

Criar um novo Documento

Um documento seria algo que envolve um Cadastro, Manutenção, DataModule e opcionalmente uma Pesquisa.

Ex.:

Clientes

Produtos

Documentos Fiscais (NFe, NFSe..)

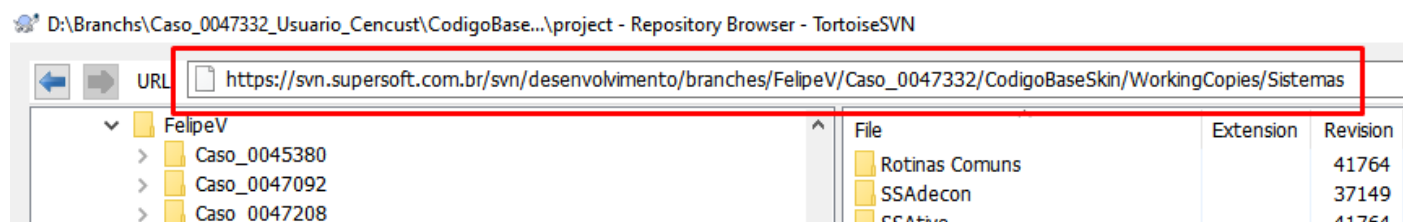
Etc.

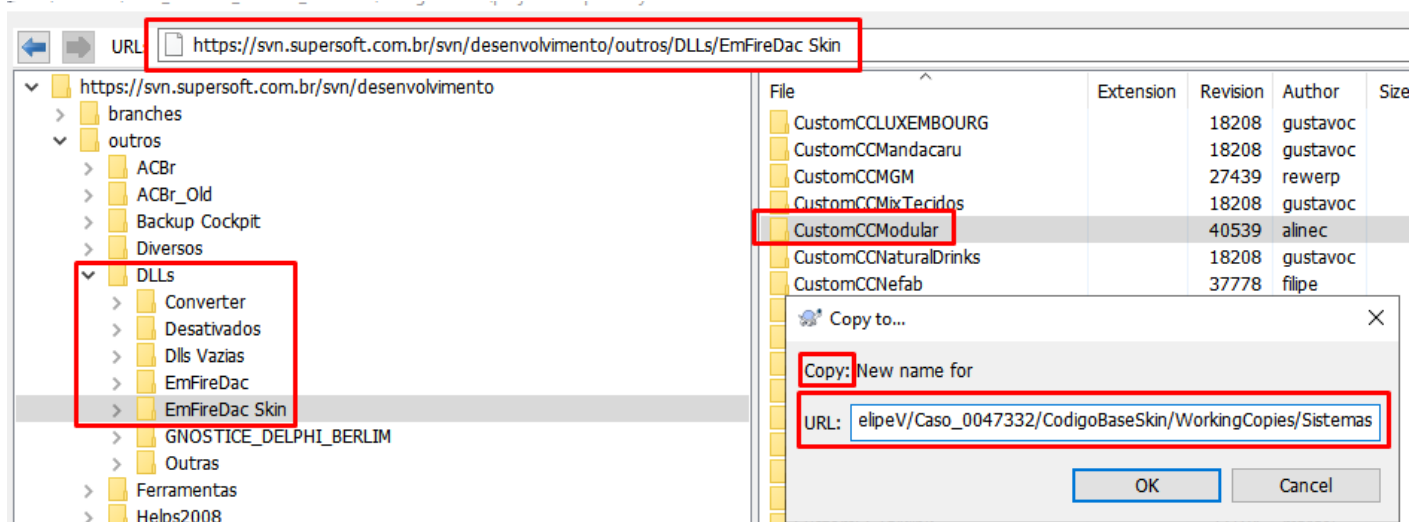
Para exemplo, usaremos a criação através da DLL, e como base o caso do Mantis 47332 onde cliente solicita a configuração de um CFOP específico por usuário.

O motivo de não ser feito direto na tela de usuários é o fato de ser customizado.

Não existe apenas **uma** forma de fazer, então o registrado aqui é uma **sugestão**.

Após fazer a cópia do trunk para uma pasta dentro do seu branch, navegue até a pasta Sistemas, copie o diretório, dê um copy to na dll em FireDac Skin ("Custom" + Sigla do módulo + Cliente), conforme as imagens abaixo:





Baixa seu código inteiro para darmos continuidade ao trabalho conforme modelo em [Local](#)

É muito importante entender bem o precisa ser feito, para desenhar a tela e as tabelas da melhor forma.

Também é importante lembrar que formulários novos são feitos através de herança, então no caso de dúvida pergunte.

Planejamento desde caso:

Nomenclatura: Como será uma informação (Centro de Custos) padrão para usuário e futuramente podem vir a acrescentar novos campos, pensamos em algo no sentido de "Padrões do Usuário", isso reflete no nome das telas e da tabela nova.

Com nome definido, inicie desenhando os campos necessários para atender o cadastro, neste caso usuário já é cadastrado, falta informar o Centro de Custos para ele.

Dica: Olhe alguns exemplos já existentes, isso ajudará muito.

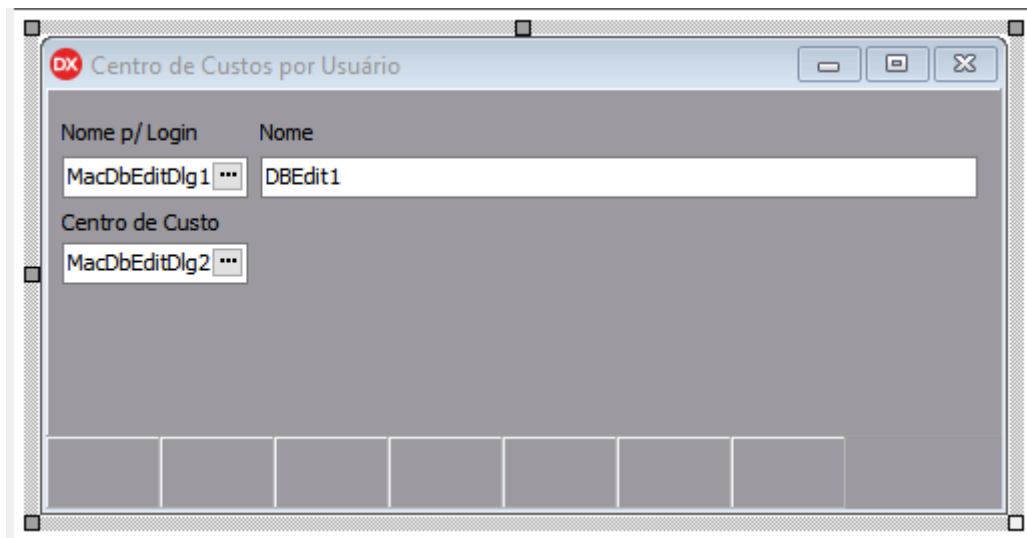
Tela de Inclusão/Edição/Visualização

Crie herdando de "TFORM_EDIT" ou "TFORM_EDIT1", o que muda é o sentido da barra de ferramentas (Vertical ou Horizontal).

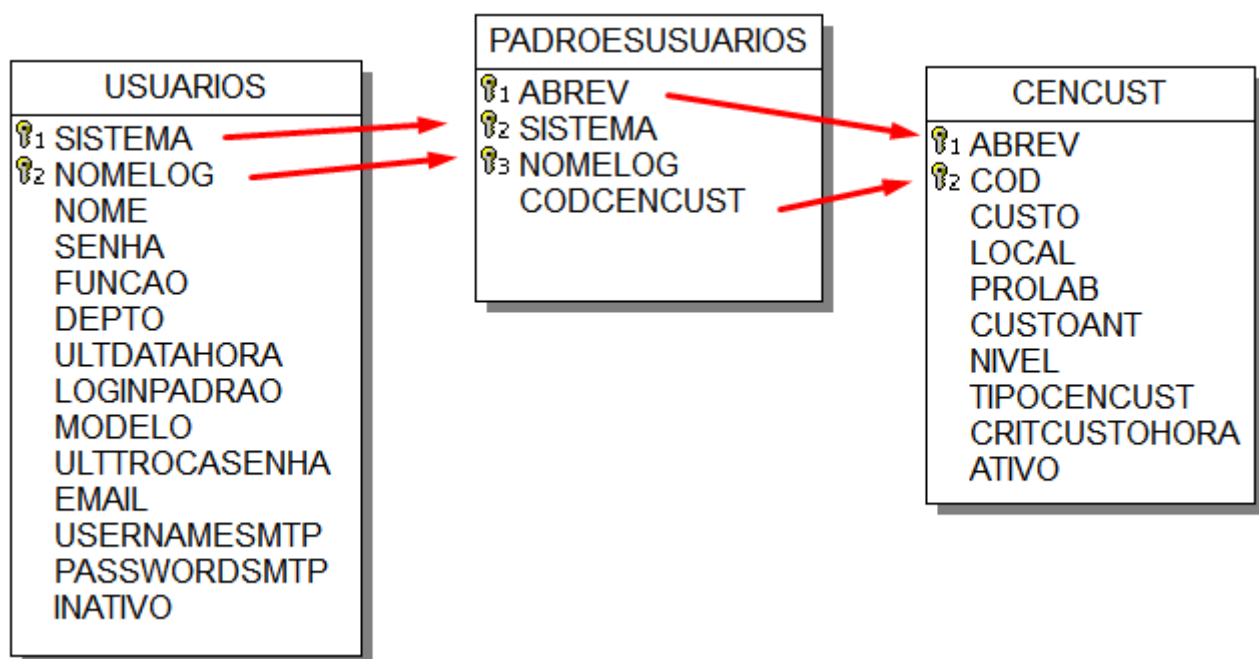
A propriedade **Name** do formulário costuma seguir alguns padrões, neste exemplo "FORM_" + Nome da tabela + "_EDIT".

A propriedade **Caption** do formulário, neste caso como se trata apenas do Centro de Custos, então chamaremos de "Centro de Custos por Usuário"

Baseado na tela de Usuários já existentes, puxamos algumas informações em ReadOnly apenas para usuário ter certeza que é o usuário correto que escolheu, e o campo que iremos cadastrar, a princípio a tela está:



Com a primeira tela feita, fica mais fácil de desenhar a tabela, que para este caso será "PadroesUsuarios":



Criar tabela:

Em alguns casos a criação estará na uit uCriaTabDll.pas dentro do fonte da DLL, em outro caso estará dentro de DllSource.pas. Caso não tenha em nenhuma das duas, então crie a uCriaTabDll.pas

A principio será necessário apenas a procedure "RegistraTabelas" e a function "Cria" + Nome da Tabela.

A procedure "RegistraTabelas" será para adicionar uma descrição da tabela nova dentro da tabela **"Sistemas"**. Já a function ""Cria" + Nome da Tabela, é usada para adicionar na tabela **"IntegRef"**

que por sua vez é responsável por **simular** as chaves estrangeiras do nosso banco.

Os métodos ficaram mais ou menos assim:

function "Cria"+ Nome da Tabela (no caso, CriaPadroesUsuarios):

```
function CriaPadroesUsuarios(AVerTipol: String; AVetor: array of String; AMatrizAlt: array of Atualizacao): TCriaTab;
var
  LAuxiliar: TFDTable;
  LNomeTab : String;
begin
  LAuxiliar := CriarFDTable(nil);
  LAuxiliar.FieldDefs.Clear();
  LAuxiliar.IndexDefs.Clear();
  LAuxiliar.FieldDefs.Add('Abrev', ftString, 8, True);      {0}
  LAuxiliar.FieldDefs.Add('Sistema', ftString, 3, True);   {1}
  LAuxiliar.FieldDefs.Add('NomeLog', ftString, 10, True);  {2}
  LAuxiliar.FieldDefs.Add('CodCenCust', ftString, 7);      {3}

  if ((High(AVetor) >= 1) and (not AVetor[1].IsEmpty)) then
    LNomeTab:= AVetor[1]
  else
    LNomeTab:= 'PADROESUSUARIOS';
  LAuxiliar.TableName := LNomeTab;
  LAuxiliar.IndexDefs.Add(LNomeTab + '_IND1', 'Abrev;Sistema;NomeLog', [ixPrimary, ixUnique]);
  try
    Result.Bool := True;
    Result.Cod := 0;
    LAuxiliar.CreateTable();
    DM_CONEXOES.CO_Principal.Commit();
  except
    Result.Bool := False;
  end;
  FreeAndNil(LAuxiliar);
end;
```

Nessa function estamos criando a tabela desejada no banco por linhas de código.

Passamos na "LAuxiliar.FieldDefs.Add" os seguintes parâmetros:

('Nome do campo', Tipo do campo, Tamanho de caracteres do campo, True se for uma PK);

Anotações importantes:

- O tipo do campo deve ser adicionado como "ft" + tipo do valor. Exemplo (Varchar no banco-> ftString);

- O ultimo parâmetro, tem um valor default de "False", sendo assim, caso não seja passado, o campo não sera uma PK;

procedure RegistraTabelas:

```

procedure RegistraTabelas(AQual: String);
var
  LMatVal: array[0..1] of String;
  LDS_Pesq: TDataSource;
  LQ_Pesq: TFDQuery;

begin
  LDS_Pesq := TDataSource.Create(nil);
  LQ_Pesq := CriarFDQuery(nil);
  LDS_Pesq.DataSet := LQ_Pesq;
  try
    if (AnsiSameText(AQual, 'SISTEMAS')) then
    begin
      if (not TProcuraRegistro.Procurar(
        LDS_Pesq, 'SISTEMAS', 'Sistema;Tabela', ['CC', 'PadroesUsuarios'], ['S', 'S'], C_ACAO_PROCURA)) then
      begin
        TGravaSistema.Gravar('NULO', 'PadroesUsuarios', 'Padrões por Usuário', 'S', 'CC', 'S');
      end;
    end
    else if (AnsiSameText(AQual, 'INTEGREF')) then
    begin
      LMatVal[0] := 'USUARIOS|2|PADROESUSUARIOS|Sistema;NomeLog|Sistema;NomeLog|N|N|CC|';
      LMatVal[1] := 'CENCUST|2|PADROESUSUARIOS|Abrev;Cod|Abrev;CodCenCust|N|N|CC|';
      TGravaIntegRef.Gravar(LMatVal);
    end;
  finally
    FreeAndNil(LDS_Pesq);
    FreeAndNil(LQ_Pesq);
  end;
end;

```

Nessa procedure, estamos fazendo o vínculo das chaves primarias (PK) e chaves estrangeiras (FK) do banco por linhas de código.

Passamos na "LMatVal" os seguintes valores:

(tabela pai, tipo da checagem que será feita, tabela filha, chaves primarias na tabela pai, chaves estrangeiras da tabela filha, se vai ser eliminado da tabela pai, se vai ser eliminada da tabela filha e a sigla do sistema)

*Lembrando que para a criação da tabela seja efetuada é necessário que o sistema ao ser aberto passe pela tela de atualizações, ou seja, é necessário mudar no banco, na tabela "Global" + sigla do módulo para versões anteriores *

Explicando o DLG nos Edits:

Basicamente o componente dlg tem algumas propriedades específicas, elas são:

-TabelaPesq:

Seleciona a tabela que o dlg do edit vai buscar os dados, por exemplo: usuários;

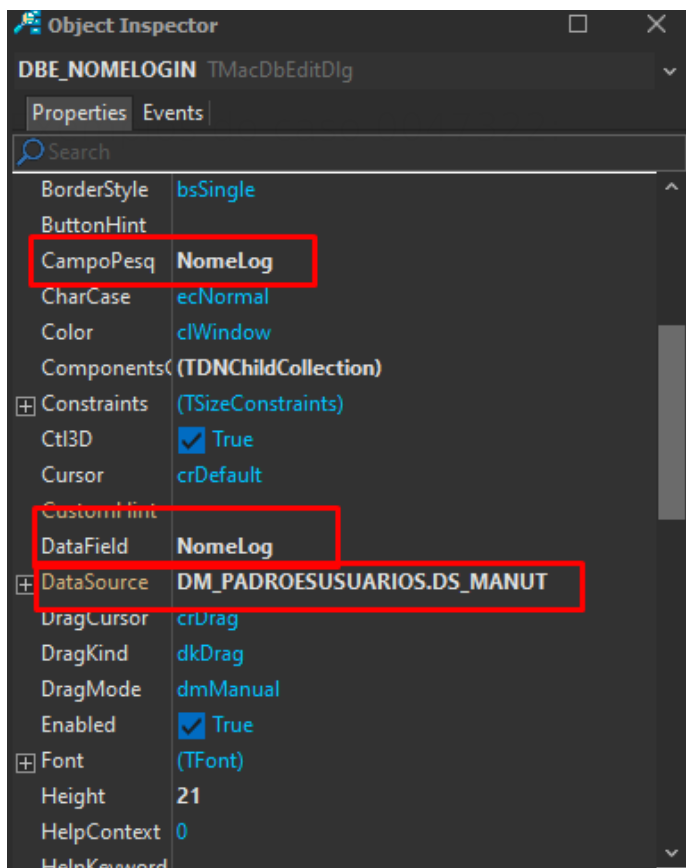
-CampoPesq:

Deve ser inserido um campo da tabela selecionada no TabelaPesq para efetuar a busca no banco, por exemplo: NomeLog;

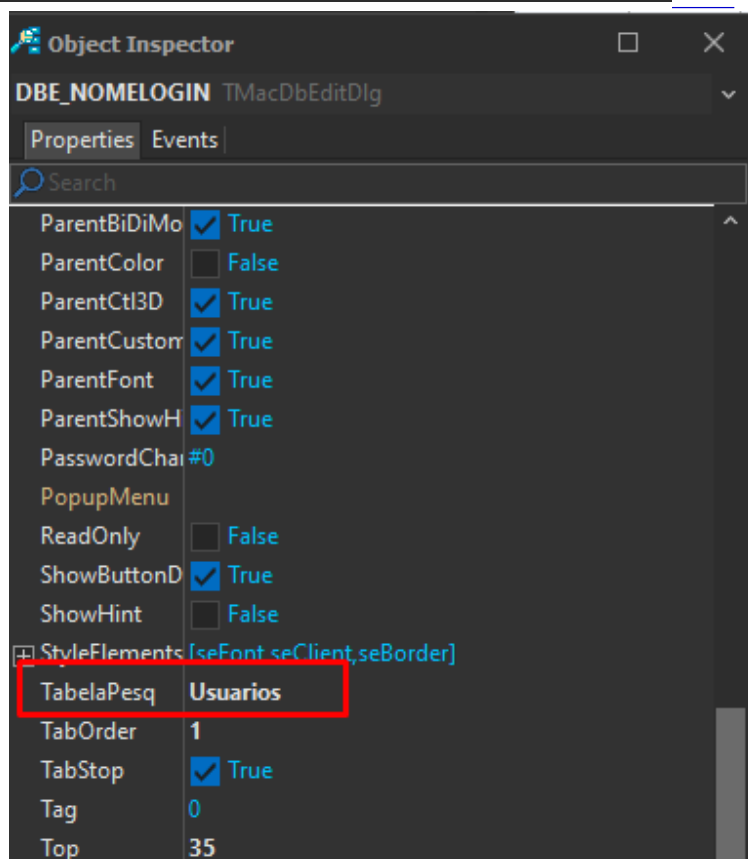
-DataSource:

Deve ser passado o DataModule.DataSource, por exemplo: DM_PADROESUSUARIOS.DS_MANUT;

-DataField:

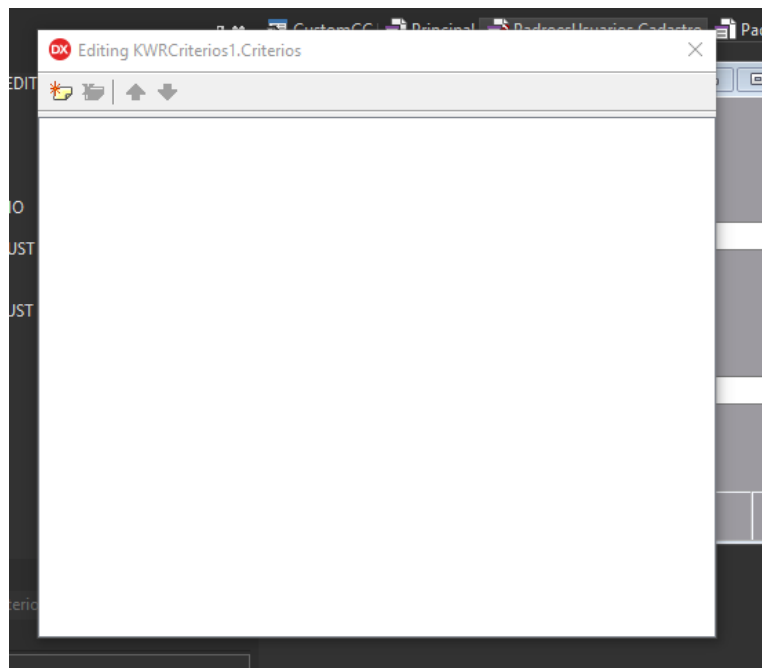


onado, por exemplo: NomeLog;

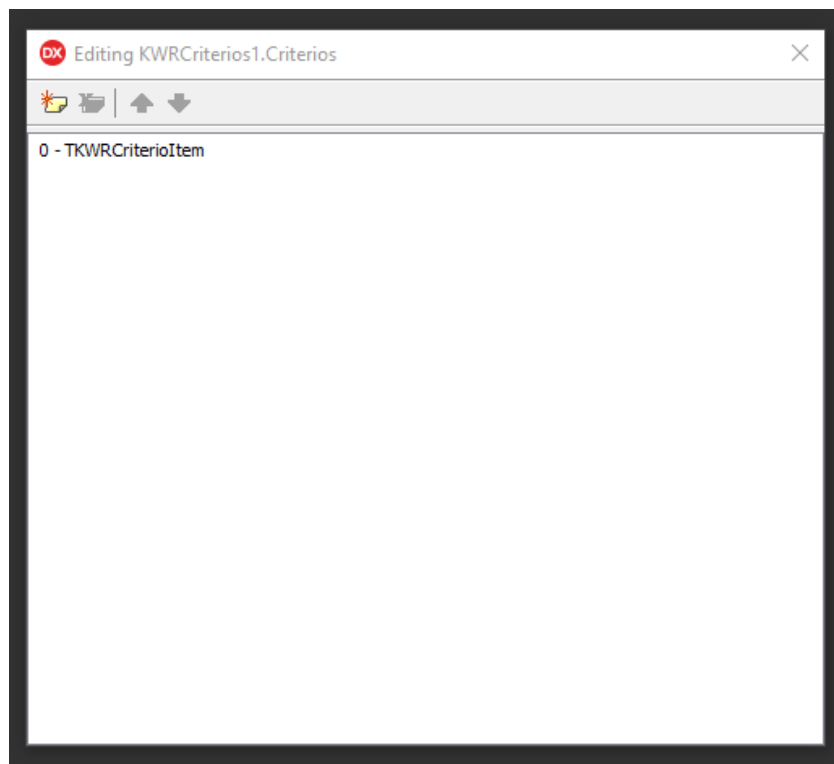


Componente TKWRCriterios

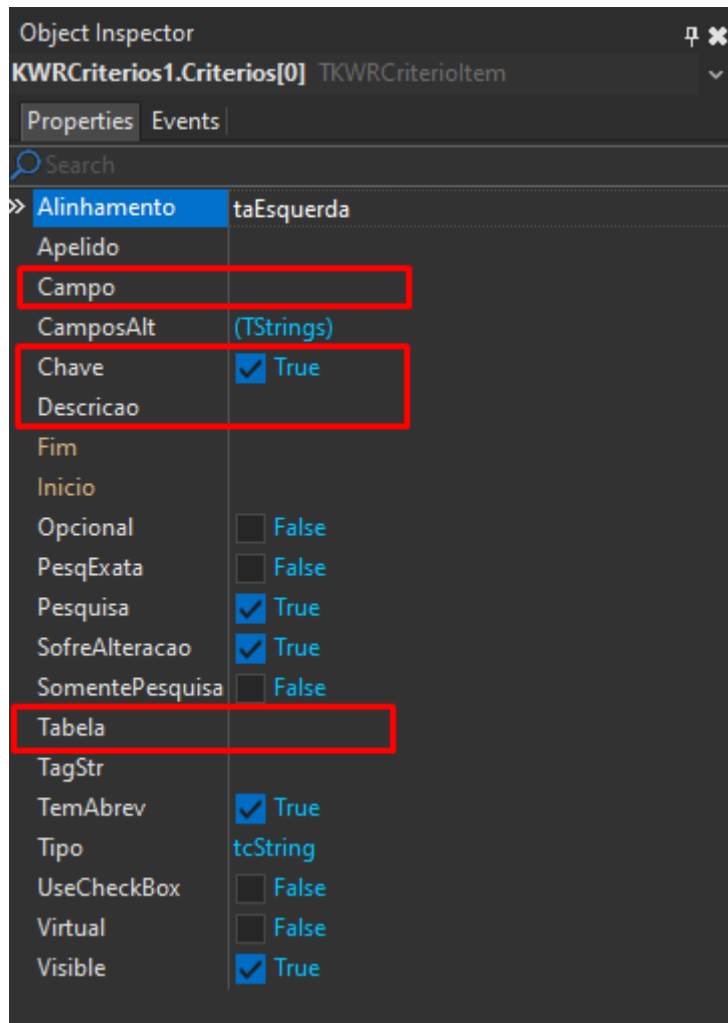
o componente Critérios tem uma propriedade chamada "Critérios", ao dar dois cliques nele, se deparamos com a seguinte tela:



Ao clicarmos no único botão disponível, criamos um novo critério, ele será adicionado com índice 0 e com um nome padrão:



Ao clicar no item adicionado, na aba "Object Inspector" irão aparecer as propriedades daquele item, iremos alterar algumas delas, sendo elas as seguintes:



Na propriedade Campo iremos adicionar o campo da tabela nova que criamos;

Na propriedade Chave iremos deixar marcado caso o campo seja uma PK e iremos desmarcar caso não seja;

Na propriedade Descrição é o título de onde os dados daquele campo vão ser inseridos;

Na propriedade Tabela vai ser inserido o nome da tabela que criamos;

Exemplo

No componente critérios do caso 0047332, as propriedades alteradas ficaram dessa maneira:

1 - PadroesUsuarios.NomeLog
2 - PadroesUsuarios.CodCenCust
3 - Usuarios.Nome
4 - CenCust.Custo

Object Inspector

CRITERIOS.Criterios[1] TKWRCriterioItem

Properties Events

Search

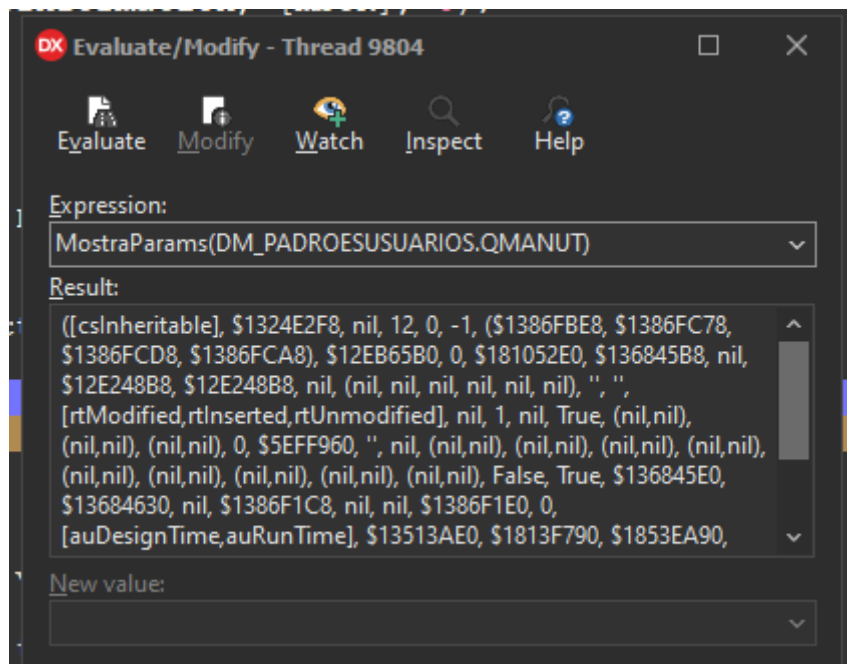
>> Alinhamento taEsquerda

Apelido	NomeLog
Campo	NomeLog
CamposAlt	(TStrings)
Chave	<input checked="" type="checkbox"/> True
Descricao	Nome de Login
Fim	
Inicio	
Opcional	<input type="checkbox"/> False
PesqExata	<input type="checkbox"/> False
Pesquisa	<input checked="" type="checkbox"/> True
SofreAlteracao	<input type="checkbox"/> False
SomentePesquisa	<input type="checkbox"/> False
Tabela	PadroesUsuarios
TagStr	
TemAbrev	<input checked="" type="checkbox"/> True
Tipo	tcString
UseCheckBox	<input type="checkbox"/> False
Virtual	<input type="checkbox"/> False
Visible	<input checked="" type="checkbox"/> True

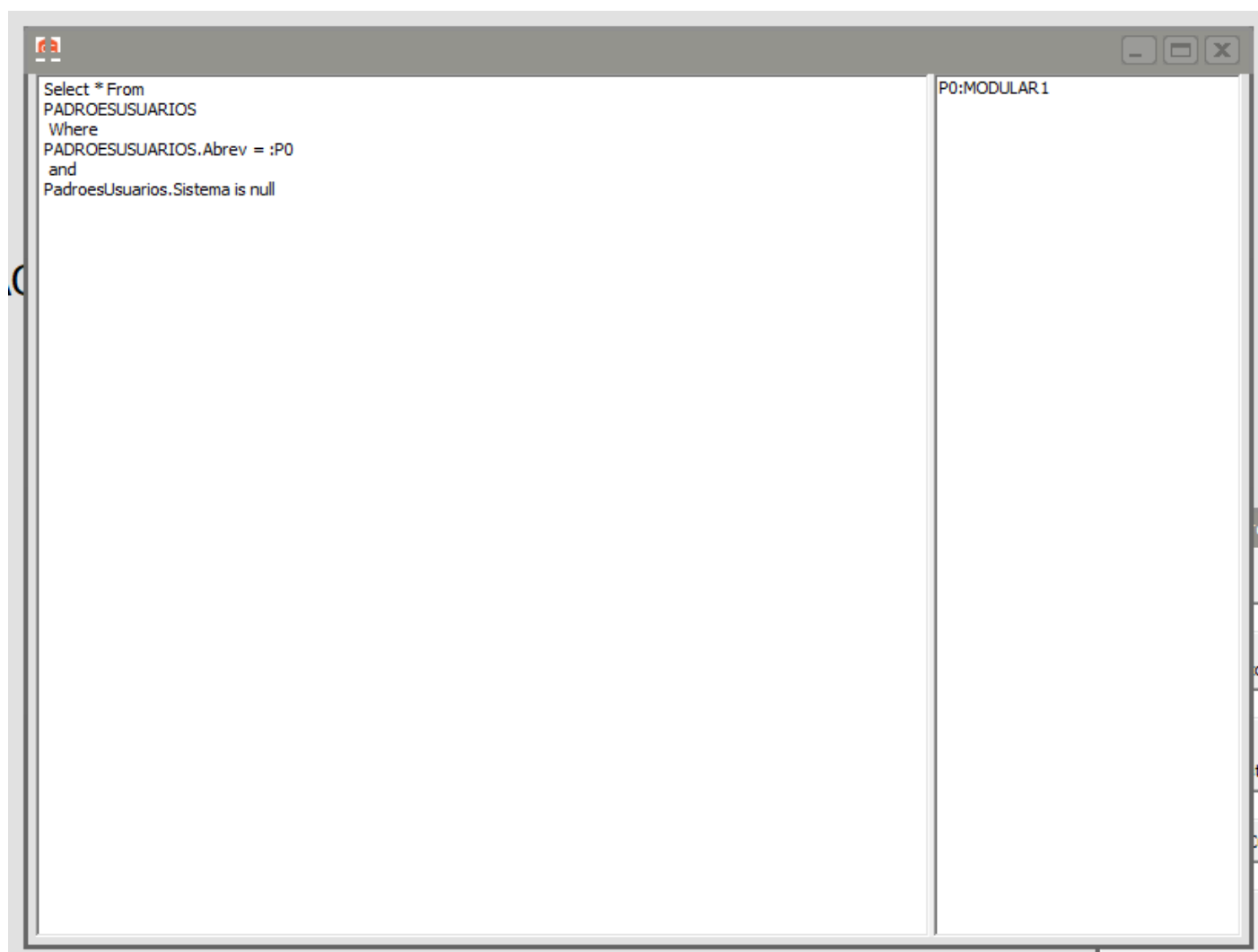
Devemos lembrar de adicionar também as chaves estrangeiras que estarão na nossa nova tabela, no exemplo citado, sendo os itens Usuarios.Nome e CenCust.Custo;

Podemos ver de maneira mais exemplificada como o critérios funciona realizando o debug do código, vamos lembrar que até o momento nenhuma query de SQL foi feita, porem ao usarmos o

"MostraParams" no nosso DataModule com a QManut por exemplo:



iremos obter esse resultado ao voltar para o sistema:



resumindo, o próprio componente ao ser configurado, monta essa query.

Evento OnSelect do Dlg:

Podemos utilizar a criação do Evento OnSelect do Dlg para escrever trechos que possam ser úteis na tela, por exemplo:

```
procedure TForm_PADROESUSUARIOS_EDIT.DB_E_NOMELOGINSelect(Sender: TObject; DataSet: TDataSet);
begin
    inherited;
    if (TExisteRegistro.Verificar(
        'PADROESUSUARIOS', 'Abrev;Sistema;NomeLog',
        [TGlobalInfoSistema.AbrevEmp, TGlobalInfoSistema.Sigla, DM_PADROESUSUARIOS.QMANUT.FieldByName('NomeLog').AsString],
        ['S', 'S', 'S'])) then
    begin
        MessageDialog.Show('Usuário já cadastrado nesta empresa', mtInformation, [mbOK], 0);
        DB_E_NOMELOGIN.SetFocus();
        DM_PADROESUSUARIOS.QMANUT.FieldByName('NomeLog').Clear();
        Abort();
    end;
    ED_NOMECompleto.Text := DataSet.FieldByName('Nome').AsString;
    DM_PADROESUSUARIOS.QMANUT.FieldByName('NomeLog').AsString := DataSet.FieldByName('NomeLog').AsString;
end;
```

Nesse trecho estamos fazendo uma validação para verificar se o usuário selecionado já tem um centro de custo vinculado a ele naquela empresa. Caso ele não tenha, trazemos não só o Campo NomeLog para o Field, mas trazemos também o campo nome(que seria o nome completo) para o Edit de nome do usuário. Exemplificando na tela do sistema:

Ao clicar no Dlg do Edit

Nos deparamos com a tela de usuários:

Nome	Nome para Login
Marcos Biagiotti	MBIAGIOTTI
Marcos Biazotto	MBIAZOTTO
Micael Costa	MCOSTA
MURILO DE FREITAS PEIGO	MFREITAS
Nelson Ferracin	NFERRACIN
Orlando Silva	OSILVA
Penoel Junior	PJUNIOR
Thainá Russo	TRUSSO
Thiago Gomes	TGOMES

Ao clicar em um usuário, ele já puxa o campo nome para o Edit:

Vincular Centro de Custos aos Usuários

Nome p/ Login
MBIAGIOTTI

Nome Completo
Marcos Biagiotti

Centro de Custos

Descrição do Custo

A mesma coisa acontece com o centro de custo:

Vincular Centro de Custos aos Usuários

Nome p/ Login
MBIAGIOTTI

Nome Completo
Marcos Biagiotti

Centro de Custos

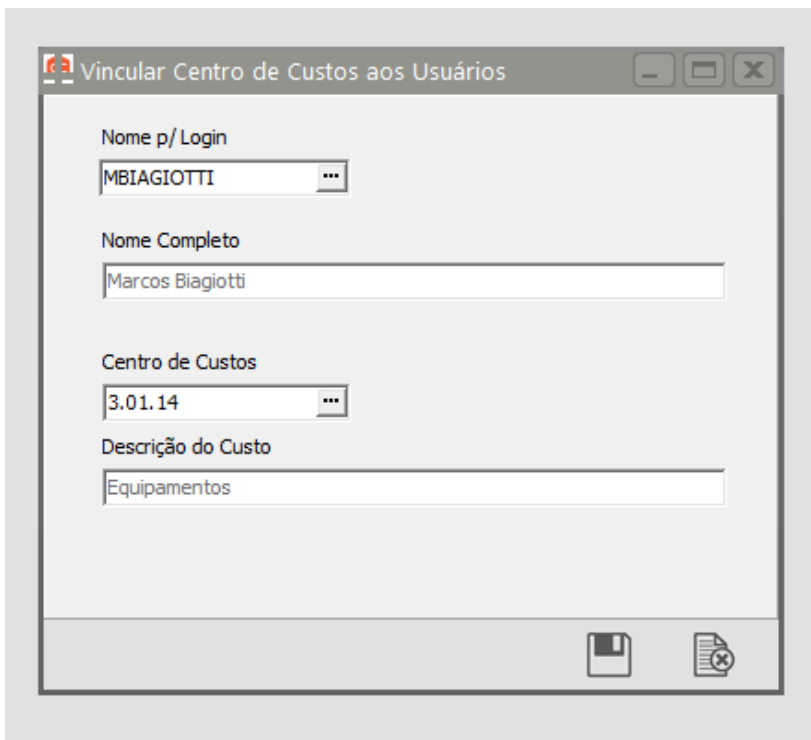
Descrição do Custo

Centros de Custo

Pesquisa por: Local
Chave:
Estruturas: Todas, Ativas
☒ Pesq.somente início da palavra
☐ Exibe todos os níveis da estrutura?
Pesquisa

Código	Custo	Local	Ativo
3.01.06	Ferramentas Fabrica		S
3.01.07	Manutenção Maquina Flow		S
3.01.10	Material de Escritorio		S
3.01.14	Equipamentos		S
3.01.15	Consultoria em Gestão ISO 9001		S
3.01.16	Exames Adm/Dem/Laudos/Analise.		S
3.01.17	Relogio de Ponto		S
3.01.21	Placas de Sinalização - Fca		S
3.01.22	Locação - Caçamba		S
3.01.25	Corpo de Prova		S
3.01.26	Uniformes		S

Ao clicarmos em um centro de custo, ele puxa para a tela, tanto o centro de custo escolhido, como a descrição do mesmo:



Vincular Centro de Custos aos Usuários

Nome p/ Login
MBIAGIOTTI

Nome Completo
Marcos Biagiotti

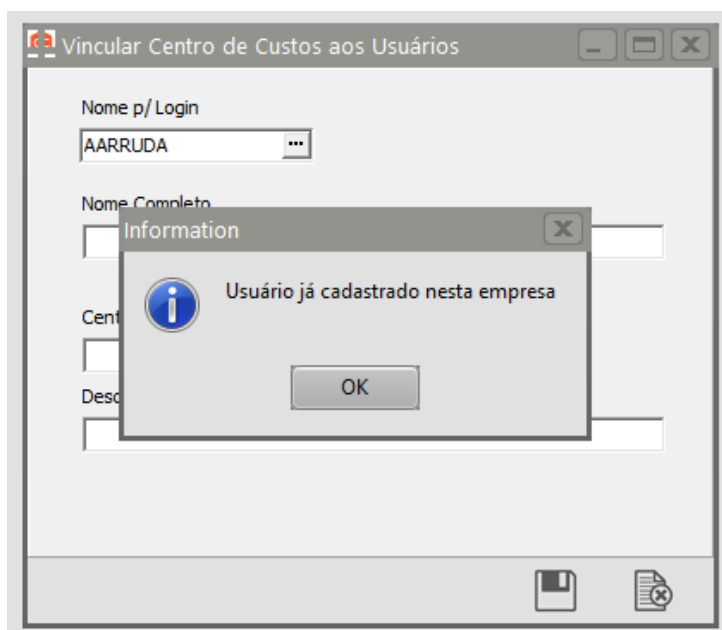
Centro de Custos
3.01.14

Descrição do Custo
Equipamentos

Validações:

É importante lembrar que mesmo já esteja funcional, devemos validar algumas informações, como o botão gravar, o botão fechar, validar se o usuário selecionado já tem um centro de custo cadastrado. Para essas validações, podemos usar eventos, como o FormClose, OnClick, OnSelect entre outros, Por exemplo:

MODULAR CC	ACRISTINA	3.01.21
MODULAR CC	AMEIRELES	3.01.14
MODULAR CC	AMENDES	3.01.15
MODULAR CC	CCARVALHO	3.01.06
MODULAR CC	CFALCHI	3.01.16
MODULAR CC	FHAIDAR	3.01.15
MODULAR CC	MDIAS	3.01.22
MODULAR CC	MROSA	3.01.16
MODULAR CC	TVENANCIO	3.01.06
MODULAR CC	WAGNER	3.01.26



Aqui vemos que ao tentar selecionar um usuário que já tinha centro de custo cadastrado, então o sistema deve dar um alerta e impedir que eu continue o processo.

Algumas functions/procedures que podem ajudar nesse passo de validações, são por exemplo:

`TVerificaCampos.Verificar();`

```
(TVerificaCampos.Verificar(DM_PADROESUSUARIOS.QMANUT, FCampos))
```

Ela basicamente verifica quais campos do DataModule foram alterados, sendo passados como parâmetros o DataModule em si e um array de strings com os nomes dos campos alterados declarada no "private"

```
private
{ Private declarations }
FCampos: Variant;
```

do tipo Variant, da seguinte forma:

DataModule.DataSource.DataSet.Modified;

Faz a verificação se houve alteração nos campos do DataModule

```
begin
if (DM_PADROESUSUARIOS.DS_MANUT.DataSet.Modified) and (TVerificaCampos.Verificar(DM_PADROESUSUARIOS.QMANUT, FCampos)) then
begin
```

TExisteRegistro.Verificar()

A função TExisteRegistro, faz a verificação se já existe um registro em uma tabela específica com campos específicos passados por parâmetros, sendo 4 ao total, os parâmetros que são passados os seguintes:

- Tabela que vai buscar;
- Os campos que serão verificados;
- Os conteúdos que serão buscados
- O tipo primitivo dos dados

Ficando no código dessa maneira:

```
if (TExisteRegistro.Verificar(
'PADROESUSUARIOS', 'Abrev;Sistema;NomeLog',
[TGlobalInfoSistema.AbrevEmp, TGlobalInfoSistema.Sigla, DM_PADROESUSUARIOS.QMANUT.FieldByName('NomeLog').AsString],
['S', 'S', 'S'])) then
```

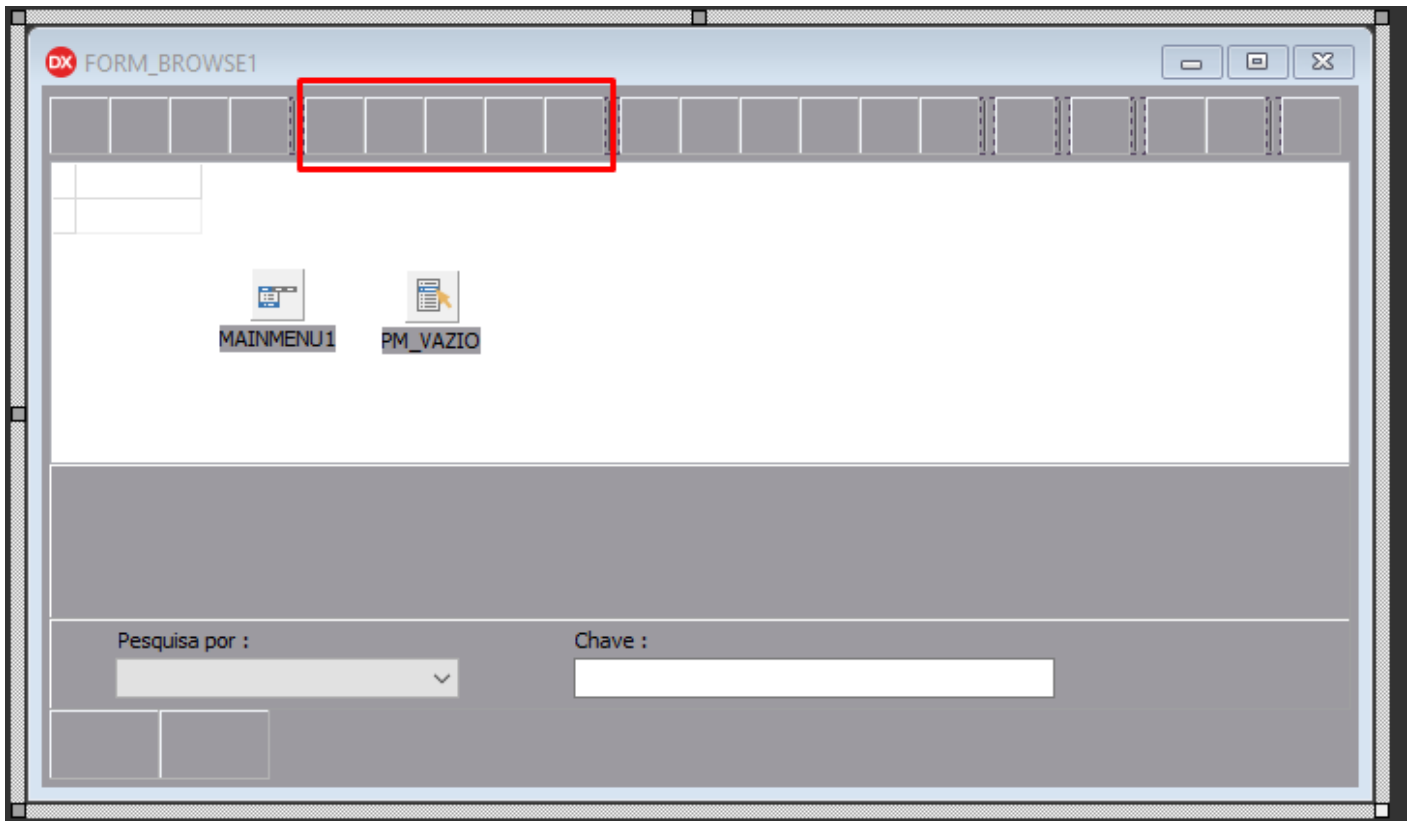
DataModule.Gravar()

É inserido no BT_GRAVA da tela, basicamente salva no banco o que foi adicionado no DataModule, pode ser utilizado dentro de um try para facilitar a manipulação:

```
procedure TForm_PADROESUSUARIOS_EDIT.BT_GRAVAClick(Sender: TObject);
begin
    inherited;
    try
        DM_PADROESUSUARIOS.Gravar();
        if (Modo = atNovo) then
        begin
            NovoRegistro();
            ED_NOMECompleto.Clear();
            ED_DESCRICAOCENCUST.Clear();
        end
        else
            Close();
    except
        ShowMessage('Erro ao gravar');
    end;
end;
```

Tela de manutenção

A tela pai FORM_BROWSE já vem com os botões criados por herança porem as funções de cada botão devem ser adicionadas manualmente, pois diversas funções usam o DataModule individual do documento, sendo assim, os botões não fazem nada quando criados.



Três desses botões (**Novo**, **Edita** e **Visualiza**), utilizam linhas de código praticamente iguais:

```
procedure TForm_PADROESUSUARIOS_BROWSE.BT_NOVOClick(Sender: TObject);
begin
    inherited;
    if (TCriaJanela.Criar(FORM_PADROESUSUARIOS_EDIT, 'TFORM_PADROESUSUARIOS_EDIT')) then
        FORM_PADROESUSUARIOS_EDIT := TForm_PADROESUSUARIOS_EDIT.Create(AtNovo, Self, False, False);
end;

procedure TForm_PADROESUSUARIOS_BROWSE.BT_VISUALIZAClick(Sender: TObject);
begin
    inherited;
    if (TCriaJanela.Criar(FORM_PADROESUSUARIOS_EDIT, 'TFORM_PADROESUSUARIOS_EDIT')) then
        FORM_PADROESUSUARIOS_EDIT := TForm_PADROESUSUARIOS_EDIT.Create(AtVisualiza, Self, False, False);
end;
```

Eles tem o mesmo código, pois abrem o mesmo formulário, a única mudança é em um dos parâmetros, mudando o Modo, usado da unit uAberturaTipo e a partir disso fazer as mudanças necessárias no Evento OnCreate da tela de cadastro, deixando informações em **ReadOnly** ou deixando todas com a propriedade **Enabled** como "False" para que o usuário apenas visualize as informações (**AtVisualiza**).

O botão de **Excluir**, é necessário muito cuidado pois muitas validações devem ser feitas antes de um registro ser excluído, como por exemplo, no caso da tela "Usuários", não é possível permitir

que o SUPERVISOR seja excluído, então essa validação é usada:

```
procedure TForm_USUARIOS_BROWSE.BT_ELIMINAClick(Sender: TObject);
begin
  if DM_USUARIOS.DS_GRID.dataset.fieldbyname('NomeLog').asString = 'SUPERVISOR' then
  begin
    MessageDialog.Show('Não é possível excluir o SUPERVISOR.', mtinformation, [mbok], 0);
    exit;
  end;
end;
```

Botão "Avançar" da tela de manutenção:

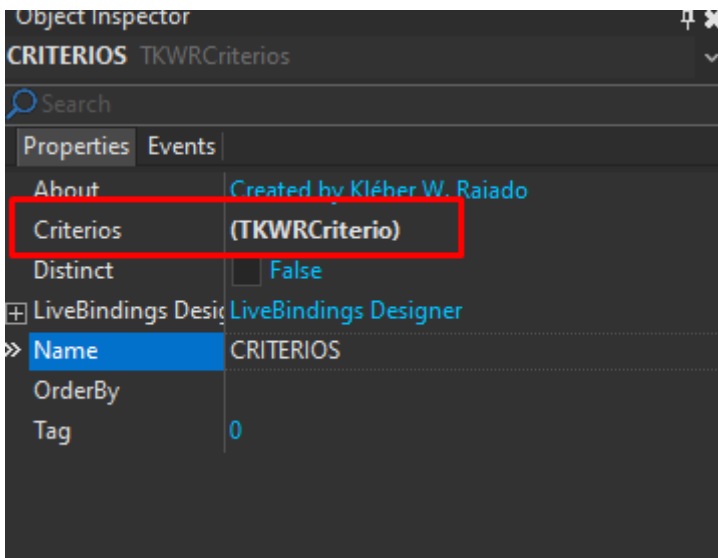
Para fazermos o botão de selecionar próximo na tela de manutenção funcionar, não é necessário montar uma query específica para isso, o próprio componente "CRITERIOS" faz isso para nós.

inicialmente temos que declarar a unit de manutenção no DataModule.

```
unit PadroesUsuarios.DataModule;
interface

uses
  System.SysUtils, System.Classes,
  DMTAB1,
  FireDAC.Stan.Intf,
  FireDAC.Stan.Option,
  FireDAC.Stan.Param,
  FireDAC.Stan.Error,
  FireDAC.DatS,
  FireDAC.Phys.Intf,
  FireDAC.DApt.Intf,
  FireDAC.Stan.Async,
  FireDAC.DApt,
  FireDAC.Comp.Client,
  KWRCriterios,
  Data.DB,
  FireDAC.Comp.DataSet,
  Data.SqlTimSt,
  uGlobalInfoSistema,
  PadroesUsuarios.Manutencao;
```

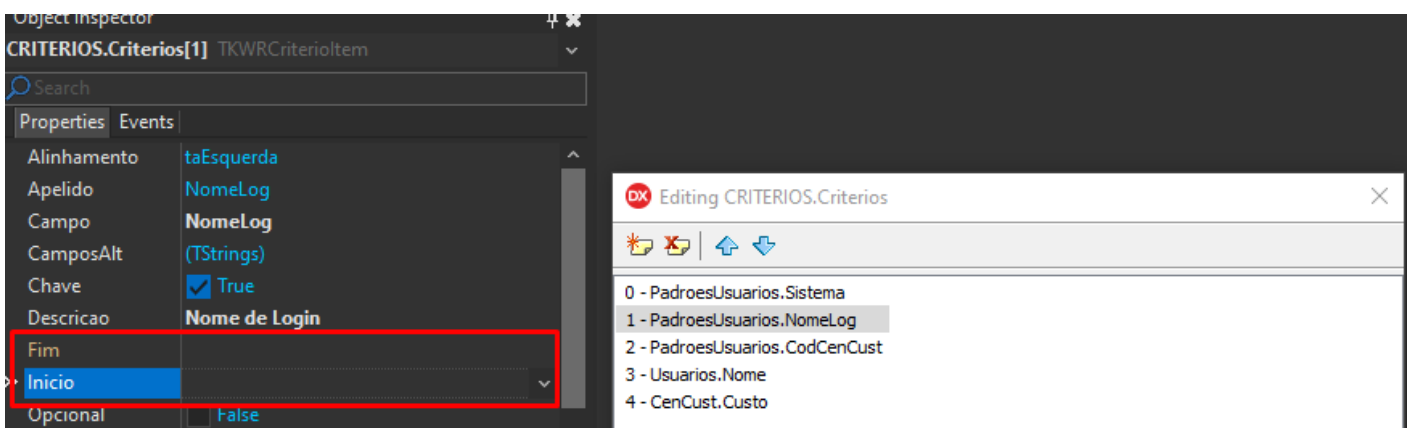
Após isso devemos clicar no componente "CRITERIOS" do DataModule e fazer algumas alterações na propriedade Criterios.



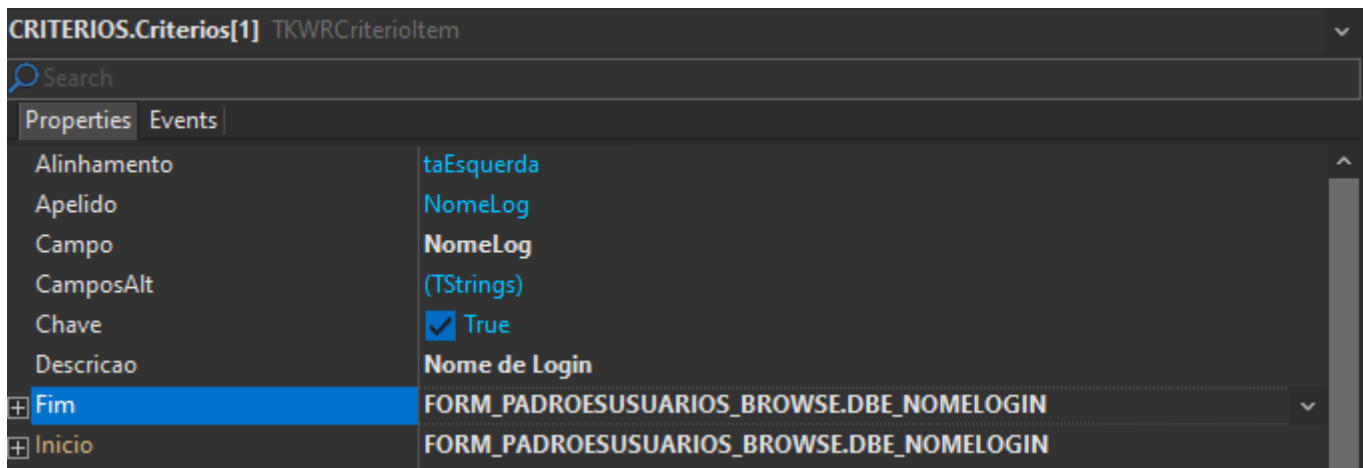
a tela que será aberta já foi documentada aqui porem vamos alterar algumas propriedades que não foram citadas ainda.

Primeiramente vamos selecionar o campo que queremos fazer a busca, no nosso exemplo, O PadroesUsuarios.NomeLog.

Agora vamos nos atentar as propriedades "**Inicio**" e "**Fim**"



Clicando na seta no fim do campo, serão exibidos todos os componentes da unit que foi declarada e a partir desse ponto, temos apenas que ligar o inicio e o fim ao mesmo componente.



e após isso, ao clicar no botão Avançar, na tela de manutenção, ele já faz a pesquisa sozinho com base nos componentes ligados ao componente que foi relacionado ao campo, não necessitando a montagem de uma query só pra isso.

Após fazer esses passos, a unit de manutenção, pode ser retirada das declarações de uses do DataModule.

Revision #34

Created 25 October 2022 15:02:04

Updated 16 November 2022 14:11:49 by Felipe De Paula Vieira Da Silva