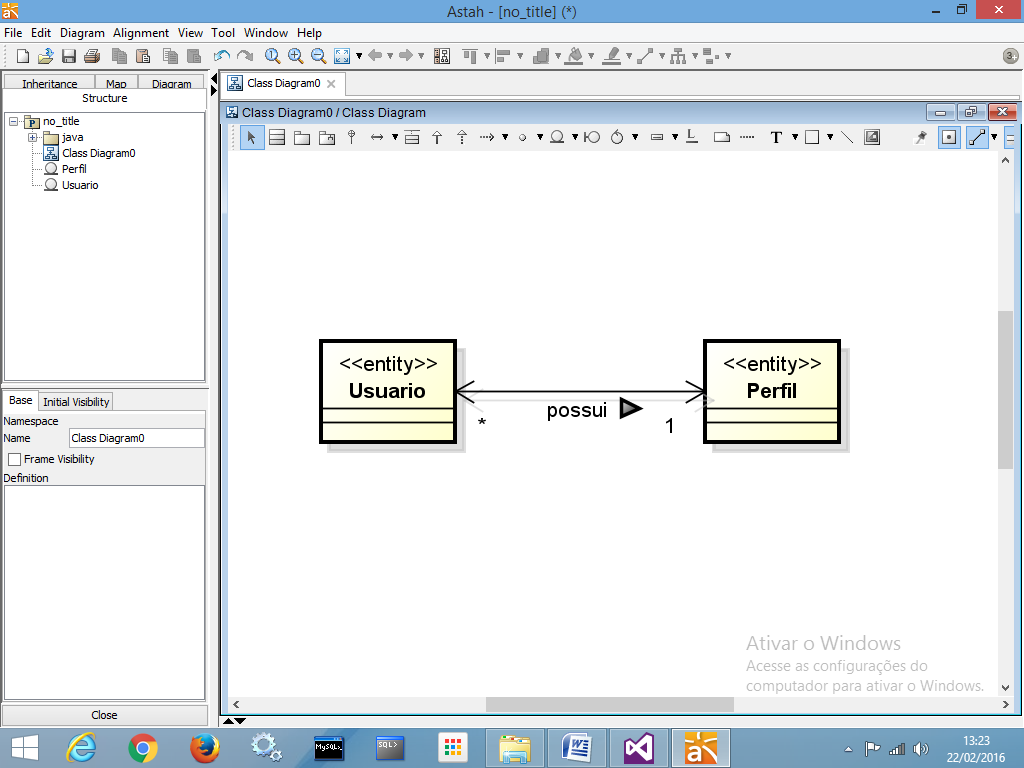
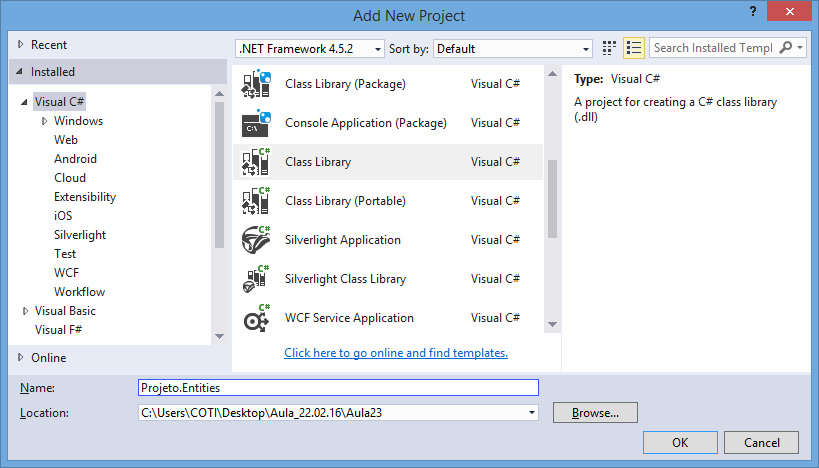
Modelagem de entidades

Diagrama de Classes



**Criando o projeto Class Library para entidades**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto.Entities

{

public class Perfil

{

public int IdPerfil { get; set; }

public string Nome { get; set; }

//Relacionamento (TER-Muitos)

public virtual ICollection<Usuario> Usuarios { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto.Entities

{

public class Usuario

{

public int IdUsuario { get; set; }

public string Nome { get; set; }

public string Email { get; set; }

public string Senha { get; set; }

public string Foto { get; set; }

public DateTime DataCadastro { get; set; }

public int IdPerfil { get; set; } //foreign key..

public virtual Perfil Perfil { get; set; }

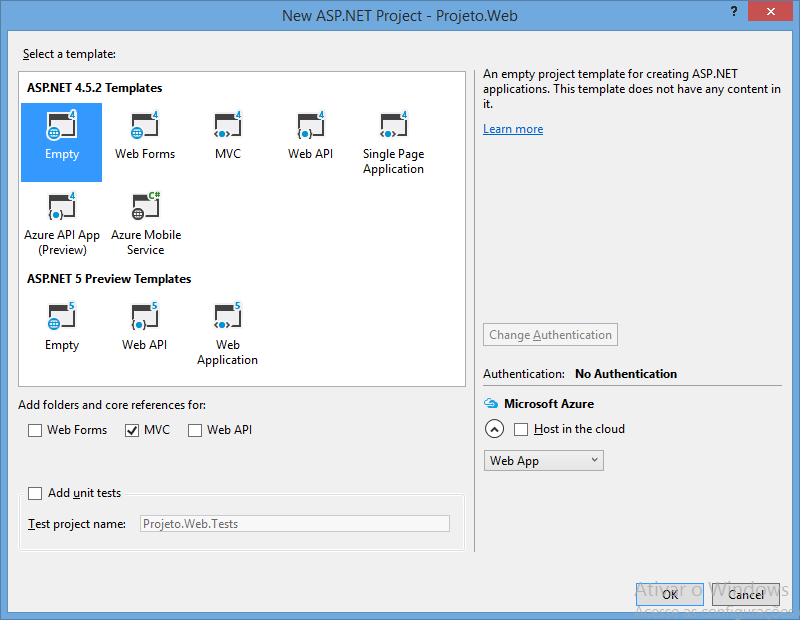
}

}

**Criando o projeto Web:**

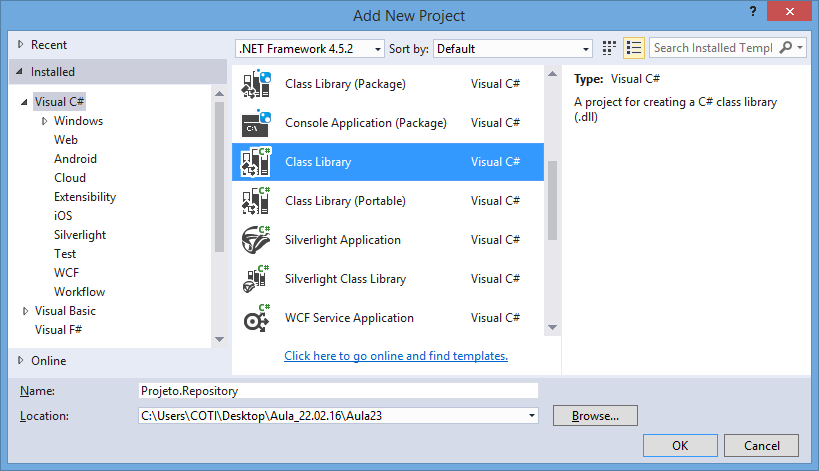
Camada de Apresentação

Empty > MVC

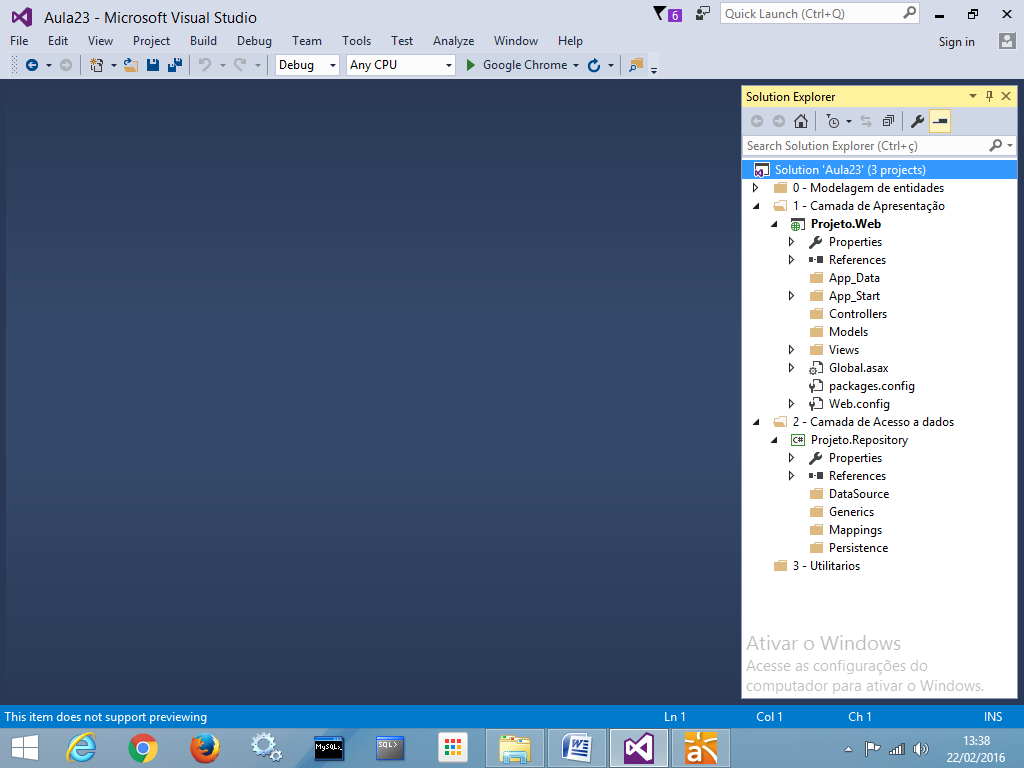


**Criando a camada de acesso a dados:**

Repositorio de dados (Repository)

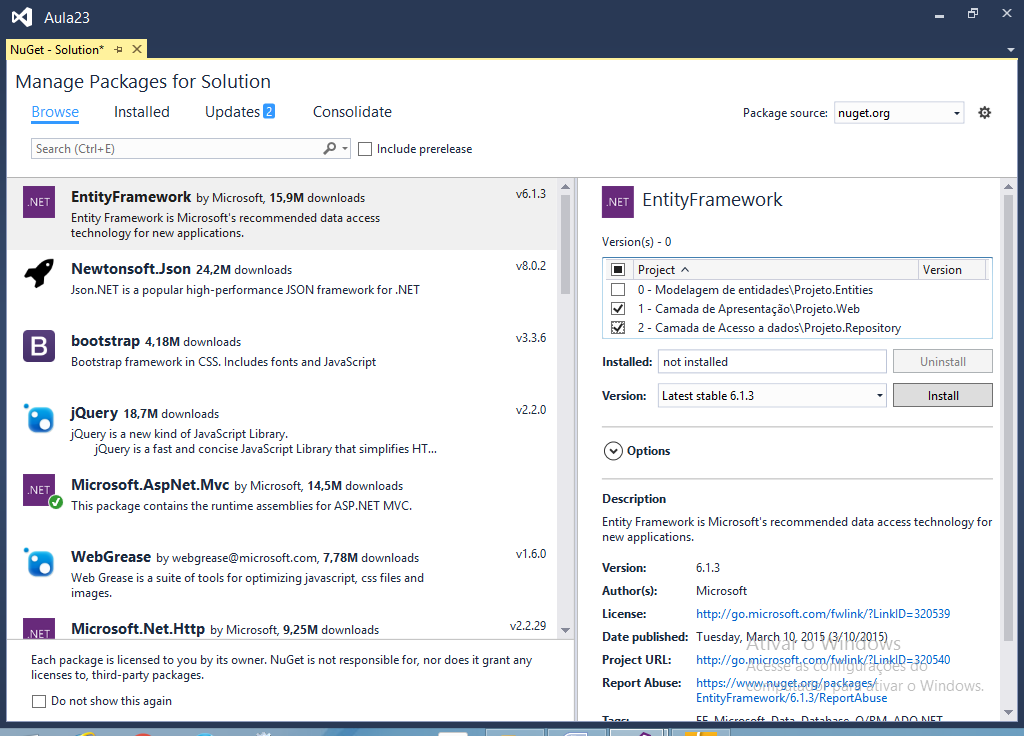


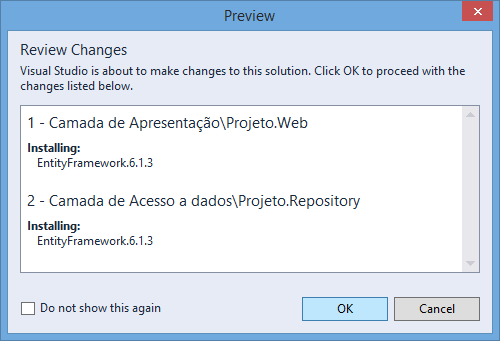
**Estrutura inicial:**



Instalando o EntityFramework:

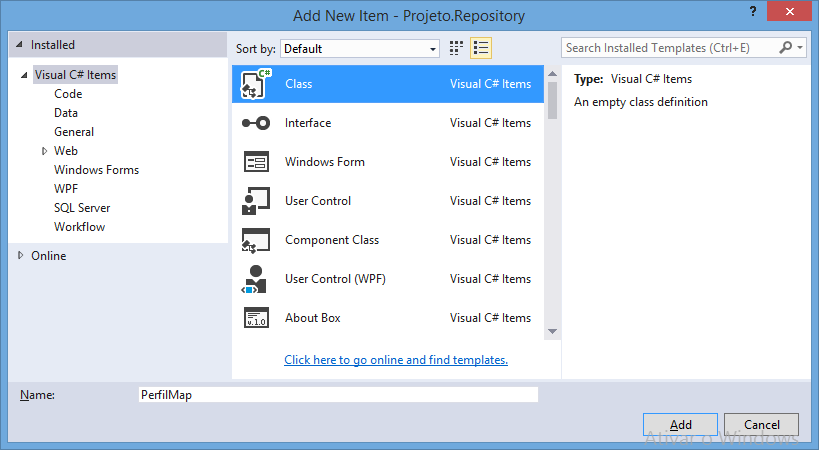
Manage NuGet Packages





Mapeamento utilizando Fluent API

Classes de mapeamento (sem annotations)



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.Entity.ModelConfiguration; //mapeamento..

using Projeto.Entities; //entidades..

namespace Projeto.Repository.Mappings

{

**public class PerfilMap : EntityTypeConfiguration<Perfil>**

{

public PerfilMap()

{

ToTable("PERFIL"); //nome da tabela

HasKey(p => p.IdPerfil); //chave primária..

//demais propriedades..

Property(p => p.IdPerfil)

.HasColumnName("IDPERFIL\_PK");

Property(p => p.Nome)

.HasColumnName("NOME")

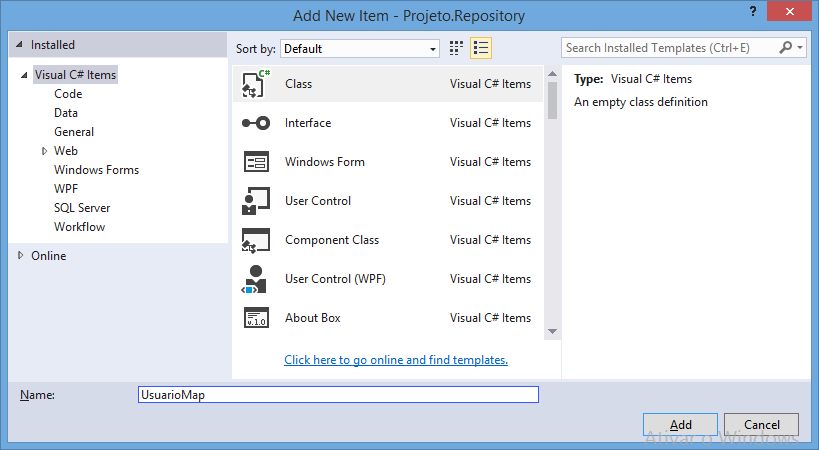
.HasMaxLength(50)

.IsRequired();

}

}

}



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.Entity.ModelConfiguration;

using System.Data.Entity.Infrastructure.Annotations;

using Projeto.Entities;

using System.ComponentModel.DataAnnotations.Schema;

namespace Projeto.Repository.Mappings

{

public class UsuarioMap : EntityTypeConfiguration<Usuario>

{

public UsuarioMap()

{

ToTable("USUARIO"); //nome da tabela..

HasKey(u => u.IdUsuario); //chave primaria..

//demais propriedades..

Property(u => u.IdUsuario)

.HasColumnName("IDUSUARIO\_PK");

Property(u => u.Nome)

.HasColumnName("NOME")

.HasMaxLength(50)

.IsRequired();

Property(u => u.Email)

.HasColumnName("EMAIL")

.HasMaxLength(50)

.IsRequired()

.HasColumnAnnotation(

IndexAnnotation.AnnotationName,

new IndexAnnotation(new IndexAttribute("IDXEMAIL", 1)

{ IsUnique = true }));

Property(u => u.Senha)

.HasColumnName("SENHA")

.HasMaxLength(50)

.IsRequired();

Property(u => u.Foto)

.HasColumnName("FOTO")

.HasMaxLength(50)

.IsRequired();

Property(u => u.DataCadastro)

.HasColumnName("DATACADASTRO")

.IsRequired();

Property(u => u.IdPerfil)

.HasColumnName("IDPERFIL\_FK")

.IsRequired();

//Mapeamento do relacionamento..

HasRequired(u => u.Perfil) //Usuario TEM 1 Perfil

.WithMany(p => p.Usuarios) //Perfil TEM Muitos Usuarios

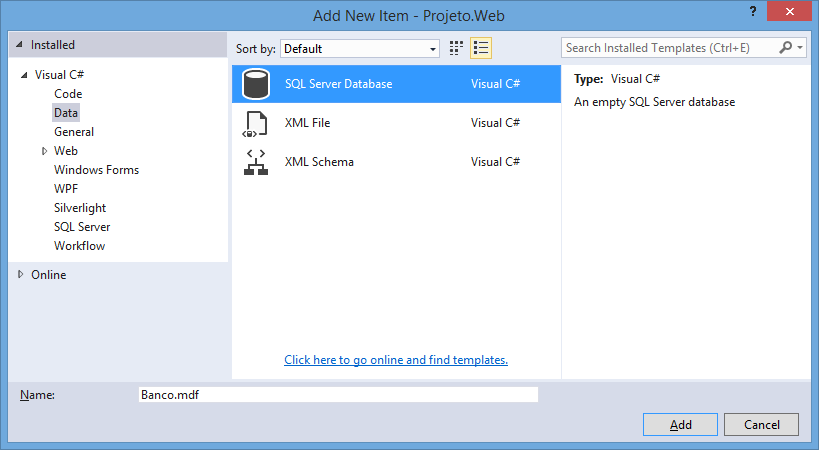
.HasForeignKey(u => u.IdPerfil); //foreign key..

}

}

}

**Criando a base de dados (MDF)**



Web.config.xml

Mapeando a connectionstring

<connectionStrings>

<add

name="aula"

connectionString="Data Source=(LocalDB)\MSSQLLocalDB;

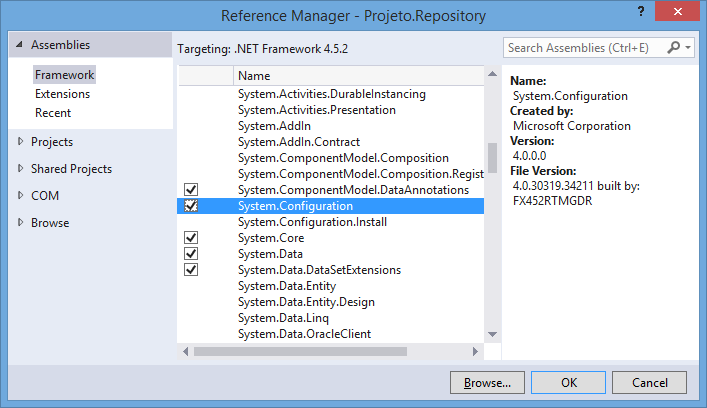
AttachDbFilename=C:\Users\COTI\Desktop\

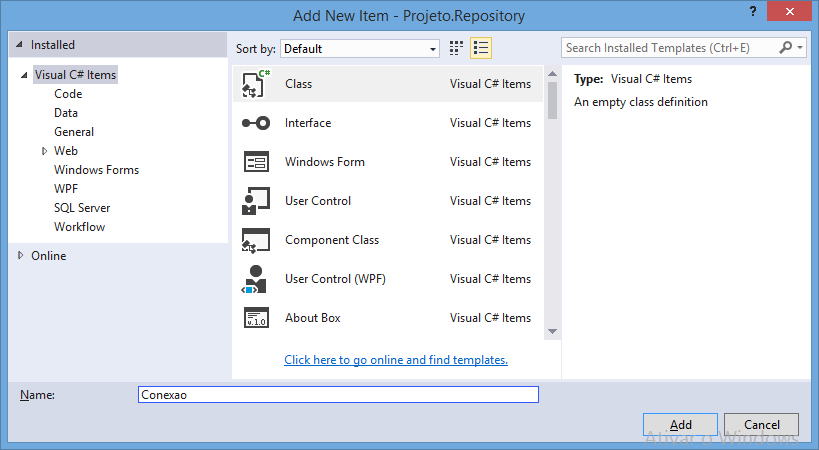
Aula\_22.02.16\Aula23\Projeto.Web\App\_Data\Banco.mdf;

Integrated Security=True"

/>

</connectionStrings>





using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.Entity; //entityframework

using System.Configuration; //connectionstring

using Projeto.Entities; //entidades..

using Projeto.Repository.Mappings; //mapeamentos..

namespace Projeto.Repository.DataSource

{

public class Conexao : DbContext

{

//construtor..

public Conexao()

: base(ConfigurationManager.ConnectionStrings

["aula"].ConnectionString)

{

}

//sobrescrever o método OnModelCreating..

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

modelBuilder.Configurations.Add(new UsuarioMap());

modelBuilder.Configurations.Add(new PerfilMap());

}

public DbSet<Usuario> Usuario { get; set; }

public DbSet<Perfil> Perfil { get; set; }

}

}

Migrando a base de dados:

Service Migrations

Enable-Migrations -Force

PM> Enable-Migrations -Force

Checking if the context targets an existing database...

Code First Migrations enabled for project Projeto.Repository.

namespace Projeto.Repository.Migrations

{

using Entities;

using System;

using System.Data.Entity;

using System.Data.Entity.Migrations;

using System.Linq;

internal sealed class Configuration : DbMigrationsConfiguration

<Projeto.Repository.DataSource.Conexao>

{

public Configuration()

{

AutomaticMigrationsEnabled = true;

}

protected override void Seed(Projeto.Repository.

DataSource.Conexao context)

{

context.Perfil.AddOrUpdate(

p => p.Nome,

new Perfil { Nome = "Administrador" },

new Perfil { Nome = "Funcionario" },

new Perfil { Nome = "Cliente" }

);

}

}

}

Gerando a base de dados:

Update-Database -Verbose

CREATE TABLE [dbo].[PERFIL] (

[IDPERFIL\_PK] [int] NOT NULL IDENTITY,

[NOME] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_dbo.PERFIL] PRIMARY KEY ([IDPERFIL\_PK])

)

CREATE TABLE [dbo].[USUARIO] (

[IDUSUARIO\_PK] [int] NOT NULL IDENTITY,

[NOME] [nvarchar](50) NOT NULL,

[EMAIL] [nvarchar](50) NOT NULL,

[SENHA] [nvarchar](50) NOT NULL,

[FOTO] [nvarchar](50) NOT NULL,

[DATACADASTRO] [datetime] NOT NULL,

[IDPERFIL\_FK] [int] NOT NULL,

CONSTRAINT [PK\_dbo.USUARIO] PRIMARY KEY ([IDUSUARIO\_PK])

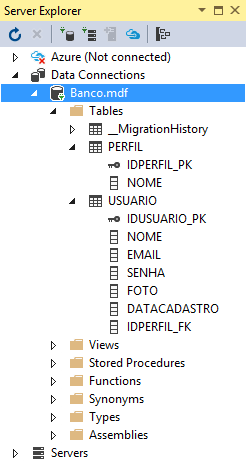
)

CREATE UNIQUE INDEX [IDXEMAIL] ON [dbo].[USUARIO]([EMAIL])

CREATE INDEX [IX\_IDPERFIL\_FK] ON [dbo].[USUARIO]([IDPERFIL\_FK])

ALTER TABLE [dbo].[USUARIO] ADD CONSTRAINT [FK\_dbo.USUARIO\_dbo.PERFIL\_IDPERFIL\_FK] FOREIGN KEY ([IDPERFIL\_FK]) REFERENCES [dbo].[PERFIL] ([IDPERFIL\_PK]) ON DELETE CASCADE

**Base de dados:**



Repositorio Generico:

(GenericDAL)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto.Repository.Generics

{

public interface IGenericRepository<TEntity>

where TEntity : class

{

void Insert(TEntity obj);

void Update(TEntity obj);

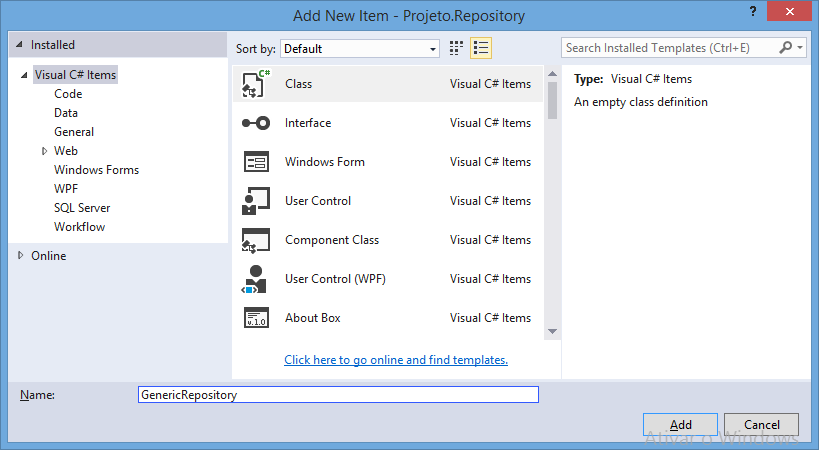
void Delete(TEntity obj);

List<TEntity> FindAll();

TEntity FindById(int id);

}

}



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.Entity; //entityframework..

using Projeto.Repository.DataSource; //conexão..

namespace Projeto.Repository.Generics

{

public abstract class GenericRepository<TEntity> : IGenericRepository<TEntity>

where TEntity : class

{

public virtual void Insert(TEntity obj)

{

using (Conexao Con = new Conexao())

{

Con.Entry(obj).State = EntityState.Added;

Con.SaveChanges(); //executando..

}

}

public virtual void Update(TEntity obj)

{

using (Conexao Con = new Conexao())

{

Con.Entry(obj).State = EntityState.Modified;

Con.SaveChanges(); //executando..

}

}

public virtual void Delete(TEntity obj)

{

using (Conexao Con = new Conexao())

{

Con.Entry(obj).State = EntityState.Deleted;

Con.SaveChanges(); //executando..

}

}

public virtual List<TEntity> FindAll()

{

using (Conexao Con = new Conexao())

{

return Con.Set<TEntity>().ToList();

}

}

public virtual TEntity FindById(int id)

{

using (Conexao Con = new Conexao())

{

return Con.Set<TEntity>().Find(id);

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto.Repository.DataSource; //conexão..

using Projeto.Repository.Generics; //generics..

using Projeto.Entities; //entidades

namespace Projeto.Repository.Persistence

{

public class UsuarioRepository

: GenericRepository<Usuario>

{

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto.Repository.DataSource; //conexão..

using Projeto.Repository.Generics; //generics..

using Projeto.Entities; //entidades

namespace Projeto.Repository.Persistence

{

public class PerfilRepository

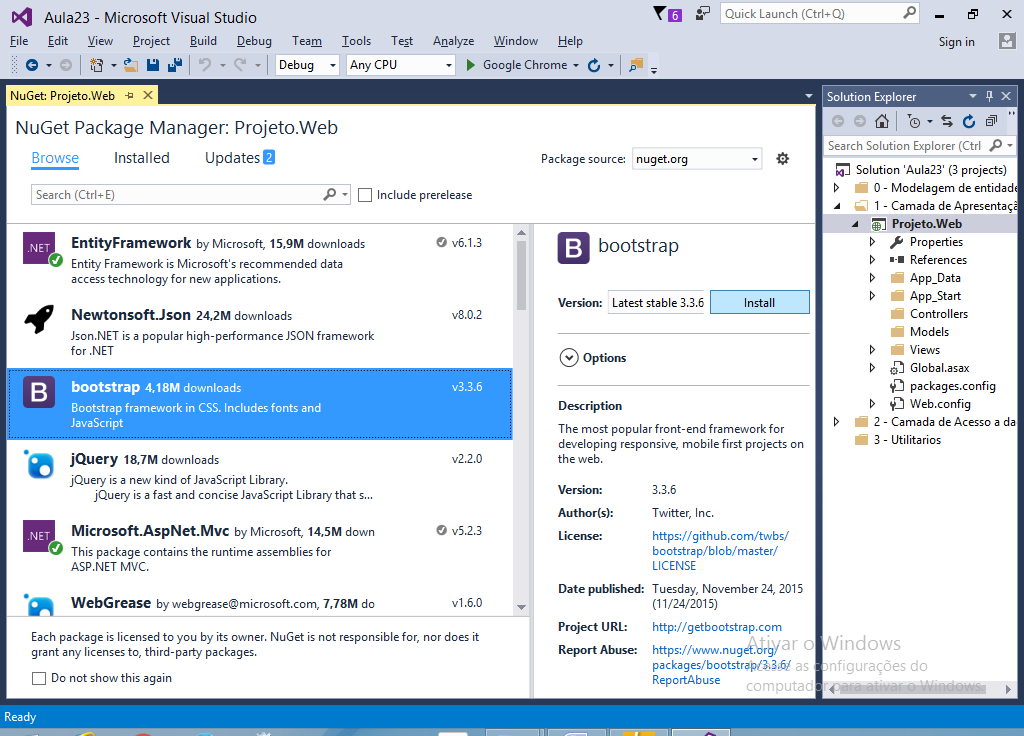
: GenericRepository<Perfil>

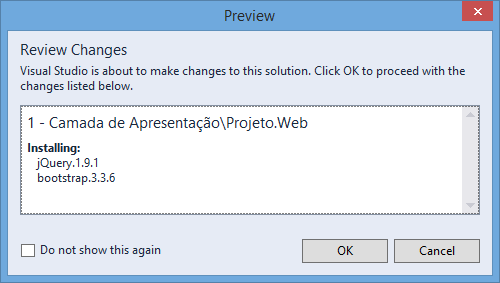
{

}

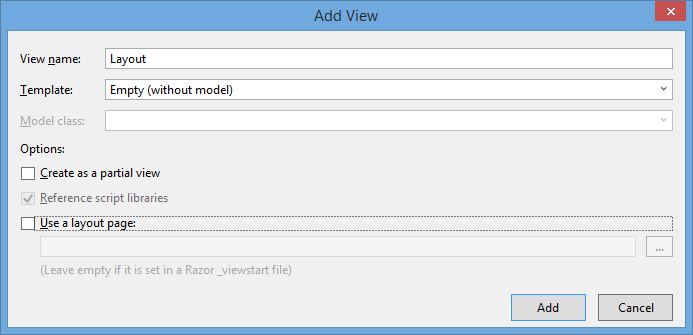
}

**Instalando o bootstrap no projeto web:**





**Criando a página de layout:**



<!DOCTYPE html>

<html>

<head>

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

<title>Projeto</title>

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

<link href="~/Content/bootstrap-theme.css" rel="stylesheet" />

</head>

<body class="container">

<div class="well">

<h3>Projeto Controle de Usuarios</h3>

<ul>

<li> <a href="/Usuario/Login">Login de Usuarios</a> </li>

<li> <a href="/Usuario/Cadastro">Criar Conta de Usuario</a> </li>

</ul>

</div>

<div class="col-md-12">

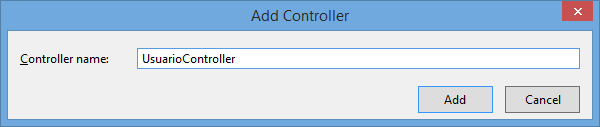
@RenderBody()

</div>

</body>

</html>

Criando o **UsuarioController**



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()

{

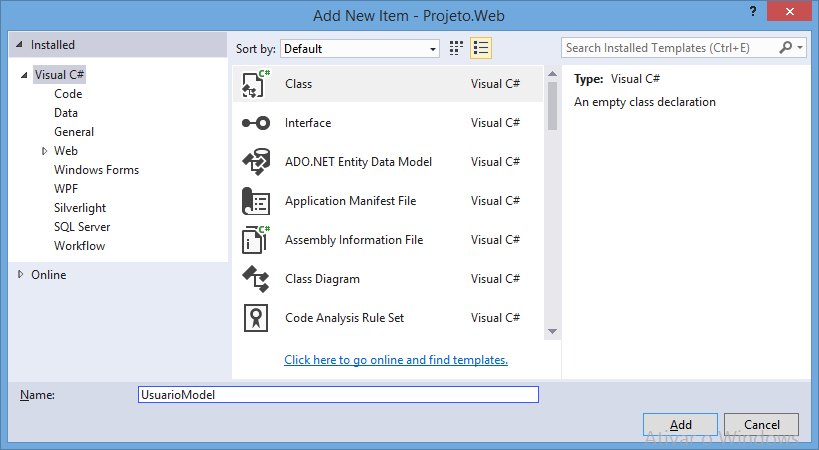
return View();

}

}

}

**Criando as models:**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.ComponentModel.DataAnnotations; //mapeamento..

namespace Projeto.Web.Models

{

public class UsuarioViewModelLogin

{

[Required(ErrorMessage = "Por favor, informe o email de acesso.")]

[Display(Name = "Email de Acesso:")]

public string Email { get; set; }

[DataType(DataType.Password)]

[Required(ErrorMessage = "Por favor, informe a senha de acesso.")]

[Display(Name = "Senha de Acesso:")]

public string Senha { get; set; }

}

public class UsuarioViewModelCadastro

{

[Required(ErrorMessage = "Por favor, infome o nome do usuario.")]

[Display(Name = "Nome do Usuario:")]

public string Nome { get; set; }

[EmailAddress(ErrorMessage = "Erro. Formato de email inválido.")]

[Required(ErrorMessage = "Por favor, infome o email do usuario.")]

[Display(Name = "Endereço de Email:")]

public string Email { get; set; }

[DataType(DataType.Password)]

[Required(ErrorMessage = "Por favor, infome a senha do usuario.")]

[Display(Name = "Senha de Acesso:")]

public string Senha { get; set; }

[DataType(DataType.Password)]

[Compare("Senha", ErrorMessage = "Erro, senhas não conferem.")]

[Required(ErrorMessage = "Por favor, confirme a senha do usuario.")]

[Display(Name = "Confirme a Senha:")]

public string SenhaConfirm { get; set; }

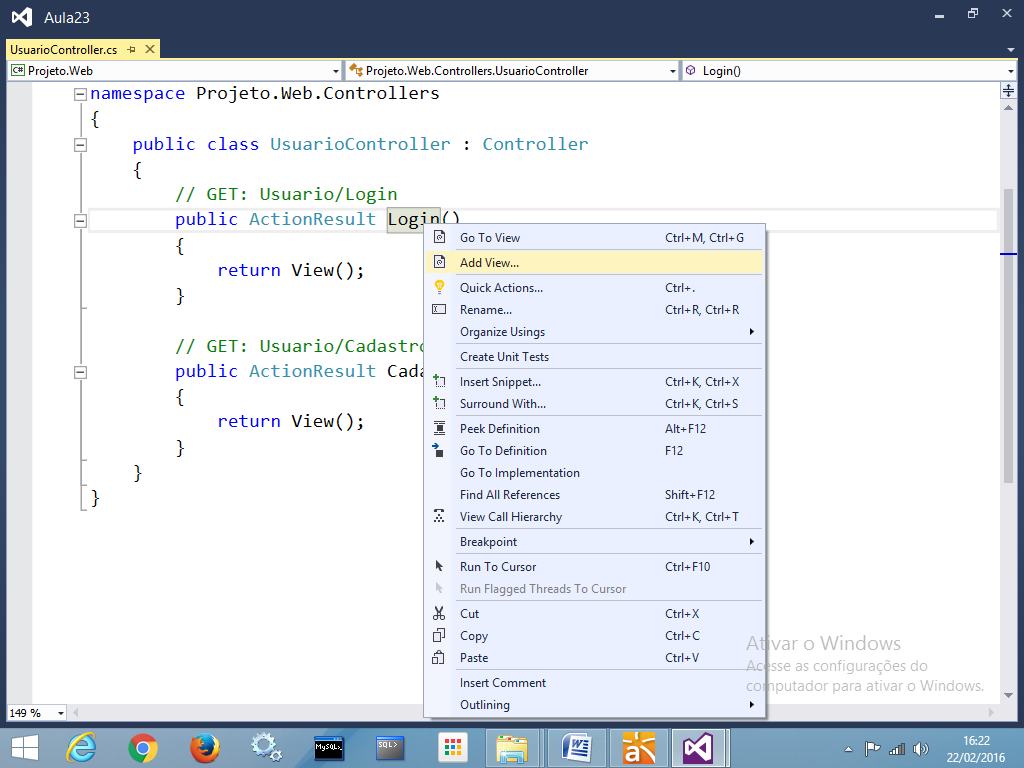
[Required(ErrorMessage = "Por favor, envie a Foto do Usuario.")]

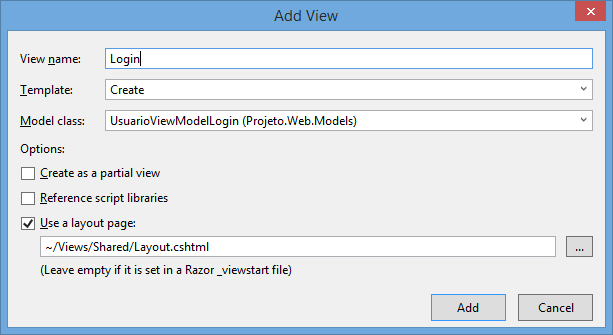
[Display(Name = "Envie sua Foto:")]

public HttpPostedFileBase Foto { get; set; }

}

}





@model Projeto.Web.Models.UsuarioViewModelLogin

@{

ViewBag.Title = "Login";

Layout = "~/Views/Shared/Layout.cshtml";

}

<h4>Login de Usuarios</h4>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.Email, htmlAttributes:

new { @class = "control-label col-md-2" })

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

@Html.EditorFor(model => model.Email, new { htmlAttributes =

new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Email, "",

new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Senha, htmlAttributes:

new { @class = "control-label col-md-2" })

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

@Html.EditorFor(model => model.Senha, new { htmlAttributes =

new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Senha, "",

new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-4">

<input type="submit" value="Acessar Sistema"

class="btn btn-default" />

</div>

</div>

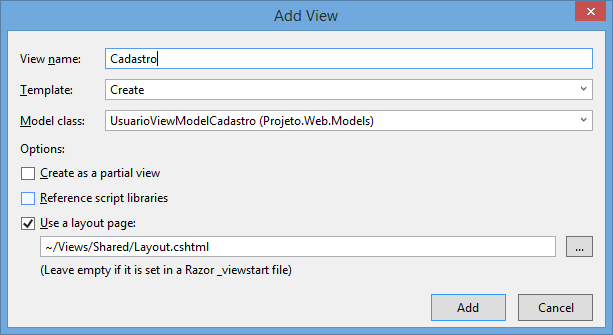
</div>

}

<div>

@ViewBag.Mensagem

</div>



@model Projeto.Web.Models.UsuarioViewModelCadastro

@{

ViewBag.Title = "Cadastro";

Layout = "~/Views/Shared/Layout.cshtml";

}

<h4>Cadastro de Usuarios</h4>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.Nome, htmlAttributes:

new { @class = "control-label col-md-2" })

<div class="col-md-5">

@Html.EditorFor(model => model.Nome, new { htmlAttributes =

new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Nome, "",

new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Email, htmlAttributes:

new { @class = "control-label col-md-2" })

<div class="col-md-5">

@Html.EditorFor(model => model.Email, new { htmlAttributes =

new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Email, "",

new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Senha, htmlAttributes:

new { @class = "control-label col-md-2" })

<div class="col-md-5">

@Html.EditorFor(model => model.Senha, new { htmlAttributes =

new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Senha, "",

new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.SenhaConfirm, htmlAttributes:

new { @class = "control-label col-md-2" })

<div class="col-md-5">

@Html.EditorFor(model => model.SenhaConfirm,

new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.SenhaConfirm, "",

new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Foto, htmlAttributes:

new { @class = "control-label col-md-2" })

<div class="col-md-5">

<input type="file" name="Foto" class="form-control" />

@Html.ValidationMessageFor(model => model.Foto, "",

new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Cadastrar Usuario"

class="btn btn-default" />

</div>

</div>

</div>

}

<div>

@ViewBag.Mensagem

</div>

**Definindo a página inicial:**

App\_Start/RouteConfig.cs

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 RouteConfig

{

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { **controller = "Usuario",**

**action = "Login",**

id = UrlParameter.Optional }

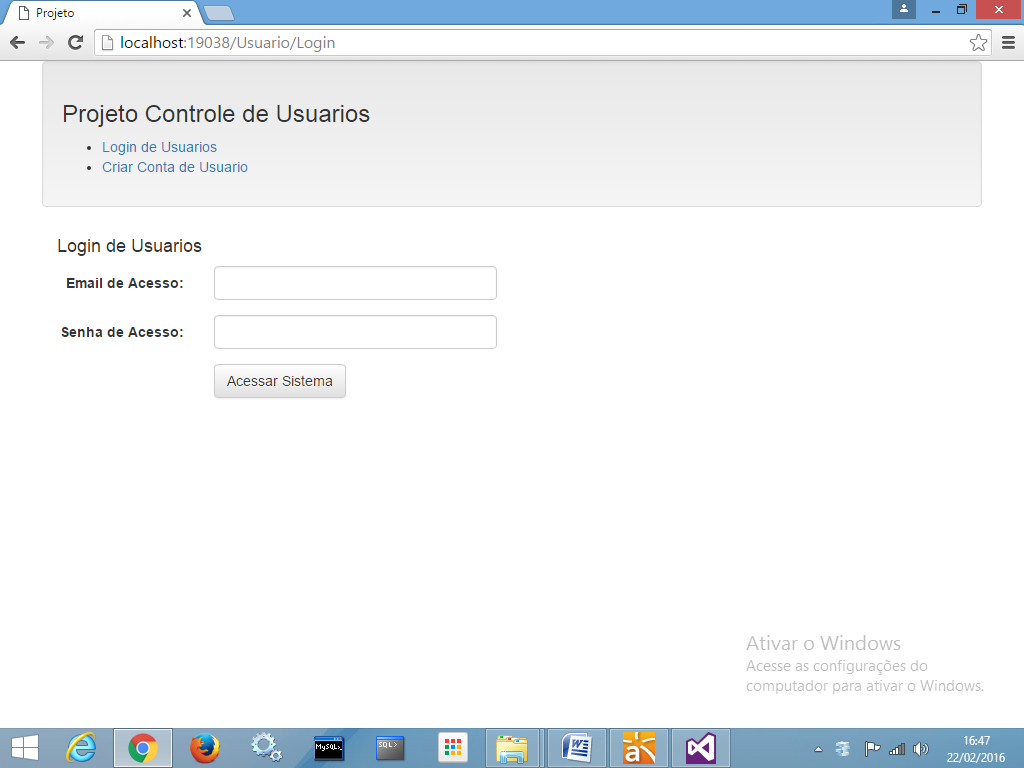
);

}

}

}

http://localhost:19038/Usuario/Login



http://localhost:19038/Usuario/Cadastro



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

**Criando no UsuarioController os métodos HTTP POST**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Projeto.Web.Models; //camada de modelo..

namespace Projeto.Web.Controllers

{

public class UsuarioController : Controller

{

// GET: Usuario/Login

public ActionResult Login()

{

return View();

}

// GET: Usuario/Cadastro

public ActionResult Cadastro()

{

return View();

}

// POST: Usuario/Login

[HttpPost]

public ActionResult Login(UsuarioViewModelLogin model)

{

return View();

}

// POST: Usuario/Cadastro

[HttpPost]

public ActionResult Cadastro(UsuarioViewModelCadastro model)

{

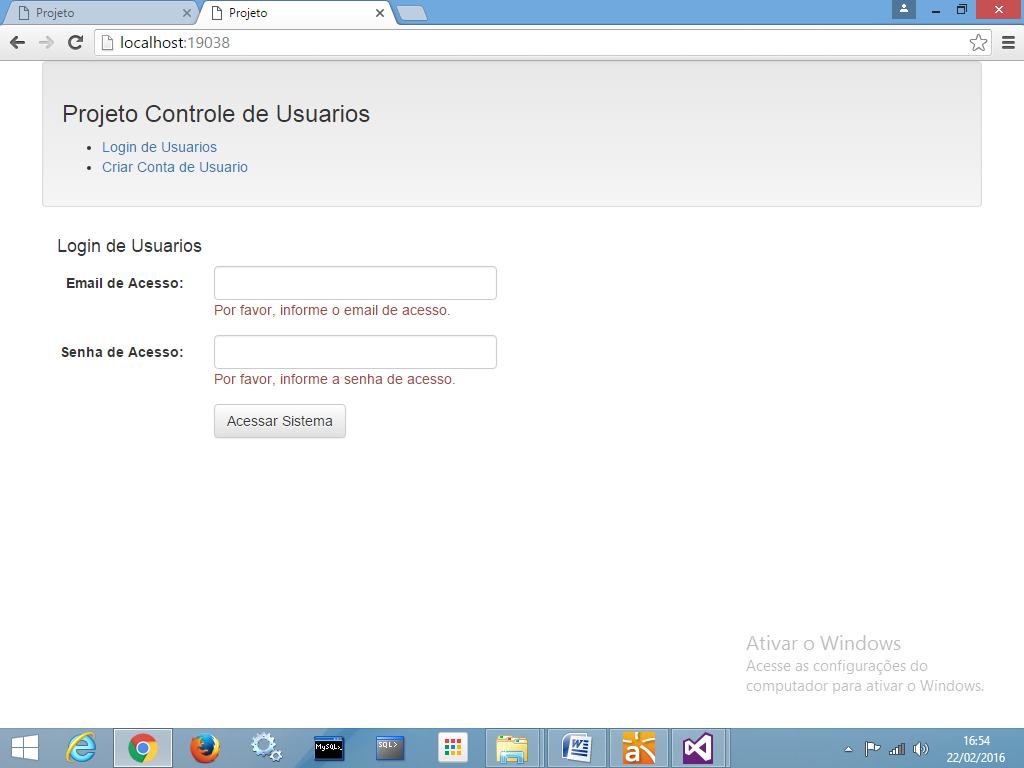
return View();

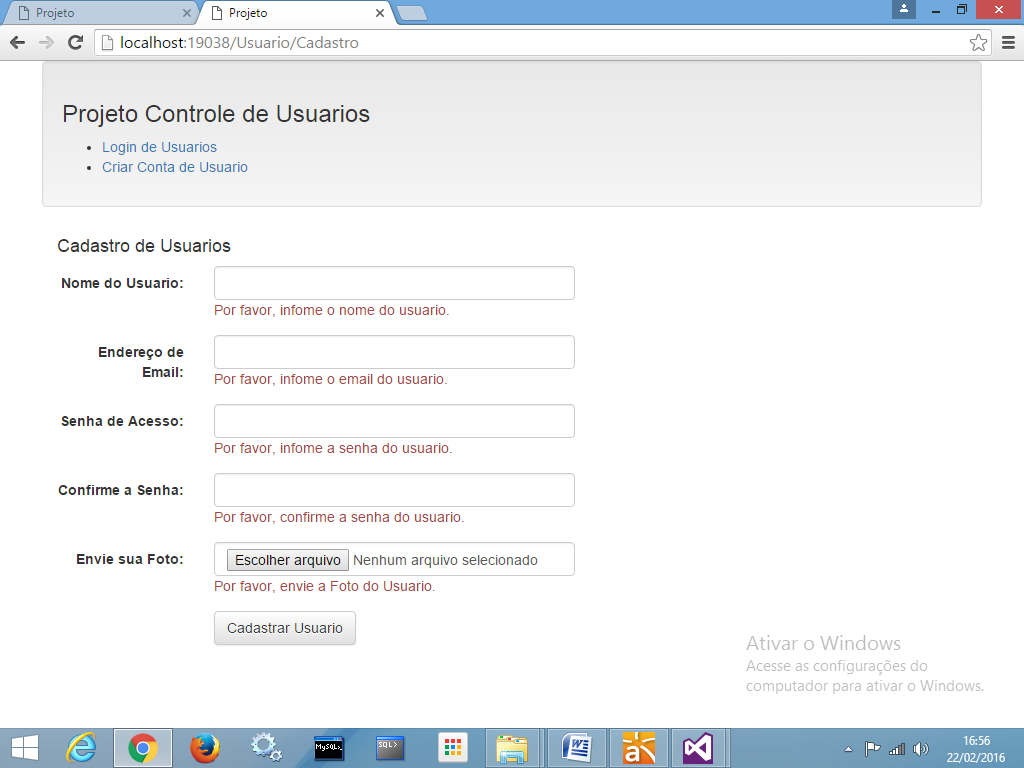
}

}

}

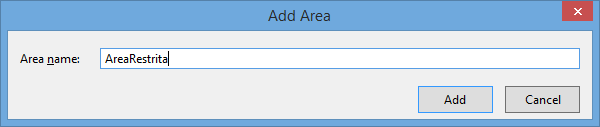
**Executando:**



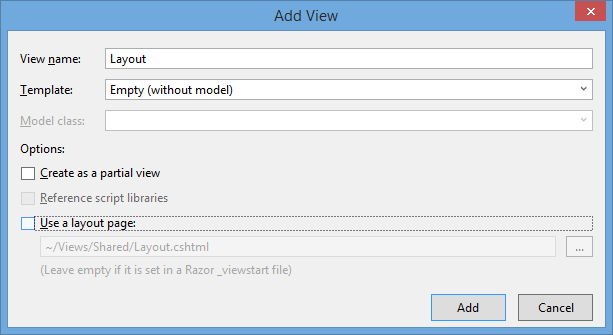


Criando a Área Restrita do projeto:

/Areas/AreaRestrita



**Layout da área restrita**



<!DOCTYPE html>

<html>

<head>

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

<title>Projeto</title>

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

<link href="~/Content/bootstrap-theme.css" rel="stylesheet" />

</head>

<body>

<div class="container">

<h2>Área Restrita do Sistema</h2>

Seja Bem vindo.

<hr/>

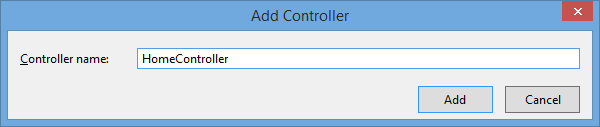
**@RenderBody()**

</div>

</body>

</html>

**Criando o controller principal da área restrita:**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace Projeto.Web.Areas.AreaRestrita.Controllers

{

public class HomeController : Controller

{

// GET: AreaRestrita/Home

public ActionResult Index()

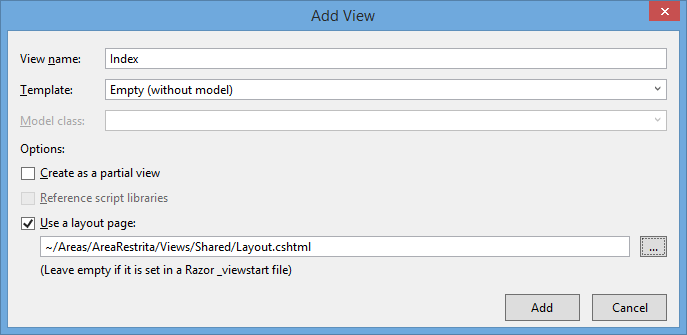
{

return View();

}

}

}



@{

Layout = "~/Areas/AreaRestrita/Views/Shared/Layout.cshtml";

}

<h4>Página inicial da área restrita</h4>

http://localhost:19038/AreaRestrita/Home/Index



[Authorize]

Define que um controller MVC só pode ser acessado   
por 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 HomeController : Controller

{

// GET: AreaRestrita/Home

public ActionResult Index()

{

return View();

}

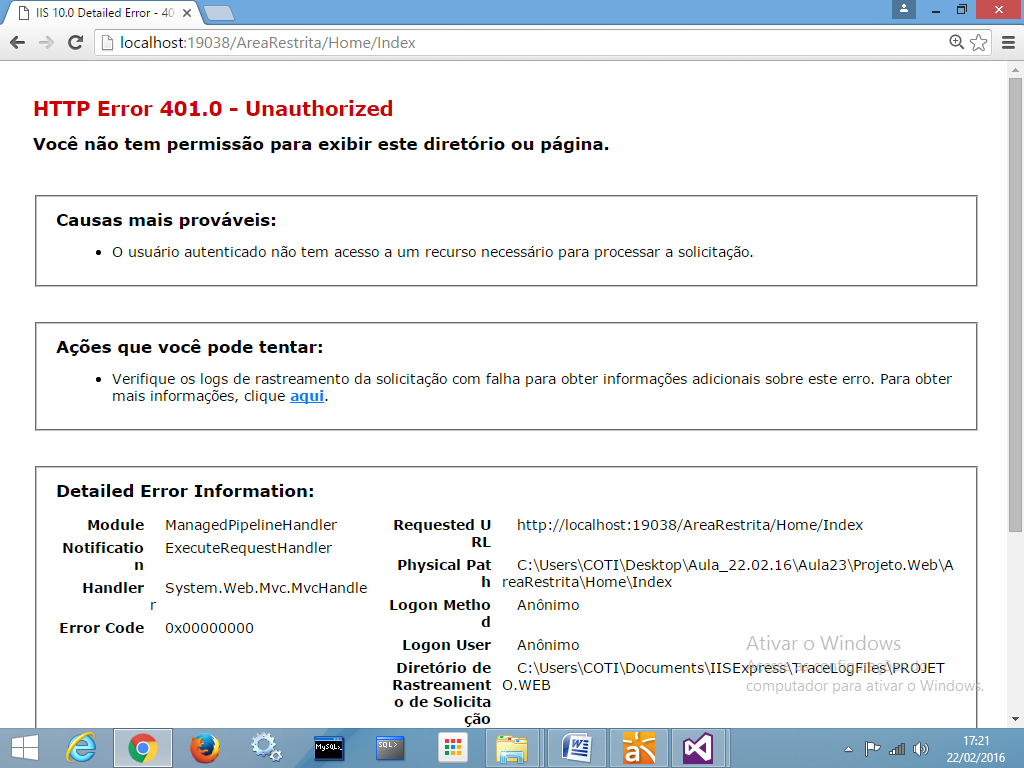
}

}

**Resultado:**

Erro de servidor: Acesso não autorizado

http://localhost:19038/AreaRestrita/Home/Index



Redicionar o Erro 401 para a   
página de login de usuarios:

**Web.config.xml**

<system.web>

<compilation debug="true" targetFramework="4.5.2" />

<httpRuntime targetFramework="4.5.2" />

**<authentication mode="Forms">**

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

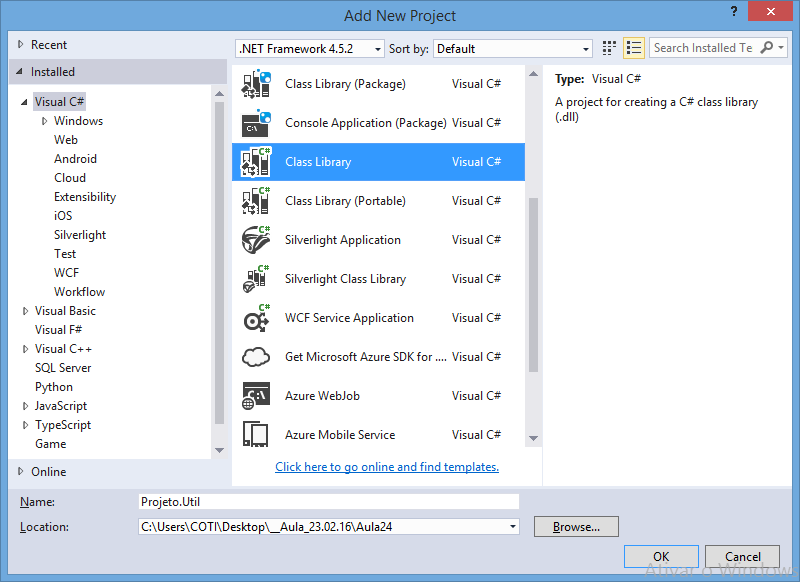
**</authentication>**

</system.web>

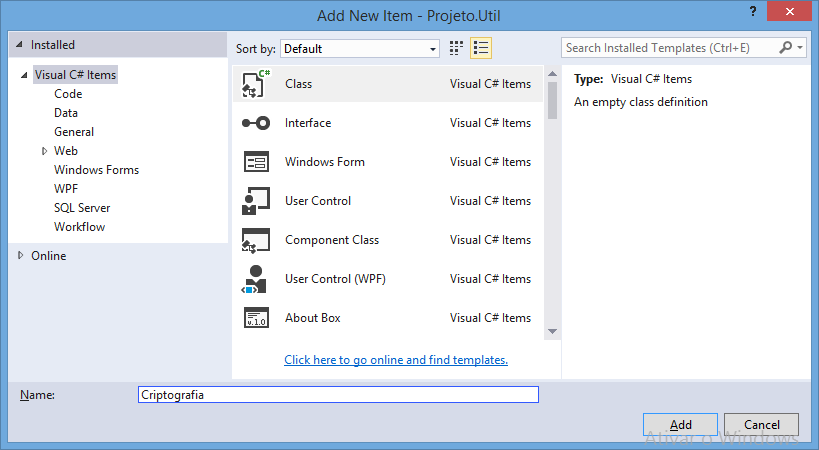
Camada de Utilitarios:

Projeto para prover recursos auxiliares ao sistema,   
tais como Criptografia, Envio de Emails, etc...

Criando a **Class Library**:



**Classe para gerar Criptografia em padrão MD5 e SHA1**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Security.Cryptography;

namespace Projeto.Util.Seguranca

{

public class Criptografia

{

//método estático para encriptar um valor para MD5

public static string EncriptarMD5(string valor)

{

MD5 md5 = new MD5CryptoServiceProvider();

byte[] hash = md5.ComputeHash(Encoding.UTF8.GetBytes(valor));

return BitConverter.ToString(hash).Replace("-", string.Empty);

}

//método estático para encriptar um valor para SHA1

public static string EncriptarSHA1(string valor)

{

SHA1 sha1 = new SHA1CryptoServiceProvider();

byte[] hash = sha1.ComputeHash(Encoding.UTF8.GetBytes(valor));

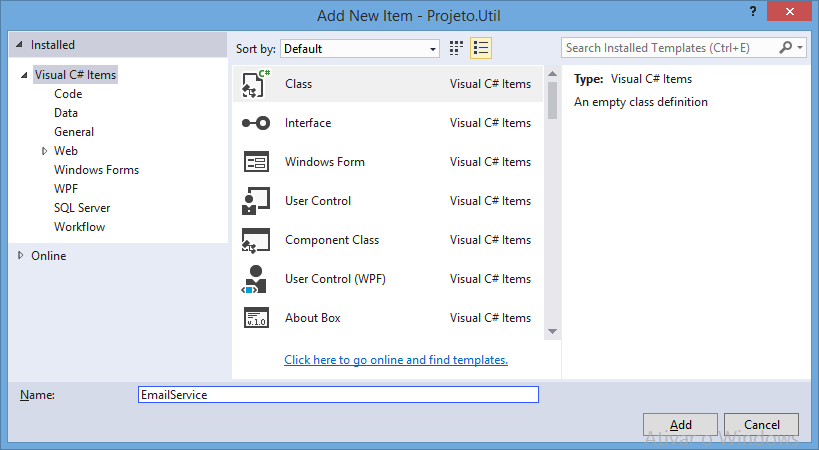
return BitConverter.ToString(hash).Replace("-", string.Empty);

}

}

}

**Classe para envio de email:**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Net; //envio de mensagem

using System.Net.Mail; //envio de mensagem

namespace Projeto.Util.Mensagens

{

public class EmailService

{

//propriedades..

public string Destinatario { get; set; }

public string Assunto { get; set; }

public string Mensagem { get; set; }

//constantes..

private readonly string Conta = "cotiexemplo@gmail.com";

private readonly string Senha = "@coticoti@";

//método para enviar email..

public void EnviarMensagem()

{

try

{

//Passo 1) Criar a mensagem...

MailMessage msg = new MailMessage(Conta, Destinatario);

msg.Subject = Assunto;

msg.Body = Mensagem;

//Passo 2) Enviar o email..

SmtpClient smtp = new SmtpClient("smtp.gmail.com", 587);

smtp.EnableSsl = true; //SSL -> Security Socket Layer (segurança)

smtp.Credentials = new NetworkCredential(Conta, Senha);

//enviando..

smtp.Send(msg);

}

catch(Exception e)

{

throw new Exception("Erro ao enviar email: " + e.Message);

}

}

}

}

**Implementando o cadastro do Usuario:**

Página de cadastro de usuarios

Habilitando o envio de arquivos pelo campo de Upload:

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

**new { enctype = "multipart/form-data" }))**

**No controller:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Projeto.Web.Models; //camada de modelo

using Projeto.Entities;

using Projeto.Repository.Persistence;

using Projeto.Util.Mensagens;

using Projeto.Util.Seguranca;

namespace Projeto.Web.Controllers

{

public class UsuarioController : Controller

{

// GET: Usuario/Login

public ActionResult Login()

{

return View();

}

// GET: Usuario/Cadastro

public ActionResult Cadastro()

{

return View();

}

// GET: Usuario/Login

[HttpPost]

public ActionResult Login(UsuarioViewModelLogin model)

{

return View();

}

**// GET: Usuario/Cadastro**

**[HttpPost]**

**public ActionResult Cadastro(UsuarioViewModelCadastro model)**

**{**

**//se a model passou nas validações..**

**if(ModelState.IsValid)**

**{**

**try**

**{**

**}**

**catch(Exception e)**

**{**

**//imprimindo mensagem de erro..**

**ViewBag.Mensagem = e.Message;**

**}**

**}**

**return View();**

**}**

}

}

**Criando um método na camada de dados para verificar se   
um Email ja esta cadastrado para algum usuario:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto.Repository.DataSource; //conexao

using Projeto.Repository.Generics;

using Projeto.Entities;

using System.Data.Entity;

namespace Projeto.Repository.Persistence

{

public class UsuarioRepository : GenericRepository<Usuario>

{

//sobrescrever os métodos..

public override void Insert(Usuario obj)

{

using (Conexao Con = new Conexao())

{

if(Con.Usuario

.Where(u => u.Email.Equals(obj.Email))

.Count() == 0) //email não existe..

{

Con.Entry(obj).State = EntityState.Added;

Con.SaveChanges();

}

else

{

throw new Exception("Erro. Este email ja encontra-se

cadastrado. Tente outro.");

}

}

}

public override List<Usuario> FindAll()

{

using (Conexao Con = new Conexao())

{

return Con.Usuario

.Include(u => u.Perfil) //JOIN..

.ToList();

}

}

public override Usuario FindById(int id)

{

using (Conexao Con = new Conexao())

{

return Con.Usuario

.Include(u => u.Perfil) //JOIN

.Where(u => u.IdUsuario == id)

.FirstOrDefault();

}

}

//método utilizado para buscar o usuario na autenticação..

public Usuario FindByEmailSenha(string Email, string Senha)

{

using (Conexao Con = new Conexao())

{

return Con.Usuario

.Include(u => u.Perfil) //JOIN..

.Where(u => u.Email.Equals(Email)

&& u.Senha.Equals(Senha))

.FirstOrDefault();

}

}

}

}

**Implementando o cadastro   
de Usuario no Controller:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Projeto.Web.Models; //camada de modelo

using Projeto.Entities;

using Projeto.Repository.Persistence;

using Projeto.Util.Mensagens;

using Projeto.Util.Seguranca;

using System.IO;

namespace Projeto.Web.Controllers

{

public class UsuarioController : Controller

{

// GET: Usuario/Login

public ActionResult Login()

{

return View();

}

// GET: Usuario/Cadastro

public ActionResult Cadastro()

{

return View();

}

// GET: Usuario/Login

[HttpPost]

public ActionResult Login(UsuarioViewModelLogin model)

{

return View();

}

// GET: Usuario/Cadastro

[HttpPost]

public ActionResult Cadastro(UsuarioViewModelCadastro model)

{

//se a model passou nas validações..

if(ModelState.IsValid)

{

try

{

//validar o arquivo enviada..

if(model.Foto.ContentType.Equals("image/jpeg")

&& model.Foto.ContentLength <= Math.Pow(1024, 2))

{

Usuario u = new Usuario(); //entidade..

u.Nome = model.Nome;

u.Email = model.Email;

u.Senha = Criptografia.EncriptarMD5(model.Senha);

u.DataCadastro = DateTime.Now;

u.Foto = Guid.NewGuid().ToString()

//gerando um GUID para o nome da foto

+ "." + Path.GetExtension(model.Foto.FileName);

//pegando a extensão..

//perfil padrão para o usuario..

u.IdPerfil = 1; //Default..

UsuarioRepository rep = new UsuarioRepository();

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

//upload da foto..

model.Foto.SaveAs(HttpContext.Server

.MapPath("/Imagens/") + u.Foto);

//envio do email..

EmailService s = new EmailService();

s.Destinatario = u.Email;

s.Assunto = "Criação de Conta de Usuario

- Aula de C# WebDeveloper";

s.Mensagem = "Seja bem vindo: " + u.Nome

+ ", sua conta foi criada com sucesso";

s.EnviarMensagem(); //enviando email..

ViewBag.Mensagem = "Usuario " + u.Nome

+ ", cadastrado com sucesso.";

ModelState.Clear(); //limpar a model..

}

else

{

throw new Exception("Erro. Envie apenas imagens

JPG de até 1MB.");

}

}

catch(Exception e)

{

//imprimindo mensagem de erro..

ViewBag.Mensagem = e.Message;

}

}

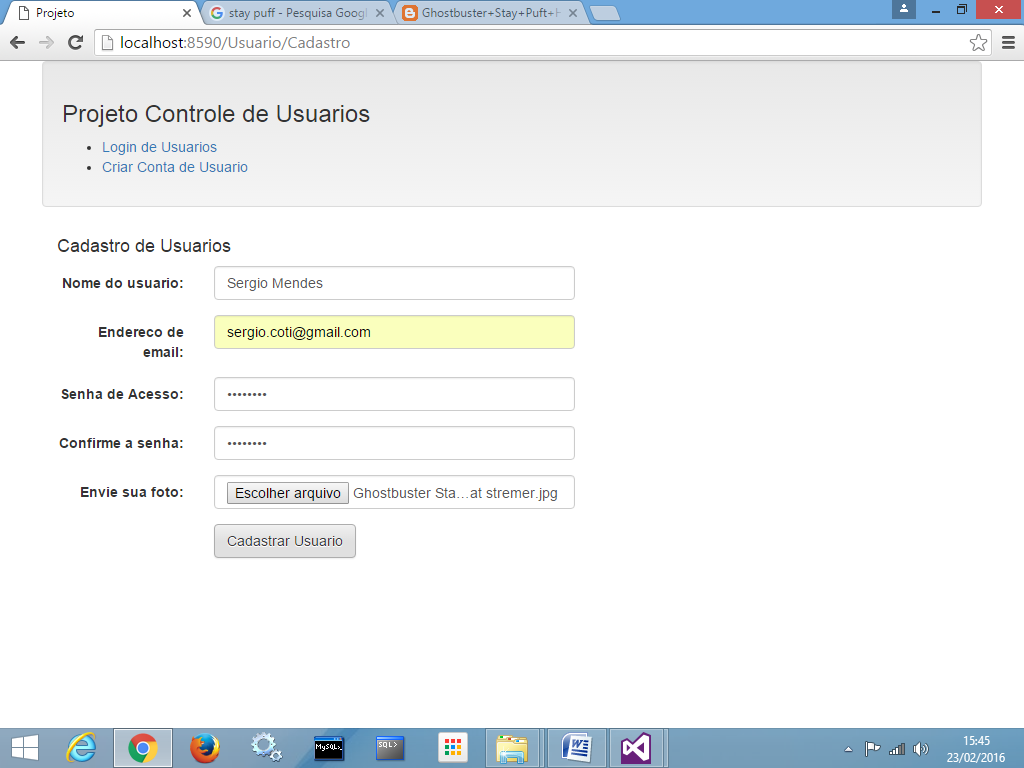
return View();

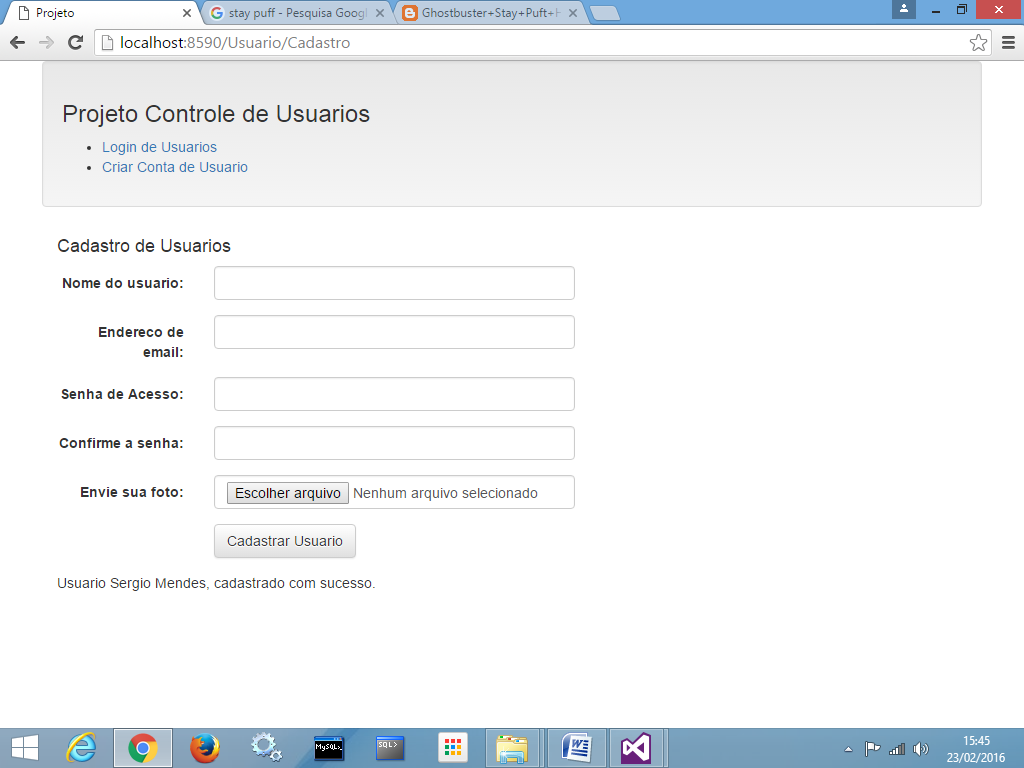
}

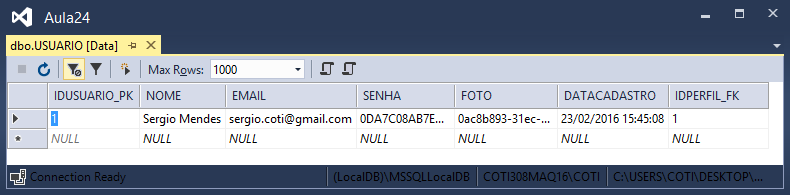
}

}

**Executando:**







Autenticar o Usuario:

UsuarioController

FormsAuthenticationTicket

Cria um ticket de autenticação para o usuario.

* **Criar uma classe de modelo para definir os dados   
  que serão gravados no ticket:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.ComponentModel.DataAnnotations; //mapeamento

namespace Projeto.Web.Models

{

public class UsuarioViewModelLogin

{

[Display(Name ="Email de Acesso:")]

[Required(ErrorMessage ="Por favor, informe o Email de acesso.")]

public string Email { get; set; }

[Display(Name ="Senha de Acesso:")]

[Required(ErrorMessage = "Por favor, informe a senha de acesso.")]

[DataType(DataType.Password)]

public string Senha { get; set; }

}

public class UsuarioViewModelCadastro

{

[Display(Name ="Nome do usuario:")]

[Required(ErrorMessage ="por favor, informe o nome do usuario")]

public string Nome { get; set; }

[EmailAddress(ErrorMessage ="Erro. formato de email invalido.")]

[Display(Name = "Endereco de email:")]

[Required(ErrorMessage = "por favor, informe o email do usuario")]

public string Email { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Senha de Acesso:")]

[Required(ErrorMessage = "por favor, informe a senha do usuario")]

public string Senha { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Confirme a senha:")]

[Required(ErrorMessage = "por favor, confirme a senha do usuario")]

[Compare("Senha", ErrorMessage ="Erro, senhas nao conferem.")]

public string SenhaConfirm { get; set; }

[Display(Name = "Envie sua foto:")]

[Required(ErrorMessage = "por favor, envie a foto do usuario")]

public HttpPostedFileBase Foto { get; set; }

}

**public class UsuarioAutenticadoModel**

**{**

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

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

**public string Email { get; set; }**

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

**public DateTime DataCadastro { get; set; }**

**public int IdPerfil { get; set; }**

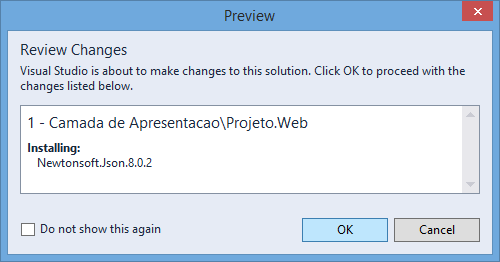
**public string NomePerfil { get; set; }**

**}**

}

JSON.NET

Biblioteca do .NET voltada para serialização e deserialização   
de objetos (classes) para json.



**FormsAuthentication.FormsCookieName**

Gera o nome do cookie que recebera o ticket de acesso

**FormsAuthentication.Encrypt(ticket)**

Grava o ticket de acesso no cookie de forma criptografada

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Projeto.Web.Models; //camada de modelo

using Projeto.Entities;

using Projeto.Repository.Persistence;

using Projeto.Util.Mensagens;

using Projeto.Util.Seguranca;

using System.IO;

using System.Web.Security; //segurança..

using Newtonsoft.Json;

namespace Projeto.Web.Controllers

{

public class UsuarioController : Controller

{

// GET: Usuario/Login

public ActionResult Login()

{

return View();

}

// GET: Usuario/Cadastro

public ActionResult Cadastro()

{

return View();

}

**// GET: Usuario/Login**

**[HttpPost]**

**public ActionResult Login(UsuarioViewModelLogin model)**

**{**

**if (ModelState.IsValid)**

**{**

**try**

**{**

**UsuarioRepository rep = new UsuarioRepository();**

**//buscar o usuario pelo Email e Senha na base de dados..**

**Usuario u = rep.FindByEmailSenha**

**(model.Email, Criptografia.EncriptarMD5(model.Senha));**

**if(u != null)**

**{**

**//gerar o objeto model que será gravado no ticket**

**UsuarioAutenticadoModel auth =**

**new UsuarioAutenticadoModel();**

**auth.IdUsuario = u.IdUsuario;**

**auth.Nome = u.Nome;**

**auth.Email = u.Email;**

**auth.Foto = u.Foto;**

**auth.DataCadastro = u.DataCadastro;**

**auth.IdPerfil = u.Perfil.IdPerfil;**

**auth.NomePerfil = u.Perfil.Nome;**

**//criando o ticket de autenticação..**

**FormsAuthenticationTicket ticket =**

**new FormsAuthenticationTicket**

**(JsonConvert.SerializeObject(auth),**

**false, 5);**

**//gravar o ticket em cookie..**

**HttpCookie cookie = new HttpCookie**

**(FormsAuthentication.FormsCookieName,**

**FormsAuthentication.Encrypt(ticket));**

**//cookie.Expires = DateTime.Now.AddYears(10); NAO FAZER!!**

**Response.Cookies.Add(cookie);**

**//gravar o cookie no navegador..**

**//redirecionar para a área restrita..**

**return RedirectToAction("Index", "Home",**

**new { area = "AreaRestrita" });**

**}**

**else**

**{**

**throw new Exception("Acesso Negado. Tente novamente.");**

**}**

**}**

**catch(Exception e)**

**{**

**//imprimir mensagem de erro..**

**ViewBag.Mensagem = e.Message;**

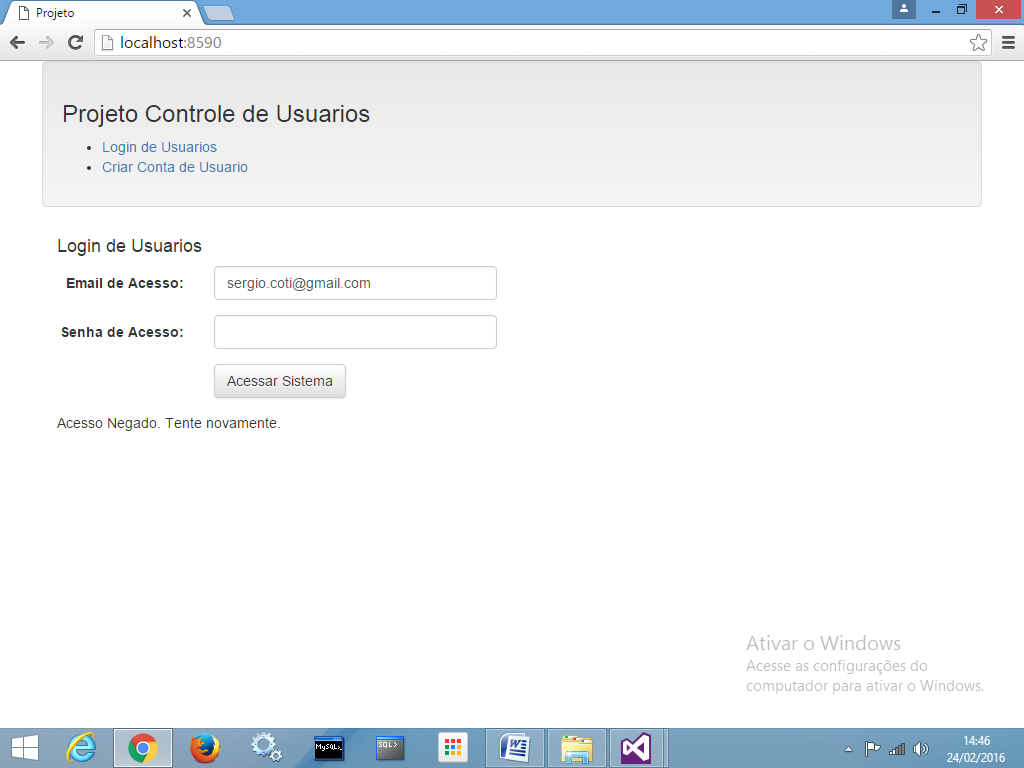
**}**

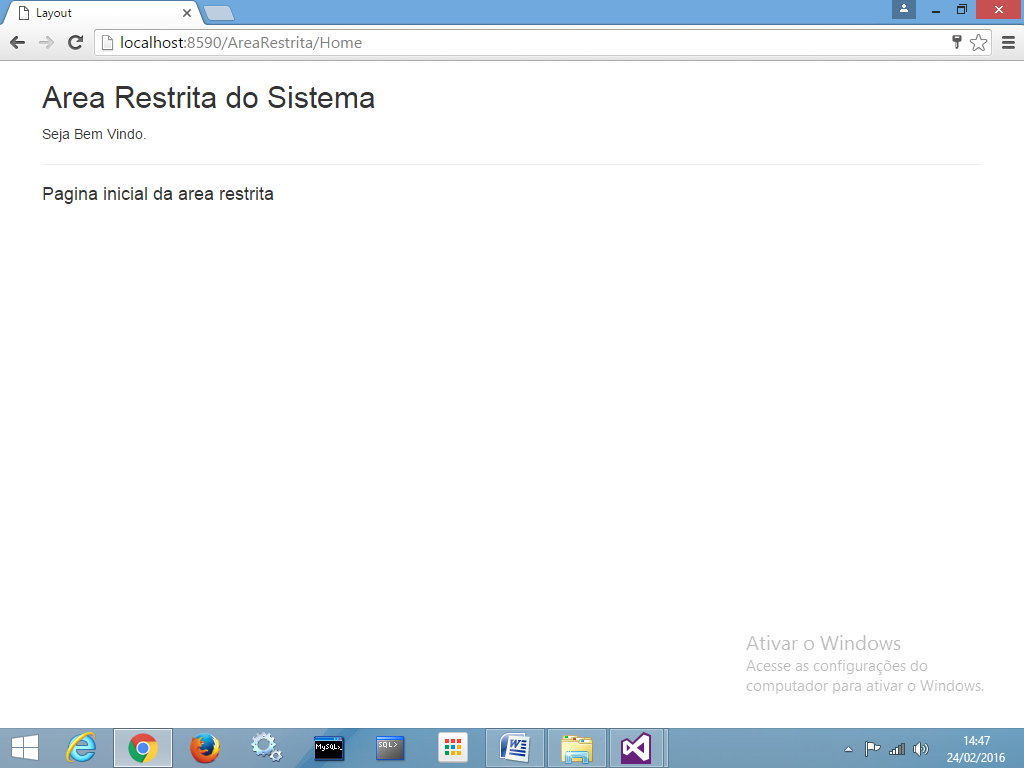
**}**

**return View();**

**}**

**Testando:**





Logout do Usuario:

Página de Layout da área restrita

<!DOCTYPE html>

<html>

<head>

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

<title>Layout</title>

<link href="~/Content/bootstrap-theme.css" rel="stylesheet" />

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

</head>

<body>

<div class="container">

<h2>Area Restrita do Sistema</h2>

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

<hr />

@RenderBody()

</div>

</body>

</html>

* **UsuarioController.cs**

//método para fazer o logout do usuario..

**public ActionResult Logout()**

{

//destruir o ticket de acesso..

FormsAuthentication.SignOut();

//voltar para a página de login..

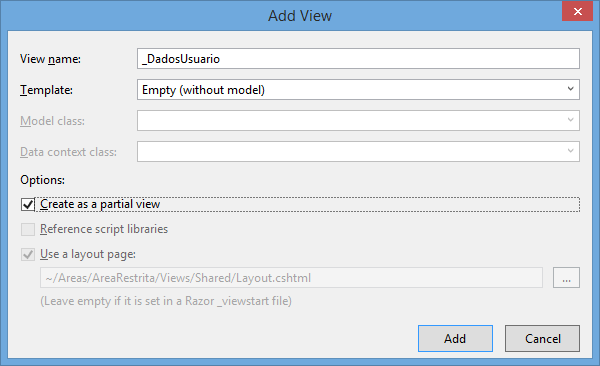
return RedirectToAction("Login", "Usuario");

}

**Exibindo os dados do Usuario Autenticado:**

PartialView

Páginas cshtml que podem ser incluidas dentro de outras páginas.



<div id="janela" class="modal fade">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header bg-primary">

Usuário Autenticado

</div>

<div class="modal-body text-center">

<!-- JSON com os dados do usuario autenticado -->

@HttpContext.Current.User.Identity.Name

</div>

<div class="modal-footer">

<button class="btn btn-default btn-block"

data-dissmiss="modal">

Fechar Janela

</button>

</div>

</div>

</div>

</div>

**Incluindo a partialview na página de layout:**

<!DOCTYPE html>

<html>

<head>

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

<title>Layout</title>

<link href="~/Content/bootstrap-theme.css" rel="stylesheet" />

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

<script src="~/Scripts/jquery-1.9.1.js"></script>

<script src="~/Scripts/bootstrap.js"></script>

</head>

<body>

<div class="container">

<h2>Area Restrita do Sistema</h2>

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

<a href="#" data-target="#janela" data-toggle="modal">

Dados do Usuario

</a>

**@Html.Partial("\_DadosUsuario")**

<hr />

@RenderBody()

</div>

</body>

</html>

**Exibindo os dados do usuario:**

<div id="janela" class="modal fade">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header bg-primary">

Usuário Autenticado

</div>

<div class="modal-body text-center">

@using Newtonsoft.Json

@using Projeto.Web.Models

@{

//transformar os dados de JSON para objeto..

var usuario = JsonConvert.DeserializeObject

<UsuarioAutenticadoModel>

(HttpContext.Current.User.Identity.Name);

}

<img src="~/Imagens/@usuario.Foto" width="140" height="160"

class="img-circle"/>

<h3> @usuario.Nome </h3>

@usuario.Email

<br /><br />

<strong>Perfil: @usuario.NomePerfil</strong>

<br/>

Cadastrado em: @usuario.DataCadastro

</div>

<div class="modal-footer">

<button class="btn btn-default btn-block"

data-dismiss="modal">

Fechar Janela

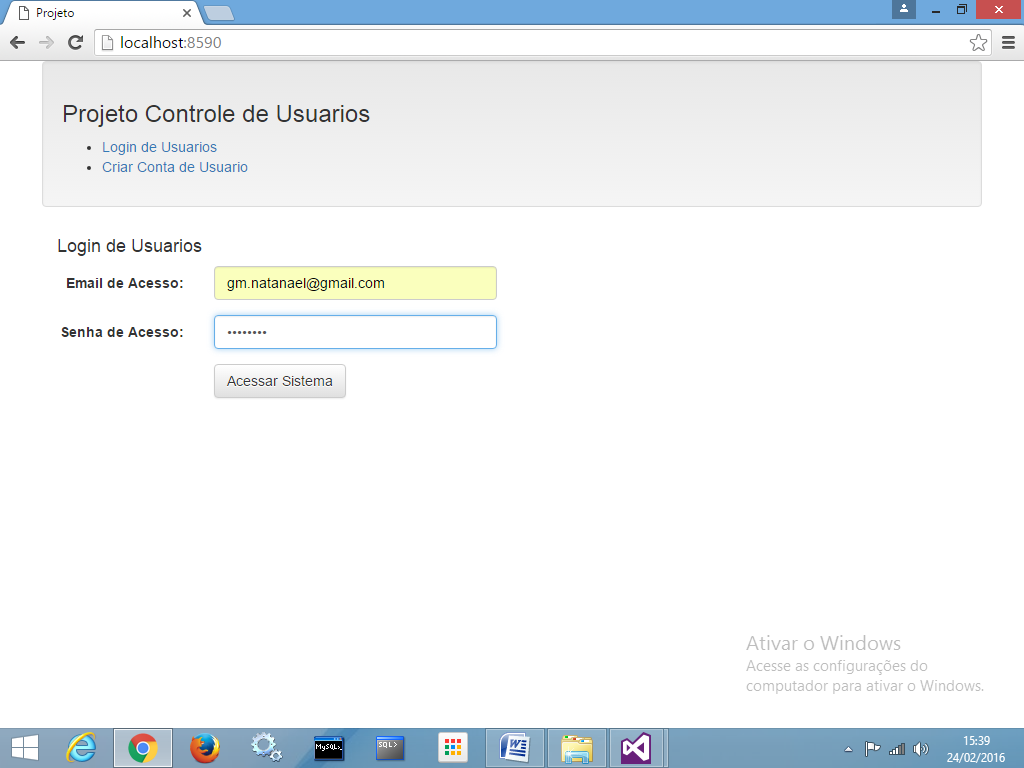
</button>

</div>

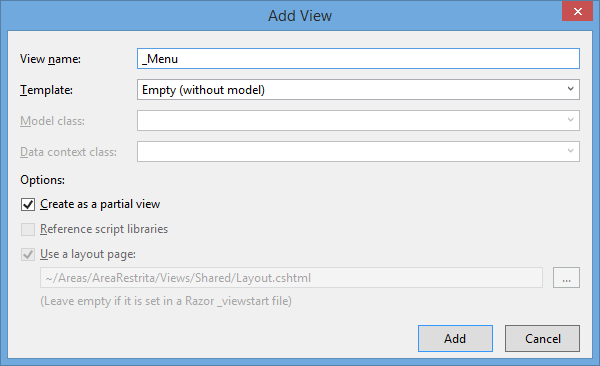
</div>

</div>

</div>



**PartialView para Menu de opções:**



**@using Newtonsoft.Json**

**@using Projeto.Web.Models**

**@{**

**//transformar os dados de JSON para objeto..**

**var usuario = JsonConvert.DeserializeObject<UsuarioAutenticadoModel>**

**(HttpContext.Current.User.Identity.Name);**

**}**

<h4>Menu de opções</h4>

@if(usuario.NomePerfil.Equals("Administrador"))

{

<ul>

<li> <a href="/AreaRestrita/Admin/Cadastro">Cadastrar Usuarios</a> </li>

<li> <a href="/AreaRestrita/Admin/Consulta">Consultar Usuarios</a> </li>

</ul>

}

else if(usuario.NomePerfil.Equals("Funcionario")

|| usuario.NomePerfil.Equals("Administrador"))

{

<ul>

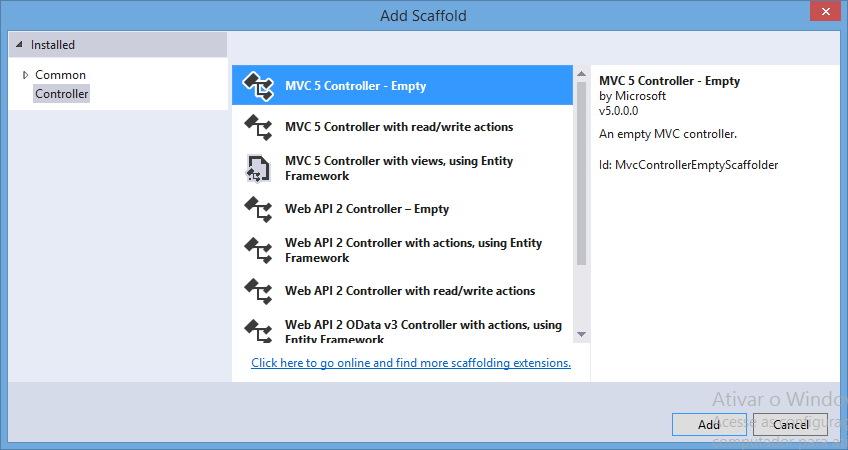
<li> <a href="/AreaRestrita/Funcionario/Cadastro">Cadastrar Vendas</a> </li>

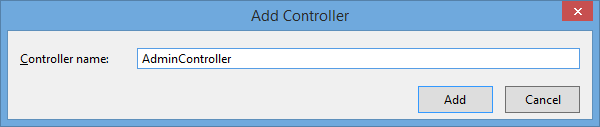
<li> <a href="/AreaRestrita/Funcionario/Consulta">Consultar Vendas</a> </li>

</ul>

}

AdminController





using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace Projeto.Web.Areas.AreaRestrita.Controllers

{

[Authorize(Roles = "Administrador")]

public class AdminController : Controller

{

// GET: AreaRestrita/Admin/Cadastro

public ActionResult Cadastro()

{

return View();

}

// GET: AreaRestrita/Admin/Consulta

public ActionResult Consulta()

{

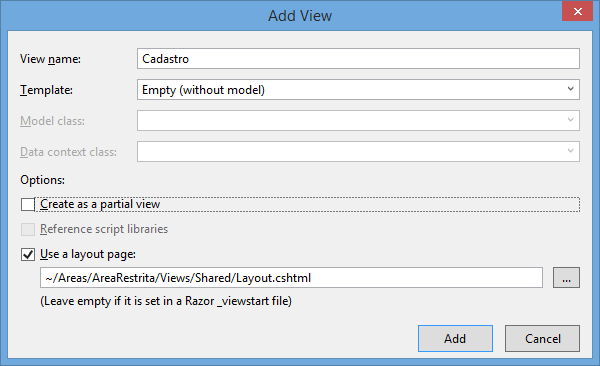
return View();

}

}

}

**Gerando as páginas do Administrador:**



@{

Layout = "~/Areas/AreaRestrita/Views/Shared/Layout.cshtml";

}

<h4>Cadastro de Usuarios</h4>

Página de Acesso do Administrador.

@{

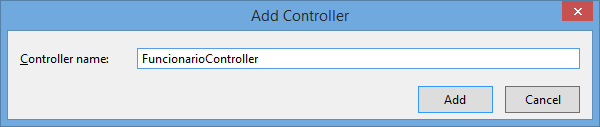
Layout = "~/Areas/AreaRestrita/Views/Shared/Layout.cshtml";

}

<h4>Consulta de Usuarios</h4>

Página de acesso do Administrador.

**Criando o Controller do Funcionario:**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace Projeto.Web.Areas.AreaRestrita.Controllers

{

[Authorize(Roles = "Funcionario, Administrador")]

public class FuncionarioController : Controller

{

// GET: AreaRestrita/Funcionario/Cadastro

public ActionResult Cadastro()

{

return View();

}

// GET: AreaRestrita/Funcionario/Consulta

public ActionResult Consulta()

{

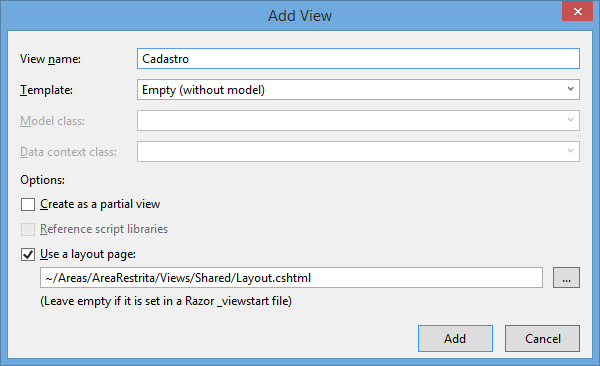
return View();

}

}

}

**Gerando as páginas do Funcionario:**



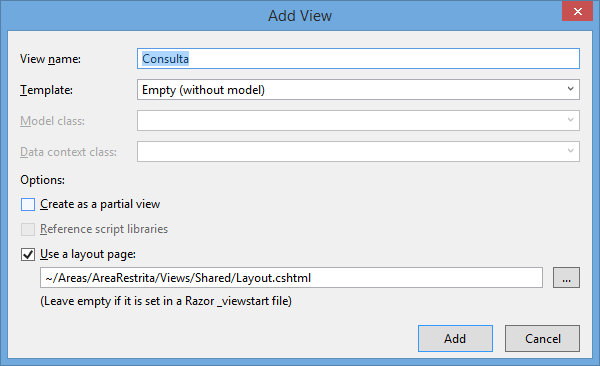
@{

Layout = "~/Areas/AreaRestrita/Views/Shared/Layout.cshtml";

}

<h4>Cadastro de Vendas</h4>

Página de acesso do Funcionario.



@{

Layout = "~/Areas/AreaRestrita/Views/Shared/Layout.cshtml";

}

<h4>Consulta de Vendas</h4>

Página de acesso do Funcionario.

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

**Tratamento para enviar o Usuario para a Area Restrita se o mesmo acessar a página de Login ja autenticado.**

**// GET: Usuario/Login**

**public ActionResult Login()**

**{**

**//verificar se o usuario ja esta autenticado..**

**if(HttpContext.User.Identity.IsAuthenticated)**

**{**

**return RedirectToAction("Index", "Home",**

**new { area = "AreaRestrita" });**

**}**

**return View();**

**}**

Global.asax

Criar um método para capturar a informação do Nome do Perfil do Usuario autenticado (contida no Ticket) e passa-la para o Asp.Net de forma que os controllers MVC possam comparar a Role corretamente.

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.NomePerfil };**

**HttpContext.Current.User =**

**new GenericPrincipal(id, roles);**

**}**

**}**

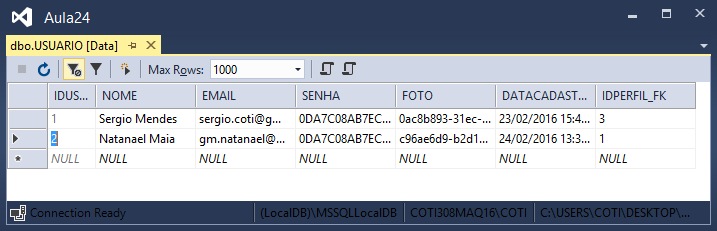
**}**

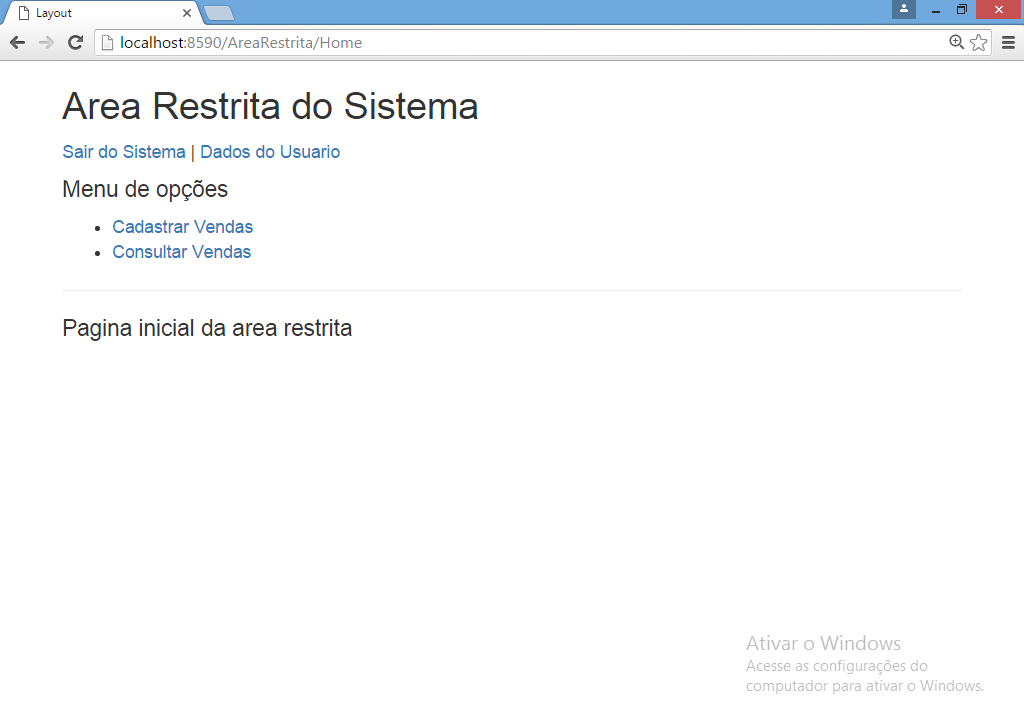
**}**

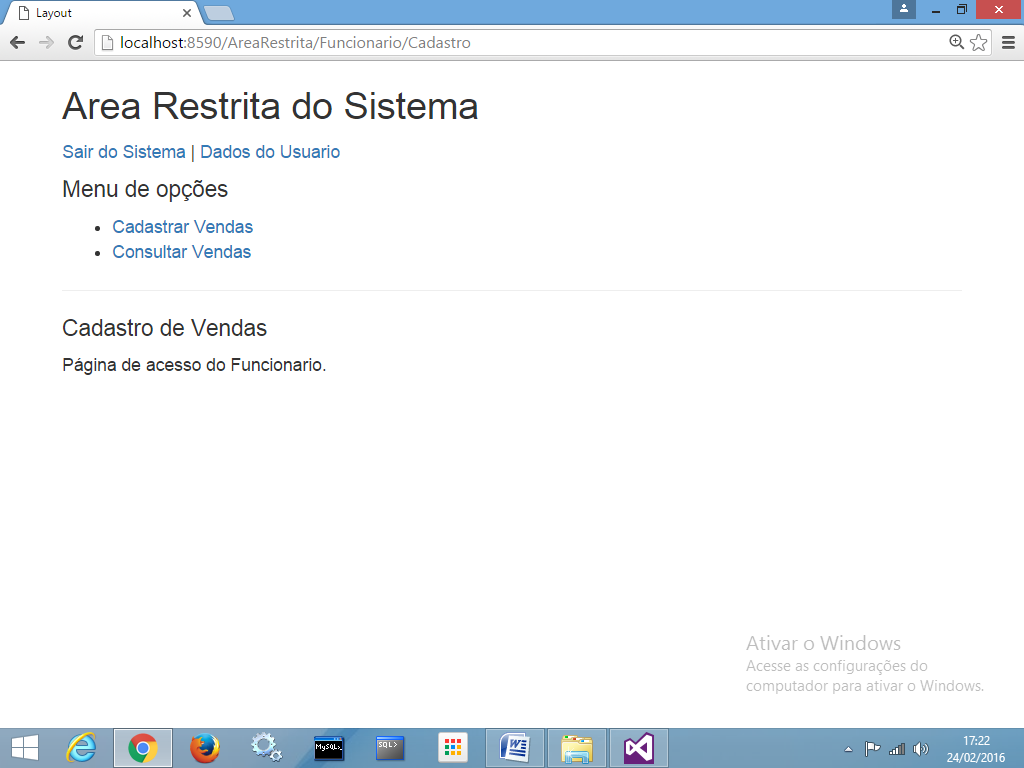
}

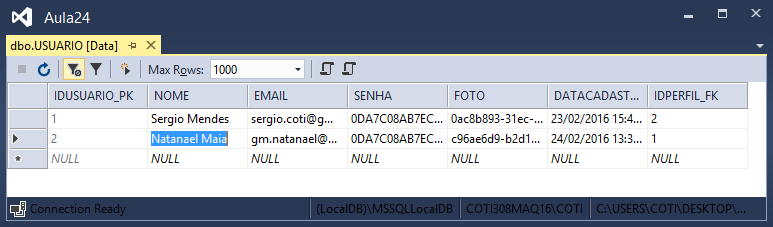
}

**Modificando o Perfil de Acesso do Usuario:**









**Usuario Administrador:**

