02

Gerando Relatórios

Relatórios

Para terminarmos nosso projeto de controle financeiro, vamos criar relatórios com os dados gravados no banco de dados, o primeiro relatório que queremos criar no projeto é uma lista de movimentações por usuário. Então dentro do `MovimentacaoController` vamos adicionar um novo método chamado `MovimentacoesPorUsuario` que deve receber qual é o usuário que queremos utilizar no filtro:

```
public ActionResult MovimentacoesPorUsuario(int? usuarioId)
{
    IList<Movimentacao> movimentacoes = // busca movimentacoes do banco

    return View(movimentacoes);
}
```

Mas com esse código não conseguimos utilizar o `HtmlHelper` para gerar o código do formulário. Para conseguirmos trabalhar com a lista tipada e montarmos um formulário fortemente tipado, utilizaremos novamente o view model.

Dentro da pasta `Models` , crie uma nova classe chamada `MovimentacoesPorUsuarioModel` :

```
public class MovimentacoesPorUsuarioModel
{
    public int? UsuarioId { get; set; }

    public IList<Movimentacao> Movimentacoes { get; set; }
}
```

No formulário, queremos utilizar um combo box para escolhermos o usuário para o relatório, então adicionaremos uma nova propriedade na view model que guardará a lista de usuários:

```
public class MovimentacoesPorUsuarioModel
{
    public int? UsuarioId { get; set; }

    public IList<Movimentacao> Movimentacoes { get; set; }

    public IList<Usuario> Usuarios { get; set; }
}
```

Agora podemos modificar a action `MovimentacoesDoUsuario` para que ela receba um `MovimentacoesPorUsuarioModel` :

```
public ActionResult MovimentacoesPorUsuario(MovimentacoesPorUsuarioModel model)
{
    model.Usuarios = usuarioDAO.Lista();
    model.Movimentacoes = movimentacaoDAO.BuscaPorUsuario(model.UsuarioId);
    return View(model);
}
```

Além do controller, precisamos também colocar o método `BuscaPorUsuario` dentro do `MovimentacaoDAO` :

```
public IList<Movimentacao> BuscaPorUsuario(int? usuarioID)
{
    return context.Movimentacoes.Where(m => m.UsuarioId == usuarioID).ToList();
}
```

Para terminar, só precisamos escrever o código da camada de visualização:

```
@model Financas.Models.MovimentacoesPorUsuarioModel

@using (Html.BeginForm("MovimentacoesPorUsuario", "Movimentacao", FormMethod.Get))
{
    @Html.LabelFor(m => m.UsuarioId, "Usuário")
    @Html.DropDownListFor(m => m.UsuarioId,
        new SelectList(Model.Usuarios, "Id", "Nome"),
        "Escolha um usuário")

    <input type="submit" value="Buscar"/>
}

<table class="table table-hover">
    <thead>
        <tr>
            <th>Usuario</th>
            <th>Tipo</th>
            <th>Valor</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var m in Model.Movimentacoes)
        {
            <tr>
                <td>@m.Usuario.Nome</td>
                <td>@m.Tipo</td>
                <td>@m.Valor</td>
            </tr>
        }
    </tbody>
</table>
```

Busca por diversos critérios

Além dessa busca por usuários, queremos ser capazes de filtrar a movimentação por diversos critérios: valor mínimo e máximo, data inicial e final, tipo e usuario.

Para implementarmos essa busca, vamos criar um view model que recebe todos os critérios do filtro:

```
public class BuscaMovimentacoesModel
{
    public decimal? ValorMinimo { get; set; }

    public decimal? ValorMaximo { get; set; }

    public DateTime? DataMinima { get; set; }

    public DateTime? DataMaxima { get; set; }

    public Tipo? Tipo { get; set; }

    public int? UsuarioId { get; set; }

    public IList<Movimentacao> Movimentacoes { get; set; }

    public IList<Usuario> Usuarios { get; set; }
}
```

Agora dentro do `MovimentacaoController`, vamos adicionar uma nova action chamada `Busca` que recebe o view model:

```
public ActionResult Busca(BuscaMovimentacoesModel model)
{
    model.Usuarios = usuarioDAO.Lista();
    model.Movimentacoes = movimentacaoDAO.Busca(model.ValorMinimo, model.ValorMaximo,
                          model.DataMinima, model.DataMaxima,
                          model.Tipo, model.UsuarioId);
    return View(model);
}
```

Agora vamos implementar o método do DAO:

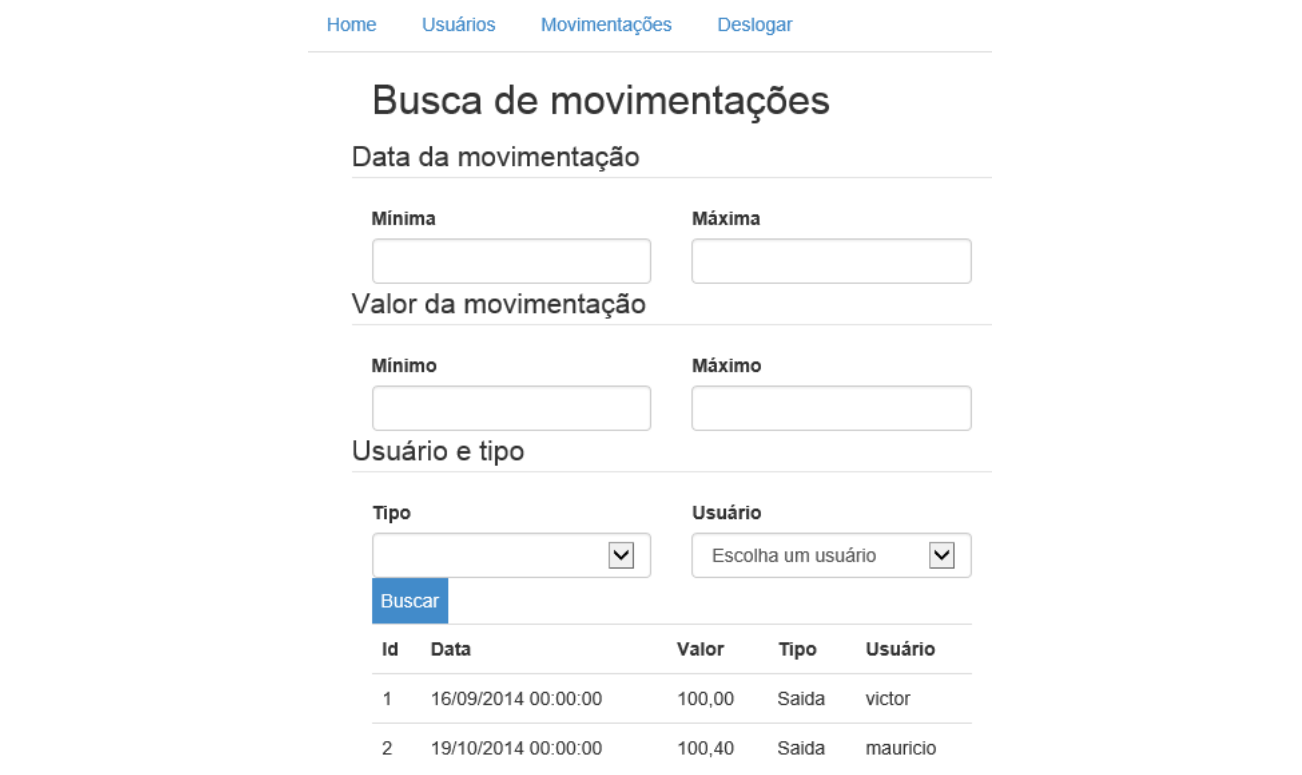
```
public IList<Movimentacao> Busca(decimal? valorMinimo, decimal? valorMaximo,
                                 DateTime? dataMinima, DateTime? dataMaxima,
                                 Tipo? tipo, int? usuario)
{
    IQueryable<Movimentacao> resultado = context.Movimentacoes;
    if (valorMinimo.HasValue)
    {
        resultado = resultado.Where(m => m.Valor >= valorMinimo);
    }
    if (valorMaximo.HasValue)
    {
        resultado = resultado.Where(m => m.Valor <= valorMaximo);
    }
    if(dataMinima.HasValue)
    {
        resultado = resultado.Where(m => m.Data >= dataMinima);
```

```
        }
        if (dataMaxima.HasValue)
        {
            resultado = resultado.Where(m => m.Data <= dataMaxima);
        }
        if (tipo.HasValue)
        {
            resultado = resultado.Where(m => m.Tipo == tipo);
        }
        if (usuario.HasValue)
        {
            resultado = resultado.Where(m => m.UsuarioId == usuario);
        }
        return resultado.ToList();
    }
```

Para terminarmos esse filtro, só precisamos criar a view:

```
@model Financas.Models.BuscaMovimentacoesModel
<h2>Busca de movimentações</h2>
@using (Html.BeginForm("Busca", "Movimentacao", FormMethod.Get))
{
    <div class="row">
        <fieldset>
            <legend>Data da movimentação</legend>
            <div class="col-sm-6">
                @Html.LabelFor(b => b.DataMinima, "Mínima")
                @Html.TextBoxFor(b => b.DataMinima, new { @class = "form-control" })
            </div>
            <div class="col-sm-6">
                @Html.LabelFor(b => b.DataMaxima, "Máxima")
                @Html.TextBoxFor(b => b.DataMaxima, new { @class = "form-control" })
            </div>
        </fieldset>
        <fieldset>
            <legend>Valor da movimentação</legend>
            <div class="col-sm-6">
                @Html.LabelFor(b => b.ValorMinimo, "Mínimo")
                @Html.TextBoxFor(b => b.ValorMinimo, new { @class = "form-control" })
            </div>
            <div class="col-sm-6">
                @Html.LabelFor(b => b.ValorMaximo, "Máximo")
                @Html.TextBoxFor(b => b.ValorMaximo, new { @class = "form-control" })
            </div>
        </fieldset>

        <fieldset>
            <legend>Usuário e tipo</legend>
            <div class="col-sm-6">
                @Html.LabelFor(b => b.Tipo, "Tipo")
                @Html.EnumDropDownListFor(b => b.Tipo, new { @class = "form-control" })
            </div>
            <div class="col-sm-6">

                @Html.LabelFor(b => b.UsuarioId, "Usuário")
                @Html.DropDownListFor(b => b.UsuarioId,
```

```
                    new SelectList(Model.Usuarios, "Id", "Nome"), "Escolha um usuário", new { @
            </div>
        </fieldset>
    </div>


    <input type="submit" value="Buscar" />
}


<table class="table table-hover">
    <tr>
        <th>Id</th>
        <th>Data</th>
        <th>Valor</th>
        <th>Tipo</th>
        <th>Usuário</th>
    </tr>
    @foreach (var m in Model.Movimentacoes)
    {
        <tr>
            <td>@m.Id</td>
            <td>@m.Data</td>
            <td>@m.Valor</td>
            <td>@m.Tipo</td>
            <td>@m.Usuario.Nome</td>
        </tr>
    }
</table>
```



Menu lateral para acessar as buscas

Com a aplicação atual, a única forma de conseguirmos acessar as buscas é colocando a url diretamente na barra de endereços do navegador. Isso não é bom para a usabilidade da aplicação, seria melhor termos um link na página que nos leva para esses relatórios.

No layout da aplicação, arquivo `_Layout.cshtml`, colocaremos o código para definir o menu lateral da aplicação. Logo abaixo da tag `main` do layout, vamos colocar uma tag chamada `aside` com o código do menu:

```
<div class="container">
    <main class="col-sm-8">
        @RenderBody()
    </main>
    <aside class="col-sm-4">
        <h3>Acesso rápido</h3>
        <ul>
            <li>@Html.ActionLink("Movimentações por usuário", "MovimentacoesPorUsuario", "Movime
            <li>@Html.ActionLink("Busca de movimentações", "Busca", "Movimentacao")</li>
        </ul>
    </aside>
</div>
```