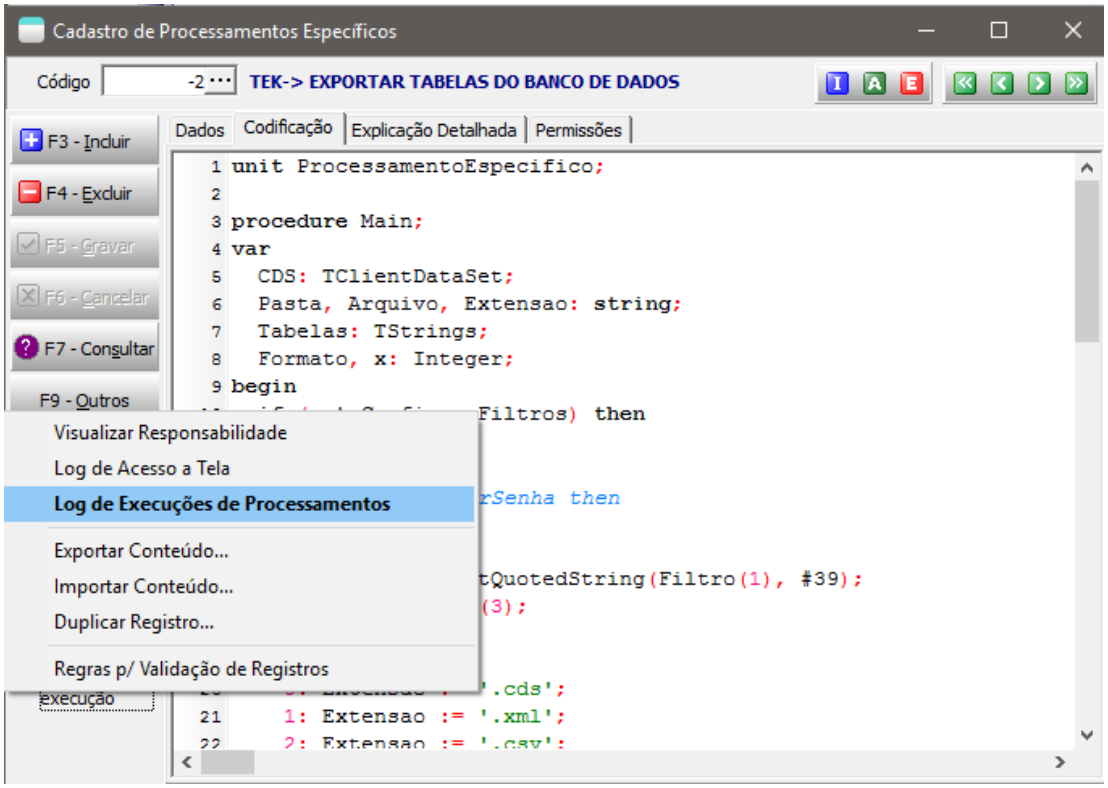
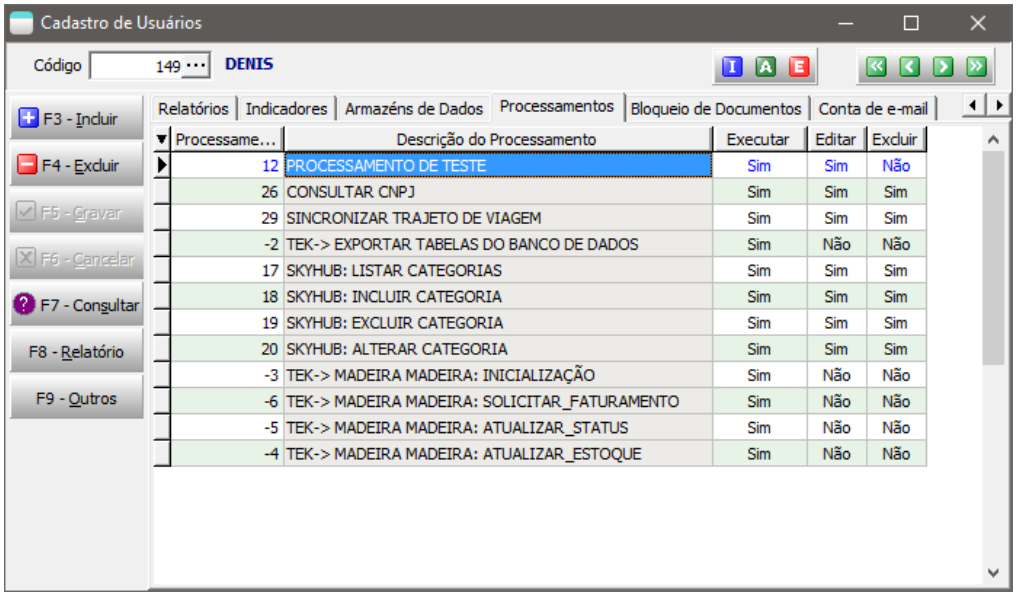
Novidades presentes no módulo de BI (Inteligência de Negócios e Gerador de Relatórios) da Tek-System

Inovações

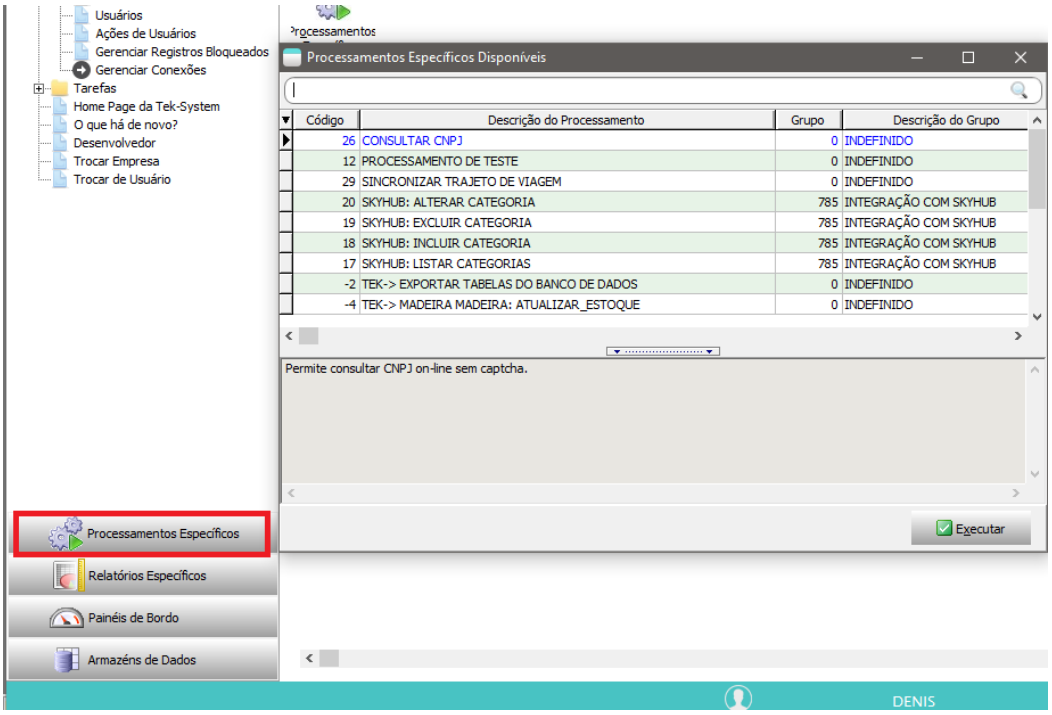
- Criado o cadastro de processamentos específicos. Que permitirá ter interações com o usuário para executar funções específicas. Este cadastro contará também com o registro de log de alterações, exclusões e execuções.



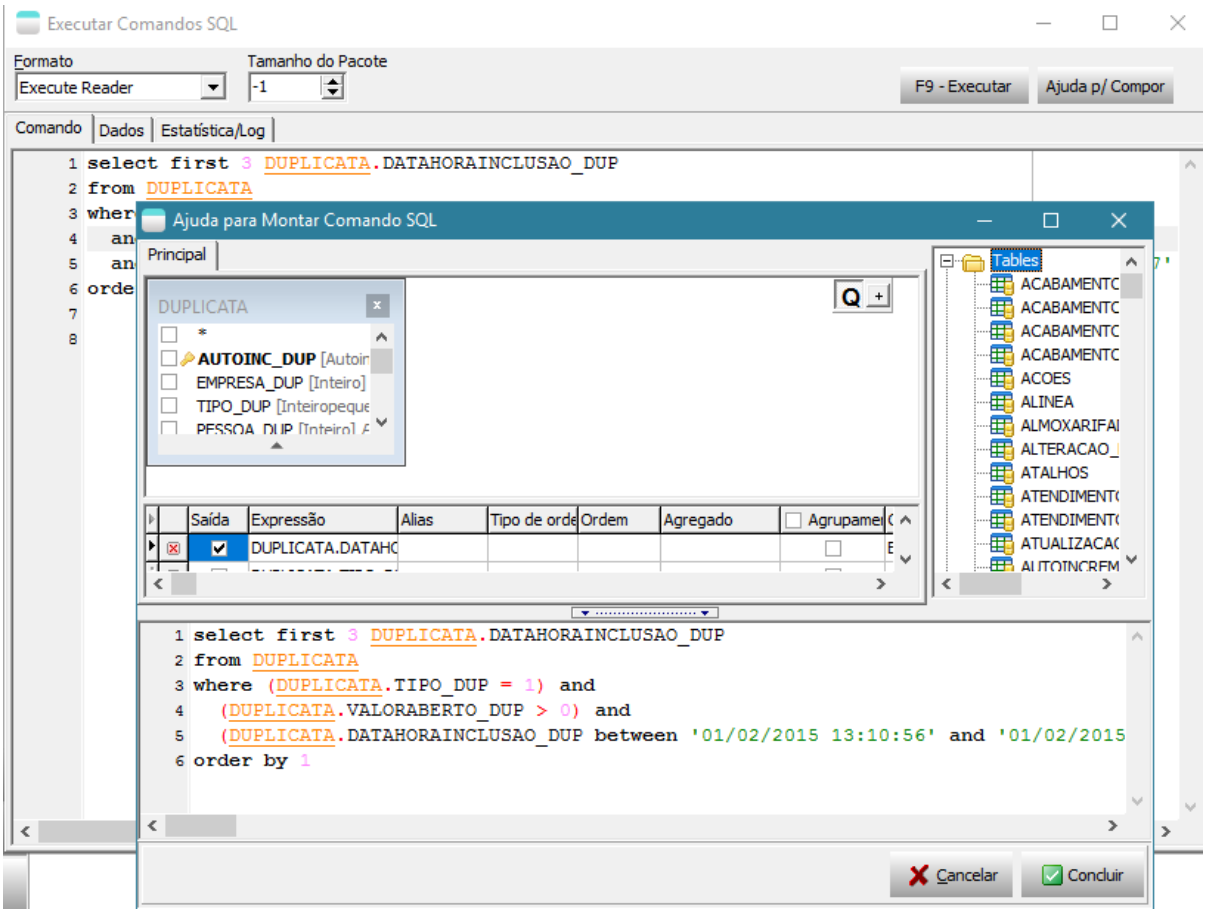
- Criada possibilidade de definir as permissões para execução, edição, exclusão de processamentos específicos, pelo cadastro de usuário:



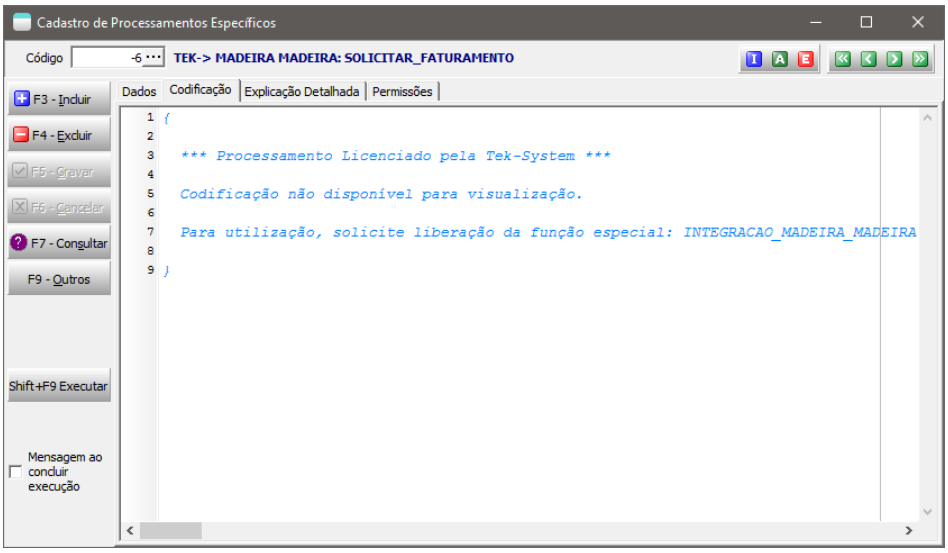
- Disponibilizado botão em todos os módulos que permitirá abrir tela para consultar e executar processamentos específicos:



- Tela para execução de comandos SQL agora está sempre disponível no menu e possui atalho para ajuda para compor.

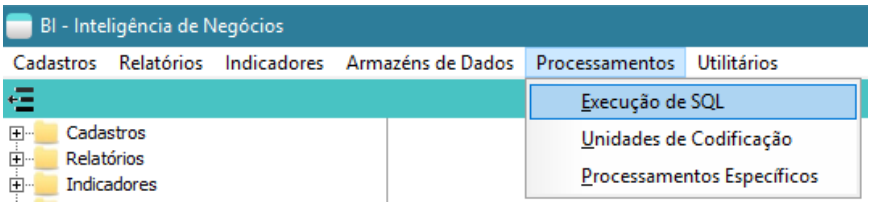


- Tek-system passará a disponibilizar algumas unidades de codificação (units) e processamentos interpetados sob licença de uso. Por exemplo, para integração com e-commerces. Estas unidades estarão nos bancos de dados dos clientes, mas somente poderão ser utilizadas após liberação financeira.

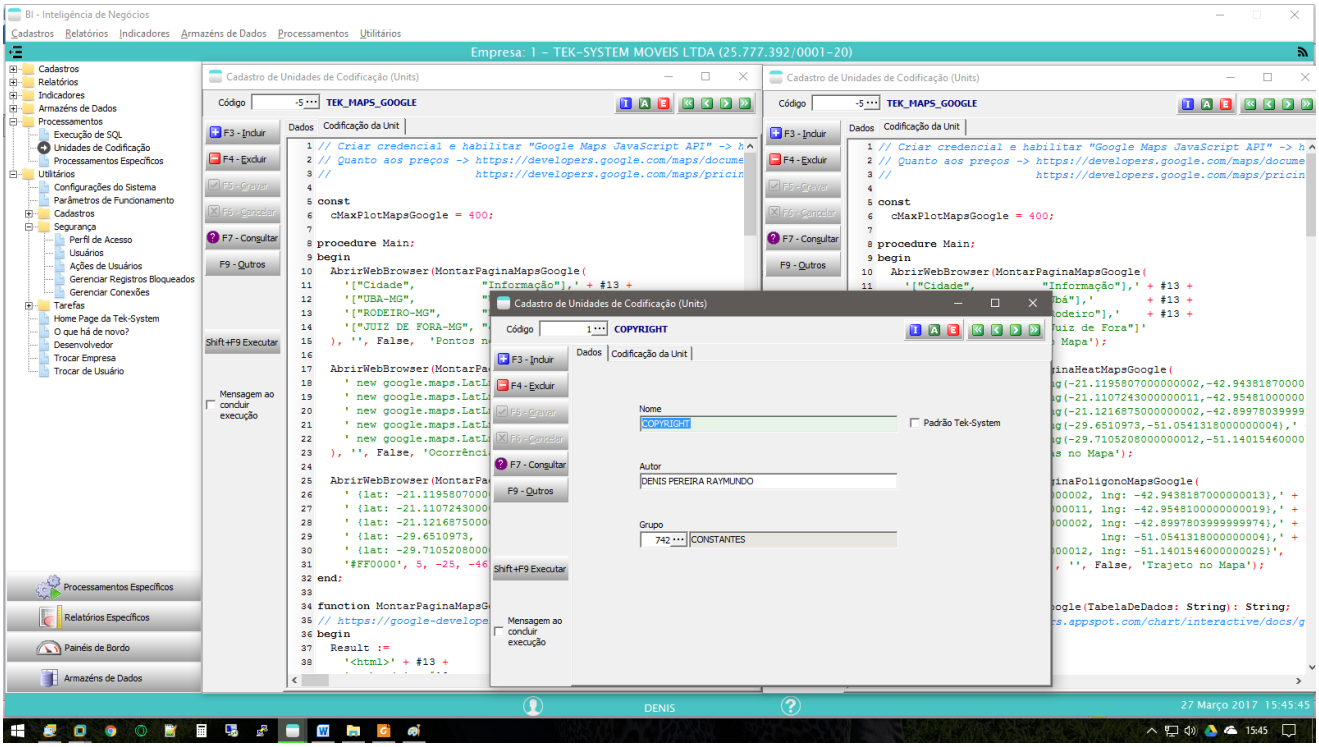


Melhorias na Interface

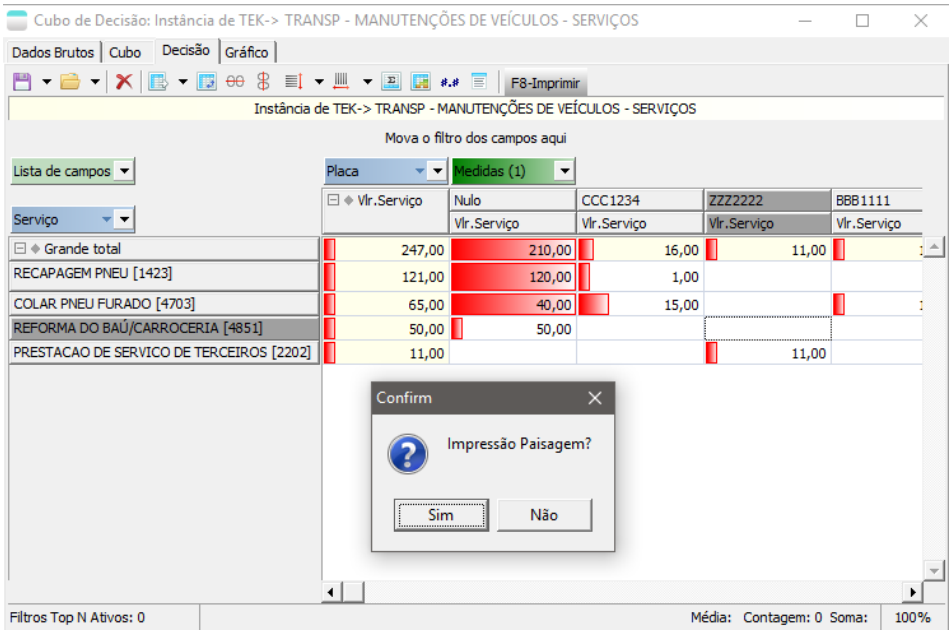
- Criação do menu Processamentos, agrupando funções relacionadas.



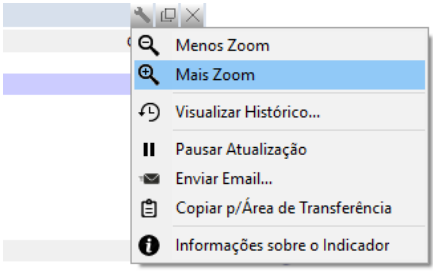
- Possibilidade de abrir o cadastro de unidades de codificação mais de uma vez simultaneamente. Facilitando a manipulação de unidades que se interagem entre si. Adição da tecla de atalho SHIFT+F9 para executar. Possibilidade de dar mensagem ao concluir execução.



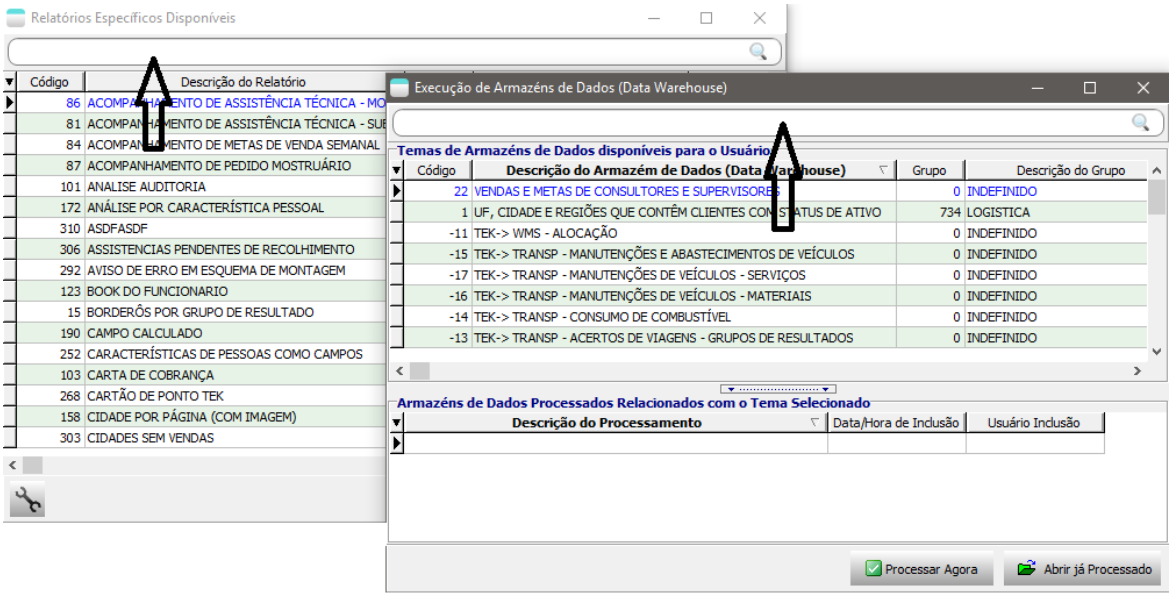
- Permitir escolher o formato (retrato/paisagem) da impressão de DW (Armazéns de Dados) antes de exibir o relatório.



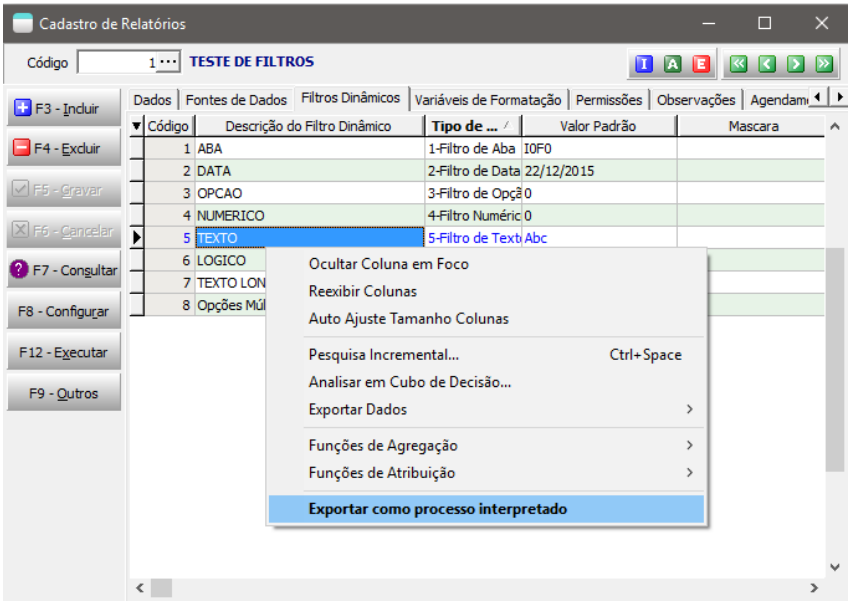
- Ao ampliar ou reduzir o Zoom em Indicadores, o menu continuará aberto. Permitindo dar mais cliques.



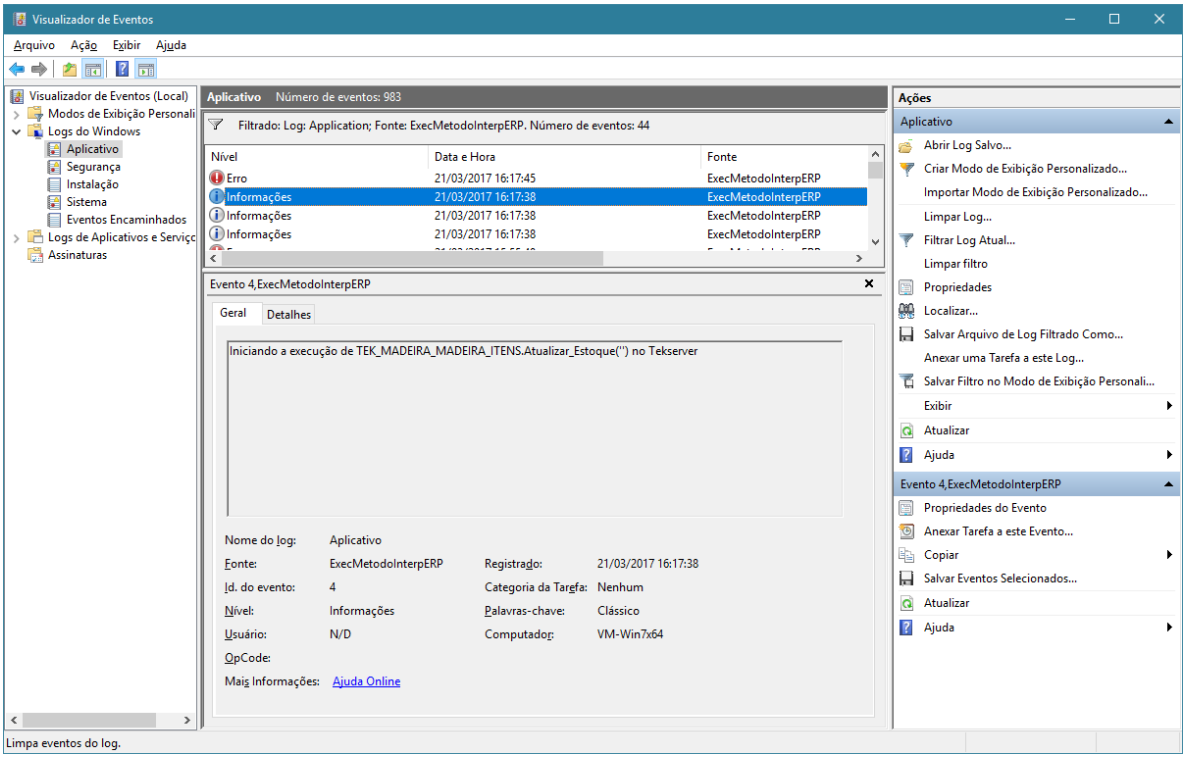
- Adicionada possibilidade de pesquisar relatórios específicos e armazéns de dados:



- Permitir exportar os filtros dinâmicos de um relatório, como processo interpretado. Isto facilitará a codificação para uso da função ExecutarFiltroDinamico. Após exportado, é só colocar da área de transferência a codificação gerada.



- Registrar nos eventos do windows as execuções do sistema **ExecMetodoInterpERP**.



Novos indicadores padrões disponibilizados:

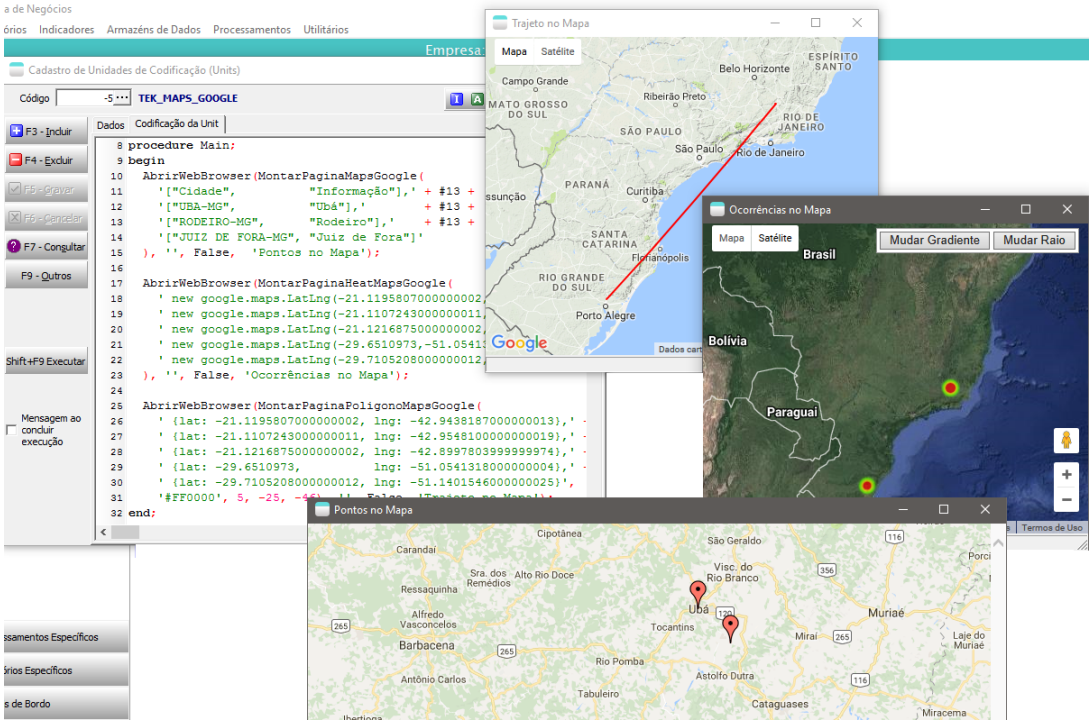
FAT - Vlr Faturamento Supervisor Mês Atual
FAT - Vlr Faturamento Supervisor Mês Atual - Por Linha
FAT - Vlr Faturamento Supervisor Mês Atual - Por Consultor
FAT - Vlr Faturamento Supervisor Mês Atual - Por Grupo
FAT - Vlr Vendas Supervisor Mês Atual
FAT - Vlr Vendas Supervisor Mês Atual - Por Consultor
FAT - Vlr Meta de Vendas Supervisor Mês Atual
FAT - Percentual Atingimento Meta de Vendas Supervisor - Mês Atual
FAT - Percentual Atingimento Meta de Vendas Por Consultor - Mês Atual
FAT - Percentual Atingimento Meta de Vendas Por Supervisor - Mês Atual
ODBC-> Vlr.Pedidos de Venda - Hoje(Comércio)
ODBC-> Vlr.Pedidos de Entrega - Hoje(Comércio)
ODBC-> Vlr.Pedidos de Venda - Últimos 30 dias(Comércio)
ODBC-> Vlr.Pedidos de Entrega - Últimos 30 dias(Comércio)
ODBC-> Top 10 Vlr. Ven. Produtos - Últimos 30 dias(Comércio)
ODBC-> Top 10 à Pagar Por Centro de Custo(Comércio) (Possui Detalhamento por Fornecedor)
ODBC-> Vlr. à Pagar Vencendo no Próx. Dia Útil(Comércio) (Possui Detalhamento por Fornecedor)
ODBC-> Vlr. à Pagar Vencendo Até Hoje(Comércio) (Possui Detalhamento por Fornecedor)
ODBC-> Vlr. de Cheques a Depositar até Hoje(Comércio) (Possui Detalhamento por Cheque)
ODBC-> Vlr.Contas a Receber Vencidas(Comércio) (Possui detalhamento por Cliente)
ODBC-> % Contas a Receber Vencidas(Comércio)

Novos modelos de relatórios padrões que podem ser agendados:

COBRANÇA DE ENTREGA DE PEDIDO DE COMPRA ATRASADO
AVISO DE PROXIMIDADE DE DATA DE ENTREGA DE PEDIDO DE COMPRA

Unidade de codificação ajustada

- TEK_MAPS_GOOGLE: disponibilizada a função MontarPaginaPoligonoMapsGoogle e ajuste no método main para abrir WebBrowsers de exemplo.



Novas classes e funções disponibilizadas para interpretação

- ExecuteCommandODBC - Permitirá executar comandos INSERT/UPDATE/DELETE em outros bancos de dados via conexão ODBC. Interessante, por exemplo, para registrar logs em outros bancos ou realizar integrações entre sistemas distintos.
- EnderecoWebProcessado2 - Agora conta com parâmetros para passar os cabeçalhos e tipo de conteúdo. Isto permitirá realizar integrações com um maior número de sites. Por exemplo, aqueles que requerem autenticação passada no cabeçalho e tipos de arquivos JSON.
- SetVariavelContexto e GetVariavelContexto - Permitirão armazenar variáveis dentro do contexto da conexão. Permitindo que o conteúdo seja reaproveitado entre mais de um processamento.
- AbrirWebBrowser - Permitirá abrir um site ou uma página HTML em um processo interpretado.
- AbrirCuboDeDecisao - Permitirá analisar uma fonte de dados em cubo de decisão em um processo interpretado.
- ExecutarFiltroDinamico - Permitirá abrir uma tela para que o usuário possa escolher os filtros para um processamento.
- SelecionarRegistros - Permitirá abrir uma tela para que o usuário possa confirmar a marcação de registros no meio de um processamento interpretado.
- DialogoAbrirArquivo, DialogoSalvarArquivo e DialogoEscolherPasta - Permitirão interagir com as telas de abertura e salvamento de arquivos e escolha de pasta.
- LogDoProcessamentoAdd - Permitirá adicionar mensagens ao log de execução de um processamento específico.
- ConfirmarSenha - Permitirá que execuções perigosas dentro de processamentos específicos sejam confirmadas através de senha.
- MostrarLogTexto, MostrarLogArquivo, MostrarLogDataSet - Permitirão exibir uma tela com logs para o usuário.
- CallBack_AbreTela, CallBack_Incremento, CallBack_FechaTela - Permitirão passar informações do progresso de processamentos.
- Classes e funções para manipulação de arquivos JSON. Arquivos JSON são muito utilizados hoje em dia para realizar integrações, por exemplo, com e-commerces. Para detalhes sobre arquivos JSON veja http://desenvolvimentoparaweb.com/javascript/json-javascript-object-notation/):
    - TJSONAncestor, TJSONString, TJSONPair, TJSONObject, TJSONArray - Classes nativas para manipulação de objetos.
    - TJSONWriter, TJSONTextWriter, TJSONObjectWriter, TStringWriter - Classes para escrita.
    - TJSONReader, TJSONTextReader, TStringReader - Classes para leitura.
    - TJSONPairEnumerator, TJSONArrayEnumerator - Classes para percorrer elementos.
    - JSONToString e StringToJSON - Converter de um formato para outro.
    - JSONFormatado - Permite formatar uma string JSON para leitura por humanos.
    - GetObjectJSON, GetArrayJSON e GetValueJSON - Capturar Objetos, Arrays e Valores em strings JSON.
    - IsJSONArray e IsJSONObject - Verificar o tipo de um TJSONValue.
    - JSONPairToField, JSONObjectToFields, JSONArrayToRecords, JSONToCDS -> Facilitadores para captura de informação e disponibilização em TClientDataSet.
    - TClientDataSet.ToJson - Função facilitadora para escrita de informação JSON a partir de ClientDataSet.
- Classe TClassProc_ImportacaoDocumento -> Facilitará ter a estrutura para inclusão de pedidos através de processos interpretados.
- Classe TClassProc_ImportacaoPessoa -> Facilitará ter a estrutura para inclusão de pessoas através de processos interpretados.

Codificações para testes das classes/funções

Deve-se criar uma unidade de codificação ou processamento específico, copiar a codificação e mandar executar.

- **ExecuteCommandODBC**

```
procedure Main;
begin
  ShowMessage(TestarFuncoesODBC);
end;

function TestarFuncoesODBC: String;
var RegAnt, RegUpdate, RegInsert, RegDelete, RegPos: Integer;
begin
  RegAnt    := ExecuteScalarODBC('ERP4g', 'select count(*) from teste');

  RegInsert := ExecuteCommandODBC('ERP4g', 'insert into teste(autoincremento, email)
values(1, ''aaa@aaa.com'')');

  RegUpdate := ExecuteCommandODBC('ERP4g', 'update teste set email = ''xxx@xxx.com''
where autoincremento = 1');

  RegDelete := ExecuteCommandODBC('ERP4g', 'delete from teste where autoincremento =
1');

  RegPos    := ExecuteScalarODBC('ERP4g', 'select count(*) from teste');

  Result :=
    'RegAnt...: ' + IntToStr(RegAnt)    + #13 +
    'RegInsert: ' + IntToStr(RegInsert) + #13 +
    'RegUpdate: ' + IntToStr(RegUpdate) + #13 +
    'RegDelete: ' + IntToStr(RegDelete) + #13 +
    'RegPos...: ' + IntToStr(RegPos);
end;
```

- **EnderecoWebProcessado2, JSONToString, StringToJSON**

```
const
  Host_Integracao  = 'https://in.skyhub.com.br/';
  ConteudoTipoJson = 'application/json';
  Path_Categorias  = 'categories';
  Integracao_Email = '' {Deve ser configurado};
  Integracao_Token = '' {Deve ser configurado};

var Cabecalho: String;

{ ***** API PRINCIPAL DE INTEGRAÇÃO (INICIO) ***** }
procedure PreencheCabecalhoAutenticacao;
var L: TStringList;
begin
  L := TStringList.Create;
  try
    L.Add('X-User-Email: ' + Integracao_Email);
    L.Add('X-User-Token: ' + Integracao_Token);
    L.Add('Accept: '       + ConteudoTipoJson);
    L.Add('Content-Type: ' + ConteudoTipoJson);

    Cabecalho := L.Text;
```

```
  finally
    L.Free;
  end;
end;

function SkyHub_Get(Arquivo: String; AcrescentarExtensao: Boolean): string;
var Link: String;
begin
  Link := Host_Integracao + Arquivo;
  if AcrescentarExtensao then
    Link := Link + '.json';
  Result := EnderecoWebProcessado2(Link, 'GET', '', 0, 0, Cabecalho,
ConteudoTipoJson, ConteudoTipoJson);
end;

function SkyHub_Post(Arquivo, CorpoPostIncluir: String; AcrescentarExtensao:
Boolean): Boolean;
var Link, Json, Retorno: String;
begin
  Link := Host_Integracao + Arquivo;
  if AcrescentarExtensao then
    Link := Link + '.json';

  Json := StringToJSON(CorpoPostIncluir);

  Retorno := EnderecoWebProcessado2(Link, 'POST', Json, 0, 0, Cabecalho,
ConteudoTipoJson, ConteudoTipoJson);

  Result := (Trim(Retorno) = '');
end;

function SkyHub_Put(Pasta, Codigo, CorpoPutAlterar: String): Boolean;
var Link: String;
begin
  Link := Host_Integracao + Pasta + '/' + Codigo;
  try
    EnderecoWebProcessado2(Link, 'PUT', StringToJSON(CorpoPutAlterar), 0, 0,
Cabecalho, ConteudoTipoJson, ConteudoTipoJson);
    Result := True;
  except
    on E: Exception do
      if (Pos('HTTP/1.1 404 Not Found', E.Message) > 0) then
        Result := False // Retorno quando não encontra o registro a alterar
      else
        raise; // Demais erros não tratados devem continuar a serem apresentados
  end;
end;

function SkyHub_Delete(Pasta, Codigo: String): Boolean;
var Link: String;
begin
  Link := Host_Integracao + Pasta + '/' + Codigo;
  try
    EnderecoWebProcessado2(Link, 'DELETE', '', 0, 0, Cabecalho, ConteudoTipoJson,
ConteudoTipoJson);
    Result := True;
  except
    on E: Exception do
```

```
      if (Pos('HTTP/1.1 404 Not Found', E.Message) > 0) then
        Result := False // Retorno quando não encontra o registro a excluir
      else
        raise; // Demais erros não tratados devem continuar a serem apresentados
  end;
end;
{ ***** API PRINCIPAL DE INTEGRAÇÃO (FIM) ***** }

{ ***** CATEGORIAS (INÍCIO) ***** }
function ListaCategorias: String;
begin
  Result := SkyHub_Get(Path_Categorias, True);
end;

function IncluirCategoria(Codigo, Descricao: String): Boolean;
begin
  Result := SkyHub_Post(Path_Categorias, Format(Categorias_JsonIncluir, [Codigo,
Descricao]), True);
end;

function AlterarCategoria(Codigo, NovaDescricao: String): Boolean;
begin
  Result := SkyHub_Put(Path_Categorias, Codigo, Format(Categorias_JsonAlterar,
[NovaDescricao]));
end;

function ExcluirCategoria(Codigo: String): Boolean;
begin
  Result := SkyHub_Delete(Path_Categorias, Codigo);
end;
{ ***** CATEGORIAS (FIM) ***** }

procedure Main;
begin
  PreencheCabecalhoAutenticacao;
  IncluirCategoria('1', 'MÓVEIS');
  AlterarCategoria('1', 'Móveis');
  ShowMessage(JSONToString(ListaCategorias));
  ExcluirCategoria('1');
end;
```

- **SetVariavelContexto e GetVariavelContexto**

```
procedure Main;
begin
  ShowMessage(GetVariavelContexto('teste'));
  SetVariavelContexto('teste', 123);
  SetVariavelContexto('DataHora', now);
  SetVariavelContexto('Teste de Caractéres Especiais', 'aàá');
  SetVariavelContexto('TesteString', 'conteúdo');
  ShowMessage(GetVariavelContexto('teste'));
end;
```

- **AbrirWebBrowser**

Veja unidade de codificação (unit) TEK_MAPS_GOOGLE disponibilizada como padrão.

- **AbrirCuboDeDecisao**

```
procedure Main;
begin
  AbrirCuboDeDecisao(
    ExecuteReader(
      'select * from DUPLICATA where DUPLICATA.TIPO_DUP = 1 and
DUPLICATA.VALORABERTO_DUP > 0'));
end;
```

- **ExecutarFiltroDinamico**

```
function Main: OleVariant;
var CDSFiltrosDisp, CDSFiltrosExec: TClientDataSet;
begin
  CDSFiltrosDisp := TClientDataSet.Create;
  CDSFiltrosExec := TClientDataSet.Create;
  try
    // Escolha uma das 3 formas de filtro abaixo e comente as demais
    //CDSFiltrosDisp.Data := FiltrosDoRelatorio(1);
    //CDSFiltrosDisp.Data := FiltrosDoDW(1);
    CDSFiltrosDisp.Data := FiltroManualDeExemplo;

    CDSFiltrosExec.Data := ExecutarFiltroDinamico(CDSFiltrosDisp.Data);

    Result := CDSFiltrosExec.Data;

    if (not CDSFiltrosExec.IsEmpty) then
      begin
        MostrarCDS(CDSFiltrosExec);
        AbrirWebBrowser(CDSFiltrosExec.XMLData);

        ShowMessage(
          Filtro('Aba')             + #13 +
          Filtro('Aba', 'campo')    + #13 +
          Filtro('Data', 1)         + #13 +
          IntToStr(Filtro('Opcao')) + #13 +
          Filtro('Numerico')        + #13 +
          Filtro('Texto')           + #13 +
          Filtro('Logico')          + #13 +
          Filtro('Memo')            + #13 +
          Filtro('Opcoes Multiplas'));
      end;
   finally
     CDSFiltrosDisp.Free;
     CDSFiltrosExec.Free;
   end;
end;

function FiltroManualDeExemplo: OleVariant;
var CDS: TClientDataSet;
begin
  CDS := TClientDataSet.Create;
  try
    CDS.Data := EstruturaDeFiltrosDinamicos;

{   // 1ª Forma: Inclusão de registros diretamente
    CDS.Insert;
    CDS.FieldByName('Codigo').AsInteger   := 1;
```

```
    CDS.FieldByName('Descricao').AsString := 'Aba';
    CDS.FieldByName('Tipo').AsInteger     := cTipoFiltro_Aba;
    CDS.FieldByName('Nome').AsString      := 'SCliente';
    CDS.FieldByName('Padrao').AsString    := 'I0F999999';
    CDS.Post;

    CDS.Insert;
    CDS.FieldByName('Codigo').AsInteger   := 2;
    CDS.FieldByName('Descricao').AsString := 'Data';
    CDS.FieldByName('Tipo').AsInteger     := cTipoFiltro_Data;
    CDS.FieldByName('Padrao').AsString    := 'INICIO_MES_ATUAL';
    CDS.Post;

    CDS.Insert;
    CDS.FieldByName('Codigo').AsInteger   := 3;
    CDS.FieldByName('Descricao').AsString := 'Opcao';
    CDS.FieldByName('Tipo').AsInteger     := cTipoFiltro_Opcao;
    CDS.FieldByName('Opcoes').AsString    := 'A'#13'B'#13'C';
    CDS.FieldByName('Padrao').AsString    := '2';
    CDS.Post;

    CDS.Insert;
    CDS.FieldByName('Codigo').AsInteger   := 4;
    CDS.FieldByName('Descricao').AsString := 'Numerico';
    CDS.FieldByName('Tipo').AsInteger     := cTipoFiltro_Numerico;
    CDS.FieldByName('Padrao').AsString    := 'ANO_ATUAL';
    CDS.Post;

    CDS.Insert;
    CDS.FieldByName('Codigo').AsInteger   := 5;
    CDS.FieldByName('Descricao').AsString := 'Texto';
    CDS.FieldByName('Tipo').AsInteger     := cTipoFiltro_Texto;
    CDS.FieldByName('Padrao').AsString    := Nome_Usuario_Atual;
    CDS.Post;

    CDS.Insert;
    CDS.FieldByName('Codigo').AsInteger   := 6;
    CDS.FieldByName('Descricao').AsString := 'Logico';
    CDS.FieldByName('Tipo').AsInteger     := cTipoFiltro_Logico;
    CDS.FieldByName('Padrao').AsString    := 'S';
    CDS.Post;

    CDS.Insert;
    CDS.FieldByName('Codigo').AsInteger   := 7;
    CDS.FieldByName('Descricao').AsString := 'Memo';
    CDS.FieldByName('Tipo').AsInteger     := cTipoFiltro_Memo;
    CDS.FieldByName('Opcoes').AsString    := 'Explique o caso aqui';
    CDS.Post;

    CDS.Insert;
    CDS.FieldByName('Codigo').AsInteger   := 8;
    CDS.FieldByName('Descricao').AsString := 'Opcoes Multiplas';
    CDS.FieldByName('Tipo').AsInteger     := cTipoFiltro_OpcaoMultipla;
    CDS.FieldByName('Opcoes').AsString    := 'AAAA'#13'BBBB'#13'CCCC';
    CDS.FieldByName('Padrao').AsString    := '1';
    CDS.Post;
}
```

```
   // 2ª Forma: Inclusão através de função genérica
    //                        Descrição          Tipo                        Padrao
Mascara  Nome da Aba  Opções
    IncluirFiltroDinamico(CDS, 'Aba',             cTipoFiltro_Aba,
'I0F999999',        '',      'SCliente',  '');
    IncluirFiltroDinamico(CDS, 'Data',            cTipoFiltro_Data,
'INICIO_MES_ATUAL', '',      '',          '');
    IncluirFiltroDinamico(CDS, 'Opcao',           cTipoFiltro_Opcao,         '2',
'',      '',          'A'#13'B'#13'C');
    IncluirFiltroDinamico(CDS, 'Numerico',        cTipoFiltro_Numerico,
'ANO_ATUAL',        '',      '',          '');
    IncluirFiltroDinamico(CDS, 'Texto',           cTipoFiltro_Texto,
Nome_Usuario_Atual, '',      '',          '');
    IncluirFiltroDinamico(CDS, 'Logico',          cTipoFiltro_Logico,        'S',
'',      '',          '');
    IncluirFiltroDinamico(CDS, 'Memo',            cTipoFiltro_Memo,          '',
'',      '',          'Explique o caso aqui');
    IncluirFiltroDinamico(CDS, 'Opcoes Multiplas', cTipoFiltro_OpcaoMultipla, '1',
'',      '',          'AAAA'#13'BBBB'#13'CCCC');


{   // 3ª Forma: Inclusão através de funções específicas
    IncluirFiltroAba            (CDS, 'Aba',              'I0F999999',
'SCliente');
    IncluirFiltroData           (CDS, 'Data',             'INICIO_MES_ATUAL');
    IncluirFiltroOpcao          (CDS, 'Opcao',            '2',
'A'#13'B'#13'C');
    IncluirFiltroNumerico       (CDS, 'Numerico',         'ANO_ATUAL');
    IncluirFiltroTexto          (CDS, 'Texto',            Nome_Usuario_Atual, '');
    IncluirFiltroLogico         (CDS, 'Logico',           'S');
    IncluirFiltroMemo           (CDS, 'Memo',             'Explique o caso aqui');
    IncluirFiltroOpcoesMultiplas(CDS, 'Opcoes Multiplas', '1',
'AAAA'#13'BBBB'#13'CCCC');
}

    Result := CDS.Data;
  finally
    CDS.Free;
  end;
end;

function IncluirFiltroAba(CDS: TClientDataSet; Descricao, Padrao, NomeAba: String):
Integer;
begin
  Result := IncluirFiltroDinamico(CDS, Descricao, cTipoFiltro_Aba, Padrao, '',
NomeAba, '');
end;

function IncluirFiltroData(CDS: TClientDataSet; Descricao, Padrao: String): Integer;
begin
  Result := IncluirFiltroDinamico(CDS, Descricao, cTipoFiltro_Data, Padrao, '', '',
'');
end;

function IncluirFiltroOpcao(CDS: TClientDataSet; Descricao, Padrao, Opcoes: String):
Integer;
begin
```

```
  Result := IncluirFiltroDinamico(CDS, Descricao, cTipoFiltro_Opcao, Padrao, '', '',
Opcoes);
end;

function IncluirFiltroNumerico(CDS: TClientDataSet; Descricao, Padrao: String):
Integer;
begin
  Result := IncluirFiltroDinamico(CDS, Descricao, cTipoFiltro_Numerico, Padrao, '',
'', '');
end;

function IncluirFiltroTexto(CDS: TClientDataSet; Descricao, Padrao, Mascara: String):
Integer;
begin
  Result := IncluirFiltroDinamico(CDS, Descricao, cTipoFiltro_Texto, Padrao, Mascara,
'', '');
end;

function IncluirFiltroLogico(CDS: TClientDataSet; Descricao, Padrao: String):
Integer;
begin
  Result := IncluirFiltroDinamico(CDS, Descricao, cTipoFiltro_Logico, Padrao, '', '',
'');
end;

function IncluirFiltroMemo(CDS: TClientDataSet; Descricao, Padrao: String): Integer;
begin
  Result := IncluirFiltroDinamico(CDS, Descricao, cTipoFiltro_Memo, '', '', '',
Padrao);
end;

function IncluirFiltroOpcoesMultiplas(CDS: TClientDataSet; Descricao, Padrao, Opcoes:
String): Integer;
begin
  Result := IncluirFiltroDinamico(CDS, Descricao, cTipoFiltro_OpcaoMultipla, Padrao,
'', '', Opcoes);
end;

function FiltrosDoRelatorio(CodigoRelatorio: Integer): OleVariant;
begin
  Result := ExecuteReader(
    ' select ' + #13 +
    '   GR_FILTROS.CODIGO_GRFILTRO            Codigo,'    + #13 +
    '   GR_FILTROS.DESCRICAO_GRFILTRO         Descricao,' + #13 +
    '   GR_FILTROS.TIPO_GRFILTRO              Tipo,'      + #13 +
    '   GR_FILTROS.NOME_GRFILTRO              Nome,'      + #13 +
    '   GR_FILTROS.DESCRICAOOPCOES_GRFILTRO   Opcoes,'    + #13 +
    '   GR_FILTROS.PADRAO_GRFILTRO            Padrao,'    + #13 +
    '   GR_FILTROS.MASCARA_GRFILTRO           Mascara'    + #13 +
    ' from GR_FILTROS' + #13 +
    ' where GR_FILTROS.RELATORIO_GRFILTRO = ' + IntToStr(CodigoRelatorio));
end;

function FiltrosDoDW(CodigoDW: Integer): OleVariant;
begin
  Result := ExecuteReader(
    ' select ' + #13 +
    '   DW_FILTROS.CODIGO_DWFILTRO            Codigo,'    + #13 +
```

```
    '   DW_FILTROS.DESCRICAO_DWFILTRO         Descricao,' + #13 +
    '   DW_FILTROS.TIPO_DWFILTRO              Tipo,'      + #13 +
    '   DW_FILTROS.NOME_DWFILTRO              Nome,'      + #13 +
    '   DW_FILTROS.DESCRICAOOPCOES_DWFILTRO   Opcoes,'    + #13 +
    '   DW_FILTROS.PADRAO_DWFILTRO            Padrao,'    + #13 +
    '   DW_FILTROS.MASCARA_DWFILTRO           Mascara'    + #13 +
    ' from DW_FILTROS' + #13 +
    ' where DW_FILTROS.TEMADW_DWFILTRO = ' + IntToStr(CodigoDW));
end;
```

- **SelecionarRegistros**

```
procedure Main;
const SQL = 'select 0 "Marque", CODIGO_BANCO "Banco", DESCRICAO_BANCO "Descrição do
Banco", CODCAMARACOMPENSACAO_BANCO from BANCO';
var CDS: TClientDataSet;
begin
  CDS := TClientDataSet.Create;
  try
    CDS.Data := ExecuteReader(SQL);

    CDS.FieldByName('CODCAMARACOMPENSACAO_BANCO').Visible := True;
    CDS.FieldByName('CODCAMARACOMPENSACAO_BANCO').DisplayLabel := 'Cód.Câmara';
    CDS.FieldByName('CODCAMARACOMPENSACAO_BANCO').DisplayWidth := 18;

    CDS.Data := SelecionarRegistros(CDS, 1, 'Escolha os bancos'); // Respeita
configurações em TFields
//    CDS.Data := SelecionarRegistros(CDS.Data, 1, 'Escolha os bancos'); // Não
respeita configurações em TFields

    if (CDS.Data <> null) then
      begin
        CDS.Filter   := 'Marque = 1';
        CDS.Filtered := True;
        MostrarCDS(CDS, True, 'Bancos Selecionados');
      end;
  finally
    CDS.Free;
  end;
end;
```

- **DialogoAbrirArquivo, DialogoSalvarArquivo e DialogoEscolherPasta**

```
procedure Main;
begin
  ShowMessage(DialogoEscolherPasta('Escolha a pasta para salvar os arquivos', 'c:',
'c:\temp'));
  ShowMessage(DialogoAbrirArquivo('c:', 'Texto Puro (*.txt)|*.txt|Arquivos de
Configuração (*.ini)|*.ini', 'c:\temp\entrada.txt', 'Quais arquivos devo processar?',
True));
  ShowMessage(DialogoSalvarArquivo('c:\', 'Texto Puro (*.txt)|*.txt|Arquivos de
Configuração (*.ini)|*.ini', 'c:\temp\saida.txt', 'Onde devo salvar o resultado?'));
end;
```

- **LogDoProcessamentoAdd**

```
unit ProcessamentoEspecifico;

procedure Main;
var F: String;
begin
  if MessageDlg('Confirma a execução deste processamento?', mtConfirmation, [mbyes,
mbno], 0) <> mrYes then Exit;

  LogDoProcessamento := LogDoProcessamento + '1-Declarou';
  F := '24695180000032108';

  LogDoProcessamentoAdd('2-Mostrou mensagem');
  ShowMessage(StrToFloat(F));

  try
    raise exception.Create('caraca, deu erro no meio da bagaça');
  except
    LogDoProcessamentoAdd('Erro engolido');
  end;

  LogDoProcessamentoAdd('3-Acabou');
  ShowMessage(LogDoProcessamento);
end;

end.
```

- **ConfirmarSenha**

```
Substitua a linha com MessageDlg no processamento anterior por:
  if not ConfirmarSenha then
```

- **MostrarLogTexto, MostrarLogArquivo, MostrarLogDataSet**

```
unit ProcessamentoEspecifico;

const NomeArquivo = 'C:\Temp\LOG.txt';

procedure Main;
var CDS: TClientDataSet;
begin
  CDS := TClientDataSet.Create;
  try
    CDS.Data := ExecuteReader('select first 10 * from ACOES order by AUTOINC_ACAO
desc');

    ExportarCDSparaTXT(CDS, NomeArquivo);

    MostrarLogArquivo(NomeArquivo, 'Log de Ações');

    MostrarLogDataSet(CDS, 'DESCRICAO_ACAO', '', False);

    MostrarLogTexto('LOG EM TEXTO', 'TITULO DA TELA');
  finally
    CDS.Free;
  end;
end;

end.
```

- **CallBack_AbreTela, CallBack_Incremento, CallBack_FechaTela**

```
procedure Main;
const
  MaxPrimeiroLoop     = 4;
  MaxSegundoLoopLongo = 1000;
  MaxSegundoLoopCurto = 20;
var I, y: integer;
begin
 // Exibe a mensagem com a barra de progresso
 CallBack_AbreTela(ClassOwner);
 for I := 1 to MaxPrimeiroLoop do
   begin
     CallBack_Incremento(ClassOwner, I, MaxPrimeiroLoop, 'Processamento Principal: '
+ IntToStr(I) + '/' + IntToStr(MaxPrimeiroLoop));

     CallBack_AbreTela(ClassOwner);
     for Y := 1 to MaxSegundoLoopLongo do
       CallBack_Incremento(ClassOwner, Y, MaxSegundoLoopLongo, 'Progresso : ' +
IntToStr(Y));
     CallBack_FechaTela(ClassOwner);
   end;

  CallBack_FechaTela(ClassOwner);

  // Exibe a mensagem sem a barra de progresso
  CallBack_AbreTela(ClassOwner);
  for I := 1 to MaxPrimeiroLoop do
    begin
     CallBack_Mensagem(ClassOwner, 'Processamento Particionado: ' + IntToStr(I) + '/'
+ IntToStr(MaxPrimeiroLoop));
     CallBack_AbreTela(ClassOwner);
     for Y := 1 to MaxSegundoLoopCurto do
       begin
         CallBack_Mensagem(ClassOwner, 'Parte: ' + IntToStr(Y));
         Sleep(100);
       end;
     CallBack_FechaTela(ClassOwner);
   end;

 CallBack_FechaTela(ClassOwner);
end;
```

- **JSONFormatado, TStringWriter, TJsonTextWriter, TJsonObjectWriter**

```
procedure Main;
var
  SW: TStringWriter;
  JTW: TJsonTextWriter;
  JOW: TJsonObjectWriter;
  CDS: TClientDataSet;
begin
  SW  := TStringWriter.Create;
  JTW := TJsonTextWriter.Create(SW);
  JOW := TJsonObjectWriter.Create(False);
  CDS := TClientDataSet.Create;
  try
    CDS.Data := ExecuteReader('select first 3 CODIGO_BANCO, DESCRICAO_BANCO from
BANCO order by 1');

    // Usando TJsonObjectWriter
    JOW.WriteStartArray;
      CDS.ToJSON(JOW);
    JOW.WriteEndArray;
    ShowMessage(JSONFormatado(JOW.JSON.ToString));

    // Usando TJsonTextWriter
    JTW.WriteStartArray;
       CDS.ToJSON(JTW);
    JTW.WriteEndArray;
    ShowMessage(JTW.Writer.ToString);
  finally
    JTW.Free;
    SW.Free;
    JOW.Free;
    CDS.Free;
  end;
end;
```

- **TJSONPairEnumerator, TJSONObject, TJSONPair, IsJSONArray, IsJSONObject**

```
procedure Main;
const
  ForcarArray = False;
  cJson       = '{"teste1": 1, "teste2": 2}';
var
  sJson: String;
  Objeto: TJSONObject;
  Enumerador: TJSONPairEnumerator;
  Par: TJSONPair;
begin
  if ForcarArray then
    sJson := '[' + cJson + ']'
  else
    sJson := cJson;

  Objeto := TJSONObject.ParseJsonValue(sJson);

  if IsJSONArray(Objeto) then
    ShowMessage('É um JSONArray')
  else if IsJSONObject(Objeto) then
    begin
      Enumerador := Objeto.GetEnumerator;
      try
        while Enumerador.MoveNext do
          begin
            Par := Enumerador.Current;
            ShowMessage(Par.ToString + #13 +
                        Par.JsonString.ToString + #13 +
                        Par.JsonValue.ToString);
          end;
      finally
        Enumerador.Free;
      end;
    end;
end;
```

- **JSONPairToField, TJSONObject, TJSONPairEnumerator**

```
procedure Main;
var
  JSONObject: TJSONObject;
  Enumerador: TJSONPairEnumerator;
  CDS: TClientDataSet;
begin
  CDS := TClientDataSet.Create;
  try
    JSONObject := TJSONObject(TJSONObject.ParseJSONValue(
      ' {' + #13 +
      '   "ITEM": 10,' + #13 +
      '   "NOME": "Descrição",' + #13 +
      '   "QTDE": 1100,' + #13 +
      '   "VLRUNIT": 1.5,' + #13 +
      '   "DATAJSON": "2017-03-22T00:00:00.00-03:00",' + #13 +
      '   "DATA1": "22/03/2017",' + #13 +
      '   "DATA2": "2017-03-22",' + #13 +
      '   "DATANULA1": null,' + #13 +
      '   "DATANULA2": "",' + #13 +
      '   "DATAHORAJSON": "2017-03-229T13:24:10.552-03:00",' + #13 +
      '   "SOBRA": "Não será capturado"' + #13 + // Existe no JSON mas não existe
campo interessado, então não será aproveitado.
      ' }'));

    CDS.FieldDefs.Add('ITEM',         ftInteger,   0, False);
    CDS.FieldDefs.Add('NOME',         ftString,   30, False);
    CDS.FieldDefs.Add('QTDE',         ftFloat,     0, False);
    CDS.FieldDefs.Add('VLRUNIT',      ftCurrency,  0, False);
    CDS.FieldDefs.Add('DATAJSON',     ftDate,      0, False);
    CDS.FieldDefs.Add('DATA1',        ftDate,      0, False);
    CDS.FieldDefs.Add('DATA2',        ftDate,      0, False);
    CDS.FieldDefs.Add('DATANULA1',    ftDate,      0, False);
    CDS.FieldDefs.Add('DATANULA2',    ftDate,      0, False);
    CDS.FieldDefs.Add('DATAHORAJSON', ftDateTime,  0, False);
    CDS.FieldDefs.Add('EXTRA',        ftString,   10, False);  // Não existe no JSON
então não será preenchido
    CDS.CreateDataSet;
    CDS.Insert;

    Enumerador := JSONObject.GetEnumerator;
    try
      while Enumerador.MoveNext do
        JSONPairToField(Enumerador.Current, CDS); // <<<<< Esta é a função criada
    finally
      Enumerador.Free;
    end;

    CDS.Post;

    MostrarCDS(CDS, True);
  finally
    CDS.Free;
  end;
end;
```

- **JSONObjectToFields, TJSONObject**

```
procedure Main;
var
  JSONObject: TJSONObject;
  CDS: TClientDataSet;
begin
  CDS := TClientDataSet.Create;
  try
    JSONObject := TJSONObject(TJSONObject.ParseJSONValue(
      ' {' + #13 +
      '   "ITEM": 10,' + #13 +
      '   "NOME": "Descrição",' + #13 +
      '   "QTDE": 1100,' + #13 +
      '   "VLRUNIT": 1.5,' + #13 +
      '   "DATAJSON": "2017-03-22T00:00:00.00-03:00",' + #13 +
      '   "DATA1": "22/03/2017",' + #13 +
      '   "DATA2": "2017-03-22",' + #13 +
      '   "DATANULA1": null,' + #13 +
      '   "DATANULA2": "",' + #13 +
      '   "DATAHORAJSON": "2017-03-229T13:24:10.552-03:00",' + #13 +
      '   "SOBRA": "Não será capturado"' + #13 + // Existe no JSON mas não existe
campo interessado, então não será aproveitado.
      ' }'));

    CDS.FieldDefs.Add('ITEM',         ftInteger,   0, False);
    CDS.FieldDefs.Add('NOME',         ftString,   30, False);
    CDS.FieldDefs.Add('QTDE',         ftFloat,     0, False);
    CDS.FieldDefs.Add('VLRUNIT',      ftCurrency,  0, False);
    CDS.FieldDefs.Add('DATAJSON',     ftDate,      0, False);
    CDS.FieldDefs.Add('DATA1',        ftDate,      0, False);
    CDS.FieldDefs.Add('DATA2',        ftDate,      0, False);
    CDS.FieldDefs.Add('DATANULA1',    ftDate,      0, False);
    CDS.FieldDefs.Add('DATANULA2',    ftDate,      0, False);
    CDS.FieldDefs.Add('DATAHORAJSON', ftDateTime,  0, False);
    CDS.FieldDefs.Add('EXTRA',        ftString,   10, False);  // Não existe no JSON
então não será preenchido
    CDS.CreateDataSet;

    CDS.Insert;
    JSONObjectToFields(JSONObject, CDS);  // <<<<< Esta é a função criada
    CDS.Post;

    MostrarCDS(CDS, True);
  finally
    CDS.Free;
  end;
end;
```

- **JSONArrayToRecords, TJSONArray**

```
const
  JSONBase =
      ' {' + #13 +
      '   "ITEM": 10,' + #13 +
      '   "NOME": "Descrição",' + #13 +
      '   "QTDE": 1100,' + #13 +
      '   "VLRUNIT": 1.5,' + #13 +
      '   "DATAJSON": "2017-03-22T00:00:00.00-03:00",' + #13 +
      '   "DATA1": "22/03/2017",' + #13 +
      '   "DATA2": "2017-03-22",' + #13 +
      '   "DATANULA1": null,' + #13 +
      '   "DATANULA2": "",' + #13 +
      '   "DATAHORAJSON": "2017-03-229T13:24:10.552-03:00",' + #13 +
      '   "SOBRA": "Não será capturado"' + #13 + // Existe no JSON mas não existe
campo interessado, então não será aproveitado.
      ' }';

procedure Main;
var
  JSONArray: TJSONArray;
  CDS: TClientDataSet;
begin
  CDS := TClientDataSet.Create;
  try
    JSONArray := TJSONArray(TJSONObject.ParseJSONValue('[' + JSONBase + ',' +
JSONBase + ',' + JSONBase + ']'));

    CDS.FieldDefs.Add('ITEM',         ftInteger,   0, False);
    CDS.FieldDefs.Add('NOME',         ftString,   30, False);
    CDS.FieldDefs.Add('QTDE',         ftFloat,     0, False);
    CDS.FieldDefs.Add('VLRUNIT',      ftCurrency,  0, False);
    CDS.FieldDefs.Add('DATAJSON',     ftDate,      0, False);
    CDS.FieldDefs.Add('DATA1',        ftDate,      0, False);
    CDS.FieldDefs.Add('DATA2',        ftDate,      0, False);
    CDS.FieldDefs.Add('DATANULA1',    ftDate,      0, False);
    CDS.FieldDefs.Add('DATANULA2',    ftDate,      0, False);
    CDS.FieldDefs.Add('DATAHORAJSON', ftDateTime,  0, False);
    CDS.FieldDefs.Add('EXTRA',        ftString,   10, False);  // Não existe no JSON
então não será preenchido
    CDS.CreateDataSet;

    JSONArrayToRecords(JSONArray, CDS);  // <<<<< Esta é a função criada

    MostrarCDS(CDS, True);
  finally
    CDS.Free;
  end;
end;
```

- **JSONToCDS**

```
const
  sJSON =
//    '{ "GERAL":' + #13 +
    '{' + #13 +
    '  "MomentoGeracaoArquivo":"2017-03-09T13:24:10.552-03:00",' + #13 +
    '  "Pedidos": [' + #13 +
    '    {' + #13 +
    '      "PEDIDO": 1,' + #13 +
    '      "DATA": "08/03/2017",' + #13 +
    '      "Prazos": [30, 60, 90],' + #13 +
    '      "Itens": [' + #13 +
    '          {"produto":' + #13 +
    '           {' + #13 +
    '            "ITEM": 10,' + #13 +
    '            "NOME": "Descrição do Item 10",' + #13 +
    '            "QTDE": 1100,' + #13 +
    '            "VLRUNIT": 1.5' + #13 +
    '           }' + #13 +
    '          },' + #13 +
    '          {"produto":' + #13 +
    '           {' + #13 +
    '            "ITEM": 20,' + #13 +
    '            "NOME": "Descrição do Item 20",' + #13 +
    '            "QTDE": 200,' + #13 +
    '            "VLRUNIT": 10.22' + #13 +
    '           }' + #13 +
    '          }' + #13 +
    '      ]' + #13 +
    '    },' + #13 +
    '    {' + #13 +
    '      "PEDIDO": 2,' + #13 +
    '      "DATA": "09/03/2017",' + #13 +
    '      "Prazos": [30, 60, 90, 120],' + #13 +
    '      "Itens": [' + #13 +
    '          {"produto":' + #13 +
    '           {' + #13 +
    '            "ITEM": 30,' + #13 +
    '            "NOME": "Descrição do Item 30",' + #13 +
    '            "QTDE": 50.3,' + #13 +
    '            "VLRUNIT": 1111.11' + #13 +
    '           }' + #13 +
    '          }' + #13 +
    '      ]' + #13 +
    '    }' + #13 +
    '  ],' + #13 +
    '  "Totalizacao": {' + #13 +
    '    "QtdePedidos": 2,' + #13 +
    '    "VlrTotalPedidos": 59582.833' + #13 +
    '  },' + #13 +
    '  "Imagens": [' + #13 +
    '   "AAA", "BBB", "CCC"' + #13 +
    '  ]' + #13 +
//    '}' + #13 +
    '}';

var
```

```
  CDSPedido, CDSProdPedido, CDSImagens, CDSTotalizacao: TClientDataSet;
  DS: TDataSource;

procedure Main;
var VetorCDS: Variant;
begin
  CDSPedido      := TClientDataSet.Create;
  CDSProdPedido  := TClientDataSet.Create;
  CDSTotalizacao := TClientDataSet.Create;
  CDSImagens     := TClientDataSet.Create;
  DS             := TDataSource.Create(nil);
  try
    PrepararEstruturaMestreDetalhe;

    VetorCDS    := VarArrayCreate([0, 3], varVariant);
    VetorCDS[0] := CDSPedido;
    VetorCDS[1] := CDSProdPedido;
    VetorCDS[2] := CDSTotalizacao;
    VetorCDS[3] := CDSImagens;

    JSONToCDS(sJSON, VetorCDS, 'Pedidos;Itens;Totalizacao;Imagens');  // <<<< ESTA É
A FUNÇÃO CRIADA

    MostrarCDS(CDSProdPedido,  False, 'Produtos do Pedido');
    MostrarCDS(CDSPedido,      True,  'Pedidos');
    MostrarCDS(CDSTotalizacao, True,  'Totalização');
    MostrarCDS(CDSImagens,     True,  'Imagens');
  finally
    DS.Free;
    CDSProdPedido.Free;
    CDSPedido.Free;
    CDSTotalizacao.Free;
    CDSImagens.Free;
  end;
end;

procedure PrepararEstruturaMestreDetalhe;
begin
  // Criação de estruturas
  CDSPedido.FieldDefs.Clear;
  CDSPedido.FieldDefs.Add('PEDIDO',    ftInteger,  0, False);
  CDSPedido.FieldDefs.Add('DATA',      ftDateTime, 0, False);
  CDSPedido.FieldDefs.Add('PRAZOS',    ftString,  50, False); // array simples
  CDSPedido.FieldDefs.Add('DESCRICAO', ftString,  30, False); // Campo que não existe
no JSON não será gravado
  CDSPedido.CreateDataSet;
  CDSPedido.IndexFieldNames := 'PEDIDO';

  CDSProdPedido.FieldDefs.Clear;
  CDSProdPedido.FieldDefs.Add('ITEM_PED', ftInteger,  0, False);
  CDSProdPedido.FieldDefs.Add('ITEM',     ftInteger,  0, False);
  CDSProdPedido.FieldDefs.Add('NOME',     ftString,  30, False);
  CDSProdPedido.FieldDefs.Add('QTDE',     ftCurrency, 0, False);
  CDSProdPedido.FieldDefs.Add('VLRUNIT',  ftCurrency, 0, False);
  CDSProdPedido.CreateDataSet;

  CDSTotalizacao.FieldDefs.Clear;
  CDSTotalizacao.FieldDefs.Add('QTDEPEDIDOS',     ftInteger,  0, False);
```

```
  CDSTotalizacao.FieldDefs.Add('VLRTOTALPEDIDOS', ftCurrency, 0, False);
  CDSTotalizacao.CreateDataSet;

  CDSImagens.FieldDefs.Clear;
  CDSImagens.FieldDefs.Add('IMAGENS', ftMemo, 0, False); // Neste caso, o campo deve
ter o mesmo nome que o elemento JSON
  CDSImagens.CreateDataSet;

  // Relacionamentos
  DS.DataSet := CDSPedido;
  CDSProdPedido.MasterSource    := DS;
  CDSProdPedido.IndexFieldNames := 'ITEM_PED'{;ITEM'};
  CDSProdPedido.MasterFields    := 'PEDIDO';
end;
```

- **JSONToString, TStringWriter, TJsonTextWriter, TJSONObjectWriter, TStringReader, TJSONTextReader, TClientDataSet.ToJSON, TJsonArray, TJsonValue, TJsonObject, TJSONArrayEnumerator, GetValueJson, GetObjectJson, GetArrayJson**

```
var
  JOW: TJsonObjectWriter;
  CDSPedido, CDSProdPedido: TClientDataSet;
  DS: TDataSource;
  VlrTotalPedidos: Currency;

procedure Main;
var sTempJSON: String;
begin
  TestarAcessoExterno;

  sTempJSON := TestarEscritaSimplesJOW;
  TestarLeituraSimples(sTempJSON);

  sTempJSON := TestarEscritaSimplesJTW;
  TestarLeituraSimples(sTempJSON);

  CDSPedido     := TClientDataSet.Create;
  CDSProdPedido := TClientDataSet.Create;
  DS            := TDataSource.Create(nil);
  JOW           := TJsonObjectWriter.Create(False);
  try
    VlrTotalPedidos := 0;
    PrepararEstruturaMestreDetalhe;
    sTempJSON := TestarEscritaMestreDetalhe;
    TestarLeituraMestreDetalhe(sTempJSON);
  finally
    DS.Free;
    CDSProdPedido.Free;
    CDSPedido.Free;
    JOW.Free;
  end;
end;

procedure TestarAcessoExterno;
begin

AbrirWebBrowser(JSONToString(EnderecoWebProcessado('http://www.nif.pt/?json=1&q=50944
2013')), '', False, 'Teste 1');
//
AbrirWebBrowser(EnderecoWebProcessado('http://in.skyhub.com.br/api/v1/attributes.json
'), '', False, 'Teste 2');
//
AbrirWebBrowser(EnderecoWebProcessado('https://in.skyhub.com.br/api/v1/attributes.jso
n'), '', False, 'Teste 3');
//
AbrirWebBrowser(EnderecoWebProcessado('https://gist.githubusercontent.com/keeguon/231
0008/raw/bdc2ce1c1e3f28f9cab5b4393c7549f38361be4e/countries.json'), '', False, 'Teste
4');
//
AbrirWebBrowser(EnderecoWebProcessado('https://gist.githubusercontent.com/jonasruth/6
1bde1fcf0893bd35eea/raw/10ce80ddeec6b893b514c3537985072bbe9bb265/paises-gentilicos-
google-maps.json'), '', False, 'Teste 5');
```

```
//
AbrirWebBrowser(EnderecoWebProcessado('https://apphom.correios.com.br/SigepMasterJPA/
AtendeClienteService/AtendeCliente?wsdl'), '', False, 'Teste 6');
end;

function TestarEscritaSimplesJOW: String;
begin
  JOW := TJsonObjectWriter.Create(False);
  try
    JOW.WriteStartObject;
      JOW.WritePropertyName('category');
      JOW.WriteStartObject;
        JOW.WritePropertyName('code');  JOW.WriteValue('1');
        JOW.WritePropertyName('name');  JOW.WriteValue('Categoria 1 com á');

        JOW.WriteProperty('code', '2');
        JOW.WriteProperty('name', 'Segunda categoria');
      JOW.WriteEndObject;
    JOW.WriteEndObject;

    Result := JOW.JSON.ToString;

    ShowMessage('TJsonObjectWriter:' + #13 + JOW.JSON.ToJSON + #13#13 +
JOW.JSON.ToString);
  finally
    JOW.Free;
  end;
end;

function TestarEscritaSimplesJTW: String;
var
  SW: TStringWriter;
  JTW: TJsonTextWriter;
begin
  SW  := TStringWriter.Create;
  JTW := TJsonTextWriter.Create(SW);
  try
    JTW.Indentation := 3;
    JTW.IndentChar  := ' '; // Outras opções testadas: '.'; #9;
    JTW.QuoteChar   := '"'; // Outra opção: '''';
    JTW.QuoteName   := False;

    // Determina se será ou não formatado:
    //   0-TJsonFormatting_None
    //   1-TJsonFormatting_Indented
    JTW.Formatting := TJsonFormatting_Indented;

    // Determina como serão mostrados caracteres especiais e tags html:
    //   0-TJsonStringEscapeHandling_Default
    //   1-TJsonStringEscapeHandling_EscapeNonAscii
    //   2-TJsonStringEscapeHandling_EscapeHtml
    JTW.StringEscapeHandling := TJsonStringEscapeHandling_Default;

    // Determina como serão mostrados números infinitos:
    //   0-TJsonFloatFormatHandling_String
    //   1-TJsonFloatFormatHandling_Symbol
    //   2-TJsonFloatFormatHandling_DefaultValue
    JTW.FloatFormatHandling := TJsonFloatFormatHandling_Symbol;
```

```
    // Determina como serão mostradas as datas:
    //   0-TJsonDateFormatHandling_Iso
    //   1-TJsonDateFormatHandling_Unix
    //   2-TJsonDateFormatHandling_FormatSettings
    JTW.DateFormatHandling := TJsonDateFormatHandling_FormatSettings;

    // Determina o comportamento para números grandes
    //   0-TJsonExtendedJsonMode_None
    //   1-TJsonExtendedJsonMode_StrictMode
    //   2-TJsonExtendedJsonMode_MongoShell
    JTW.ExtendedJsonMode := TJsonExtendedJsonMode_None;

    // Determina o funcionamento do fuso horário
    //   0-TJsonDateTimeZoneHandling_Local
    //   1-TJsonDateTimeZoneHandling_Utc
    JTW.DateTimeZoneHandling := TJsonDateTimeZoneHandling_Utc;

    // Determina como serão escritas as propriedades com valores vazios ''
    //   0-TJsonEmptyValueHandling_Empty
    //   1-TJsonEmptyValueHandling_Null
    JTW.EmptyValueHandling := TJsonEmptyValueHandling_Empty;

    JTW.WriteComment('Comentário inicial');
    JTW.WriteStartObject;
      JTW.WritePropertyName('cores');
        //ShowMessage(JTW.Path + #13 + JTW.ContainerPath);
        JTW.WriteStartArray;
          //ShowMessage(JTW.Path + #13 + JTW.ContainerPath);
          JTW.WriteStartObject;
            //ShowMessage(JTW.Path + #13 + JTW.ContainerPath);
            JTW.WritePropertyName('nome');      JTW.WriteValue('vermelho');
            JTW.WritePropertyName('hex');       JTW.WriteValue('#f00');
            JTW.WritePropertyName('vazio');     JTW.WriteValue('');
            JTW.WriteProperty('numero',         12345678901234567);
//            JTW.WriteProperty('numeroinfinito', 1/0);
            JTW.WriteProperty('Data',           Today);
            JTW.WriteProperty('Hora',           Now);
            JTW.WriteProperty('html',           '<html>Texto Complexão</html>');
          JTW.WriteEnd{Object};
        JTW.WriteEnd{Array};
    JTW.WriteEnd{Object};

    Result := JTW.Writer.ToString;

    ShowMessage('TJsonTextWriter: ' + #13 + Result);
  finally
    JTW.Free;
    SW.Free;
  end;
end;

procedure TestarLeituraSimples(S: String);
var
  SR: TStringReader;
  JTR: TJsonTextReader;
begin
  SR  := TStringReader.Create(S);
```

```
  JTR := TJsonTextReader.Create(SR);
  try
    S := '';
    try
      JTR.Rewind; {First}
      while JTR.Read do
        case JTR.TokenType of
          4{TJsonToken.PropertyName}: S := S + #13 + JTR.Value;
          7{TJsonToken.Integer}:      S := S + ': ' + IntToStr(JTR.Value);
          9{TJsonToken.String}:       S := S + ': ' + JTR.Value;
        end;
    except
      on E: Exception do
        raise Exception.Create(
          E.Message + #13 +
          'Linha: ' + IntToStr(JTR.LineNumber) + #13 +
          'Posição: ' + IntToStr(JTR.LinePosition));
    end;

    ShowMessage('TStringReader:' + #13 + S);
  finally
    JTR.Free;
    SR.Free;
  end;
end;

procedure PrepararEstruturaMestreDetalhe;
begin
  // Criação da estrutura
  CDSPedido.FieldDefs.Clear;
  CDSPedido.FieldDefs.Add('PEDIDO',    ftInteger,  0, False);
  CDSPedido.FieldDefs.Add('DESCRICAO', ftString,  30, False);
  CDSPedido.FieldDefs.Add('DATA',      ftDate,     0, False);
  CDSPedido.CreateDataSet;
  CDSPedido.IndexFieldNames := 'PEDIDO';

  CDSProdPedido.FieldDefs.Clear;
  CDSProdPedido.FieldDefs.Add('ITEM_PED', ftInteger,  0, False);
  CDSProdPedido.FieldDefs.Add('ITEM',     ftInteger,  0, False);
  CDSProdPedido.FieldDefs.Add('NOME',     ftString,  30, False);
  CDSProdPedido.FieldDefs.Add('QTDE',     ftCurrency, 0, False);
  CDSProdPedido.FieldDefs.Add('VLRUNIT',  ftCurrency, 0, False);
  CDSProdPedido.CreateDataSet;

  // Relacionamento
  DS.DataSet := CDSPedido;
  CDSProdPedido.MasterSource    := DS;
  CDSProdPedido.IndexFieldNames := 'ITEM_PED'{;ITEM};
  CDSProdPedido.MasterFields    := 'PEDIDO';

  // Inclusão de Registros Fictícios
  CDSPedido.Insert;
  CDSPedido.FieldByName('PEDIDO').AsInteger   := 1;
  CDSPedido.FieldByName('DESCRICAO').AsString := 'Descrição do Pedido 1';
  CDSPedido.FieldByName('DATA').AsDateTime    := Yesterday;
  CDSPedido.Post;
  CDSProdPedido.Insert;
  CDSProdPedido.FieldByName('ITEM_PED').AsInteger := 1;
```

```
  CDSProdPedido.FieldByName('ITEM').AsInteger     := 10;
  CDSProdPedido.FieldByName('NOME').AsString      := 'Descrição do Item 10';
  CDSProdPedido.FieldByName('QTDE').AsCurrency    := 1100;
  CDSProdPedido.FieldByName('VLRUNIT').AsCurrency := 1.50;
  CDSProdPedido.Post;
  CDSProdPedido.Insert;
  CDSProdPedido.FieldByName('ITEM_PED').AsInteger := 1;
  CDSProdPedido.FieldByName('ITEM').AsInteger     := 20;
  CDSProdPedido.FieldByName('NOME').AsString      := 'Descrição do Item 20';
  CDSProdPedido.FieldByName('QTDE').AsCurrency    := 200;
  CDSProdPedido.FieldByName('VLRUNIT').AsCurrency := 10.22;
  CDSProdPedido.Post;

  CDSPedido.Insert;
  CDSPedido.FieldByName('PEDIDO').AsInteger   := 2;
  CDSPedido.FieldByName('DESCRICAO').AsString := 'Descrição do Pedido 2';
  CDSPedido.FieldByName('DATA').AsDateTime    := Today;
  CDSPedido.Post;
  CDSProdPedido.Insert;
  CDSProdPedido.FieldByName('ITEM_PED').AsInteger := 2;
  CDSProdPedido.FieldByName('ITEM').AsInteger     := 30;
  CDSProdPedido.FieldByName('NOME').AsString      := 'Descrição do Item 30';
  CDSProdPedido.FieldByName('QTDE').AsCurrency    := 50.3;
  CDSProdPedido.FieldByName('VLRUNIT').AsCurrency := 1111.11;
  CDSProdPedido.Post;
end;

function TestarEscritaMestreDetalhe: String;
begin
  JOW.DateFormatHandling := 0;

  JOW.WriteStartObject;
    JOW.WriteProperty('MomentoGeracaoArquivo', Now);
    JOW.WritePropertyName('Pedidos');
      JOW.WriteStartArray;
        CDSPedido.ToJson(JOW, 'EscreveDetalhesDoPedido', True, '', ['DESCRICAO']);
      JOW.WriteEndArray;

    JOW.WritePropertyName('Totalizacao');
      JOW.WriteStartObject;
        JOW.WriteProperty('QtdePedidos',     CDSPedido.RecordCount);
        JOW.WriteProperty('VlrTotalPedidos', VlrTotalPedidos);
      JOW.WriteEndObject;
  JOW.WriteEndObject;

  Result := JOW.ToString;

  ShowMessage('Mestre Detalhe:' + #13#13 +
              JOW.ToString + #13#13 +
              JOW.ToJSON);
end;

procedure EscreveDetalhesDoPedido;
begin
  JOW.WritePropertyName('Itens');
    JOW.WriteStartArray;
      CDSProdPedido.ToJson(JOW, 'TotalizaItens', True, 'produto');
    JOW.WriteEndArray;
```

```
end;

procedure TotalizaItens;
begin
  VlrTotalPedidos := VlrTotalPedidos + CDSProdPedido.FieldByName('QTDE').AsCurrency *
CDSProdPedido.FieldByName('VLRUNIT').AsCurrency;
end;

procedure TestarLeituraMestreDetalhe(sJSON: String);
var
  sMomentoGeracaoArquivo, sTotalizacao, sPedidos, sPedido, sItens, sItem, sNomeItem:
String;
  iQtdePedidos, iPedido, iItem: Integer;
  cVlrTotalPedidos, cQtde, cVlrUnit: Currency;
  ArrayPedidos, ArrayItens: TJsonArray;
  jvItem: TJsonValue;
  dData: TDateTime;
  ObjetoGeral: TJsonObject;
  Enumerador: TJSONArrayEnumerator;
begin
  // Teste capturando diversos tipos através de Values
  ObjetoGeral := TJsonObject.ParseJsonValue(sJSON);
  ShowMessage(
    ObjetoGeral.Values('MomentoGeracaoArquivo').Value + #13#13 +
    ObjetoGeral.Values('Pedidos').ToJSON + #13#13 +
    ObjetoGeral.Values('Totalizacao').ToString);

  // Teste da função GetValueJson
  sMomentoGeracaoArquivo := GetValueJson(sJSON, 'MomentoGeracaoArquivo');
  ShowMessage('GetValueJson => Momento de Geração do Arquivo: ' +
sMomentoGeracaoArquivo);

  // Teste da função GetObjectJson
  sTotalizacao := GetObjectJson(sJSON, 'Totalizacao');
  ShowMessage('GetObjectJson => ' + sTotalizacao);

  // Interpretação de dados, segundo teste da função GetValueJson
  iQtdePedidos     := StrToInt (GetValueJson(sTotalizacao, 'QtdePedidos'));
  cVlrTotalPedidos := StrToCurr(Troca(GetValueJson(sTotalizacao, 'VlrTotalPedidos'),
'.', ','));
  ShowMessage('Existem ' + IntToStr(iQtdePedidos) + ' pedido(s) totalizando R$ ' +
FormatCurr('#,##0.00', cVlrTotalPedidos));

  // Teste da função GetArrayJson
  sPedidos := GetArrayJson(sJSON, 'Pedidos');
  ShowMessage('GetArrayJson => ' + sPedidos);

  // Teste de loop em TJsonArray
  ArrayPedidos := TJsonArray(TJsonObject.ParseJsonValue(sPedidos));
  while (ArrayPedidos.Count > 0) do
    begin
      sPedido := ArrayPedidos.Remove(0).ToString; {extração de TJsonValue e conversão
para string}
      ShowMessage('Pedido Extraído do Array ==> ' + sPedido);

      iPedido := StrToInt(GetValueJson(sPedido, 'PEDIDO'));
      dData   := StrToDateTime(GetValueJson(sPedido, 'DATA'));
```

```
      ShowMessage('Código do Pedido: ' + IntToStr(iPedido) + #13 + 'Data do Pedido..:
' + DateTimeToStr(dData));

      sItens := GetArrayJson(sPedido, 'Itens');
      ShowMessage('Array de Itens => ' + sItens);

      ArrayItens := TJsonArray(TJsonObject.ParseJsonValue(sItens));
      while (ArrayItens.Count > 0) do
        begin
          jvItem := ArrayItens.Remove(0) ; {extração de TJsonValue}
          sItem  := GetObjectJson(jvItem.ToString, 'produto');
          ShowMessage('Item Extraído do Array ==> ' + sItem);

          iItem     := StrToInt(GetValueJson(sItem, 'ITEM'));
          sNomeItem := GetValueJson(sItem, 'NOME');
          cQtde     := StrToCurr(Troca(GetValueJson(sItem, 'QTDE'), '.', ','));
          cVlrUnit  := StrToCurr(Troca(GetValueJson(sItem, 'VLRUNIT'), '.', ','));
          ShowMessage('Código do Item: ' + IntToStr(iItem) + #13 +
                      'Nome do Item: ' + sNomeItem + #13 +
                      'Qtde: ' + FormatCurr('#,##0.00', cQtde) + #13 +
                      'Vlr.Unit: ' + FormatCurr('#,##0.00', cVlrUnit));
        end;
    end;

  // Segunda forma de percorrer array, através de TJSONArrayEnumerator
  ArrayPedidos := TJsonArray(TJsonObject.ParseJsonValue(sPedidos));
  Enumerador := TJSONArrayEnumerator.Create(ArrayPedidos);
  try
     while Enumerador.MoveNext do
       begin
         sPedido := Enumerador.Current.ToString;
         ShowMessage(sPedido);
       end;
  finally
    Enumerador.Free;
  end;
end;
```

- **TClassProc_ImportacaoDocumento, DialogoAbrirArquivo**

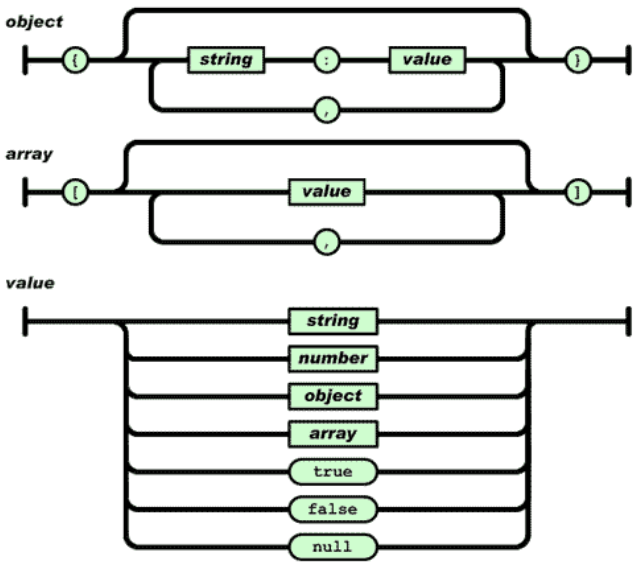
```
unit ProcessamentoEspecifico;

procedure Main;
var
  ImportacaoDoc: TClassProc_ImportacaoDocumento;
  CaminhoArquivo: String;
begin
 ImportacaoDoc := TClassProc_ImportacaoDocumento.Create;
 try
   CaminhoArquivo := DialogoAbrirArquivo('C:\', '', '', 'Selecione o arquivo para
importação!');
   if(CaminhoArquivo <> '') then
   begin
     ImportacaoDoc.ProcessarArquivo(14,               {Código do Layout}
                                    9,                {Destino dos dados / 9 - Pedido
de Venda}
                                    CaminhoArquivo); {Caminho do Arquivo}

     if(ImportacaoDoc.Log = '') then
     begin
       MostrarCDS(ImportacaoDoc.CDSDocumento, True);
       MostrarCDS(ImportacaoDoc.CDSItem, True);
       MostrarCDS(ImportacaoDoc.CDSPrazo, True);
     end
     else
      ShowMessage('Erros durante o processo:' + #13 + ImportacaoDoc.Log);
   end;
 finally
   ImportacaoDoc.Free;
 end;
end;
```

- **TClassProc_ImportacaoPessoa**

```
unit ProcessamentoEspecifico;

procedure Main;
var
  ImportaPessoa: TClassProc_ImportacaoPessoa;
begin
  ImportaPessoa := TClassProc_ImportacaoPessoa.Create;
  try
    ImportaPessoa.AtribuirLigacoes;

    ImportaPessoa.Processando := True;

    // Para conhecer as estruturas de campos disponíveis, descomente as linhas abaixo
    //MostrarCDS(ImportaPessoa.CDSPessoa);
    //MostrarCDS(ImportaPessoa.CDSEndereco);

    // 1ª Pessoa
    ImportaPessoa.CDSPessoa.Insert;
    ImportaPessoa.CDSPessoa.FieldByName('RAZAOSOCIAL_PESSOA').AsString := 'DANILO
CANESCHI';
    ImportaPessoa.CDSPessoa.FieldByName('NOMEFANTASIA_PESSOA').AsString := 'DANILO';
    ImportaPessoa.CDSPessoa.FieldByName('FORNECEDOR_PESSOA').AsString := 'S';
    ImportaPessoa.CDSPessoa.FieldByName('EMAIL_PESSOA_EMAIL').AsString :=
'danilocaneschi@teksystem.com.br';
    ImportaPessoa.CDSPessoa.FieldByName('TIPO_PESSOA').AsString := 'F';
    ImportaPessoa.CDSPessoa.FieldByName('CPF_PESSOA_FIS').AsString := '11722491647';
    ImportaPessoa.CDSPessoa.FieldByName('IDENTIDADE_PESSOA_FIS').AsString :=
'15667587';

    ImportaPessoa.CDSEndereco.Insert;
    ImportaPessoa.CDSEndereco.FieldByName('ENDERECO_PESSOA_END').AsString := 'RUA
TESTE DE ATUALIZACAO';
    ImportaPessoa.CDSEndereco.FieldByName('NUMERO_PESSOA_END').AsString := '98';
    ImportaPessoa.CDSEndereco.FieldByName('BAIRRO_PESSOA_END').AsString := 'BAIRRO
TESTE';
    ImportaPessoa.CDSEndereco.FieldByName('DESCRICAO_CIDADE').AsString := 'UBÁ';
    ImportaPessoa.CDSEndereco.FieldByName('CEP_PESSOA_END').AsString := '36500-000';
    ImportaPessoa.CDSEndereco.FieldByName('SIGLA_UF').AsString := 'MG';
    ImportaPessoa.CDSEndereco.FieldByName('TELEFONE_PESSOA_TEL').AsString :=
'(32)98867-6580';
    ImportaPessoa.CDSEndereco.Post;

    ImportaPessoa.CDSPessoa.Post;

    // 2ª Pessoa
    ImportaPessoa.CDSPessoa.Insert;
    ImportaPessoa.CDSPessoa.FieldByName('RAZAOSOCIAL_PESSOA').AsString  := 'DANILO
CANESCHI';
    ImportaPessoa.CDSPessoa.FieldByName('FORNECEDOR_PESSOA').AsString := 'S';
    ImportaPessoa.CDSPessoa.FieldByName('EMAIL_PESSOA_EMAIL').AsString :=
'danilocaneschi@teksystem.com.br';
    ImportaPessoa.CDSPessoa.FieldByName('TIPO_PESSOA').AsString := 'J';
    ImportaPessoa.CDSPessoa.FieldByName('CNPJCEI_PESSOA_JUR').AsString :=
'11820231000168';
    ImportaPessoa.CDSPessoa.FieldByName('INSCRICAO_PESSOA_JUR').AsString := 'ISENTO';
```

```
    ImportaPessoa.CDSEndereco.Insert;
    ImportaPessoa.CDSEndereco.FieldByName('TIPOENDERECO_PESSOA_END').AsString := 1;
{principal}
    ImportaPessoa.CDSEndereco.FieldByName('ENDERECO_PESSOA_END').AsString := 'R.u
TESTE DE ATUALIZACAO';
    ImportaPessoa.CDSEndereco.FieldByName('NUMERO_PESSOA_END').AsString := '98';
    ImportaPessoa.CDSEndereco.FieldByName('BAIRRO_PESSOA_END').AsString := 'BAIRRO
TESTE';
    ImportaPessoa.CDSEndereco.FieldByName('DESCRICAO_CIDADE').AsString := 'UBÁ';
    ImportaPessoa.CDSEndereco.FieldByName('CEP_PESSOA_END').AsString := '36500-000';
    ImportaPessoa.CDSEndereco.FieldByName('SIGLA_UF').AsString := 'MG';
    ImportaPessoa.CDSEndereco.FieldByName('TELEFONE_PESSOA_TEL').AsString :=
'(32)98867-6580';
    ImportaPessoa.CDSEndereco.Post;
    ImportaPessoa.CDSEndereco.Insert;
    ImportaPessoa.CDSEndereco.FieldByName('TIPOENDERECO_PESSOA_END').AsString := 2;
{Entrega}
    ImportaPessoa.CDSEndereco.FieldByName('ENDERECO_PESSOA_END').AsString := 'MEU
DEPÓSITO';
    ImportaPessoa.CDSEndereco.FieldByName('NUMERO_PESSOA_END').AsString := '888';
    ImportaPessoa.CDSEndereco.FieldByName('COMPLEMENTO_PESSOA_END').AsString :=
'GALPÃO';
    ImportaPessoa.CDSEndereco.FieldByName('BAIRRO_PESSOA_END').AsString := 'AQUELE
OUTRO';
    ImportaPessoa.CDSEndereco.FieldByName('DESCRICAO_CIDADE').AsString :=
'CATAGUASES';
    ImportaPessoa.CDSEndereco.FieldByName('SIGLA_UF').AsString := 'MG';
    ImportaPessoa.CDSEndereco.Post;

    ImportaPessoa.CDSPessoa.Post;

    ImportaPessoa.Processando := False;

    if ImportaPessoa.Log <> '' then
      MostrarLogTexto(ImportaPessoa.Log)
    else
      begin
        //MostrarCDS(ImportaPessoa.CDSEndereco, False);
        //MostrarCDS(ImportaPessoa.CDSPessoa);

        {Gravação no Banco de Dados de Acordo com o Cadastro}
        ExecuteMethods('TSMCadPessoa_Fornecedor.ImportarPessoa',
[ImportaPessoa.EstruturaParaPersistencia]);
        ShowMessage('Sincronização da Pessoa realizada com sucesso');
      end;
  finally
    ImportaPessoa.Free;
  end;
end;

end.
```

Estrutura de Objetos JSON



Novidades no APP

- Exibição de Indicadores do Tipo WebBrowser
- Possibilidade de Detalhar os valores dos indicadores
- Pesquisa nas listas de painéis e relatórios

Denis Pereira Raymundo
Certified Delphi Developer
Professional Coach of Life Coaching
Especialista em Gestão e Manutenção de Tecnologia da Informação
Bacharel em Ciência da Computação
Licenciado em Matemática
Técnico em Processamento de Dados

Gerente de Sistemas
www.teksystem.com.br
Prêmios: Top Móbile - Segmento: Fornecedores de Softwares p/Setor Moveleiro
- 1ª lugar (2013)
- 2ª lugar (2012, 2014, 2015 e 2016)
- 3ª lugar (2009)

"Não que sejamos capazes, por nós, de pensar alguma coisa, como de nós mesmos; mas a nossa capacidade vem de Deus." 2 Co 3.5