ETEC PROF. HORÁCIO AUGUSTO DA SILVEIRA

DESENVOLVIMENTO DE SISTEMAS – 2023/1

Luiz Fernando Souza / Quitéria Danno

Aula 18 – Bibliotecas Javascript AJAX - Partial Views – Modal interno

1. Abra a classe **UsuarioViewModel** e crie a propriedade a seguir

```
public DateTime? DataAcesso { get; set; }
```

2. Crie os links abaixo na view _layout (Views/Shared/_layout.cshtml) para abrir a listagem de disputas que programaremos mais a frente e a página de informações do usuário

```
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Disputas"
        asp-action="IndexDisputas">Disputas</a>
</li>
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Usuarios"
        asp-action="IndexInformacoes">Informações do Usuário</a>
</li>
```

3. Altere o método **AutenticacaoAsync** na Controller de Usuário, acrescentando o trecho onde o login do usuário também ficará guardado em uma variável de sessão

```
if (response.StatusCode == System.Net.HttpStatusCode.OK)
{
    HttpContext.Session.SetString("SessionTokenUsuario", serialized);

    HttpContext.Session.SetString("SessionUsername", u.Username);

    TempData["Mensagem"] = string.Format("Bem-vindo {0}!!!", u.Username);
    return RedirectToAction("Index", "Personagens");
}
```

4. Ainda na controller de usuário, adicione o método para buscar informações do usuário através do login

```
[HttpGet]
        public async Task<ActionResult> IndexInformacoesAsync()
        {
            try
            {
                HttpClient httpClient = new HttpClient();

                //Novo: Recuperação informação da sessão
                string login = HttpContext.Session.GetString("SessionUsername");
                string uriComplementar =
$"GetByLogin/{login}";
                HttpResponseMessage response = await httpClient.GetAsync(uriBase +
uriComplementar);
                string serialized = await response.Content.ReadAsStringAsync();

                if (response.StatusCode == System.Net.HttpStatusCode.OK)
                {
                    UsuarioViewModel u = await Task.Run(() =>
JsonConvert.DeserializeObject<UsuarioViewModel>(serialized));
                    return View(u);
                }
                else
                {
                    throw new System.Exception(serialized);
                }
            }
            catch (System.Exception ex)
            {
                TempData["MensagemErro"] = ex.Message;
                return RedirectToAction("Index");
            }
        }
```

5. Clique com o direito em Views/Usuarios, crie o arquivo **IndexInformacoes.cshtml** e adicione o html abaixo

```
@model RpgMvc.Models.UsuarioViewModel
@{
    ViewBag.Title = "Informações do Usuário";
}
@if (@TempData["Mensagem"] != null)
{
    <div class="alert alert-success" role="alert">@TempData["Mensagem"]</div>
}
<!--Configuração para exibir mensagem de erro -->
@if (@TempData["MensagemErro"] != null)
{
    <div class="alert alert-danger" role="alert">@TempData["MensagemErro"]</div>
}
<h2>Detalhes do Usuário</h2>
<div>
    <hr />
    <dl class="dl-horizontal">
        <dt>@Html.DisplayNameFor(model => model.Id)</dt>
        <dd>@Html.DisplayFor(model => model.Id)</dd>

        <dt>@Html.DisplayNameFor(model => model.Username)</dt>
        <dd>@Html.DisplayFor(model => model.Username)</dd>

        <dt>@Html.DisplayNameFor(model => model.Perfil)</dt>
        <dd>@Html.DisplayFor(model => model.Perfil)</dd>

        <dt>@Html.DisplayNameFor(model => model.Email)</dt>
        <dd>@Html.DisplayFor(model => model.Email)</dd>

        <dt>@Html.DisplayNameFor(model => model.DataAcesso)</dt>
        <dd>@Html.DisplayFor(model => model.DataAcesso)</dd>
    </dl>
    <button onclick="AlterarEmail();" type="button" class="btn btn-primary">Alterar E-
mail</button>
```

- Execute para atestar que a página está funcionando

6. No final da View **IndexInformacoes**, acrescente o html que servirá como um modal

```
<!--Conteúdo do modal popup-->
<div class="modal fade bd-example-modal-lg" id="modalInterno" tabindex="-1" role="dialog"
aria-labelledby="modalInternoLabel" aria-hidden="true">
    <div class="modal-dialog modal-lg" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="modalInternoLabel">Alteração de E-mail</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
            </div>
            <div class="modal-body">
                @using (Html.BeginForm("AlterarEmail", "Usuarios", FormMethod.Post))
                {
                    <div class="form-horizontal">
                        <div class="form-group">
                            @Html.LabelFor(model => model.Id, htmlAttributes: new { @class =
"control-label col-md-2" })
                            <div class="col-md-6">
                                @Html.EditorFor(model => model.Id, new { htmlAttributes = new
{ @class = "form-control",
                            @readonly = "readonly" }})
                            </div>
                        </div><br />
                        <div class="form-group">
                            @Html.LabelFor(model => model.Email, htmlAttributes: new { @class
= "control-label col-md-2" })
                            <div class="col-md-6">
                                @Html.EditorFor(model => model.Email, new { htmlAttributes =
new { @class = "form-control" }})
                            </div>
                        </div><br />
                        <div class="form-group">
                            <div class="col-md-offset-2 col-md-6">
                                <input type="submit" value="Salvar" class="btn btn-primary" />
                            </div>
                        </div>
                    </div>
                }
            </div>
        </div>
    </div>
</div>
```

7. Ainda na view **IndexInformacoes**, crie uma área de scripts após o html e adicione uma função para que o modal apareça

```
<script type="text/javascript">

    function AlterarEmail() {
        $("#modalInterno").modal("show");
    }

</script>
```

8. Volte até a controller de usuário e crie um método para alterar o e-mail

```
[HttpPost]
0 references
public async Task<ActionResult> AlterarEmail(UsuarioViewModel u)
{
    try
    {
        HttpClient httpClient = new HttpClient();
        string token = HttpContext.Session.GetString("SessionTokenUsuario");
        httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);

        string uriComplementar = "AtualizarEmail";
        var content = new StringContent(JsonConvert.SerializeObject(u));
        content.Headers.ContentType = new MediaTypeHeaderValue("application/json");

        HttpResponseMessage response = await httpClient.PutAsync(uriBase + uriComplementar, content);
        string serialized = await response.Content.ReadAsStringAsync();

        if (response.StatusCode == System.Net.HttpStatusCode.OK)
            TempData["Mensagem"] = "E-mail alterado com sucesso.";
        else
            throw new System.Exception(serialized);
    }
    catch (System.Exception ex)
    {
        TempData["MensagemErro"] = ex.Message;

    }
    return RedirectToAction("IndexInformacoes");
}
```

ETEC PROF. HORÁCIO AUGUSTO DA SILVEIRA

DESENVOLVIMENTO DE SISTEMAS – 2023/1

Luiz Fernando Souza / Quitéria Danno

Aula 18 - Partial Views – Modais externos

São ideais para cenários de reusabilidade de código. Com este recurso podemos criar leiautes que podem ser chamados a partir de outras views, o padrão de criação conterá o _ no início do nome da view

1. Abra a Controller de Usuários e crie o método Get que trará a informação do usuário logado

```
 [HttpGet]

        public async Task<ActionResult> ObterDadosAlteracaoSenha()
        {
            UsuarioViewModel viewModel = new UsuarioViewModel();
            try
            {
                HttpClient httpClient = new HttpClient();
                string login = HttpContext.Session.GetString("SessionUsername");
                string uriComplementar = $"GetByLogin/{login}";
                HttpResponseMessage response = await httpClient.GetAsync(uriBase +
uriComplementar);
                string serialized = await response.Content.ReadAsStringAsync();

                TempData["TituloModalExterno"] = "Alteração de Senha";

                if (response.StatusCode == System.Net.HttpStatusCode.OK)
                {
                    viewModel = await Task.Run(() =>
JsonConvert.DeserializeObject<UsuarioViewModel>(serialized));
                    return PartialView("_AlteracaoSenha", viewModel);
                }
                else
                    throw new System.Exception(serialized);

            }
            catch (System.Exception ex)
            {
                TempData["MensagemErro"] = ex.Message;
                return RedirectToAction("IndexInformacoes");
            }
        }
```

2. Abra a View **Usuarios/IndexInformacoes.cshtml** e adicione o botão que chamará o modal através de uma função Javascript. O botão ficará próximo ao que abre o modal de alteração de e-mail

```
<button onclick=" AbrirModalAlteracaoSenha();" type="button" class="btn btn-primary">
    Alterar Senha
</button>
```

3. Ainda na view **IndexInformacoes.cshtml**, programe no bloco de funções Javascript a abertura do modal

```
function AbrirModalAlteracaoSenha() {
        $.ajax({

            url: "@Url.Action("ObterDadosAlteracaoSenha", "Usuarios")",
            //data: { nomeProp1: valorProp1, nomeProp2: valorProp2,  },
            success: function (retorno) {
                $("#modalExterno .modal-dialog .modal-content .modal-body").html(retorno);
                $("#modalExterno").modal("show");
            }
        });
    }
```

4. Inclua abaixo do modal interno as divs referente ao modal externo

```
<!-- Modal externo -->
<div class="modal" id="modalExterno" tabindex="-1" role="dialog" aria-
labelledby="modalExternoLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="modalExternoLabel">
                    @TempData["TituloModalExterno"]
                </h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close">
                </button>
            </div>
            <div class="modal-body">
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Fechar</button>
            </div>
        </div>
    </div>
</div>
```

- Perceba que neste caso não existe campos criados dentro da div modal body, pois faremos com que o conteúdo html da partial view se acople a esta div.

5. Clique com o direito na pasta Views/Usuarios e crie o arquivo **_AlteracaoSenha.cshtml**, que será uma partial view

```
@model RpgMvc.Models.UsuarioViewModel
@{
    Layout = ""; //Fará com que a view não herde o leiaute padrão com menus e rodapés
}
@using (Html.BeginForm("AlterarSenha", "Usuarios", FormMethod.Post, new { id = "formSenha" }))
{
    <div class="form-horizontal">
        <div class="form-group">
            @Html.LabelFor(model => model.Id, htmlAttributes: new { @class = "control-label
col-md-2" })
            <div class="col-md-6">
                @Html.EditorFor(model => model.Id, new { htmlAttributes = new { @class =
"form-control", @readonly =
            "readonly" }})
            </div>
        </div><br/>
        <div class="form-group">
            @Html.LabelFor(model => model.PasswordString, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-6">
                @Html.EditorFor(model => model.PasswordString, new { htmlAttributes = new {
@class = "form-control" }})
            </div>
        </div><br/>
        <div class="form-group">
            <div class="col-md-offset-2 col-md-6">
                <input onclick="AlterarSenha();" type="submit" value="Salvar" class="btn btn-
primary" />
            </div>
        </div>
    </div>
}
```

- O botão de salvamento será acionado através da função Javascript “AlterarSenha” que será definida na próxima etapa.

6. Abaixo do hmtl da partial view **_AlteracaoSenha.cshtml**, realize a inserção da função Javascript que será responsável por acionar a controller para salvar os dados e depois exibirá uma mensagem ao usuário e por fim, vai atualizar ao chamar o método IndexInformacoes da controller de usuários.

```
<script type="text/javascript">

    function AlterarSenha() {

        var url = "@Url.Action("AlterarSenha", "Usuarios")";
        var valdata = $("#formSenha").serialize();

        $.ajax({
            url: url,
            type: "POST",
            dataType: 'json',
            contentType: 'application/x-www-form-urlencoded; charset=UTF-8',
            //data: { nomeProp1: valorProp1, nomeProp2: valorProp2,  },
            data: valdata,
            success: function (retorno) {
                alert(retorno);
                window.location.href = "@Url.Action("IndexInformacoes", "Usuarios")";
            }
        });
    }
</script>
```

7. Insira na controller de usuários o método que vai fazer a alteração da senha

```
[HttpPost]
        public async Task<ActionResult> AlterarSenha(UsuarioViewModel u)
        {
            try
            {
                HttpClient httpClient = new HttpClient();
                string token = HttpContext.Session.GetString("SessionTokenUsuario");
                httpClient.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
                string uriComplementar = "AlterarSenha";
                u.Username = HttpContext.Session.GetString("SessionUsername");

                var content = new StringContent(JsonConvert.SerializeObject(u));
                content.Headers.ContentType = new MediaTypeHeaderValue("application/json");
                HttpResponseMessage response = await httpClient.PutAsync(uriBase +
uriComplementar, content);
                string serialized = await response.Content.ReadAsStringAsync();

                if (response.StatusCode == System.Net.HttpStatusCode.OK)
                {
                    string mensagem = "Senha alterada com sucesso.";
                    TempData["Mensagem"] = mensagem; //Mensagem guardada do TempData que
aparcerá na página pai do modal
                    return Json(mensagem); //Mensagem que será exibida no alert da Função que
chamou este método
                }
                else
                    throw new System.Exception(serialized);
            }
            catch (System.Exception ex)
            {
                return Json(ex.Message);
            }
        }
```