Recapitulando a última aula:
Criamos uma página chamada **/Account/Login**. E definimos essa página como a página inicial do projeto. Para isso, alteramos a classe **Program.cs** da aplicação.

/Program.cs
Configuramos o projeto para MVC e definimos a página inicial do projeto.

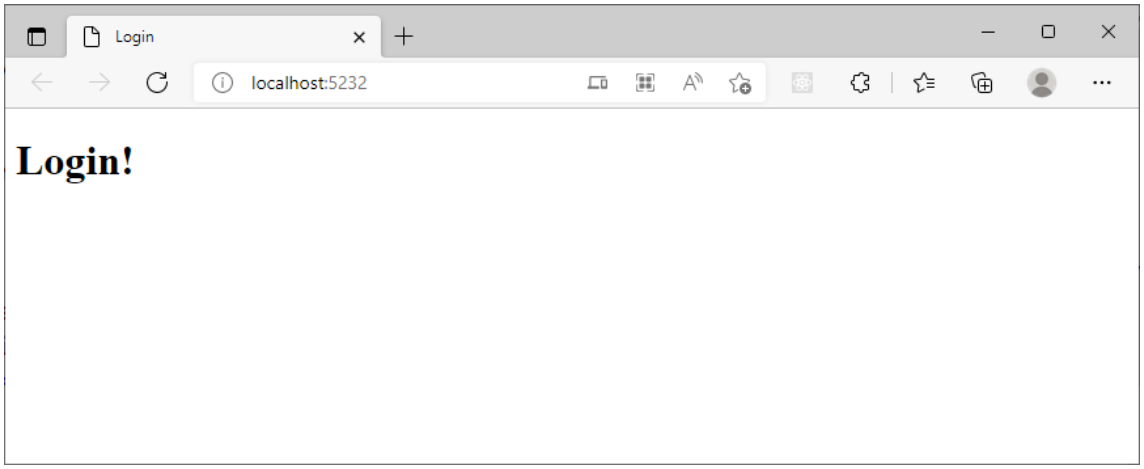
```
var builder = WebApplication.CreateBuilder(args);

// Configurando para MVC
builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");
}
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

//Definir a página inicial do projeto (/Account/Login)
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Account}/{action=Login}"
);

app.Run();
```

/Controllers/AccountController.cs
Neste controlador, criamos o método para abrir a página inicial do sistema.

```
using Microsoft.AspNetCore.Mvc;


namespace UsuariosWeb.Presentation.Controllers
{
    public class AccountController : Controller
    {
        public IActionResult Login()
        {
            return View();
        }
    }
}
```

A página foi criada em:
/Views/Account/Login.cshtml

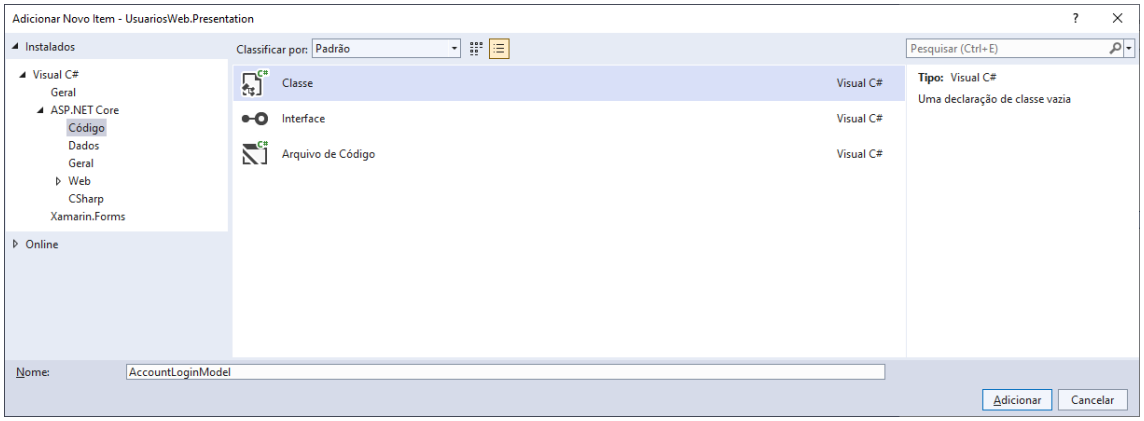
```
@{
    Layout = null;
}

<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Login</title>
</head>
<body>
    <h1>Login!</h1>
</body>
</html>
```



Criando uma página para autenticação do usuário:
Campos: Email e Senha

Primeiro, iremos criar uma classe Model para definir os campos que serão preenchidos na página de login:

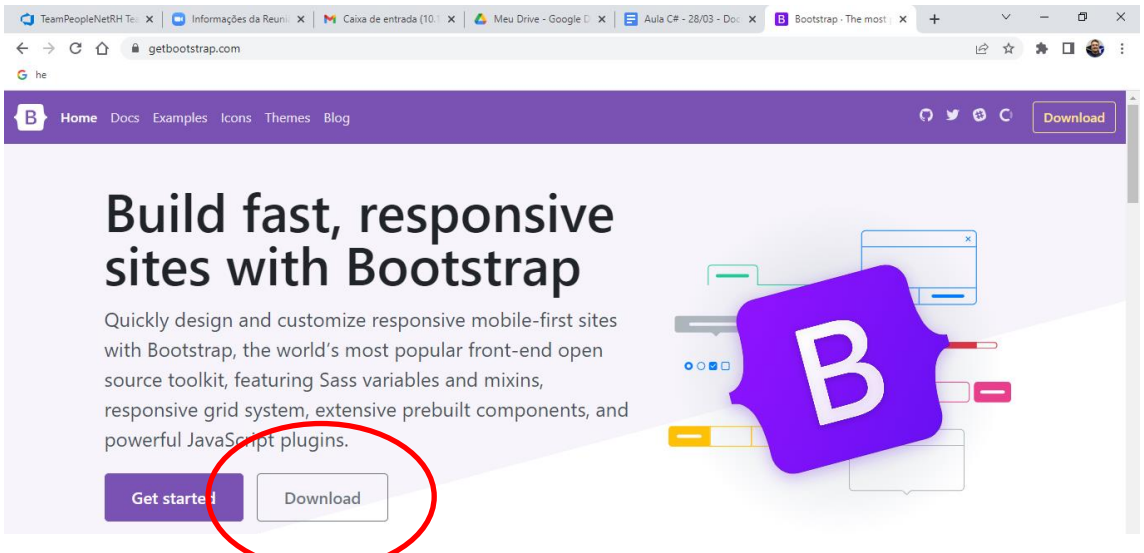
/AccountLoginModel.cs



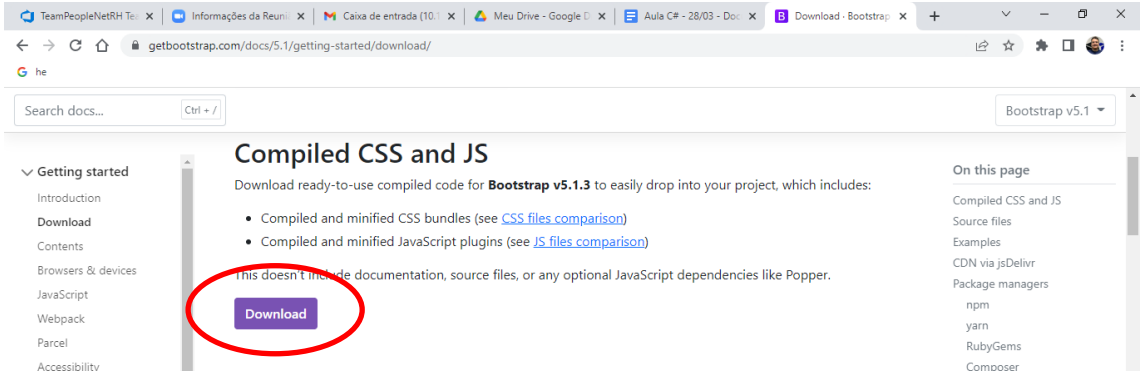
```
using System.ComponentModel.DataAnnotations;

namespace UsuariosWeb.Presentation.Models
{
    public class AccountLoginModel
    {
        [EmailAddress(ErrorMessage = "Por favor, informe um email válido.")]
        [Required(ErrorMessage = "Por favor, informe seu email.")]
        public string Email { get; set; }

        [MinLength(6, ErrorMessage = "Por favor, informe
                                      no mínimo {1} caracteres.")]
        [MaxLength(20, ErrorMessage = "Por favor, informe
                                      no máximo {1} caracteres.")]
        [Required(ErrorMessage = "Por favor, informe sua senha.")]
        public string Senha { get; set; }
    }
}
```

Obtendo os arquivos do Bootstrap

Biblioteca para desenvolvimento CSS e JS.
https://getbootstrap.com/



https://getbootstrap.com/docs/5.1/getting-started/download/



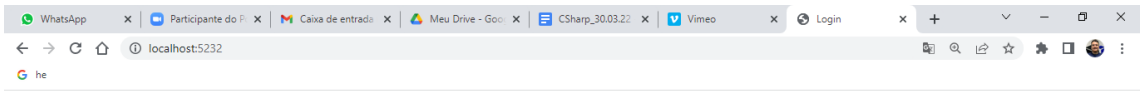
/Views/Account/Login.cshtml

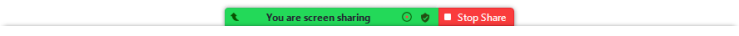
Página de autenticação do sistema.

```
@* definindo a classe de modelo da página *@
@model UsuariosWeb.Presentation.Models.AccountLoginModel

@{
    Layout = null;
}

<!DOCTYPE html>

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

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

</head>
<body>

    <div class="container">
        <div class="row mt-5">
            <div class="col-md-4 offset-4">
                <div class="card">
                    <div class="card-body">

                        <div class="text-center">
                            <h1>UsuariosWeb!</h1>
                            <h5>Acesso ao Sistema</h5>
                        </div>

                        <hr />

                        @* formulário para captura dos campos *@
                        @using (Html.BeginForm())
                        {
                            <div class="mb-3">
                                <label>Email de acesso:</label>
                                @Html.TextBoxFor(model => model.Email,
                                    new { @class = "form-control" })
                                <span class="text-danger">
                                    @Html.ValidationMessageFor
                                      (model => model.Email)
                                </span>
                            </div>

                            <div class="mb-3">
                                <label>Senha de acesso:</label>
                                @Html.PasswordFor(model => model.Senha,
                                    new { @class = "form-control" })
                                <span class="text-danger">
                                    @Html.ValidationMessageFor
                                      (model => model.Senha)
                                </span>
                            </div>
```

```
                        <div class="mb-3">
                            <div class="d-grid">
                                <input type="submit"
                                    value="Acessar Sistema"
                                    class="btn btn-primary"/>
                            </div>
                        </div>

                        <div class="mb-3">
                            @if(TempData["MensagemErro"] != null)
                            {
                                <strong class="text-danger">
                                    @TempData["MensagemErro"]
                                </strong>
                            }
                        </div>
                    }

                </div>

            </div>

        </div>

    </div>

    <!-- arquivos javascript -->
    <script src="/js/bootstrap.min.js"></script>

</body>
</html>
```

Executando:
http://localhost:5232/



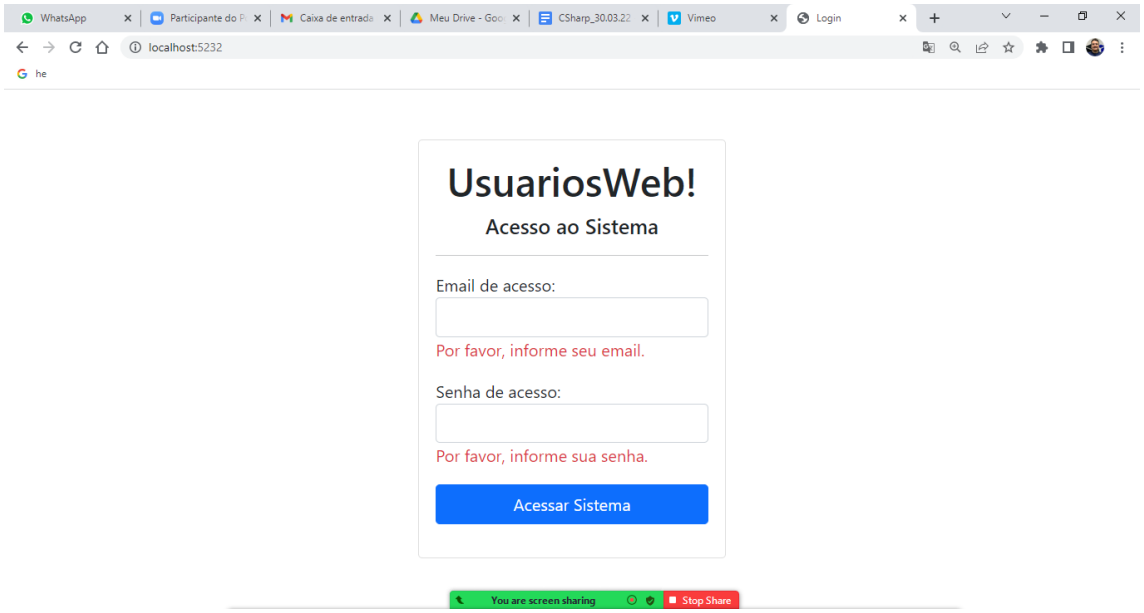


/Controllers/AccountController.cs

Criando o método **HTTP POST** para receber o SUBMIT da página.

```
using Microsoft.AspNetCore.Mvc;
using UsuariosWeb.Presentation.Models;

namespace UsuariosWeb.Presentation.Controllers
{
    public class AccountController : Controller
    {
        public IActionResult Login()
        {
            return View();
        }

        [HttpPost]
        public IActionResult Login(AccountLoginModel model)
        {
            if(ModelState.IsValid)
            {

            }

            return View();
        }
    }
}
```

Testando:
Visualizando as mensagens de erro de validação.
http://localhost:5232/

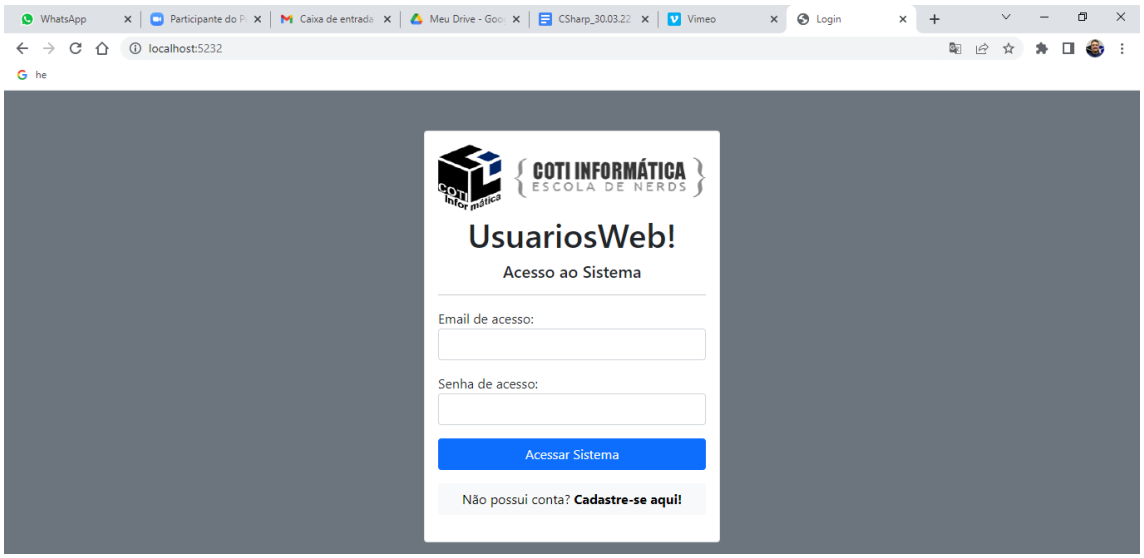


Finalizando a página de login do sistema:

```
@* definindo a classe de modelo da página *@
@model UsuariosWeb.Presentation.Models.AccountLoginModel

@{
    Layout = null;
}

<!DOCTYPE html>

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

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

</head>
<body class="bg-secondary">

    <div class="container">
        <div class="row mt-5">
            <div class="col-md-4 offset-4">
                <div class="card">
                    <div class="card-body">

                        <div class="text-center">

                            <img src="https://www.cotiinformatica.com.br
                                /imagens/logo-coti-informatica.png"
                                class="img-fluid mb-1"/>

                            <h1>UsuariosWeb!</h1>
                            <h5>Acesso ao Sistema</h5>
                        </div>

                        <hr />

                        @* formulário para captura dos campos *@
                        @using (Html.BeginForm())
                        {
                            <div class="mb-3">
                                <label>Email de acesso:</label>
                                @Html.TextBoxFor(model => model.Email,
                                    new { @class = "form-control" })
                                <span class="text-danger">
                                    @Html.ValidationMessageFor
                                      (model => model.Email)
                                </span>
                            </div>

                            <div class="mb-3">
                                <label>Senha de acesso:</label>
                                @Html.PasswordFor(model => model.Senha,
                                    new { @class = "form-control" })
                                <span class="text-danger">
                                    @Html.ValidationMessageFor
                                      (model => model.Senha)
                                </span>
```

```
                        </div>

                        <div class="mb-3">
                            <div class="d-grid">
                                <input type="submit"
                                    value="Acessar Sistema"
                                    class="btn btn-primary"/>
                            </div>
                        </div>

                        <div class="mb-3 text-center">
                            <div class="d-grid">
                                <a href="/Account/Register"
                                   class="btn btn-light">
                                    Não possui conta?
                                   <strong>Cadastre-se aqui!</strong>
                                </a>
                            </div>
                        </div>

                        <div class="mb-3">
                            @if(TempData["MensagemErro"] != null)
                            {
                                <strong class="text-danger">
                                    @TempData["MensagemErro"]
                                </strong>
                            }
                        </div>
                    }
                </div>
            </div>
        </div>
    </div>
    <!-- arquivos javascript -->
    <script src="/js/bootstrap.min.js"></script>
</body>
</html>
```

Resultado:
http://localhost:5232/



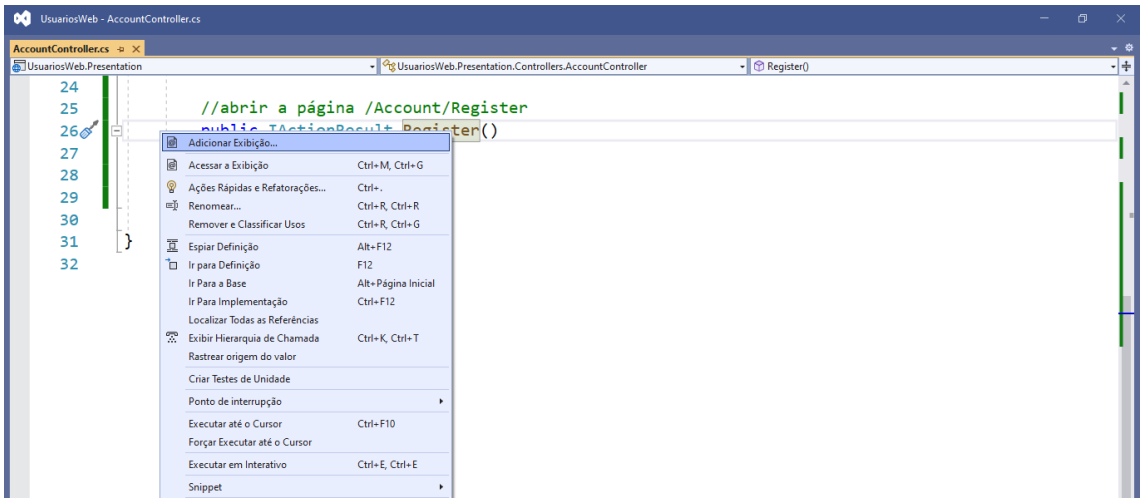
Criando uma página para que o usuário possa cadastrar uma conta de acesso:

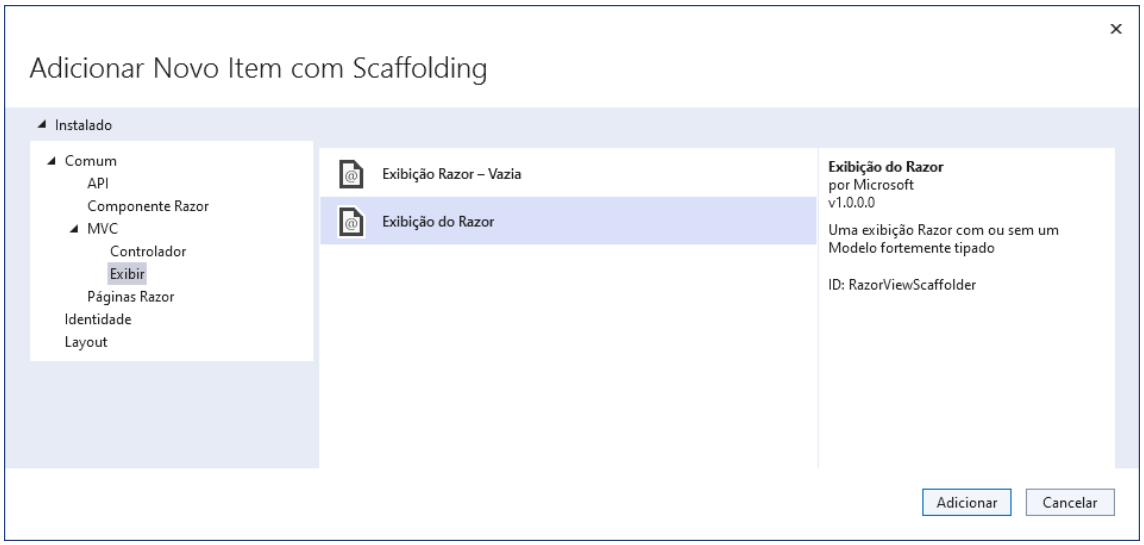
/Controllers/AccountController.cs

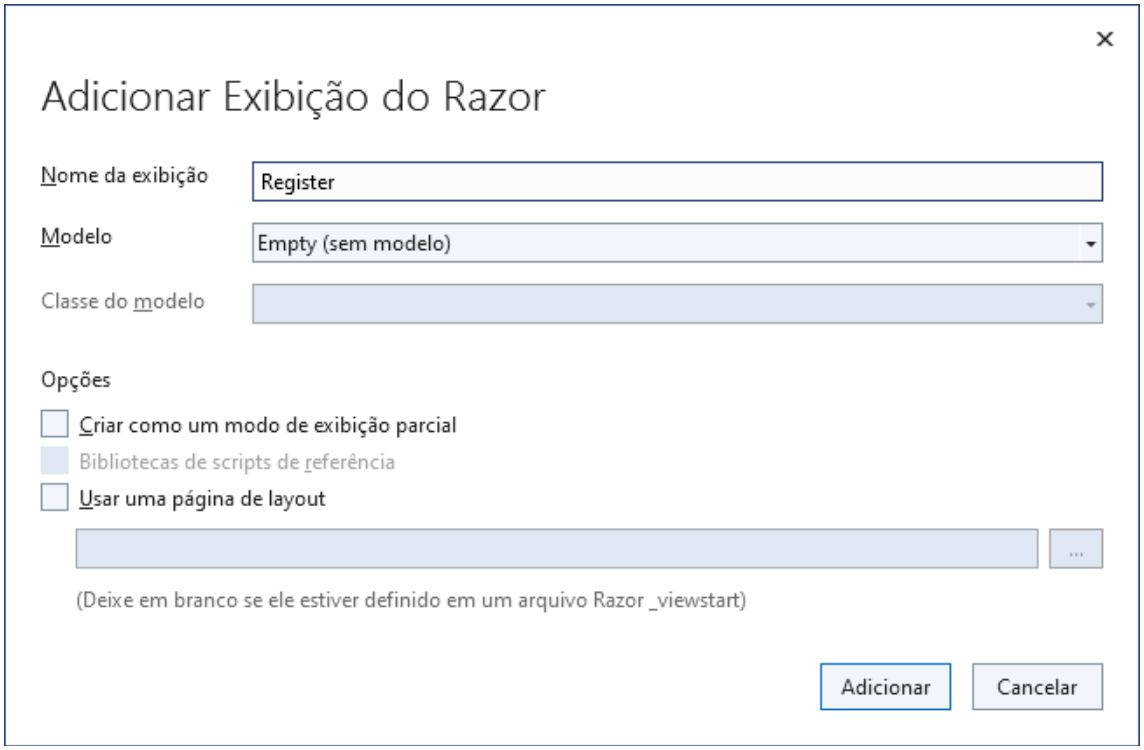
Primeiro, precisamos criar o método **IActionResult** no controlador para permitir que a página seja aberta.

```
using Microsoft.AspNetCore.Mvc;
using UsuariosWeb.Presentation.Models;

namespace UsuariosWeb.Presentation.Controllers
{
    public class AccountController : Controller
    {
        //abrir a página /Account/Login
        public IActionResult Login()
        {
            return View();
        }

        [HttpPost] //recebe o submit do formulário da página /Account/Login
        public IActionResult Login(AccountLoginModel model)
        {
            if(ModelState.IsValid)
            {

            }

            return View();
        }

        //abrir a página /Account/Register
        public IActionResult Register()
        {
             return View();
        }
    }
}
```

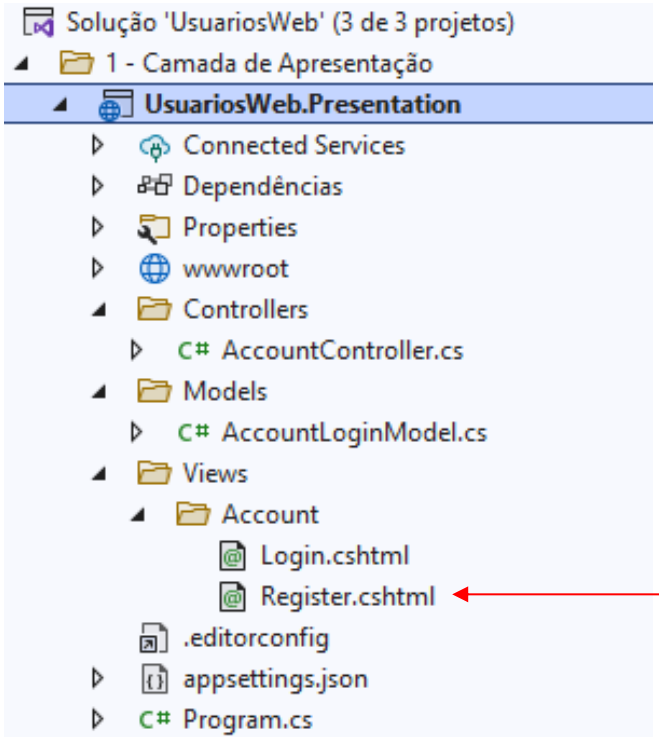
Criando a página:








```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Register</title>
</head>
<body>
</body>
</html>
```



/Models/AccountRegisterModel.cs

Classe de modelo de dados do MVC para capturar os campos do formulário da página de cadastro de usuário **/Account/Register**.

```
using System.ComponentModel.DataAnnotations;

namespace UsuariosWeb.Presentation.Models
{
    public class AccountRegisterModel
    {
        [MinLength(6, ErrorMessage = "Por favor,
            informe no mínimo {1} caracteres.")]
        [MaxLength(150, ErrorMessage = "Por favor,
            informe no máximo {1} caracteres.")]
        [Required(ErrorMessage = "Por favor, informe o nome do usuário.")]
        public string Nome { get; set; }

        [EmailAddress(ErrorMessage = "Por favor,
            informe um endereço de email válido.")]
        [Required(ErrorMessage = "Por favor, informe o email do usuário.")]
        public string Email { get; set; }

        [MinLength(8, ErrorMessage = "Por favor,
            informe no mínimo {1} caracteres.")]
        [MaxLength(20, ErrorMessage = "Por favor,
            informe no máximo {1} caracteres.")]
        [Required(ErrorMessage = "Por favor, informe a senha do usuário.")]
        public string Senha { get; set; }

        [Compare("Senha", ErrorMessage = "Senhas não conferem.")]
        [Required(ErrorMessage = "Por favor, confirme a senha do usuário.")]
        public string SenhaConfirmacao { get; set; }
    }
}
```

Desenhando a página de cadastro de usuário

/Views/Account/Register.cshtml

```
@* definindo a classe de modelo desta página *@
@model UsuariosWeb.Presentation.Models.AccountRegisterModel

@{
    Layout = null;
}

<!DOCTYPE html>

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

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

</head>
<body class="bg-secondary">

    <div class="container">
        <div class="row mt-5">
            <div class="col-md-4 offset-4">
                <div class="card">
                    <div class="card-body">

                        <div class="text-center">
                            <h1>UsuariosWeb!</h1>
                            <h5>Cadastro de Conta de Usuário</h5>
                        </div>

                        <hr />

                        @* formulário para captura dos campos *@
                        @using (Html.BeginForm())
                        {
                            <div class="mb-3">
                                <label>Nome do usuário:</label>
                                @Html.TextBoxFor(model => model.Nome,
                                    new { @class = "form-control" })
                                <span class="text-danger">
                                    @Html.ValidationMessageFor
                                      (model => model.Nome)
                                </span>
                            </div>

                            <div class="mb-3">
                                <label>Email de acesso:</label>
                                @Html.TextBoxFor(model => model.Email,
                                    new { @class = "form-control" })
                                <span class="text-danger">
                                    @Html.ValidationMessageFor
                                      (model => model.Email)
                                </span>
                            </div>
```

```
                        <div class="mb-3">
                            <label>Senha de acesso:</label>
                            @Html.PasswordFor(model => model.Senha,
                                new { @class = "form-control" })
                            <span class="text-danger">
                                @Html.ValidationMessageFor
                                  (model => model.Senha)
                            </span>
                        </div>

                        <div class="mb-3">
                            <label>Confirme a sua senha:</label>
                            @Html.PasswordFor
                                  (model => model.SenhaConfirmacao,
                                new { @class = "form-control" })
                            <span class="text-danger">
                                @Html.ValidationMessageFor
                                  (model => model.SenhaConfirmacao)
                            </span>
                        </div>

                        <div class="mb-3">
                            <div class="d-grid">
                                <input type="submit"
                                    value="Realizar Cadastro"
                                    class="btn btn-success"/>
                            </div>
                        </div>

                        <div class="mb-3 text-center">
                            <div class="d-grid">
                                <a href="/Account/Login"
                                   class="btn btn-light">
                                    Já possui conta? <strong>
                                        Acesse aqui!</strong>
                                </a>
                            </div>
                        </div>

                        <div class="mb-3">
                            @if(TempData["MensagemSucesso"] != null)
                            {
                                <strong class="text-success">
                                    @TempData["MensagemSucesso"]
                                </strong>
                            }
                            else if(TempData["MensagemErro"] != null)
                            {
                                <strong class="text-danger">
                                    @TempData["MensagemErro"]
                                </strong>
                            }
                        </div>
                    }

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

```
    <!-- arquivos javascript -->
    <script src="/js/bootstrap.min.js"></script>

</body>
</html>
```

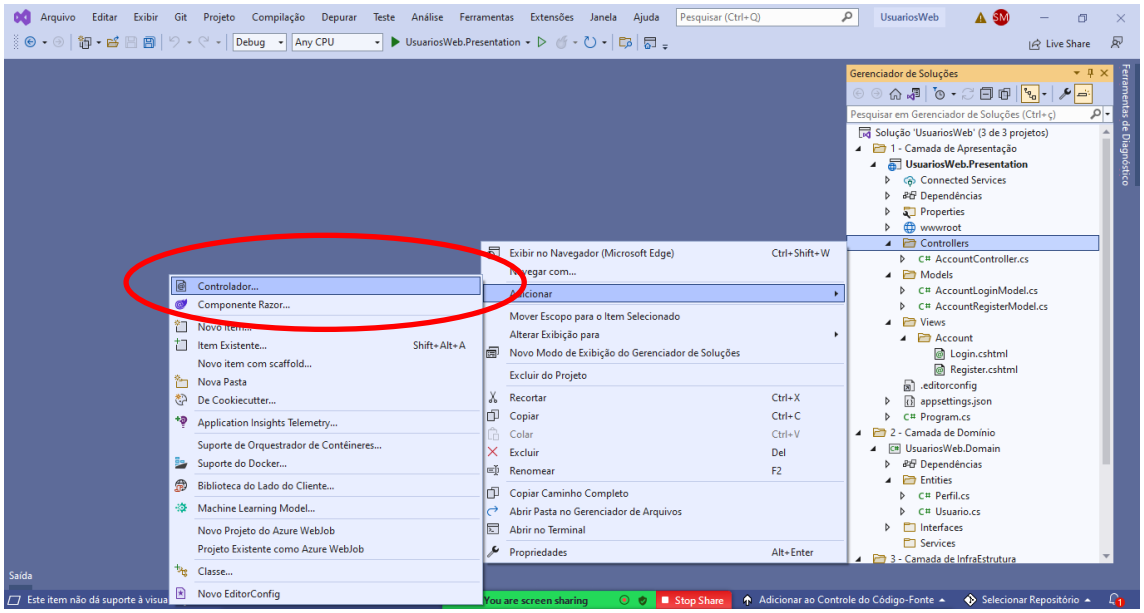
Criando o método HTTP POST para receber o envio do SUBMIT do formulário da página /Account/Register

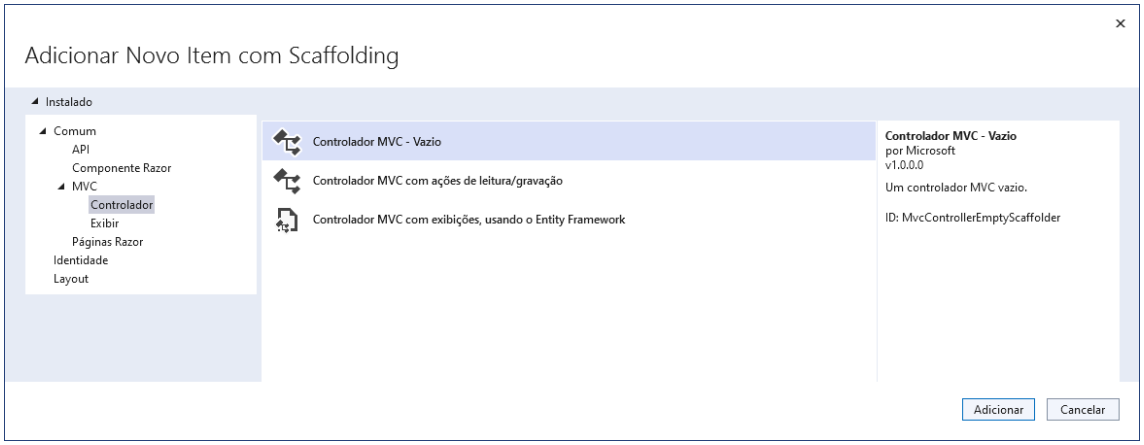
```
using Microsoft.AspNetCore.Mvc;
using UsuariosWeb.Presentation.Models;

namespace UsuariosWeb.Presentation.Controllers
{
    public class AccountController : Controller
    {
        //abrir a página /Account/Login
        public IActionResult Login()
        {
            return View();
        }

        [HttpPost] //recebe o submit do formulário da página /Account/Login
        public IActionResult Login(AccountLoginModel model)
        {
            if(ModelState.IsValid)
            {

            }

            return View();
        }

        //abrir a página /Account/Register
        public IActionResult Register()
        {
            return View();
        }

        [HttpPost] //recebe o submit do formulário
       //da página /Account/Register
        public IActionResult Register(AccountRegisterModel model)
        {
             if(ModelState.IsValid)
             {

             }

             return View();
        }

    }

}
```

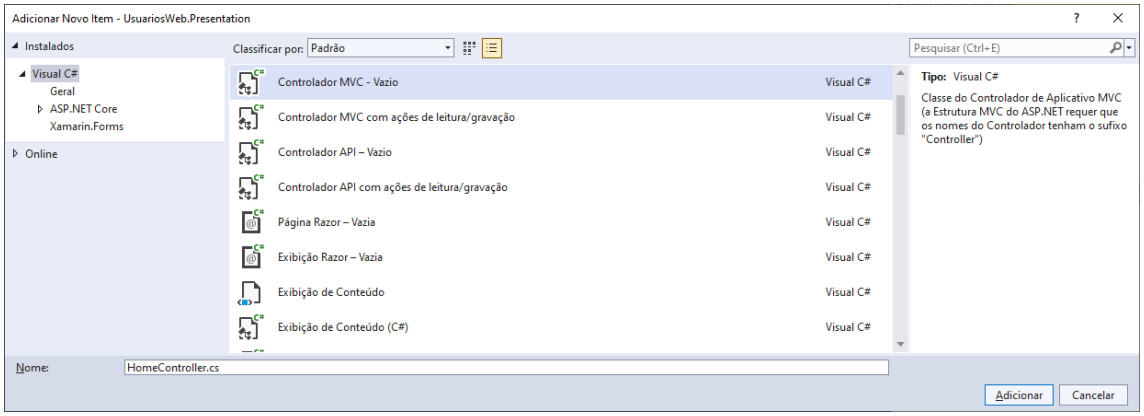
Criando a página para onde o usuário será redirecionado após a sua autenticação.

/Home/Index

Página inicial do projeto após a autenticação do usuário.



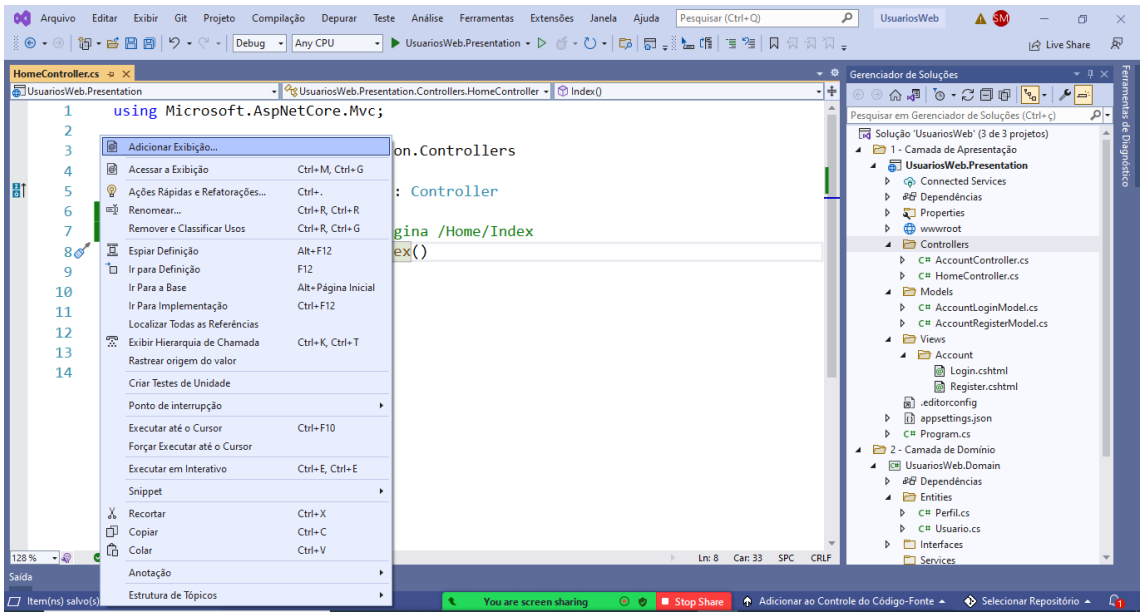


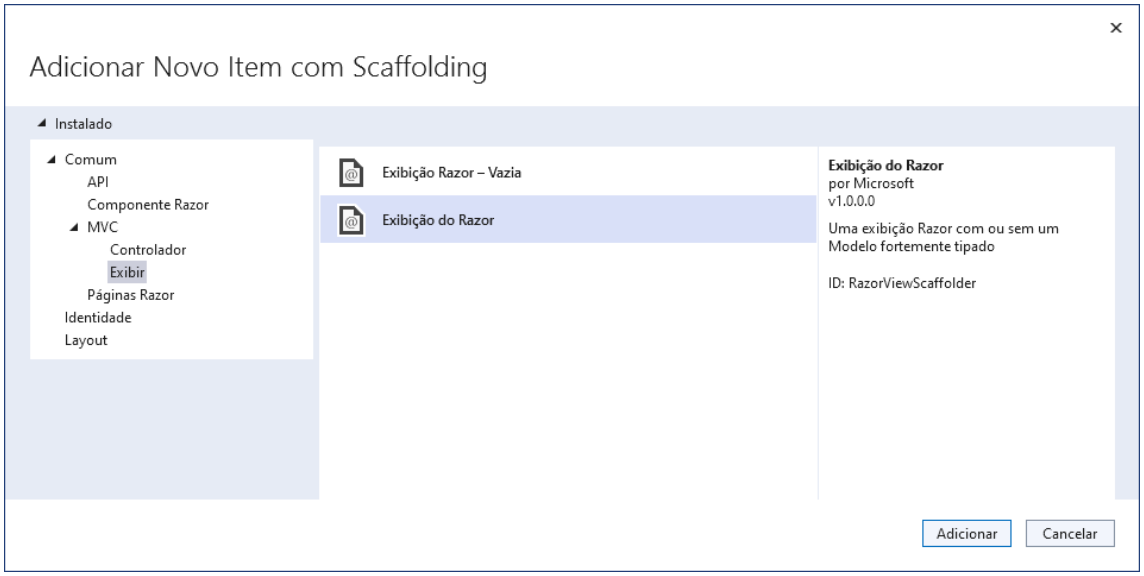


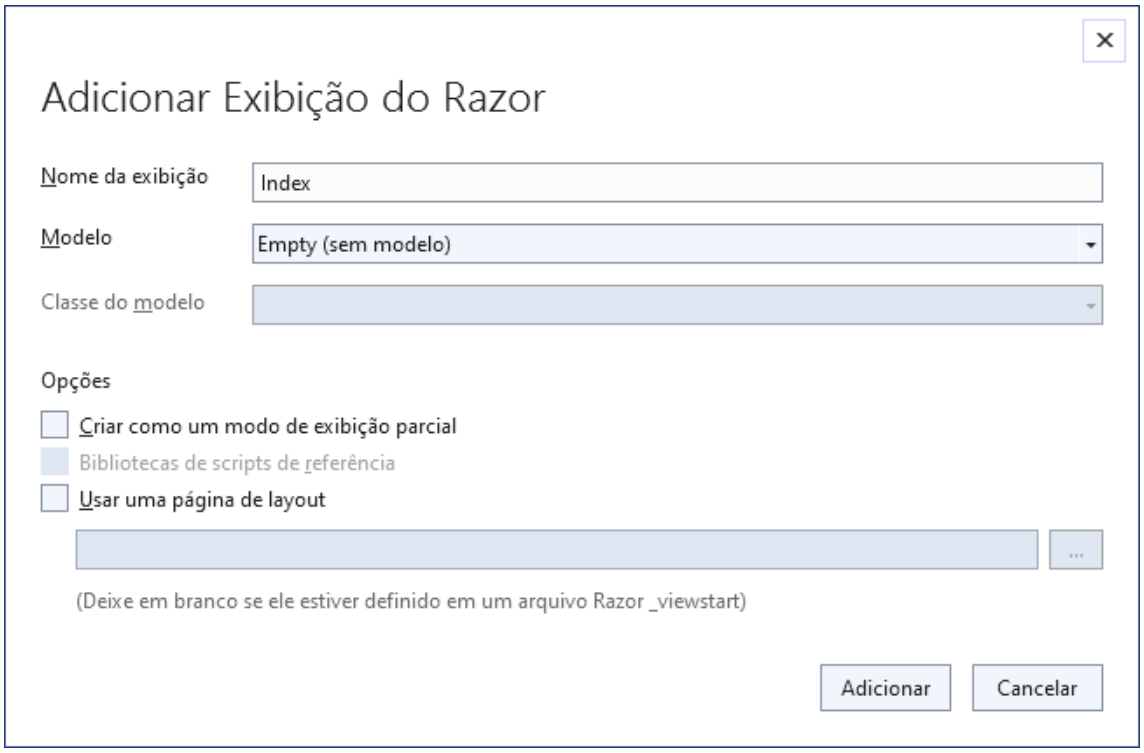

```
using Microsoft.AspNetCore.Mvc;
```

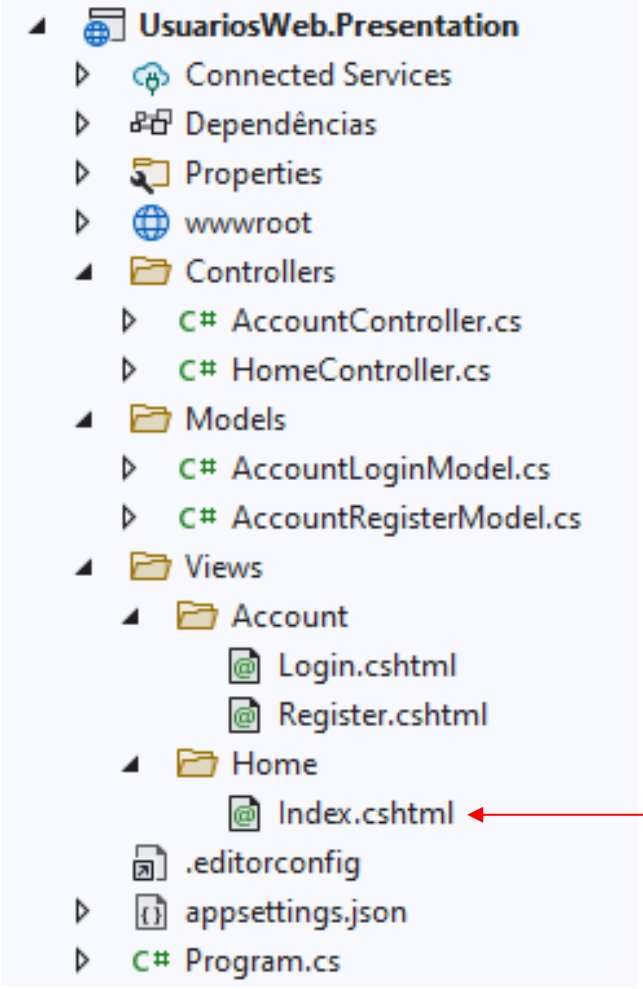
```
namespace UsuariosWeb.Presentation.Controllers
{
    public class HomeController : Controller
    {
        //método para abrir a página /Home/Index
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Criando a página:






Adicionar Exibição do Razor
Nome da exibição
Index
Modelo
Empty (sem 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


UsuariosWeb.Presentation
Connected Services
Dependências
Properties
wwwroot
Controllers
AccountController.cs
HomeController.cs
Models
AccountLoginModel.cs
AccountRegisterModel.cs
Views
Account
Login.cshtml
Register.cshtml
Home
Index.cshtml
.editorconfig
appsettings.json
Program.cs

Criando uma página de layout mestre:
/Views/Shared/Layout.cshtml

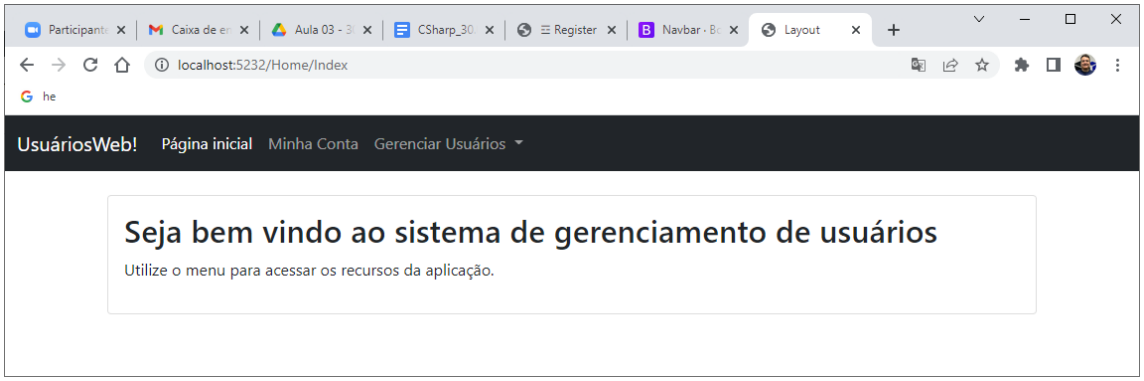
```
@{
    Layout = null;
}

<!DOCTYPE html>

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

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

</head>
<body>

    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <div class="container-fluid">
            <a class="navbar-brand" href="#">UsuáriosWeb!</a>
            <button class="navbar-toggler" type="button" data-bs-
                    toggle="collapse" data-bs-
                    target="#navbarSupportedContent" aria-
                    controls="navbarSupportedContent" aria-
                    expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse"
                 id="navbarSupportedContent">
                <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page"
                           href="/Home/Index">Página inicial</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="#">Minha Conta</a>
                    </li>
                    <li class="nav-item dropdown">
                        <a class="nav-link dropdown-toggle" href="#"
                                id="navbarDropdown" role="button" data-bs-
                                toggle="dropdown" aria-expanded="false">
                            Gerenciar Usuários
                        </a>
                        <ul class="dropdown-menu" aria-
                             labelledby="navbarDropdown">
                            <li><a class="dropdown-item"
                                href="/Usuario/Cadastro">Cadastrar
                                Usuários</a></li>
                            <li><a class="dropdown-item"
                                href="/Usuario/Consulta">Consultar
                                Usuários</a></li>
                            <li><hr class="dropdown-divider"></li>
                            <li><a class="dropdown-item"
                                href="/Usuario/Relatorio">Relatório de
                                Usuários</a></li>
                        </ul>
                    </li>
```

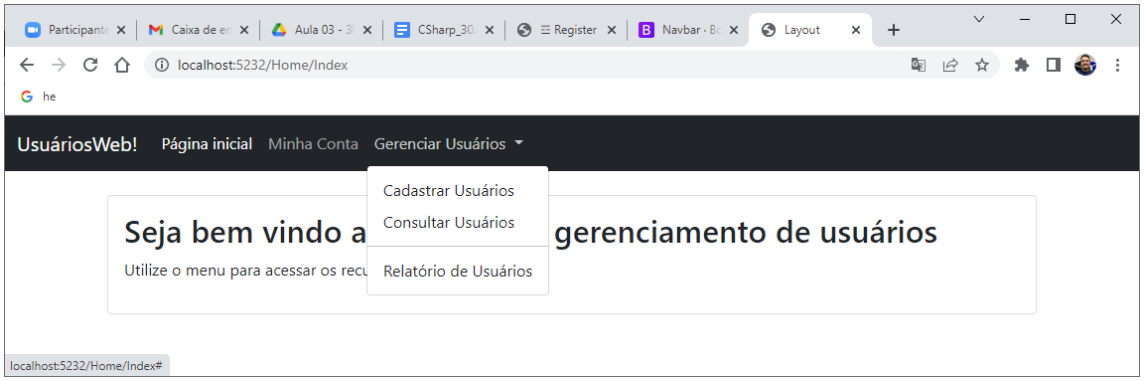
```
                </ul>
            </div>
        </div>
    </nav>

    <div class="container mt-4">
        @* Local para entrada do conteudo das demais páginas do projeto. *@
        @RenderBody()
    </div>

    <!-- Arquivos de extensão JavaScript (JS) -->
    <script src="~/js/bootstrap.min.js"></script>

</body>
</html>
```

Fazendo com que a página /Home/Index possa HERDAR a página de Layout:

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

<div class="card">
    <div class="card-body">
        <h2>Seja bem vindo ao sistema de gerenciamento de usuários</h2>
        <p>Utilize o menu para acessar os recursos da aplicação.</p>
    </div>
</div>
```

http://localhost:5232/Home/Index





Processo de Autenticação em aplicações AspNet MVC:

Passo a passo para autenticação de usuários:

1. O usuário irá acessar a página **/Account/Login** e informar seu email e senha e solicitar o acesso (SUBMIT).

2. O controlador **AccountController** deverá validar os dados do usuário e se estiverem corretos, ele deverá criar uma autorização de acesso para o usuário. Esta autorização será gravada em um arquivo de Cookie no navegador do usuário.

3. O controlador **AccountController** irá redirecionar o usuário para a página /Home/Index. Porém, esta página /Home/Index e todas as demais só permitirão que usuários autorizados possam acessá-las.

Primeiro, precisamos configurar o projeto para permitir o uso de Cookies e para permitir a implementação da autenticação dos usuários.

/Program.cs

Classe de inicialização e configuração do projeto.

```
using Microsoft.AspNetCore.Authentication.Cookies;

var builder = WebApplication.CreateBuilder(args);

// Configurando para MVC
builder.Services.AddControllersWithViews();

//habilitando o projeto para usar cookies e autenticação de acesso
builder.Services.Configure<CookiePolicyOptions>
    (options => { options.MinimumSameSitePolicy = SameSiteMode.None; });
builder.Services.AddAuthentication
    (CookieAuthenticationDefaults.AuthenticationScheme).AddCookie();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");
}
app.UseStaticFiles();
app.UseRouting();

//autenticação e autorização
app.UseCookiePolicy();
app.UseAuthentication();
app.UseAuthorization();

//Definir a página inicial do projeto (/Account/Login)
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Account}/{action=Login}"
);
app.Run();
```

Definindo a classe HomeController como autorizada somente para usuários autenticados.

[Authorize]

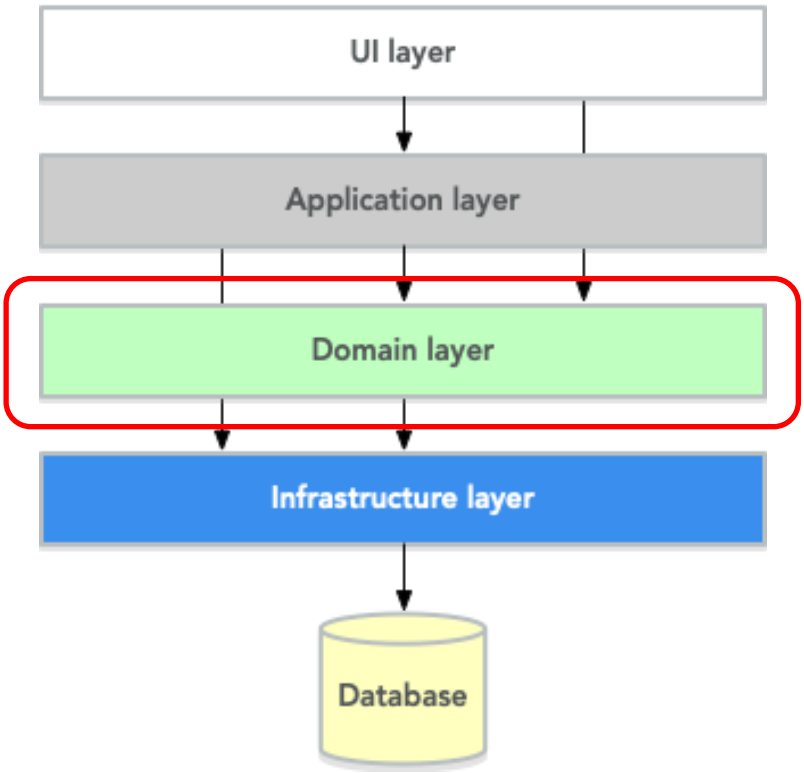
Annotation do AspNet que faz com que o acesso a um controlador só seja permitido para usuários que estejam autenticados na aplicação.

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace UsuariosWeb.Presentation.Controllers
{
    [Authorize] //só permite acesso de usuários autenticados!
    public class HomeController : Controller
    {
        //método para abrir a página /Home/Index
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Dessa forma, só conseguiremos acessar a página /Home/Index se estivermos autenticados na aplicação.

Próximo passo:

Vamos criar as regras de negócio para autenticação e cadastro do usuário:



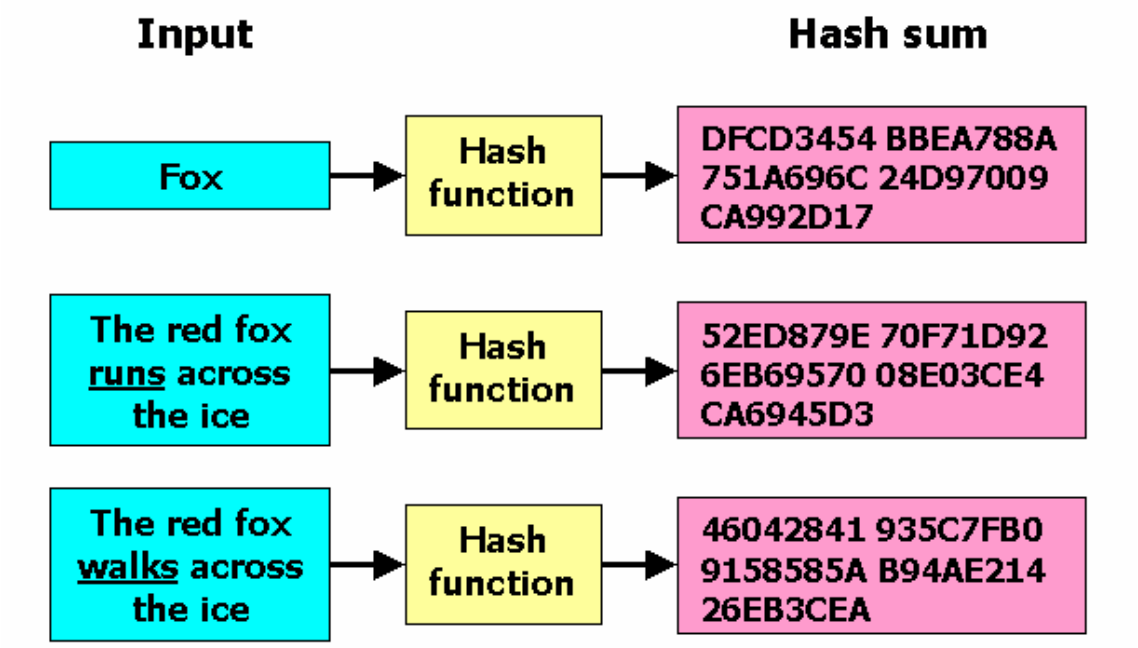
Antes de fazermos as regras de negócio da camada de domínio, vamos alterar a classe **UsuarioRepository** para criptografar a senha do usuário no banco de dados do SqlServer.

/Repositories/UsuarioRepository.cs

Camada de InfraEstrutura de banco de dados.

MD5

Algoritmo de criptografia de dados baseado em 128bits. Consegue converter qualquer valor para um HASH de 32 caracteres hexadecimal. Um valor criptografado em MD5 não pode ser descriptografado.




```
using Dapper;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using UsuariosWeb.Domain.Entities;
using UsuariosWeb.Domain.Interfaces.Repositories;

namespace UsuariosWeb.Infra.Data.Repositories
{
    /// <summary>
    /// Classe para implementar o repositorio da entidade Usuário
    /// </summary>
    public class UsuarioRepository : IUsuarioRepository
    {
        //atributo para armazenar a string de conexão do banco de dados
        private string _connectionString;
```

```
//construtor para receber a string de conexão do banco de dados
public UsuarioRepository(string connectionString)
{
    _connectionString = connectionString;
}

public void Inserir(Usuario entity)
{
    var query = @"
                INSERT INTO USUARIO(
                    IDUSUARIO,
                    NOME,
                    EMAIL,
                    SENHA,
                    DATACADASTRO,
                    IDPERFIL)
                VALUES(
                    @IdUsuario,
                    @Nome,
                    @Email,
                    CONVERT(VARCHAR(32),
                       HASHBYTES('MD5', @Senha), 2),
                    @DataCadastro,
                    @IdPerfil)
            ";

    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Execute(query, entity);
    }
}

public void Alterar(Usuario entity)
{
    var query = @"
                UPDATE USUARIO
                SET
                    NOME = @Nome,
                    EMAIL = @Email,
                    IDPERFIL = @IdPerfil
                WHERE
                    IDUSUARIO = @IdUsuario
            ";

    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Execute(query, entity);
    }
}
```

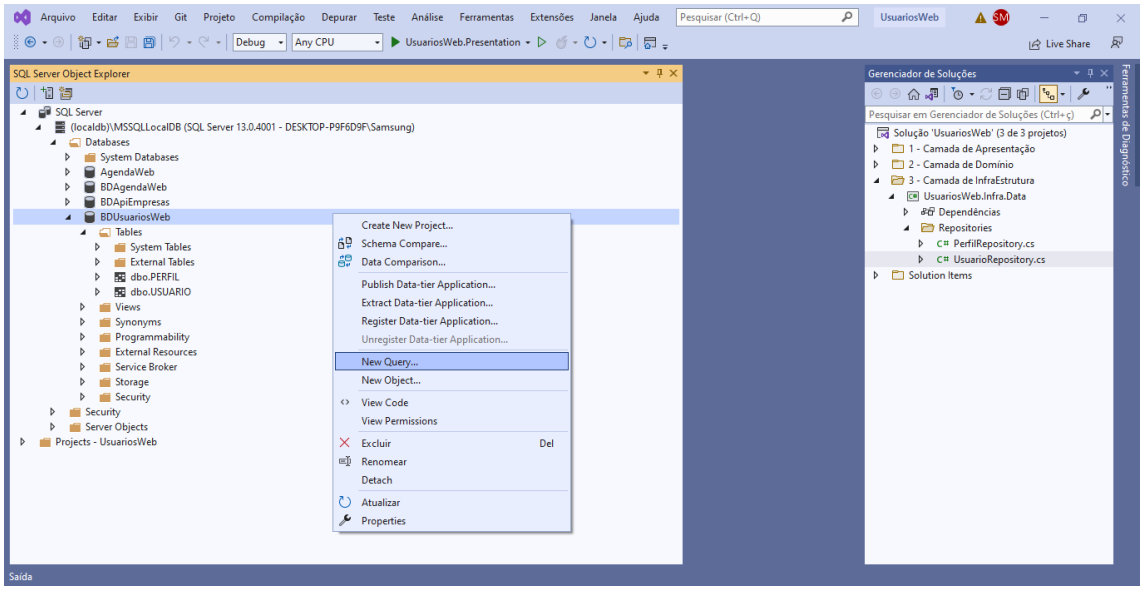
```
public void Excluir(Usuario entity)
{
    var query = @"
                DELETE USUARIO
                WHERE IDUSUARIO = @IdUsuario
            ";

    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Execute(query, entity);
    }
}

public List<Usuario> Consultar()
{
    var query = @"
            SELECT * FROM USUARIO
            ORDER BY NOME
        ";

    using (var connection = new SqlConnection(_connectionString))
    {
        return connection
            .Query<Usuario>(query)
            .ToList();
    }
}

public Usuario ObterPorId(Guid id)
{
    var query = @"
            SELECT * FROM USUARIO
            WHERE IDUSUARIO = @id
        ";

    using (var connection = new SqlConnection(_connectionString))
    {
        return connection
            .Query<Usuario>(query, new { id })
            .FirstOrDefault();
    }
}

public Usuario Obter(string email)
{
    var query = @"
            SELECT * FROM USUARIO
            WHERE EMAIL = @email
        ";

    using (var connection = new SqlConnection(_connectionString))
    {
        return connection
```

```
                    .Query<Usuario>(query, new { email })
                    .FirstOrDefault();
            }
        }

        public Usuario Obter(string email, string senha)
        {
            var query = @"
                    SELECT * FROM USUARIO
                    WHERE EMAIL = @email
                    AND SENHA = CONVERT(VARCHAR(32),
                                    HASHBYTES('MD5', @senha), 2),
                ";

            using (var connection = new SqlConnection(_connectionString))
            {
                return connection
                    .Query<Usuario>(query, new { email, senha })
                    .FirstOrDefault();
            }

        }

    }

}
```

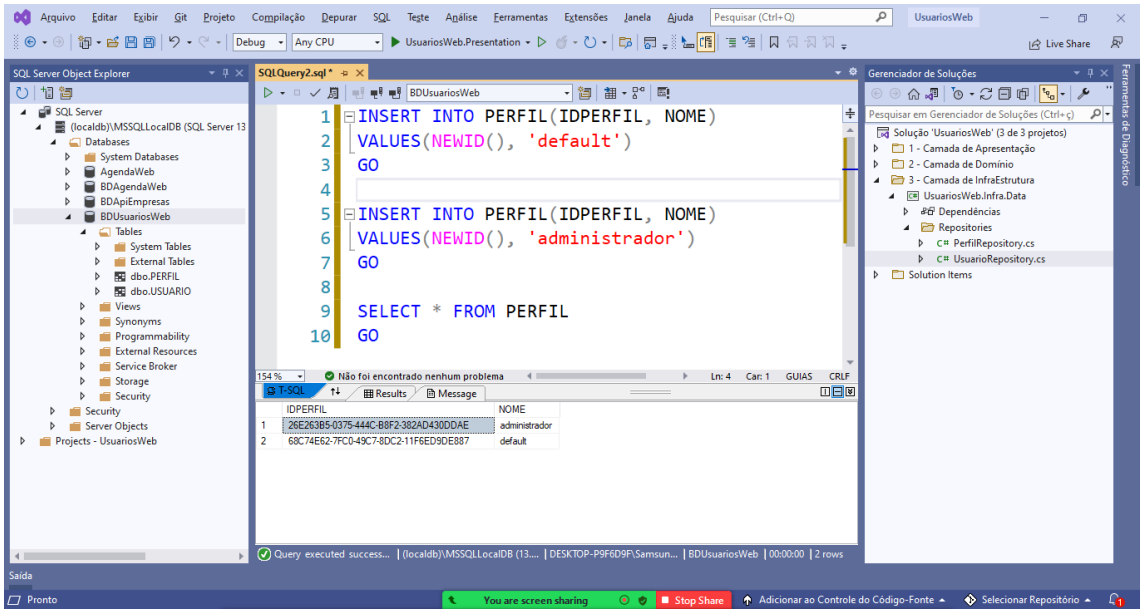
Modificando a tabela de Perfil do banco de dados para não permitir o cadastro de perfis com nomes iguais (campo único).

```
CREATE TABLE [dbo].[PERFIL] (
    [IDPERFIL] UNIQUEIDENTIFIER NOT NULL,
    [NOME]     NVARCHAR (50)    NOT NULL UNIQUE,
    PRIMARY KEY CLUSTERED ([IDPERFIL] ASC)
);

CREATE TABLE [dbo].[USUARIO] (
    [IDUSUARIO]    UNIQUEIDENTIFIER NOT NULL,
    [NOME]         NVARCHAR (150)   NOT NULL,
    [EMAIL]        NVARCHAR (100)   NOT NULL,
    [SENHA]        NVARCHAR (40)    NOT NULL,
    [DATACADASTRO] DATETIME         NOT NULL,
    [IDPERFIL]     UNIQUEIDENTIFIER NOT NULL,
    PRIMARY KEY CLUSTERED ([IDUSUARIO] ASC),
    UNIQUE NONCLUSTERED ([EMAIL] ASC),
    FOREIGN KEY ([IDPERFIL])
      REFERENCES [dbo].[PERFIL] ([IDPERFIL])
);
```

Cadastrando perfis no banco de dados:




```
INSERT INTO PERFIL(IDPERFIL, NOME)
VALUES(NEWID(), 'default')
GO

INSERT INTO PERFIL(IDPERFIL, NOME)
VALUES(NEWID(), 'administrador')
GO

SELECT * FROM PERFIL
GO
```



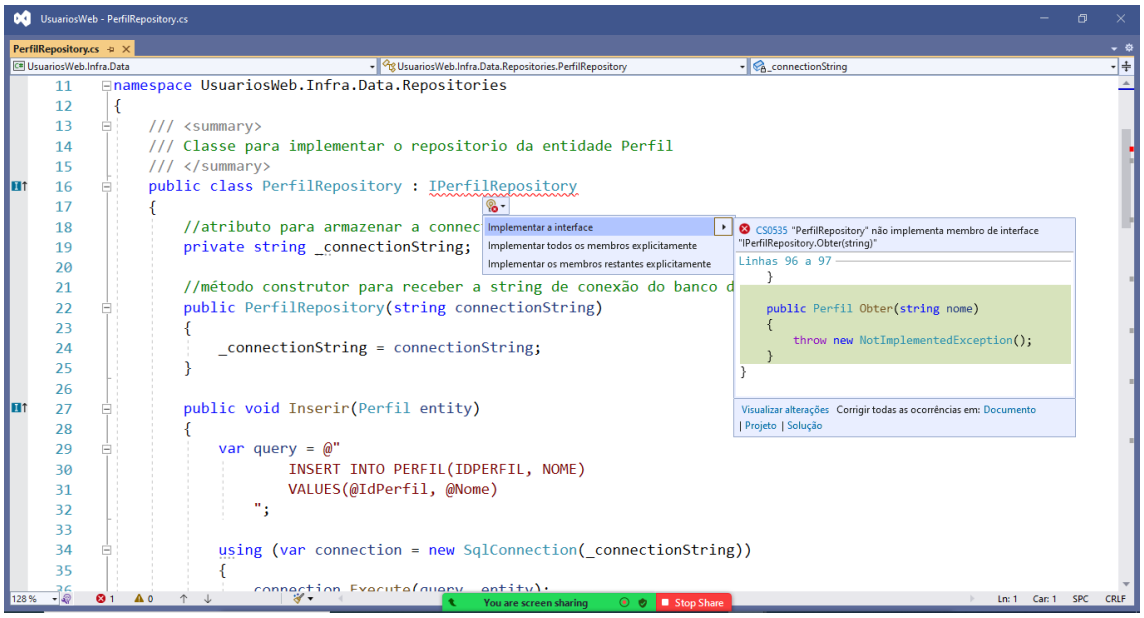
Criando um método no repositório para consultar 1 perfil através do nome:

/Domain/Interfaces/Repositories/IPerfilRepository.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using UsuariosWeb.Domain.Entities;

namespace UsuariosWeb.Domain.Interfaces.Repositories
{
    /// <summary>
    /// Interface de repositório para a entidade Perfil
    /// </summary>
    public interface IPerfilRepository : IBaseRepository<Perfil>
    {
        /// <summary>
        /// Método para consultar 1 perfil no banco de dados atraves do nome
        /// </summary>
        Perfil Obter(string nome);
    }
}
```

Implementando a interface:

/Infra.Data/Repositories/PerfilRepository.cs




```
using Dapper;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;
using UsuariosWeb.Domain.Entities;
using UsuariosWeb.Domain.Interfaces.Repositories;

namespace UsuariosWeb.Infra.Data.Repositories
{
    /// <summary>
    /// Classe para implementar o repositorio da entidade Perfil
    /// </summary>
    public class PerfilRepository : IPerfilRepository
    {
        //atributo para armazenar a connectionstring do banco de dados
        private string _connectionString;

        //método construtor para receber
        //a string de conexão do banco de dados
        public PerfilRepository(string connectionString)
        {
            _connectionString = connectionString;
        }

        public void Inserir(Perfil entity)
        {
            var query = @"
                    INSERT INTO PERFIL(IDPERFIL, NOME)
                    VALUES(@IdPerfil, @Nome)
                ";

            using (var connection = new SqlConnection(_connectionString))
            {
                connection.Execute(query, entity);
            }
        }

        public void Alterar(Perfil entity)
        {
            var query = @"
                    UPDATE PERFIL SET
                        NOME = @Nome
                    WHERE
                        IDPERFIL = @IdPerfil
                ";

            using (var connection = new SqlConnection(_connectionString))
            {
                connection.Execute(query, entity);
            }
        }

        public void Excluir(Perfil entity)
        {
            var query = @"
                    DELETE FROM PERFIL
```

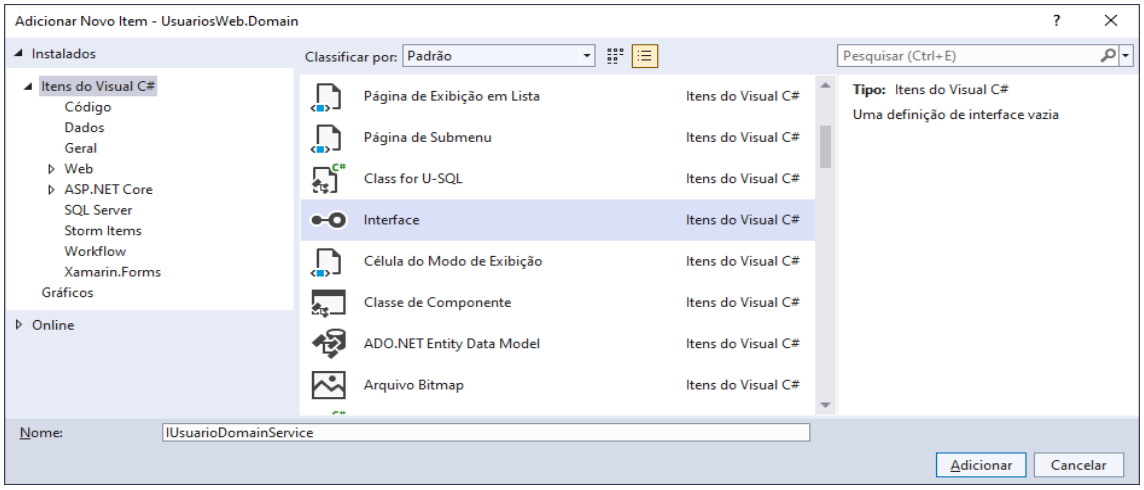
```
                    WHERE IDPERFIL = @IdPerfil
                ";

            using (var connection = new SqlConnection(_connectionString))
            {
                connection.Execute(query, entity);
            }
        }

        public List<Perfil> Consultar()
        {
            var query = @"
                    SELECT * FROM PERFIL
                    ORDER BY NOME
                ";

            using (var connection = new SqlConnection(_connectionString))
            {
                return connection
                    .Query<Perfil>(query)
                    .ToList();
            }
        }

        public Perfil ObterPorId(Guid id)
        {
            var query = @"
                    SELECT * FROM PERFIL
                    WHERE IDPERFIL = @id
                ";

            using (var connection = new SqlConnection(_connectionString))
            {
                return connection
                    .Query<Perfil>(query, new { id })
                    .FirstOrDefault();
            }
        }

        public Perfil Obter(string nome)
        {
            var query = @"
                    SELECT * FROM PERFIL
                    WHERE NOME = @nome
                ";

            using (var connection = new SqlConnection(_connectionString))
            {
                return connection
                    .Query<Perfil>(query, new { nome })
                    .FirstOrDefault();
            }
        }
    }
}
```

DOMAIN SERVICES

Interfaces e Classes que serão desenvolvidas na camada de Domínio do sistema e voltadas para implementação de regras de negócio da aplicação.

Domain Services (Serviços de domínio)
Os serviços do domínio (*Domain Services*) são classes que tem como objetivo serem uma alternativa para o **desacoplamento de código**. Os serviços do domínio surgem em cenários onde a escolha de dar responsabilidade a uma classe ou outra poderia causar problemas com **acoplamento do código**.
Ou ainda, quando uma responsabilidade nova não se encaixa em nenhuma das Entidades já definidas.

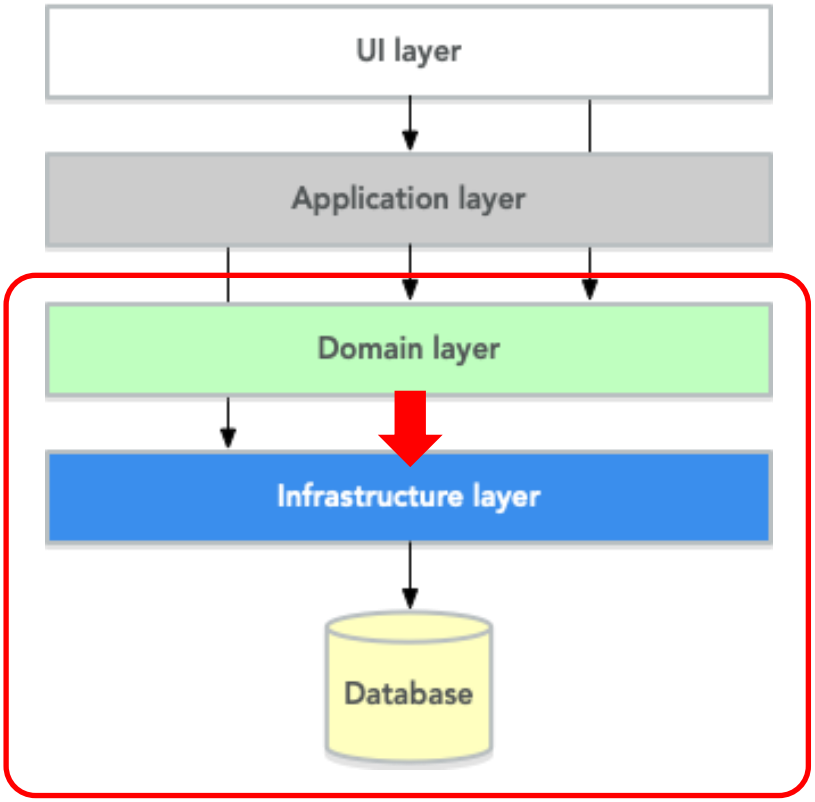
**** Primeiro, iremos criar uma interface para cada DomainService:**




```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using UsuariosWeb.Domain.Entities;

namespace UsuariosWeb.Domain.Interfaces.Services
{
    /// <summary>
    /// Interface para regras de negócio da aplicação
    /// </summary>
    public interface IUsuarioDomainService
    {
        /// <summary>
        /// Método para realização do cadastro do usuário
        /// </summary>
        void CadastrarUsuario(Usuario usuario);

        /// <summary>
        /// Método para fazer a verificação de autenticação do usuário
        /// </summary>
        Usuario AutenticarUsuario(string email, string senha);
    }
}
```

Implementando a interface:
/Services/UsuarioDomainService.cs

** A camada de domínio irá depender do repositório para poder fazer as suas operações.




```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using UsuariosWeb.Domain.Entities;
using UsuariosWeb.Domain.Interfaces.Repositories;
using UsuariosWeb.Domain.Interfaces.Services;

namespace UsuariosWeb.Domain.Services
{
    /// <summary>
    /// Classe para implementação das regras de negócio de usuário
    /// </summary>
    public class UsuarioDomainService : IUsuarioDomainService
    {
        //declarar atributos para utilizarmos os repositorios
        private readonly IUsuarioRepository _usuarioRepository;
        private readonly IPerfilRepository _perfilRepository;

        //construtor para fazer a injeção
       //de dependência (inicialização) dos repositorios
```

```
        public UsuarioDomainService(IUsuarioRepository usuarioRepository,
                                    IPerfilRepository perfilRepository)
        {
            _usuarioRepository = usuarioRepository;
            _perfilRepository = perfilRepository;
        }

        public void CadastrarUsuario(Usuario usuario)
        {
            throw new NotImplementedException();
        }

        public Usuario AutenticarUsuario(string email, string senha)
        {
            throw new NotImplementedException();
        }
    }
}
```

Desenvolvendo a regra de negócio para cadastro do usuário:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using UsuariosWeb.Domain.Entities;
using UsuariosWeb.Domain.Interfaces.Repositories;
using UsuariosWeb.Domain.Interfaces.Services;

namespace UsuariosWeb.Domain.Services
{
    /// <summary>
    /// Classe para implementação das regras de negócio de usuário
    /// </summary>
    public class UsuarioDomainService : IUsuarioDomainService
    {
        //declarar atributos para utilizarmos os repositorios
        private readonly IUsuarioRepository _usuarioRepository;
        private readonly IPerfilRepository _perfilRepository;

        //construtor para fazer a injeção de dependência
       //(inicialização) dos repositorios
        public UsuarioDomainService(
                    IUsuarioRepository usuarioRepository,
                    IPerfilRepository perfilRepository)
        {
            _usuarioRepository = usuarioRepository;
            _perfilRepository = perfilRepository;
        }
```

```
        public void CadastrarUsuario(Usuario usuario)
        {
            //verificar se o email já está cadastrado no banco de dados
            //REGRA: Não permitir o cadastro de usuários com o mesmo email

            if(_usuarioRepository.Obter(usuario.Email) != null)
                //retornar um erro
                throw new Exception($"O email '{usuario.Email}'
                          já está cadastrado na aplicação.");

            //REGRA: Todo usuário cadastrado na aplicação deverá
            //ter o perfil 'default' como padrão
            var perfil = _perfilRepository.Obter("default");
            usuario.IdPerfil = perfil.IdPerfil;

            //cadastrando o usuário
            _usuarioRepository.Inserir(usuario);
        }

        public Usuario AutenticarUsuario(string email, string senha)
        {
            throw new NotImplementedException();
        }
    }
}
```

Continua...