ETEC PROF. HORÁCIO AUGUSTO DA SILVEIRA

DESENVOLVIMENTO DE SISTEMAS – 2023/1

Luiz Fernando Souza / Quitéria Danno

Aula 19 - Salvando a foto do usuário

1. Abra a view **Usuarios/IndexInformacoes.cshtml** e adicione o leiaute a seguir abaixo da exibição da data de acesso. Veja que temos um elemento html image que carregará a imagem através da conversão dos dados da propriedade Foto, além de um link para download.

```
@if(Model.Foto != null)
        {
            var base64 = Convert.ToBase64String(Model.Foto);
            var imgSrc = String.Format("{0}", base64);

            <dt>@Html.DisplayNameFor(model => model.Foto)</dt>
            <img src="@imgSrc" title="@Model.Username" width="100" height="132" />
            <br/>
            @Html.ActionLink("Baixar Foto", "BaixarFoto", "Usuarios",  null, new { @title =
"Baixar foto" })
        }
```

2. Insira abaixo do botão de alteração da senha o trecho responsável pelo upload da foto.

```
<hr/>
    @using (Html.BeginForm("EnviarFoto", "Usuarios", FormMethod.Post, new { enctype =
"multipart/form-data", id = "formFoto" }))
    {
        <div class="row">
            <div class="col-sm-3">
                  <h3>Envio de foto</h3>
             </div>
        </div>
        <div class="row">
            <div class="col-sm-3">
                @Html.TextBox("file", "", new { type = "file" })
                @Html.HiddenFor(model => model.Id, new { htmlAttributes = new { @class =
"form-control" } })
            </div>
            <div class="col-sm-3">
                <button id="FotoSubmit" type="submit" value="Enviar Foto" class="btn btn-
primary">
                    Enviar Foto
                </button>
            </div>
        </div>
    }
    <hr>
```

3. Abra a controller de usuário e programe o método que fará o upload da foto. O trecho comentado não precisa ser feito, pois é um exemplo para caso fosse necessário salvar a imagem em uma pasta do Sistema Operacional ou na rede.

```
[HttpPost]
0 references
public async Task<ActionResult> EnviarFoto(UsuarioViewModel u)
{
    try
    {
        if (Request.Form.Files.Count == 0)
            throw new System.Exception("Selecione o arquivo.");
        else
        {
            var file = Request.Form.Files[0];
            var fileName = Path.GetFileName(file.FileName);
            string nomeArquivoSemExtensao = Path.GetFileNameWithoutExtension(fileName);
            var extensao = Path.GetExtension(fileName);

            if (extensao != ".jpg" && extensao != "jpeg" && extensao != ".png")
                throw new System.Exception("O Arquivo selecionado não é uma foto.");

            //var pastaUpload = @"\" + "";
            //var path = Path.Combine(pastaUpload, fileName);
            using (var ms = new MemoryStream())
            {
                file.CopyTo(ms);
                u.Foto = ms.ToArray();
                //string s = Convert.ToBase64String(fileBytes);//Escrever bytes numa string
                //System.IO.File.WriteAllBytes(path, ms.ToArray()); //Escrever arquivo em uma pasta
            }
        }
        //Próximo trecho referente ao envio para a API será aqui
    }
    catch (System.Exception ex)
    {
        TempData["MensagemErro"] = ex.Message;
    }
    return RedirectToAction("IndexInformacoes");
}
```

4. Insira antes do fechamento do bloco try do método programado anteriormente, a sintaxe referente ao envio dos dados para a API.

```
HttpClient httpClient = new HttpClient();
                string token = HttpContext.Session.GetString("SessionTokenUsuario");
                httpClient.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);

                string uriComplementar = "AtualizarFoto";
                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"] = "Foto enviada com sucesso";
                else
                    throw new System.Exception(serialized);
```

- Tente realizar o salvamento de uma imagem no formato png

5. Insira o método responsável por baixar a foto

```
 [HttpGet]
        public async Task<ActionResult> BaixarFoto()
        {
            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();

                if (response.StatusCode == System.Net.HttpStatusCode.OK)
                {
                    UsuarioViewModel viewModel = await
                        Task.Run(() =>
JsonConvert.DeserializeObject<UsuarioViewModel>(serialized));

                    //string contentType = "application/image";
                    string contentType = System.Net.Mime.MediaTypeNames.Application.Octet;

                    byte[] fileBytes = viewModel.Foto;
                    string fileName =
$"Foto{viewModel.Username}_{DateTime.Now:ddMMyyyyHHmmss}.png"; // + extensao;
                    return File(fileBytes, contentType, fileName);
                }
                else
                    throw new System.Exception(serialized);
            }
            catch (System.Exception ex)
            {
                TempData["MensagemErro"] = ex.Message;
                return RedirectToAction("IndexInformacoes");
            }
        }
```

Luiz Fernando Souza / Quitéria Danno

Aula 19 - Gerenciando a Sessão e Perfil do usuário

1. Abra a controller de Usuário e modifique o método de autenticação, adicionando antes da mensagem de boas-vindas o trecho que guardará o id e o perfil do usuário em mais duas sessões:

```
if (response.StatusCode == System.Net.HttpStatusCode.OK)
{
    UsuarioViewModel uLogado = JsonConvert.DeserializeObject<UsuarioViewModel>(serialized);

    HttpContext.Session.SetString("SessionTokenUsuario", uLogado.Token);
    HttpContext.Session.SetString("SessionUsername", uLogado.Username);

    HttpContext.Session.SetString("SessionPerfilUsuario", uLogado.Perfil);
    HttpContext.Session.SetString("SessionIdUsuario", uLogado.Id.ToString());

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

2. Abra a view **_Layout.cshtml** e insira o bloco @{ ... } conforme a seguir. A abertura de chave ficará imediatamente depois da abertura do header, e o fechamento da chave ficará antes do fechamento da tag header. Com esse bloco será possível realizar programação C# dentro do arquivo cshtml.

```
14      <header>
15          @{
16  >           <nav class="navbar navbar-expand-sm navbar-toggleable-sm navba
48              </nav>
49          }
50      </header>
```

3. Insira a programação abaixo da abertura da chave que será responsável por recuperar os dados da sessão

```
<header>
    @{
        var nomeSessao = new Byte[20];
        bool nomeOK = Context.Session.TryGetValue("SessionIdUsuario", out nomeSessao);
        string sessao = string.Empty;

        var nomeSessaoPerfil = new Byte[20];
        bool perfilOK = Context.Session.TryGetValue("SessionPerfilUsuario", out nomeSessaoPerfil);
        string sessaoPerfil = string.Empty;

        if (nomeOK)
            sessao = System.Text.Encoding.UTF8.GetString(nomeSessao);

        if (perfilOK)
            sessaoPerfil = System.Text.Encoding.UTF8.GetString(nomeSessaoPerfil);

        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
```

4. Procure a ***ul*** que apresenta os itens de menu e faça a verificação das sessões. Nestes casos, o menu ficará apresentável apenas se a sessão não estiver vazia e o item disputas ficará disponível apenas se o usuário for ***Admin***

```
<ul class="navbar-nav flex-grow-1">
    @if (!string.IsNullOrEmpty(sessao))
    {
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-actic
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Personagens"
        asp-action="Index">Personagens</a>
        </li>

        @if (sessaoPerfil == "Admin")
        {
            <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>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home"
        asp-action="Privacy">Privacy</a>
        </li>
    }
</ul>
```

5. Adicione também uma verificação para apresentar o menu “sair” ou a partial view de registro e login, de acordo com o estado da sessão.

```
            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="" asp-controller="Home"
            asp-action="Privacy">Privacy</a>
            </li>
        }
    </ul>
    @if (string.IsNullOrEmpty(sessao))
    {
        <partial name="_LoginPartialRpg.cshtml" />
    }
    else
    {
        <a class="" asp-area="" asp-controller="Usuarios" asp-action="Sair">Sair</a>
    }
</div>
```

6. Abra a controller de usuário e faça a programação do método sair. Perceba que neste trecho, as sessões criadas são removidas e o usuário será direcionado para a página inicial.

```
[HttpGet]
0 references
public ActionResult Sair()
{
    try
    {
        HttpContext.Session.Remove("SessionTokenUsuario");
        HttpContext.Session.Remove("SessionUsername");
        HttpContext.Session.Remove("SessionPerfilUsuario");
        HttpContext.Session.Remove("SessionIdUsuario");

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

7. Abra a controller de personagens, procure o método IndexAsync e comente a string urlComplementar, deixando a programação como sinalizada a seguir. O intuito é verificar se o usuário for administrador exibir todos os personagens, caso contrário, exibir apenas que pertence a ele. Secundariamente, estamos enviando uma ViewBag com o perfil do usuário para a View. Futuramente vamos fazer uma comparação para exibir ou não determinados links do personagem.

```
if (response.StatusCode == System.Net.HttpStatusCode.OK)
{
    UsuarioViewModel uLogado = JsonConvert.DeserializeObject<UsuarioViewModel>(serialized);

    HttpContext.Session.SetString("SessionTokenUsuario", uLogado.Token);
    HttpContext.Session.SetString("SessionUsername", uLogado.Username);

    HttpContext.Session.SetString("SessionPerfilUsuario", uLogado.Perfil);
    HttpContext.Session.SetString("SessionIdUsuario", uLogado.Id.ToString());

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

8. Para não permitir que um usuário logado acesse os métodos Get das controllers, podemos fazer uma verificação de sessão como a que está abaixo, feita no início do bloco try do método Index da classe PersonagensController.cs. Neste caso se a sessão chamda de SessionIdUsuario estiver nula ou vazia, o usuário será direcionado para o método sair, consequentemente voltando para a tela de login.

```
[HttpGet]
0 references
public async Task<ActionResult> IndexAsync()
{
    try
    {
        if(string.IsNullOrEmpty(HttpContext.Session.GetString("SessionIdUsuario")))
        {
            return RedirectToAction("Sair", "Usuarios");
        }
```