
Aula 16 - Configuração da página de disputas

1. Crie o arquivo IndexDisputas.cshtml dentro da pasta Views/Disputas e insira a codificação abaixo

```
<!--Namespace da classe de Modelo para esta view-->
@model IEnumerable<RpgMvc.Models.DisputaViewModel>

@if (@TempData["Mensagem"] != null)
{
    <div class="alert alert-success" role="alert">
       @Html.Raw(@TempData["Mensagem"])</div>
}
<!--Configuração para exibir mensagem de erro -->
@if (@TempData["MensagemErro"] != null)
{
    <div class="alert alert-danger" role="alert">
        @TempData["MensagemErro"]</div>
}
@{ViewBag.Title = "Disputas"; }<!--Título da página para o navegador-->
<h2>Relação de Disputas</h2><!--Título da página-->
<table class="table">
    <tr><!--Títulos das colunas da tabela-->
        <th>@Html.DisplayNameFor(model => model.Id)</th>
        <th>@Html.DisplayNameFor(model => model.DataDisputa)</th>
        <th>@Html.DisplayNameFor(model => model.Narracao)</th>
        <th>@Html.DisplayNameFor(model => model.Resultados)</th>
        <th></th>
    </tr>
    <!--Looping para escrever os dados na tabela-->
    @foreach (var item in Model)
    {
        <tr>
            <td>@Html.DisplayFor(modelItem => item.Id)</td>
            <td>@Html.DisplayFor(modelItem => item.DataDisputa)</td>
            <td>@Html.DisplayFor(modelItem => item.Narracao)</td>
            <td>@Html.DisplayFor(modelItem => item.Resultados)</td>
        </tr>
    }
</table>
@Html.ActionLink("Apagar Disputas" ,"ApagarDisputas" ,"Disputas", null
                    ,new { @class = "btn btn-danger", onclick = "return confirm('Deseja
realmente apagar as disputas?');"})
```

2. Abra a classe Disputas Controller e adicione o método IndexDisputas.

```
[HttpGet]
        public async Task<ActionResult> IndexDisputasAsync()
        {
            try
            {
                string uriComplementar = "Listar";

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

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

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

                    return View(lista);
                }
                else
                    throw new System.Exception(serialized);
            }
            catch (System.Exception ex)
            {
                TempData["MensagemErro"] = ex.Message;
                return RedirectToAction("Index");
            }
        }
```

3. Ainda na Controller DisputasController adicione o método ApagarDisputas.

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

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

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

                if (response.StatusCode == System.Net.HttpStatusCode.OK)
                    TempData["Mensagem"] = "Disputas apagadas com sucessso.";
                else
                    throw new System.Exception(serialized);
            }
            catch (System.Exception ex)
            {
                TempData["MensagemErro"] = ex.Message;
            }
            return RedirectToAction("IndexDisputas", "Disputas");
        }
```

4. Abra o arquivo de layout Shared/_Layout.cshtml para adicionar o menu para as disputas. Execute o projeto para confirmar que as disputas serão listadas.

```
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Personagens"
        asp-action="Index">Personagens</a>
</li>

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

ETEC PROF. HORÁCIO AUGUSTO DA SILVEIRA

DESENVOLVIMENTO DE SISTEMAS – 2023/1

Luiz Fernando Souza / Quitéria Danno

Criação de Models, Controllers e Views para PersonagemHabilidades

1. Crie uma classe chamada **PersonagemHabilidadesViewModel.cs** conforme abaixo

```
namespace RpgMvc.Models
{
    public class PersonagemHabilidadeViewModel
    {
        public int PersonagemId { get; set; }
        public PersonagemViewModel Personagem { get; set; }
        public int HabilidadeId { get; set; }
        public HabilidadeViewModel Habilidade { get; set; }
    }
}
```

2. Acrescente uma lista de PersonagemHabilidade como propriedade na classe **HabilidadeViewModel** e na classe **PersonagemViewModel**

```
public List<PersonagemHabilidadeViewModel> PersonagemHabilidades {get; set;}
```

3. Crie a controller PersonagemHabilidadesController dentro da pasta Controllers

```
namespace RpgMvc.Controllers
{
    public class PersonagemHabilidadesController : Controller
    {
         public string uriBase = "http://xyz.somee.com/RpgApi/PersonagemHabilidades/";
        //xyz será substituído pelo nome do seu site na API.
    }
}
```

4. Crie o método para consumir a API que retornará a listagem de habilidades

```
[HttpGet("PersonagemHabilidades/{id}")]
0 references
public async Task<ActionResult> IndexAsync(int id)
{
    try
    {
        HttpClient httpClient = new HttpClient();
        string token = HttpContext.Session.GetString("SessionTokenUsuario");
        httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);

        HttpResponseMessage response = await httpClient.GetAsync(uriBase + id.ToString());
        string serialized = await response.Content.ReadAsStringAsync();

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

            return View(lista);
        }
        else
            throw new System.Exception(serialized);
    }
    catch (System.Exception ex)
    {
        TempData["MensagemErro"] = ex.Message;
        return RedirectToAction("Index");
    }
}
```

- Faça os seguintes usings no topo da controller: RpgMvc.Models, System.Net.Http.Headers e Newtonsoft.Json .

- Perceba que acabamos de usar o método Get que é chamado com uma rota personalizada. Até então nos outros métodos, as rotas tinham sido padrão.

5. Crie a pasta **PersonagemHabilidades** dentro da pasta Views e dentro da pasta *PersonagemHabilidades* recém-criada crie a view **Index.cshtml** que terá o html/razor abaixo

```
@model IEnumerable<RpgMvc.Models.PersonagemHabilidadeViewModel>
@{ ViewBag.Title = "Index"; }
@if (@TempData["Mensagem"] != null) {
    <div class="alert alert-success" role="alert">
        @TempData["Mensagem"]</div>
}
@if (@TempData["MensagemErro"] != null) {
    <div class="alert alert-danger" role="alert">
        @TempData["MensagemErro"]</div>
}
<h2>Habilidades do Personagem</h2>
<table class="table">
    <tr>
        <th>
            @Html.DisplayName("Personagem")
        </th>
        <th>
            @Html.DisplayName("Habilidade")
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Habilidade.Dano)
        </th>
    </tr>
    @foreach (var item in Model)
    {
        <tr>
            <td>@Html.DisplayFor(modelItem => item.Personagem.Nome) </td>
            <td>@Html.DisplayFor(modelItem => item.Habilidade.Nome) </td>
            <td>@Html.DisplayFor(modelItem => item.Habilidade.Dano) </td>
                <td>@Html.ActionLink("Deletar", "Delete"

       , new {habilidadeId = item.HabilidadeId, personagemId = item.PersonagemId }
       , new { onclick = "return confirm('Deseja realmente deletar esta habilidade ?');"})
            </td>
        </tr>
    }
</table>
<div>
    @Html.ActionLink("Retornar aos Personagens", "Index", "Personagens")
</div>
```

6. Abra a View Index de Personagens e adicione o link para exibir a palavra “Listar Habilidades”, que carregará o método Index da controller de PersonagemHabilidades, passando o Id do personagem

```
@Html.ActionLink("Deletar", "Delete", new { id = item.Id }
    , new { onclick = "return confirm('Deseja realmente deletar este personagem ?');"})

@Html.ActionLink("Listar Habilidades", "Index", "PersonagemHabilidades", new {id = item.Id})
```

7. Crie um método na controller de PersonagemHabilidades para receber os ids da habilidade e do personagem para que a API remova os dados da base.

```
[HttpGet("Delete/{habilidadeId}/{personagemId}")]
0 references
public async Task<ActionResult> DeleteAsync(int habilidadeId, int personagemId)
{
    try
    {
        HttpClient httpClient = new HttpClient();
    A   string uriComplentar = "DeletePersonagemHabilidade";
    B   string token = HttpContext.Session.GetString("SessionTokenUsuario");
        httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);

        PersonagemHabilidadeViewModel ph = new PersonagemHabilidadeViewModel();
    C   ph.HabilidadeId = habilidadeId;
        ph.PersonagemId = personagemId;

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

        if(response.StatusCode == System.Net.HttpStatusCode.OK)
    E       TempData["Mensagem"] = "Habilidade removida com sucesso";
        else
            throw new System.Exception(serialized);
    }
    catch (System.Exception ex)
    {
        TempData["MensagemErro"] = ex.Message;
    }
F   return RedirectToAction("Index", new {Id = personagemId});
}
```

(A) Rota da API sendo guardada para concatenar com o endereço base
(B) Recuperação do Token da sessão e armazenamento dela no httpClient
(C) Objeto sendo preenchido pelo que foi passado nos parâmetros do método
(D) Serialização do objeto, envio para a endereço completo da API e armazenamento do retorno em serialized.
(E) If/Else de acordo com o código de retorno
(F) O retorno do método será para a mesma listagem de habilidades do personagem.

8. Crie mais um link na Index de Personagens

```
@Html.ActionLink("Listar Habilidades", "Index", "PersonagemHabilidades", new {id = item.Id}) |
@Html.ActionLink("Atribuir Habilidade", "Create", "PersonagemHabilidades", new {id = item.Id, nome = item.Nome })
```

- Perceba que estamos coletando o Id e nome do Personagem.

9. Crie o método que vai receber o Id e nome do personagem na controller de PersonagemHabilidades. Este método também consumirá a lista de habilidades na API pra poder apresentar um DropDownList que será carregado através de uma ViewBag.

```
[HttpGet]
0 references
public async Task<ActionResult> CreateAsync(int id, string nome)
{
    try
    {
        string uriComplentar = "GetHabilidades";
        HttpClient httpClient = new HttpClient();
        string token = HttpContext.Session.GetString("SessionTokenUsuario");
        httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);
        HttpResponseMessage response = await httpClient.GetAsync(uriBase + uriComplentar);

        string serialized = await response.Content.ReadAsStringAsync();
        List<HabilidadeViewModel> habilidades = await Task.Run(() =>
        JsonConvert.DeserializeObject<List<HabilidadeViewModel>>(serialized));
        ViewBag.ListaHabilidades = habilidades;

        PersonagemHabilidadeViewModel ph = new PersonagemHabilidadeViewModel();
        ph.Personagem = new PersonagemViewModel();
        ph.Habilidade = new HabilidadeViewModel();
        ph.PersonagemId = id;
        ph.Personagem.Nome = nome;

        return View(ph);
    }
    catch (System.Exception ex)
    {
        TempData["MensagemErro"] = ex.Message;
        return RedirectToAction("Create", new { id, nome });
    }
}
```

10. Crie a view Create.cshtml dentro da pasta PersonagemHabilidades com a codificação abaixo:

```
@model RpgMvc.Models.PersonagemHabilidadeViewModel
@{    ViewBag.Title = "Create";}
<h2>Cadastro de Habilidade para @Model.Personagem.Nome </h2>
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    <div class="form-horizontal">
        <hr />
        <div class="form-group">
            @Html.LabelFor(model => model.Personagem.Id, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-6">
                @Html.EditorFor(model => model.PersonagemId, new { htmlAttributes = new { @class = "form-
control", @readonly = "readonly" } })
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.Personagem, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-6">
@Html.EditorFor(model => model.Personagem.Nome, new { htmlAttributes = new { @class = "form-
control", @readonly = "readonly" } })
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.Habilidade, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-6">
                @Html.DropDownListFor(model => model.HabilidadeId, new SelectList(@ViewBag.ListaHabilidades, "Id", "Nome"),
                    "---Selecione---", 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>
@Html.ActionLink("Retornar às habilidades ", "Index", "PersonagemHabilidades", new{id = Model.PersonagemId}) |
@Html.ActionLink("Retornar aos Personagens", "Index", "Personagens")
</div>
```

- Execute para testar se o cadastro está aparecendo juntamente como DropDownList Carregado. Adicione e remova uma habilidade

11. Contrua o método que vai salvar a habilidade para o personagem. Vamos criar um método Post chamado Create na controller de PersonagemHabilidades.

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

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

        if (response.StatusCode == System.Net.HttpStatusCode.OK)
            TempData["Mensagem"] = "Habilidade cadastrada com sucesso";
        else
            throw new System.Exception(serialized);
    }
    catch (System.Exception ex)
    {
        TempData["MensagemErro"] = ex.Message;
    }
    return RedirectToAction("Index", new { id = ph.PersonagemId});
}
```

- Atualmente temos apenas 3 habilidades, mas você poderia inserir outras na base de dados para que possamos variar bastante na vinculação delas aos personagens.

- Execute e tente realizar o salvamento da habilidade. Perceba que depois de salvar, redirecionaremos o usuário para a controller que lista as habilidades de acordo com o personagem.

12. Abra a view Personagens/Index.cshtml e inclua na tabela de personagens os links a seguir para restaurar pontos e zerar o ranking de um personagem individualmente

```
    @Html.ActionLink("Listar Habilidades", "Index", "PersonagemHabilidades", new {id = item.Id}) |
    @Html.ActionLink("Atribuir Habilidade", "Create", "PersonagemHabilidades", new {id = item.Id, nome =
item.Nome}) |

    @Html.ActionLink("Restaurar Pontos Vida", "RestaurarPontosVida", new { id = item.Id }
        , new { onclick = "return confirm('Deseja realmente restaurar pontos vida ?');"}) |

    @Html.ActionLink("Zerar Ranking", "ZerarRanking", new { id = item.Id }
    , new { onclick = "return confirm('Deseja realmente zerar ranking ?');"})
</td>
```

13. Abra a controller de Personagens e Insira o método que restaurará os pontos de vida de um personagem

```
[HttpGet]
        public async Task<ActionResult> RestaurarPontosVidaAsync(int id)
        {
            try
            {
                string uriComplementar = "RestaurarPontosVida";
                PersonagemViewModel p = new PersonagemViewModel();
                p.Id = id;

                HttpClient httpClient = new HttpClient();
                string token = HttpContext.Session.GetString("SessionTokenUsuario");
                httpClient.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
                var content = new StringContent(JsonConvert.SerializeObject(p));
                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"] = "Pontos de vida do personagem restaurados com
sucesso";
                }
                else
                    throw new System.Exception(serialized);
            }
            catch (System.Exception ex)
            {
                TempData["MensagemErro"] = ex.Message;
            }
            return RedirectToAction("Index");
        }
```

14. Insira o método que vai zerar o ranking de um personagem

```
[HttpGet]
        public async Task<ActionResult> ZerarRankingAsync(int id)
        {
            try
            {
                string uriComplementar = "ZerarRanking";
                PersonagemViewModel p = new PersonagemViewModel();
                p.Id = id;

                HttpClient httpClient = new HttpClient();
                string token = HttpContext.Session.GetString("SessionTokenUsuario");
                httpClient.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
                var content = new StringContent(JsonConvert.SerializeObject(p));
                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"] = "Ranking do personagem zerado com
sucesso";
                }
                else
                    throw new System.Exception(serialized);
            }
            catch (System.Exception ex)
            {
                TempData["MensagemErro"] = ex.Message;
            }
            return RedirectToAction("Index");
        }
```