02

Layout e cadastro de usuários

Layout da aplicação

Agora que já fizemos a configuração inicial que será utilizada para a aplicação, vamos começar a trabalhar no layout que será utilizado, para isso modificaremos o código que está no arquivo `Views/Shared/_Layout.cshtml` :

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="~/Content/bootstrap.css" rel="stylesheet" />
    <link href="~/Content/Site.css" rel="stylesheet" />
    <title>@ViewBag.Title - My ASP.NET Application</title>
</head>
<body>
    @Html.Partial("_MenuSuperior")
    <div class="container">
        <main class="col-sm-8">
            @RenderBody()
        </main>
    </div>
</body>
</html>
```

O menu superior da aplicação será criado dentro de uma view parcial do projeto chamada `_MenuSuperior.cshtml` que ficará dentro da pasta `Views/Shared` :

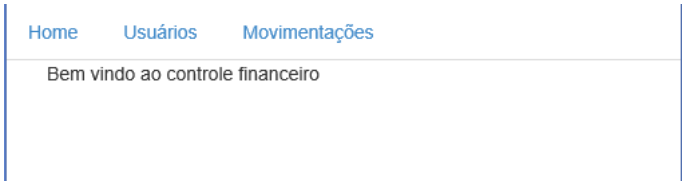
```
<nav>
    <div class="container-fluid">
        <ul class="nav nav-tabs">
            <li>@Html.ActionLink("Home", "Index", "Home")</li>
            <li>@Html.ActionLink("Usuários", "Index", "Usuario")</li>
            <li>@Html.ActionLink("Movimentações", "Index", "Movimentacao")</li>
        </ul>
    </div>
</nav>
```

Além disso, vamos também modificar o arquivo de estilo padrão da aplicação (`Content/Site.css`), substituindo seu conteúdo com o seguinte código:

```
input{
    margin: 0.5em;
}

input[type=submit]{
    background-color: rgb(66,139,202);
    border:0;
    padding: .5em;
```

```
    color: white;
}

select {
    margin: 0.5em 0.5em 0;
    padding: .5em;
}

.field-validation-error {
    padding: 0.5em;
    width: 100%;
    display: block;
    color: red;
}

.input-validation-error{
    background-color: #ffd6d6;
}

aside ul{
    padding: 0;
}

aside ul li{
    list-style-type: none;
    margin: .5em 1em;
}
```

A página inicial da aplicação deve ficar parecida com a abaixo:



Criando o DAO e configurando o Ninject

Agora que já criamos o layout da aplicação, estamos interessados implementar o cadastro e a lista de usuários da aplicação. Para isso, precisamos inicialmente implementar a classe que será responsável por acessar a tabela de usuários do banco de dados, o `UsuarioDAO`, que será criado dentro da pasta DAO:

```
public class UsuarioDAO
{
    public void Adiciona(Usuario usuario)
    {
        // implementação
    }

    public IList<Usuario> Lista()
    {
        // implementação
    }
}
```

Mas na implementação dos métodos desse DAO, precisaremos do contexto do Entity Framework, para conseguirmos o contexto, podemos utilizar novamente a injeção de dependências, assim como fizemos no curso de Entity Framework. Para isso, utilizaremos novamente o Nuget para instalar o plugin Ninject.MVC3:

```
Install-Package Ninject.MVC3 -Version 3.2.1
```

Agora precisamos configurar o Ninject para fazer com que o contexto do Entity Framework seja criado no máximo uma vez por requisição web. Para isso implementaremos o método `RegisterServices` do arquivo `App_Start/NinjectWebCommon.cs`:

```
private static void RegisterServices(IKernel kernel)
{
    kernel.Bind<FinancasContext>().ToSelf().InRequestScope();
}
```

Depois de configurarmos Ninject, podemos pedir a instância do contexto do Entity Framework como argumento do construtor do DAO:

```
public class UsuarioDAO
{
    private FinancasContext context;

    public UsuarioDAO(FinancasContext context)
    {
        this.context = context;
    }
}
```

Utilizando o contexto injetado no construtor, podemos implementar os métodos `Adiciona` e `Lista` do `UsuarioDAO`:

```
public class UsuarioDAO
{
    private FinancasContext context;
    // código do construtor da classe

    public void Adiciona(Usuario usuario)
    {
        context.Usuarios.Add(usuario);
        context.SaveChanges();
    }
    public IList<Usuario> Lista()
    {
        return context.Usuarios.ToList();
    }
}
```

Para finalizar, vamos criar o controller que cuidará do cadastro e listagem de usuários, o `UsuarioController`:

```
public class UsuarioController : Controller
{
```

```
    private UsuarioDAO usuarioDAO;
    public UsuarioController(UsuarioDAO usuarioDAO)
    {
        this.usuarioDAO = usuarioDAO;
    }
}
```

Dentro desse controller, colocaremos um novo método chamado `Form` que mostrará o formulário de cadastro para o usuário:

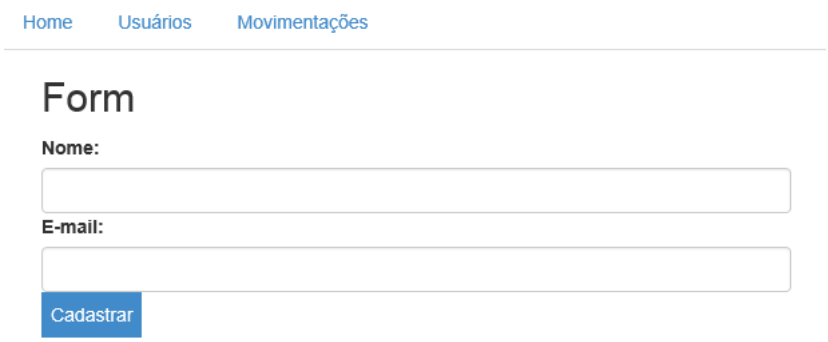
```
public ActionResult Form()
{
    return View();
}
```

Para implementarmos o código Html do formulário de cadastro, utilizaremos os métodos do `HtmlHelper` :

```
@model Financas.Entidades.Usuario

@using(Html.BeginForm("Adiciona", "Usuario", FormMethod.Post))
{
    @Html.LabelFor(u => u.Nome, "Nome")
    @Html.TextBoxFor(u => u.Nome, new { @class = "form-control" })
    @Html.ValidationMessageFor(u => u.Nome)

    @Html.LabelFor(u => u.Email, "E-mail")
    @Html.TextBoxFor(u => u.Email, new { @class = "form-control" })
    @Html.ValidationMessageFor(u => u.Email)

    <input type="submit" value="Cadastrar"/>
}
```

Agora implementaremos o método `Adiciona` dentro do `UsuarioController` recebendo o usuário do formulário de cadastro:

```
public class UsuarioController : Controller
{
    private UsuarioDAO usuarioDAO;
    // resto do código da classe

    public ActionResult Adiciona(Usuario usuario)
    {
        if(ModelState.IsValid)
        {
            usuarioDAO.Adiciona(usuario);
            return RedirectToAction("Index");
        }
        else
        {
            return View("Form", usuario);
        }
    }
}
```



Para terminar, vamos fazer com que a action `Index` do controller liste os usuários do banco de dados:

```
public ActionResult Index()
{
    return View(usuarioDAO.Lista());
}
```

A view para essa action mostrará os itens da lista utilizando uma tabela do Html:

```
@model IList<Financas.Entidades.Usuario>

@Html.ActionLink("Novo usuário", "Form")

<table class="table table-hover">
    <thead>
        <tr>
            <th>Id</th>
            <th>Nome</th>
            <th>E-mail</th>
        </tr>
    </thead>
    <tbody>
        @foreach(var u in Model)
        {
            <tr>
                <td>@u.Id</td>
                <td>@u.Nome</td>
                <td>@u.Email</td>
            </tr>
        }
    </tbody>
</table>
```