09T15...	21
ReqSpecPopupWi...	698	/	2018-03-09T15...	20
TCEditPopupHeig...	532	/	2018-03-09T16...	20
TCEditPopupHei...	532	/	2018-03-11T13...	20
TL_execSetResults...	0	/	2018-03-31T13...	33
TL_execSetResults...	0	/	2018-03-31T13...	45
TL_execSetResults...	0	/	2018-03-31T13...	34
TL_execSetResults...	0	/	2018-03-31T13...	40

## Exibindo os dados do usuario autenticado no layout da área restrita



@using Newtonsoft.Json

```
@using Projeto.WEB.Models
```

```
@{
```

```
//recuperar o usuario que esta gravado no ticket..
```

```
string json = User.Identity.Name;
```

```
//deserializar o json para objeto..
```

```
UsuarioAutenticadoModel auth =
```

```
    JsonConvert.DeserializeObject<UsuarioAutenticadoModel>(json);
```

```
}
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

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

```
    <title>COTI Informática</title>
```

```
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
alpha.6/css/bootstrap.min.css" integrity="sha384-
rwoIResjU2yc3z8GV/NPeZWAv56rSmlldC3R/AZzGRnGxQQKnKkoFVhFQhNUwEyJ"
crossorigin="anonymous">
```

```
</head>
```

```
<body class="container">
```

```
    <div class="row">
```

```
        <div class="col-md-8">
```

```
            <h2>Área Restrita do Sistema</h2>
```

```
            PAINEL DE CONTROLE ADMINISTRATIVO <br/>
```

```
            <a href="/Usuario/Logout">Sair do Sistema</a>
```

```
        </div>
```

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

```
            <div class="row">
```

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

```
                    
```

```
                </div>
```

```
                <div class="col-md-8">
```

```
                    <strong>@auth.Nome</strong> <br/>
```

```
                    Login: @auth.Login <br/>
```

```
                    Acesso: @auth.DataHoraAcesso
```

```
                </div>
```

```
            </div>
```

```
        </div>
```

```
    </div>
```

```
<hr/>
```

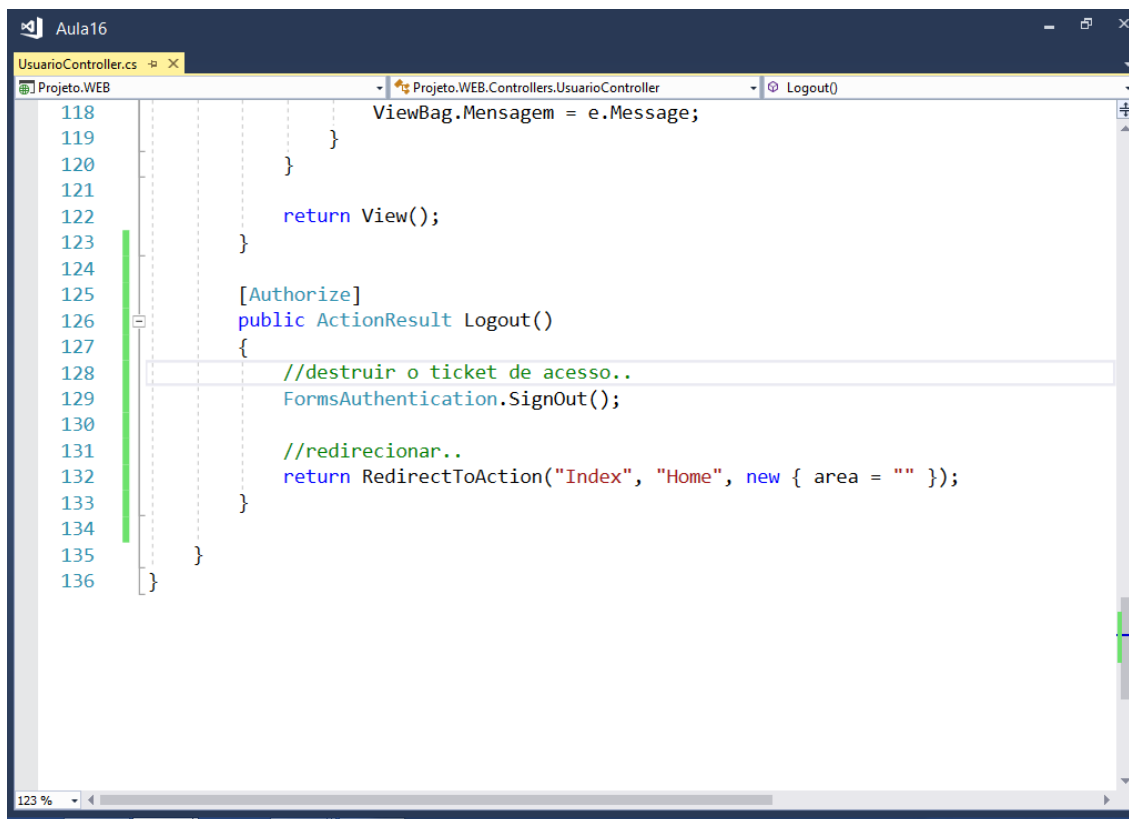
```
<!-- local para entrada do conteudo das demais páginas -->
```

```
@RenderBody()
```

```
</body>
```

```
</html>
```

## Logout do usuario:



```

118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136

```

```

[Authorize]
public ActionResult Logout()
{
    //destruir o ticket de acesso..
    FormsAuthentication.SignOut();

    //redirecionar..
    return RedirectToAction("Index", "Home", new { area = "" });
}

```

-----

Criando uma classe em MVC para limpar o cache das páginas da area restrita quando elas forem acessadas.

### Criando um Filter

Classe em MVC que intercepta uma ação de uma página e realiza uma atividade (por ser utilizado para auditoria, controle de cache, etc..)

```

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

namespace Projeto.WEB.Filters
{

```

```
public class NoCache : ActionFilterAttribute
{
    //filtro será executado sempre que a página for carregada..
    //sobrescrever o método OnResultExecuting
    public override void OnResultExecuting
        (ResultExecutingContext filterContext)
    {
        //limpar o cache da página..
        filterContext.HttpContext.Response.Cache.SetExpires
            (DateTime.UtcNow.AddDays(-1));

        filterContext.HttpContext.Response.Cache
            .SetValidUntilExpires(false);

        filterContext.HttpContext.Response.Cache.SetRevalidation
            (HttpCacheRevalidation.AllCaches);

        filterContext.HttpContext.Response.Cache
            .SetCacheability(HttpCacheability.NoCache);

        filterContext.HttpContext.Response.Cache.SetNoStore();

        base.OnResultExecuting(filterContext);
    }
}
```

## Aplicando o filtro:

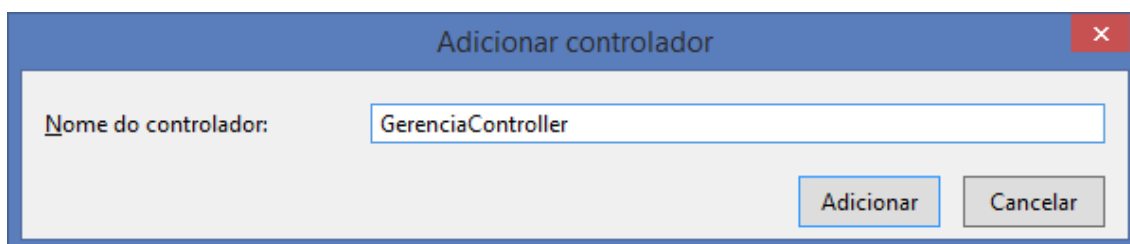
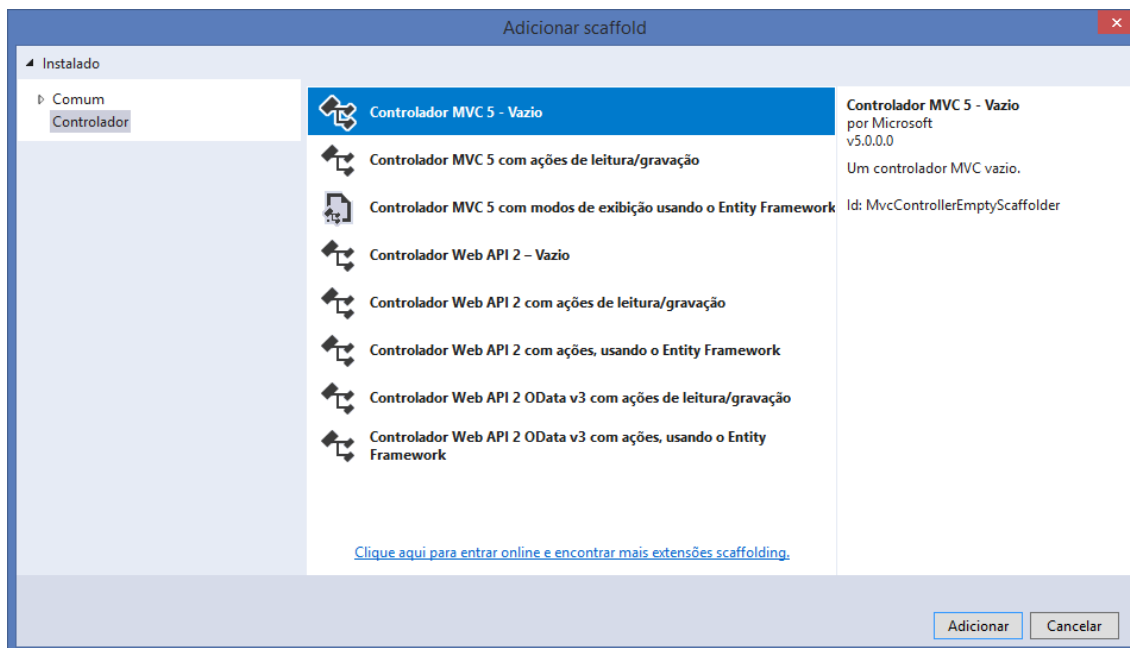
/PrincipalController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Projeto.WEB.Filters;

namespace Projeto.WEB.Areas.AreaRestrita.Controllers
{
    [Authorize]
    public class PrincipalController : Controller
    {
        // GET: AreaRestrita/Principal
        [NoCache]
        public ActionResult Index()
        {
            return View();
        }
    }
}
```



## Exibindo conteudos conforme o perfil do usuario:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Projeto.WEB.Filters;

namespace Projeto.WEB.Areas.AreaRestrita.Controllers
{
    [Authorize(Roles = "Gerente")]
    public class GerenciaController : Controller
    {
        // GET: AreaRestrita/Gerencia
        [NoCache]
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

## Página de Layout da área restrita:

```
@using Newtonsoft.Json
@using Projeto.WEB.Models

@{
    //recuperar o usuario que esta gravado no ticket..
    string json = User.Identity.Name;
    //deserializar o json para objeto..
    UsuarioAutenticadoModel auth =
        JsonConvert.DeserializeObject<UsuarioAutenticadoModel>(json);
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>COTI Informática</title>

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
alpha.6/css/bootstrap.min.css" integrity="sha384-
rwoIResjU2yc3z8GV/NPeZWAv56rSmLldC3R/AZzGRnGxQQKnKkoFVhFQhNUwEyJ"
crossorigin="anonymous">

</head>
<body class="container">

    <div class="row">
        <div class="col-md-8">
            <h2>Área Restrita do Sistema</h2>
            PAINEL DE CONTROLE ADMINISTRATIVO <br/>
            <a href="/Usuario/Logout">Sair do Sistema</a>
        </div>
        <div class="col-md-4">
            <div class="row">
                <div class="col-md-4">
                    
                </div>
                <div class="col-md-8">
                    <strong>@auth.Nome</strong> <br/>
                    Login: @auth.Login <br/>
                    Acesso: @auth.DataHoraAcesso
                </div>
            </div>
        </div>
    </div>
    <hr/>
    @if(User.IsInRole("Gerente"))
    {
        <a href="/AreaRestrita/Gerencia/Index">Área do Gerente</a>
        <br/>
        <br/>
    }

    <!-- local para entrada do conteudo das demais páginas -->
    @RenderBody()
</body>
</html>
```

## Página de login de usuarios:

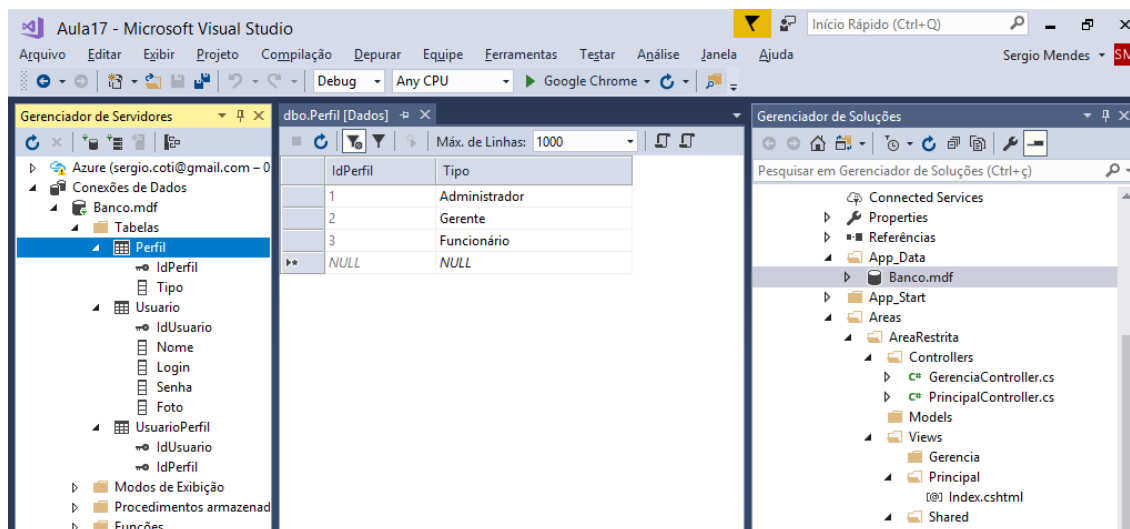
```

1
2 @if(User.Identity.IsAuthenticated)
3 {
4     Response.Redirect("/AreaRestrita/Principal/Index");
5 }
6
7 @{
8     ViewBag.Title = "Login";
9     Layout = "~/Views/Shared/Layout.cshtml";
10 }
11
12 <h4>Login de Usuários</h4>
13 <a href="/Home/Index">Voltar</a>
14 <hr />
15
16 <div class="row">
17     <div class="col-md-3">
18
19         <!-- Declarando a classe de modelo.. -->
20         @model Projeto.WEB.Models.UsuarioLoginModel
21
22         <!-- Abrindo a área do formulário -->
23         @using (Html.BeginForm("Login", "Usuario", FormMethod.Post))
    
```

```

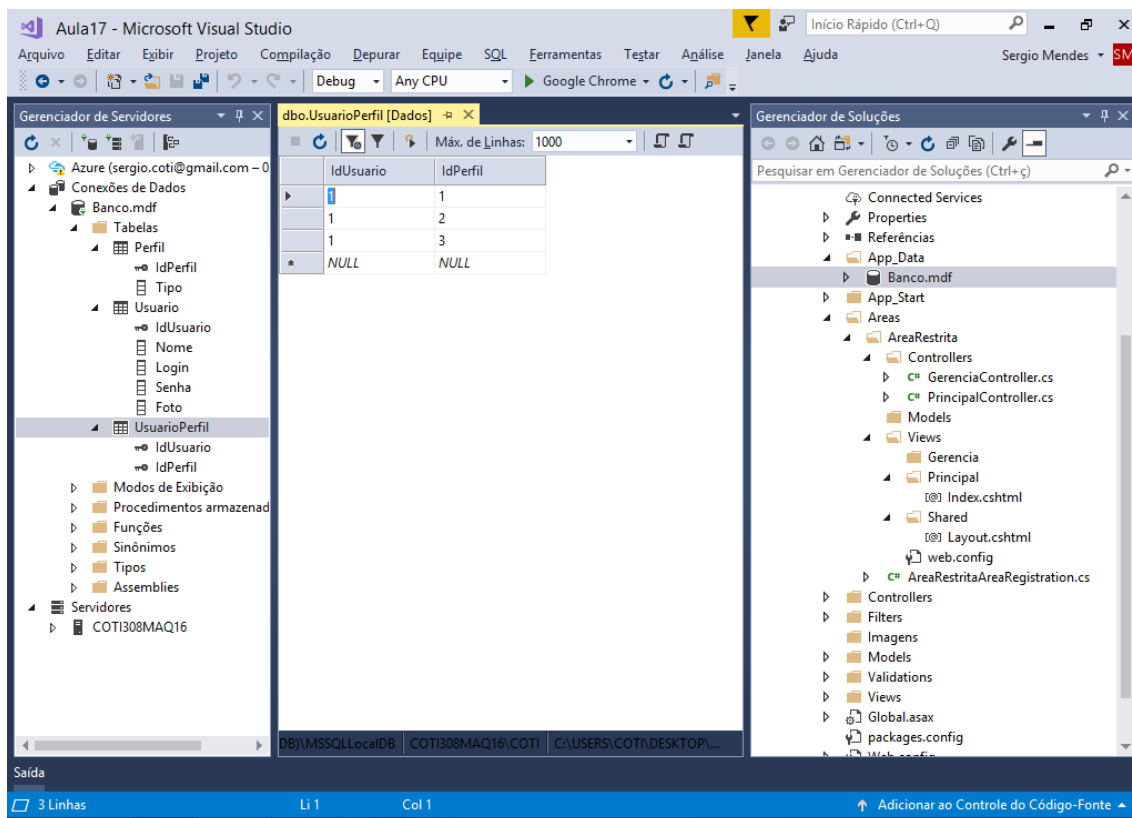
@if(User.Identity.IsAuthenticated)
{
    Response.Redirect("/AreaRestrita/Principal/Index");
}
    
```

## Cadastrando perfis na tabela:



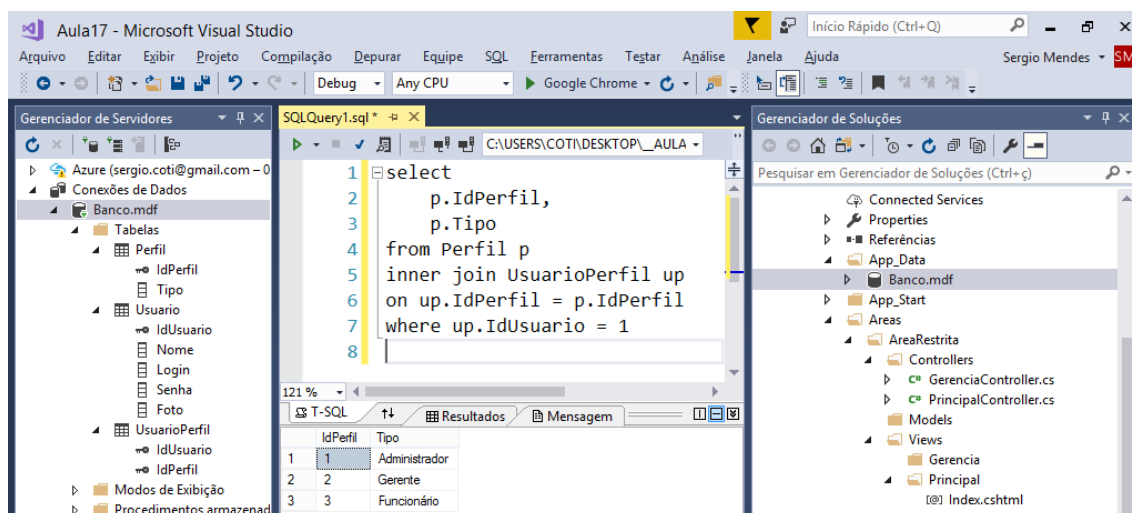
IdPerfil	Tipo
1	Administrador
2	Gerente
3	Funcionário
NULL	NULL

## Associando perfis para um usuario na base de dados:



Criando uma consulta no banco de dados para obter os perfis de um determinado usuario:

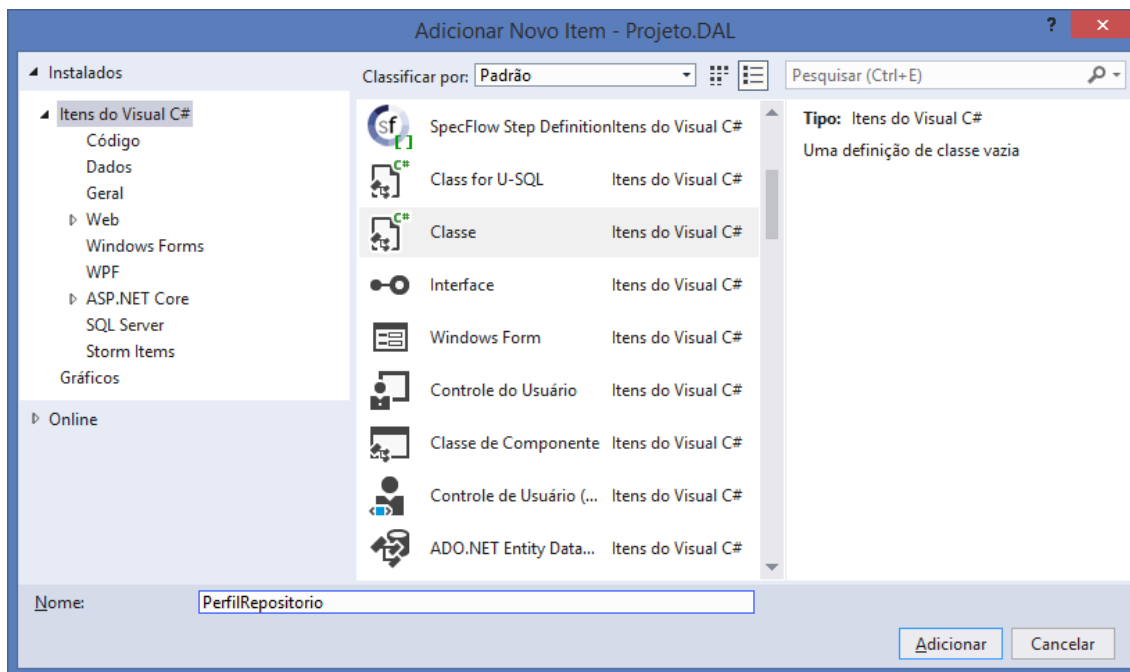
```
select
    p.IdPerfil,
    p.Tipo
from Perfil p
inner join UsuarioPerfil up
on up.IdPerfil = p.IdPerfil
where up.IdUsuario = 1
```



IdPerfil	Tipo
1	Administrador
2	Gerente
3	Funcionário

## Classe de persistencia de dados para a entidade Perfil:

/PerfilRepositorio.cs



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient; //sqlserver..
using Projeto.Entidades; //classes de entidade..

namespace Projeto.DAL
{
    public class PerfilRepositorio : Conexao
    {
        //método para retornar os perfis de um usuario..
        public List<Perfil> FindByUsuario(int idUsuario)
        {
            OpenConnection();

            string query = "select p.IdPerfil, p.Tipo "
                + "from Perfil p "
                + "inner join UsuarioPerfil up "
                + "on up.IdPerfil = p.IdPerfil "
                + "where up.IdUsuario = @IdUsuario";

            cmd = new SqlCommand(query, con);
            cmd.Parameters.AddWithValue("@IdUsuario", idUsuario);
            dr = cmd.ExecuteReader();

            List<Perfil> lista = new List<Perfil>(); //instanciando..

            while(dr.Read()) //enquanto houver registros..
            {
                Perfil p = new Perfil();
            }
        }
    }
}
```

```

        p.IdPerfil = Convert.ToInt32(dr["IdPerfil"]);
        p.Tipo = Convert.ToString(dr["Tipo"]);

        lista.Add(p); //adicionar na lista..
    }

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

```

## Classe de modelo que armazena os dados do usuario autenticado:

```

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

namespace Projeto.WEB.Models
{
    public class UsuarioAutenticadoModel
    {
        public int IdUsuario { get; set; }
        public string Nome { get; set; }
        public string Login { get; set; }
        public string Foto { get; set; }
        public DateTime DataHoraAcesso { get; set; }
        public string HostOrigem { get; set; }
        public List<string> Perfis { get; set; }
    }
}

```

Método de autenticação do usuario:

## /UserController.cs

```

// POST: Usuario/Login
[HttpPost]
public ActionResult Login(UsuarioLoginModel model)
{
    if(ModelState.IsValid)
    {
        try
        {
            //buscar o usuario no banco de dados..
            UsuarioRepositorio rep = new UsuarioRepositorio();
            Usuario u = rep.Find(model.Login, model.Senha);

            if(u != null) //se o usuario foi encontrado..
            {
                UsuarioAutenticadoModel auth
                    = new UsuarioAutenticadoModel();
            }
        }
    }
}

```

```

auth.IdUsuario = u.IdUsuario;
auth.Nome = u.Nome;
auth.Login = u.Login;
auth.Foto = u.Foto;
auth.DataHoraAcesso = DateTime.Now;
auth.HostOrigem = Request.UserHostAddress;

//buscar os perfis do usuario..
PerfilRepositorio perfilRep = new PerfilRepositorio();
List<Perfil> lista
    = perfilRep.FindByUsuario(u.IdUsuario);

//armazenar o nome da cada perfil na classe de modelo..
auth.Perfis = lista.Select(p => p.Tipo).ToList();

string json = JsonConvert.SerializeObject(auth);

//criando o ticket do usuario..
FormsAuthenticationTicket ticket
    = new FormsAuthenticationTicket(json, false, 10);

//criando o cookie que irá armazenar o ticket..
HttpCookie cookie = new HttpCookie(FormsAuthentication
    .FormsCookieName,
    FormsAuthentication.Encrypt(ticket));
Response.Cookies.Add(cookie); //gravando no navegador..

//redirecionamento..
return RedirectToAction("Index", "Principal",
    new { area = "AreaRestrita" });
}
else
{
    ViewBag.Mensagem = "Acesso Negado. Usuario
        não encontrado.";
}
}
catch(Exception e)
{
    //mensagem de erro..
    ViewBag.Mensagem = e.Message;
}
}

return View();
}

```

## Exibindo os perfis do usuario autenticado:

/Areas/AreaRestrita/Shared/Layout.cshtml

```

@using Newtonsoft.Json
@using Projeto.WEB.Models

```

```

@{
    //recuperar o usuario que esta gravado no ticket..
    string json = User.Identity.Name;
}

```

```
//deserializar o json para objeto..
UsuarioAutenticadoModel auth =
    JsonConvert.DeserializeObject<UsuarioAutenticadoModel>(json);

}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>COTI Informática</title>

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.6/css/bootstrap.min.css" integrity="sha384-rwoIREsjU2yc3z8GV/NPeZWAv56rSmLldC3R/AZzGRnGxQQKnKkoFVhFQhNUwEyJ" crossorigin="anonymous">

</head>
<body class="container">

    <div class="row">
        <div class="col-md-8">
            <h2>Área Restrita do Sistema</h2>
            PAINEL DE CONTROLE ADMINISTRATIVO <br/>
            <a href="/Usuario/Logout">Sair do Sistema</a>
        </div>
        <div class="col-md-4">
            <div class="row">
                <div class="col-md-4">
                    
                </div>
                <div class="col-md-8">
                    <strong>@auth.Nome</strong> <br/>
                    Login: @auth.Login <br/>
                    Perfis: @string.Join("|", auth.Perfis) <br/>
                    Acesso: @auth.DataHoraAcesso
                </div>
            </div>
        </div>
    </div>

    <hr/>

    @if(User.IsInRole("Gerente"))
    {
        <a href="/AreaRestrita/Gerencia/Index">Área do Gerente</a>
        <br/>
        <br/>
    }

    <!-- local para entrada do conteudo das demais páginas -->
    @RenderBody()

</body>
</html>
```





No ticket gravado pelo MVC, o objeto `UsuarioAutenticadoModel` possui uma propriedade "Perfis" que contem o nome de cada Perfil do usuario obtido do banco de dados.

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

```
namespace Projeto.WEB.Models
```

```
{
```

```
public class UsuarioAutenticadoModel
```

```
{
```

```
public int IdUsuario { get; set; }
```

```
public string Nome { get; set; }
```

```
public string Login { get; set; }
```

```
public string Foto { get; set; }
```

```
public DateTime DataHoraAcesso { get; set; }
```

```
public string HostOrigem { get; set; }
```

```
public List<string> Perfis { get; set; }
```

```
}
```

```
}
```

Como podemos associar para o MVC que é esta propriedade acima (Perfis) que contem os nomes dos perfis exigidos pelas classes de controle?

```
[Authorize(Roles = "Gerente")]
```

```
public class GerenciaController : Controller
```

```
{
```

```
}
```

## Para isso, iremos incluir uma configuração na classe Global.asax do projeto

\*\* Global.asax: Classe que configura a maioria das tecnologias utilizadas no projeto bem como suas configurações.

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

namespace Projeto.WEB
{
    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();
            RouteConfig.RegisterRoutes(RouteTable.Routes);
        }
    }
}

-----

using Newtonsoft.Json;
using Projeto.WEB.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Principal;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;
using System.Web.Security;

namespace Projeto.WEB
{
    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();
            RouteConfig.RegisterRoutes(RouteTable.Routes);
        }

        protected void Application_AuthenticateRequest(
            Object sender, EventArgs e)
        {
            if (HttpContext.Current.User != null)
            {
                if (HttpContext.Current.User.Identity.IsAuthenticated)
                {
                    if (HttpContext.Current.User.Identity is FormsIdentity)
                    {
                        FormsIdentity id = (FormsIdentity)
                            HttpContext.Current.User.Identity;
                    }
                }
            }
        }
    }
}
```

```
FormsAuthenticationTicket ticket = id.Ticket;

UsuarioAutenticadoModel model
    = JsonConvert.DeserializeObject
        <UsuarioAutenticadoModel>(ticket.Name);

string[] roles = model.Perfis.ToArray();

HttpContext.Current.User =
    new GenericPrincipal(id, roles);
}
}
}
}
}
}
```

