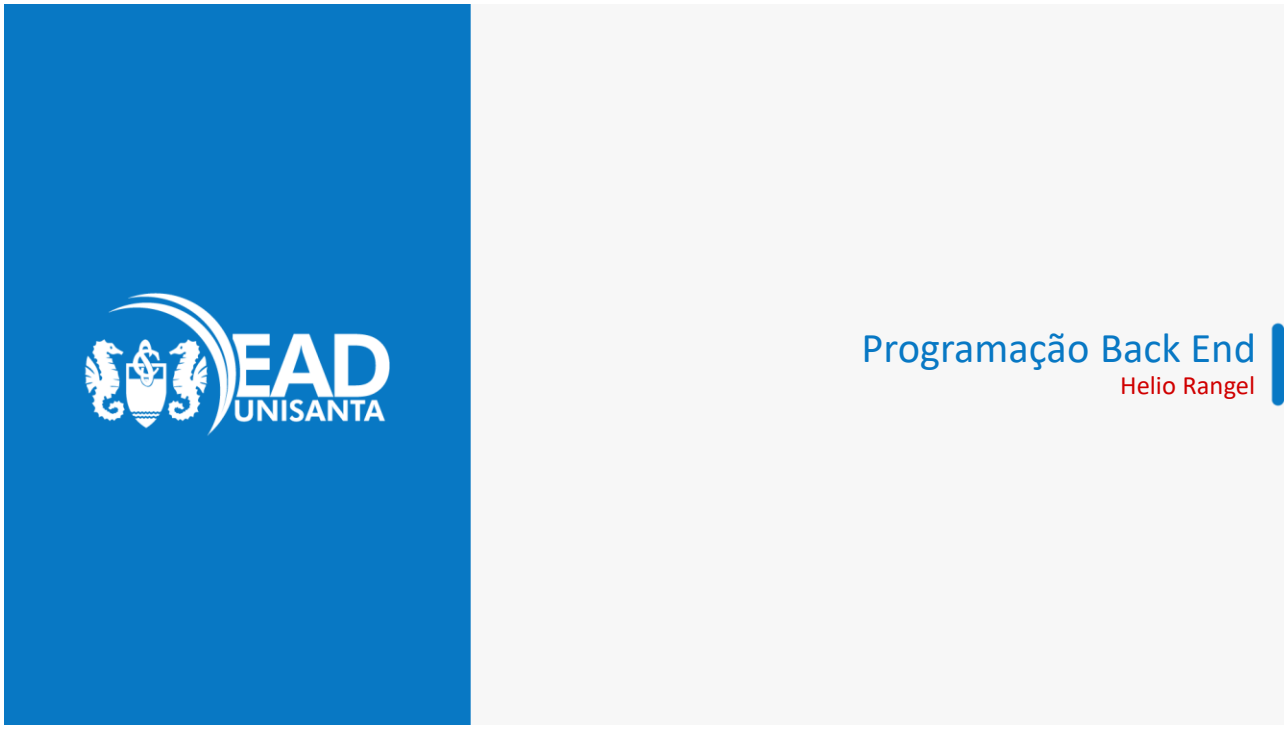
1



Programação Back End

Helio Rangel

2

AULA 8

O Livro Caixa

3

Livro Caixa – Requisitos

- Principal funcionalidade do sistema, foi projetada para controlar o caixa.
- Deve operar por períodos mensais que podem ser fechados fora do último dia do mês, se necessário.
- Quando o usuário fecha o período, o sistema cria o período seguinte, transportando o saldo do período que foi fechado para o novo que está abrindo.
- Os períodos anteriores podem ser consultados livremente mas não podem receber novos lançamentos.

4

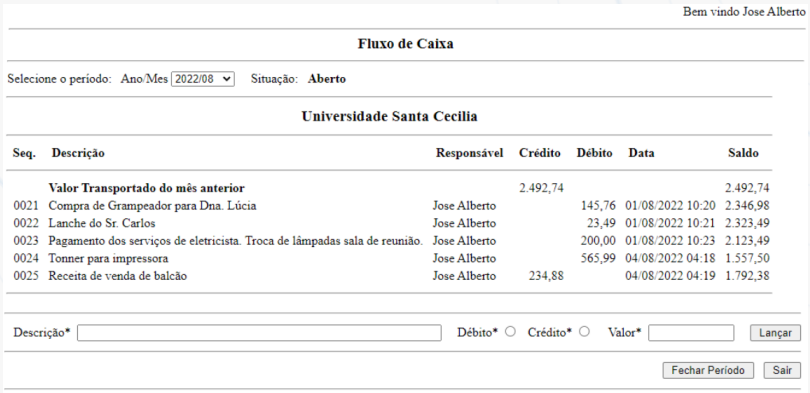
Usando Javascript

- Temos uma oportunidade para mostrar como usar uma função *Javascript* chamada no campo texto onde digitamos o valor do lançamento. Esta função é muito útil porquê além de formatar o campo valor com duas casas decimais, facilita a consistência impedindo que o usuário digite caracteres não numéricos.
- Normalmete o código Javascript é carregado na página diretamente de um ou mais arquivos extensão *.js* que devem ser anexados ao projeto. No nosso caso, o código *Javascript* foi inorporado diretamente no Form *FluxoDeCaixa.aspx* para facilitar o estudo. Na realidade, temos uma coleção de métodos que se chamam entre si.
- O estudo de *Javascript* está mais voltado ao estudo e desenvolvimento do ***Front End*** mas, como dissemos, não é possível falar de ***Back End*** sem jamais se referir ao ***Front End***.
- No objeto TextBox txValor, colocamos no evento *onkeyup* a chamada para o método *formataValor()* que está definido no corpo do form *FluxoDeCaixa.aspx*

```
<asp:TextBox ID="txValor" runat="server"  Width="100px" onkeyup='formataValor(this,event);' />
```


Os lançamentos

- Os campos descrição, um dos *Radio Buttons* se Débito ou Crédito e o valor do lançamento, são campos obrigatórios. O usuário logado e a data atual são fornecidos diretamente pelo sistema.



O evento do botão Lançar

- Como os lançamentos não podem ser editados nem excluídos, é importante que o usuário tenha chance de verificar os dados do lançamento antes de prosseguir. Por isso mostramos uma caixa de confirmação. O método ***valida()***, que já mostra a mensagem no caso de inconsistência, logo no início do evento, aborta o processo de inserção de lançamentos caso exista alguma inconsistência.

```
protected void btLancar_Click(object sender, EventArgs e)
{
    if (!valida()) return;

    StringBuilder myScript = new StringBuilder(String.Empty);
    myScript.Append("<script type='text/javascript' language='javascript'> ");
    myScript.Append("var result = window.confirm('Confirma realizar o lançamento? Valor: R$" +
        txValor.Text + " - " + (rbCredito.Checked ? "Crédito":"Débito") + "'); ");
    myScript.Append("__doPostBack('lancamento', result); ");
    myScript.Append("</script> ");
    ScriptManager.RegisterStartupScript(Page, Page.GetType(), "msg", myScript.ToString(), false);
}
```

- O método ScriptManager. RegisterStartupScript(), monta na página retornada pelo server, um script que mostra a caixa de diálogo permitindo o usuário conferir o valor lançado se está ok.
- O método que efetivamente vai realizar o lançamento é visto na chamada __doPostBack('lancamento',result)

O evento do botão Lançar

- Quando o formulário é recarregado, o método ***FormLoad()***, "percebe" que foi lançado um evento. Se o evento laçado for "lancamento", é instanciado um novo lançamento, passando para o construtor todos os dados digitados. Lembrando que foram validado anteriormente. O Lançamento é inserido e em seguida os campos da tela são limpos e o formulário de lançamentos do período, já atualizado é mostrado.

```
String evento = (this.Request["__EVENTTARGET"] == null) ? String.Empty : this.Request["__EVENTTARGET"];
if (evento == "fecharPeriodo")
{
    livro.fechaPeriodo(con);
    btFecharPeriodo.Enabled = false;
    comboAnoMes.Items.Clear();
    montaComboAnoMes(comboAnoMes, con);
}
if (evento == "lancamento")
{
    Lancamento l = new Lancamento(txDescricao.Text, livro.idLivroCaixa, Double.Parse(txValor.Text),
                                  rbCredito.Checked ? 'C' : 'D', usuario, DateTime.Now);
    l.insere(con);
    livro.listaLancamentos(con);
    txValor.Text = txDescricao.Text = String.Empty;
    rbDebito.Checked = rbCredito.Checked = false;
}
```

Checando o método *valida()*

- Todos os campos do form são obrigatórios e por isso nenhum pode ser deixado em branco.
- O campo *txValor* precisa ser validado, com respeito a validade dele como número, se número, deve ser maior que zero e, o mais importante:
- Se existe saldo disponível para o saque.
    - A última consistência pergunta se estamos procedendo um **débito** e se este valor é superior ao saldo disponível. Para saber que saldo é este, utilizamos a propriedade saldo do objeto livro.

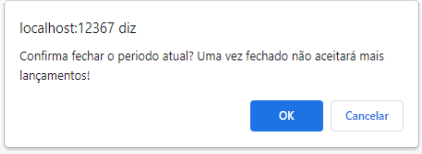
```
private bool valida()
{
    if (txDescricao.Text.Trim() == String.Empty)
    {
        lbMensagem.Text = "Descrição é campo obrigatório!";
        return false;
    }
    if (txValor.Text.Trim() == String.Empty)
    {
        lbMensagem.Text = "Valor é campo obrigatório!";
        return false;
    }
    double valor;
    if (!Double.TryParse(txValor.Text, out valor))
    {
        lbMensagem.Text = "Valor digitado não é um número válido!";
        return false;
    }
    if (valor <= 0)
    {
        lbMensagem.Text = "Valor digitado deve ser maior que zero e positivo!";
        return false;
    }
    if (rbCredito.Checked == false && rbDebito.Checked == false)
    {
        lbMensagem.Text = "Selecione se o lançamento é de crédito ou débito";
        return false;
    }
    if (rbDebito.Checked && valor > livro.saldo)
    {
        lbMensagem.Text = "Saldo em caixa insuficiente para retirada!";
        return false;
    }
    return true;
}
```


Fechando o período

- Analisando os requisitos, eles especificam que os períodos de lançamentos são definidos pelo mês/ano selecionado e esta ação seria feita pelo usuário em um dia arbitrário do período. Isto foi implentado inserindo novos registros na tabela LivroCaixa.
- Ao clicar no botão de Fechar Periodo, o evento *btFecharPeriodo_Click()* é executado:

```
protected void btFecharPeriodo_Click(object sender, EventArgs e)
{
    StringBuilder myScript = new StringBuilder(String.Empty);
    myScript.Append("<script type='text/javascript' language='javascript'> ");
    myScript.Append("var result = window.confirm('Confirma fechar o periodo atual? Uma vez " +
        "fechado não aceitará mais lançamentos!'); ");
    myScript.Append("__doPostBack('fecharPeriodo', result); ");
    myScript.Append("</script> ");
    ScriptManager.RegisterStartupScript(Page, Page.GetType(), "msg", myScript.ToString(), false);
}
```

- Como vimos, a caixa de diálogo permite que o usuário possa se certificar que o fechamento é “pra valer”.



Fechando o período

- De forma análoga ao que vimos no Cadastro de Usuários, precisamos tratar o retorno do post back da resposta a caixa de diálogo.

```
String evento = (this.Request["__EVENTTARGET"] == null) ? String.Empty : this.Request["__EVENTTARGET"];
if (evento == "fecharPeriodo")
{
    livro.fechaPeriodo(con);
    btFecharPeriodo.Enabled = false;
    comboAnoMes.Items.Clear();
    montaComboAnoMes(comboAnoMes, con);
}
```

- Atendendo o evento para o caso do usuário clicar em **OK**, o método fecharPeriodo da classe LivroCaixa é chamado. Em seguida o botão de fechar caixa é desabilitado, o combo de períodos é limpo e logo em seguida remontado para que aconteça a sua atualização.

O método **fecharPeriodo()** da classe **LivroCaixa**

- A primeira providência é iniciar a transação, fazemos isso porque estamos alterando/criando dois registros em momentos diferentes. Para garantir que não teremos inconsistências no par de registros, usamos a transação para garantir a integridade dos dados.
- Na primeira *query*, alteramos o *flag stLivroFechado* para 1 sinalizando que o registro atual está sendo fechado.
- Em seguida, montamos uma data, *proximoPeriodo*, que aponta sempre para o mês seguinte ao mês do período que esta sendo fechado.
- A segunda *query*, insere um novo registro no Banco de Dados.
- Finalmente a transação é finalizada e os dados são fisicamente atualizados no banco com o *commit()*.
- No caso de erro, o tratamento de excessão deixa os registros no banco inalterados com o *rollback()*.

```
1 reference
public void fechaPeriodo(Conexao con)
{
  try
  {
    con.beginTransaction(); //Afeta dois registros, por isso foi implementada transação
    string sql = String.Concat("UPDATE " , nomeTabela , " SET stLivroFechado=1 Where ",
        nomeId , "=" ,  idLivroCaixa);
            Conexao.executaQuery(con, sql, nomeTabela, null);
    DateTime proximoPeriodo = new DateTime(Int16.Parse(ano),
                                           Int16.Parse(mes), 1).AddMonths(1);
    sql = String.Concat("INSERT INTO  ", nomeTabela, " (", campos, ") Values (",
            (saldo + "").Replace(",", "."),",", proximoPeriodo.Month , ",",
            proximoPeriodo.Year , ",0)");
    Conexao.executaQuery(con, sql, nomeTabela,null);
    con.commit();
  }
  catch (Exception)
  {
     con.rollback();
     throw;
  }
}
```