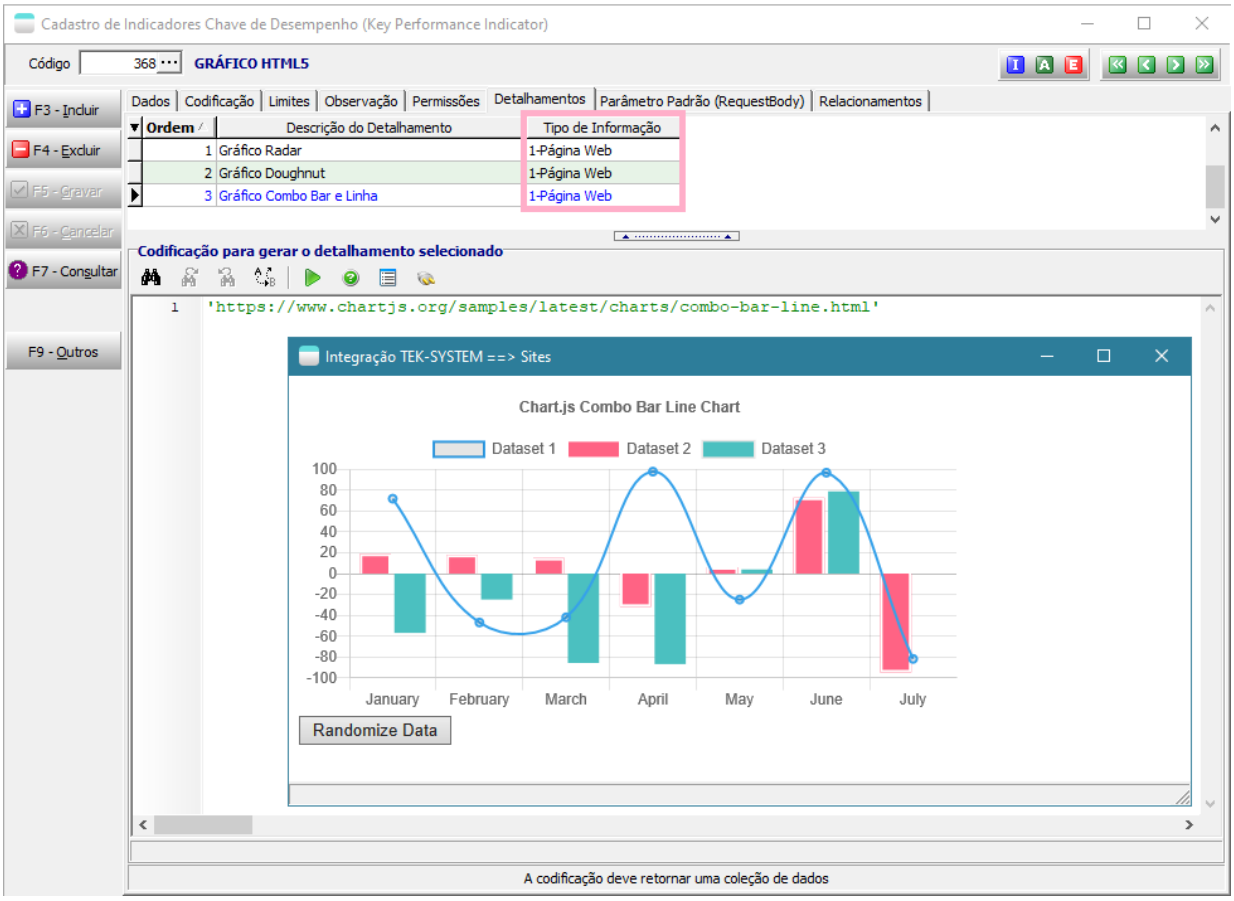
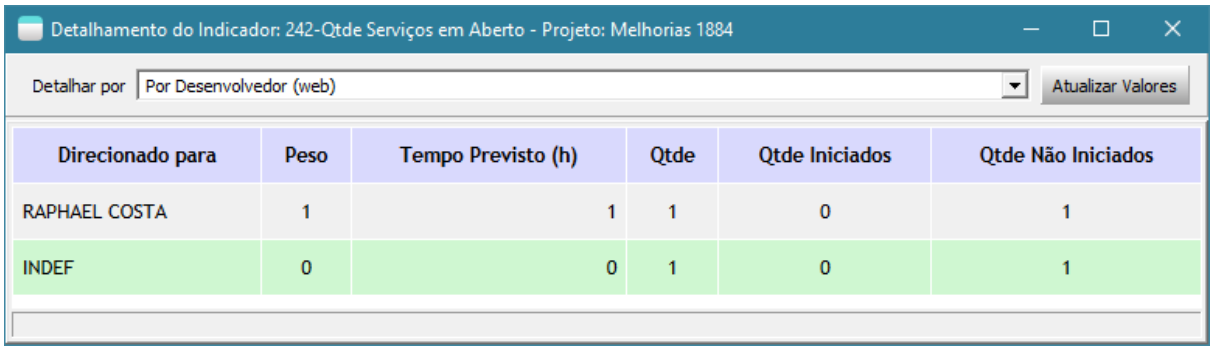
Novidades no módulo BI (Inteligência de Negócios e Gerador de Relatórios) da Tek-System

Inovações

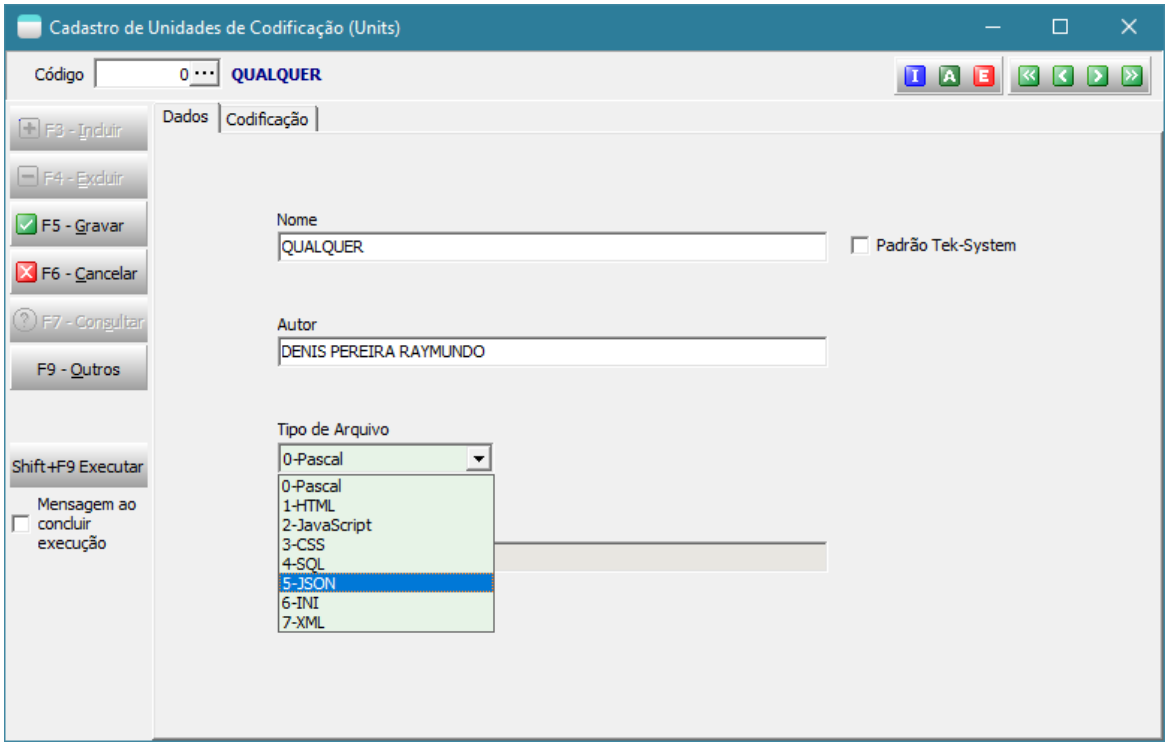
- Detalhamentos de indicadores agora, além de coleções de dados, podem ser páginas web. Desta forma poderemos criar diversas visualizações para detalhar um indicador, como por exemplo, gráficos diversos.



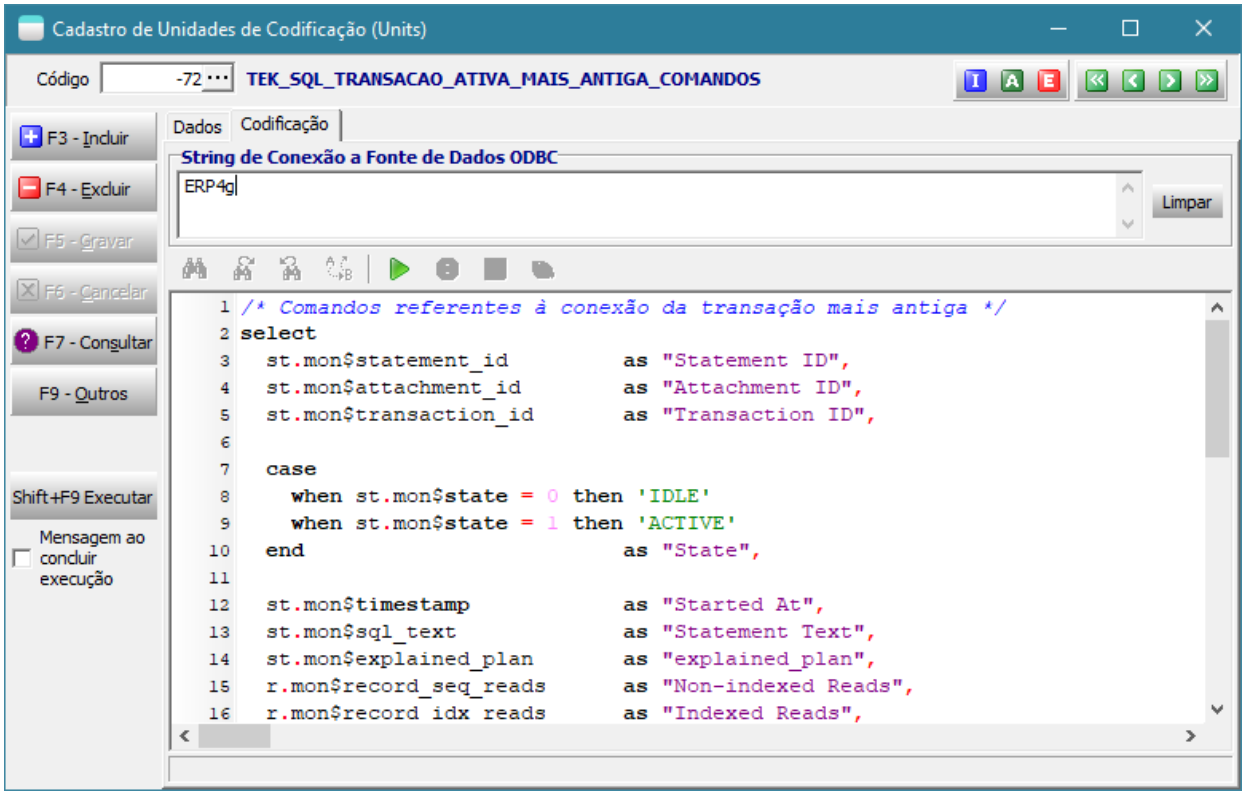


- Melhorias no cubo de decisão:
  - Execução de cubos de decisão relativos a temas de armazéns de dados, passa a salvar o último esquema de análise utilizado pelo usuário. Desta forma quando abrir o cubo novamente, a visualização já estará disponível.
  - Não será mais solicitada confirmação sobre onde salvar/carregar um esquema (disco/banco de dados). O padrão será sempre banco de dados. Se quiser forçar a gravação em disco deverá pressionar a tecla CTRL antes de solicitar a ação.
  - Sempre abrirá maximizado, aproveitando o tamanho maior dos monitores atuais.

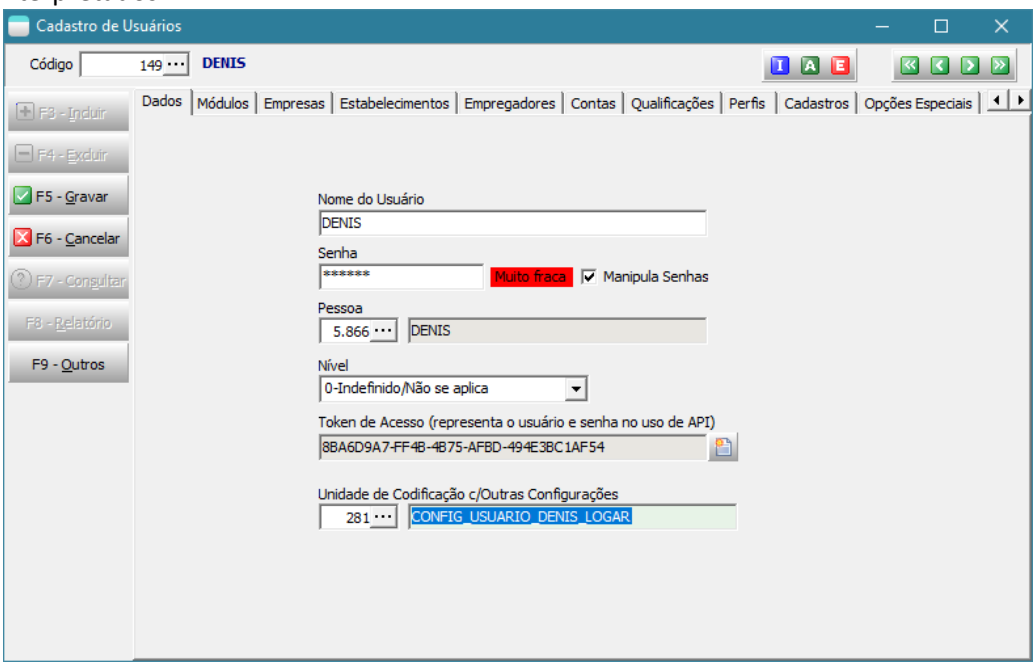
- Os tipos de unidades de codificação foram ampliados para comportar outras sintaxes: JSON, INI, XML. O formato JSON passa a realizar a validação antes de aceitar a gravação, assim como nos parâmetros de indicadores. Dica: Os formatos INI e XML também podem ser usados como configuração/parâmetros.



- Quando o cadastro de unidade de codificação for uma SQL, o sistema permitirá informar a string de conexão com uma fonte de dados ODBC facilitando a sua execução em outro banco de dados e também a "ajuda para compor" com o diagrama do banco de dados selecionado. Isto facilitará para que os usuários tenham suas pesquisas SQLs sempre à mão, sem a necessidade de manter repositórios individuais e separados, com risco de perda da inteligência da empresa.



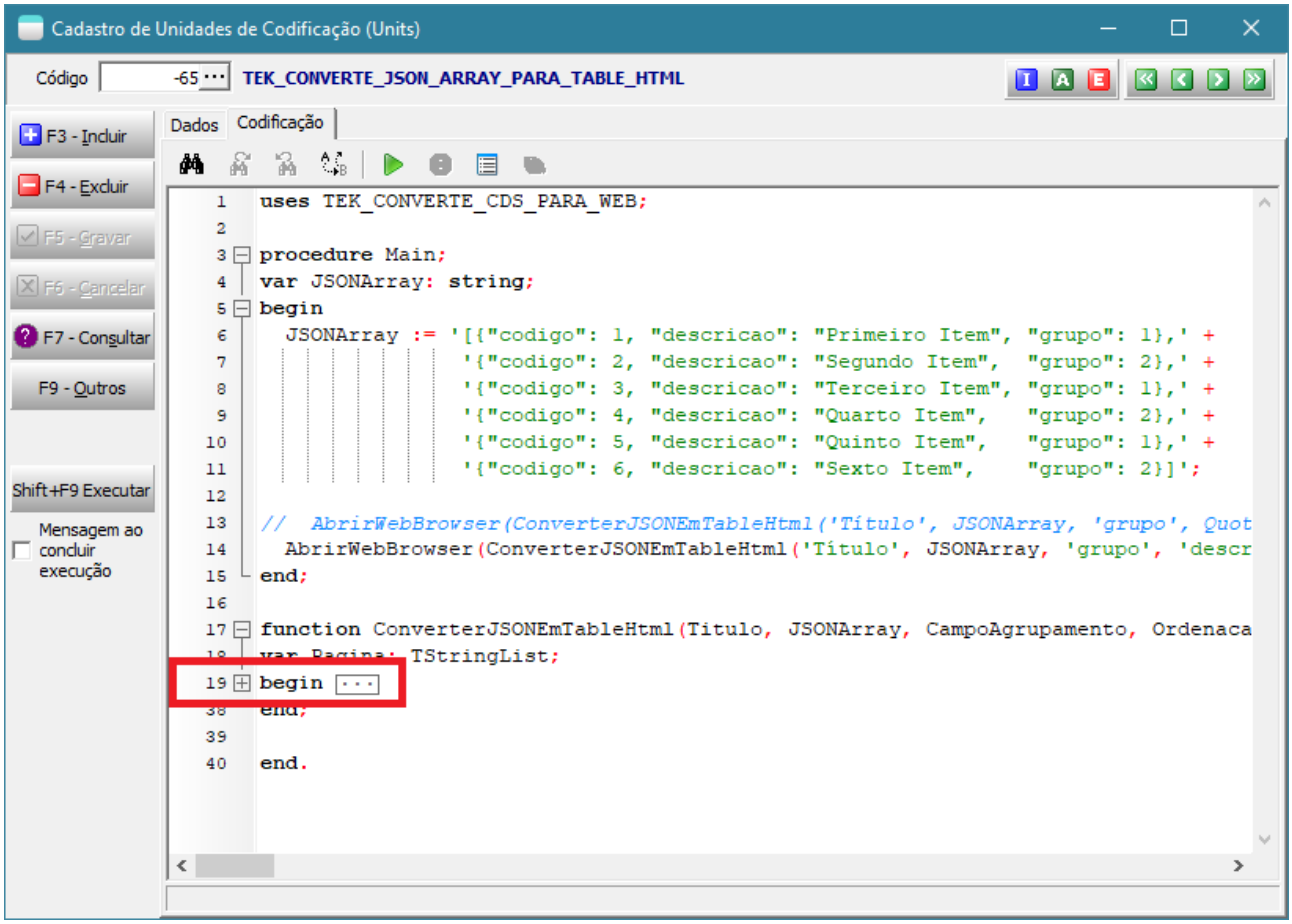
- No cadastro de usuário passa a ser permitido informar uma unidade de codificação que será executada (.pas) ou carregada (json) quando o usuário se conectar. Desta forma será possível, por exemplo:
    - Impedir que um usuário se conecte ao sistema fora do horário de trabalho
    - Receber um aviso com informações de onde a conexão foi feita
    - Declarar variáveis para a seção do usuário, que poderão ser utilizadas em outros locais interpretados.




```
function Main: string;
begin
  if (GetValueJson(SecaoAtualJson, 'Plataforma') = 'Win32') then
    if (not HorarioExpediente) then
      raise Exception.Create(MensagemPersonalizada + 'Não é permitido acessar o sistema
desktop fora do expediente.'#13#13'VÁ DESCANSAR!');

  if (Nome_Usuario_Atual = 'A') then
    EnviarMensagemLog;

  SetVariavelContexto('UsuarioPossuiPrivilegiosSuperiores', 'SIM');

  Result := '{"OutrasConfiguracoesParaASecao": true}';
end;

function HorarioExpediente: Boolean;
var
  Ctrl: TClassCntrlDiasUteis;
  DiaUtil: Boolean;
  Horario: String;
begin
  Ctrl := TClassCntrlDiasUteis.Create(0, 0);
  try
    DiaUtil := Ctrl.DiaUtil(HOJE);
  finally
    Ctrl.Free;
  end;

  Horario := FormatDateTime('hh:nn:ss', Now - Today);

  Result := (DiaUtil) and
            ( ((Horario >= '07:15:00') and (Horario <= '11:13:00')) or
              ((Horario >= '13:00:00') and (Horario <= '17:50:00')) );
end;
```

```
procedure EnviarMensagemLog;
var Detalhes: string;
begin
  Detalhes :=
    'Host: ' + GetValueJson(SecaoAtualJson, 'Host') + #13 +
    'IP: ' + GetValueJson(SecaoAtualJson, 'IP') + #13 +
    'Plataforma: ' + GetValueJson(SecaoAtualJson, 'Plataforma') + #13 +
    'Sistema: ' + GetValueJson(SecaoAtualJson, 'Sistema');

  ExecuteMethods(
    'TSMMensagem.EnviaMensagemSimples',
    ['DENIS',
     'MONITORAMENTO DE USUÁRIOS',
      Nome_Usuario_Atual + ' logou na empresa ' + IntToStr(EmpresaAtual) + #13#13 +
Detalhes]);
end;
```

- Melhoria no editor de codificação: Disponibilizada a funcionalidade para expandir / recolher codificações, semelhante ao Delphi melhorando a navegação em grandes codificações.



Observe que na barra lateral esquerda deverá aparecer símbolo de "subtração", ao lado das palavras procedure, function, begin, region, try...

Ao clicar neste símbolo de "subtração", a codificação deve ser recolhida, e o símbolo mudar para "adição". Ao clicar no símbolo de "adição", a codificação deverá ser expandida novamente.

* Pode-se também usar a tecla de atalho CTRL+SHIFT+_ para recolher tudo de uma vez e CTRL+SHIFT++ para expandir tudo novamente. Esta combinação funciona apenas se você puder editar a codificação.

- **Possibilidade de interpretação recursiva**, isto amplia consideravelmente o que pode ser feito com o sistema! Disponibilizada a função "interpretar". Esta função permitirá interpretar codificações passadas como parâmetro string, como por exemplo cálculos. Ela rodará dentro do contexto de sua execução.

```
const
  UnitRecursiva =
    'unit Recursiva;' + #13 +
    'function main: OleVariant;' + #13 +
    'begin' + #13 +
    ' Result := ExecuteReader(''select first 1 CODIGO_EMP, NOME_EMP from EMPRESA'')' +
#13 +
    'end;' + #13 +
    'end.';

procedure Main;
var
  Codificacao: TStringList;
  NomeArquivo: string;
begin
  // A) Interpretação de cálculos
  ShowMessage(Interpretar('5 * 4 + 2'));

  // B) Interpretação de métodos
  Interpretar('ShowMessage(HOJE)');

  // C) Interpretação de units inteiras
  MostrarCDS(Interpretar(UnitRecursiva));

  // D) Interpretação de units vindas de arquivo no disco.
  NomeArquivo := DialogoAbrirArquivo('c:\', 'Pascal (*.pas)|*.pas');
  if (NomeArquivo <> '') then
    begin
      Codificacao := TStringList.Create;
      try
        Codificacao.LoadFromFile(NomeArquivo);
        Interpretar(Codificacao.Text);
      finally
        Codificacao.Free;
      end;
    end;

   // E) Interpretação de units on-line
Interpretar(EnderecoWebProcessado2('http://www.teksystem.com.br/api/bi/samples/UnitInterp
retadaOnLine.pas'));
end;
```

* Com este recurso será possível ter parâmetros de indicadores com codificações ou fórmulas de cálculo!

* Pode-se também criar, por exemplo, uma dll de recursos com as suas codificações, evitando que sejam expostas facilmente.

Novas funções/métodos disponibilizados para interpretação

- DadosOrdenados_ApenasCampos - invocado a partir de um objeto ClientDataSet e passando um array com no máximo 32 strings com a lista dos campos desejados. Isto é útil quando se cria indicadores do tipo tabela de dados ou armazéns de dados.

- NivelMaximoBuscaComposicao - recebe o arquivo_item (cTipoVolume, cTipoPeca...) e retorna o nível máximo de busca configurado no sistema.

- ExecuteScalarODBCServ, ExecuteReaderODBCServ, ExecuteCommandODBCServ – Executam comandos SQL através de conexões ODBC criadas no servidor de aplicação não sendo necessário instalar/configurar o driver nas estações. Serão úteis em processamentos específicos e em locais onde se queira testar a interpretação nas estações dos usuários e não apenas na estação de desenvolvimento. A partir desta versão, dê preferência por usar estas funções no lugar de ExecuteScalarODBC, ExecuteReaderODBC, ExecuteCommandODBC.

- ExecutarProcessoSO - Permite executar processos e aplicativos pelo sistema operacional e ainda aguardar o término da execução.

```
// Abre a a calculadora do windows
ExecutarProcessoSO('calc');

// Abre o bloco de notas
ExecutarProcessoSO('notepad.exe', '', 1{SW_SHOW}, false, false);

// Cria o arquivo c:\temp\dir.txt com a lista de executáveis na pasta do
aplicativo. A pasta temp deve existir!
ExecutarProcessoSO('cmd /c dir *.exe > c:\temp\dir.txt');

// Cria o arquivo c:\temp\dir.txt com a lista de executáveis da pasta windows. A
pasta temp deve existir!
ExecutarProcessoSO('cmd /c dir *.exe > c:\temp\dir.txt', 'c:\windows');

// Apresenta um command listando arquivos exe da pasta windows e subpastas e só
depois exibe a mensagem de "acabou"
ExecutarProcessoSO('cmd /c dir/s *.exe', 'c:\windows', 1{SW_SHOW}, true, false);
ShowMessage('Acabou de listar');
```

- ExecutarComandoSOCapturandoOutput - Permite executar comandos do sistema operacional, ou outros aplicativos que escrevam no console e ir capturando a execução linha a linha.

```
var
  F: TForm;
  M: TMemo;

procedure Main;
var RetornoCompleto: String;
begin
  // Apresenta mensagem com a execuçao do ping para o google
  ShowMessage(ExecutarComandoSOCapturandoOutput('ping www.google.com'));

  // Executa comando, passando parâmetro e capturando retorno ao final
  MostrarLogTexto(ExecutarComandoSOCapturandoOutput('tree', 'c:\.'));

  // Executa comando capturando retorno durante execução e também ao final
  CallBack_AbreTela(ClassOwner);
  try
    RetornoCompleto := ExecutarComandoSOCapturandoOutput('dir /s *.com',
'c:\windows', 0{SW_HIDE}, 'CapturaOutPutEscreveCallBack');
  finally
```

```
      CallBack_FechaTela(ClassOwner);
    end;
    MostrarLogTexto(RetornoCompleto);

    // Executa comando capturando retorno durante a execução e plotando em um memo.
    F := TForm.Create(nil);
    M := TMemo.Create(F);
    try
      M.Parent := F;
      M.Align := alClient;
      F.Width := 800;
      F.Height := 600;
      F.Show;
      ExecutarComandoSOCapturandoOutput('dir /s *.com', 'c:\windows', 0{SW_HIDE},
'CapturaOutPutEscreveMemo');
      ShowMessage('Terminou');
    finally
      M.Free;
      F.Free;
    end;
  end;

  procedure CapturaOutPutEscreveCallBack(S: string);
  begin
    CallBack_Mensagem(ClassOwner, S);
  end;

  procedure CapturaOutPutEscreveMemo(S: string);
  begin
    M.Lines.Add(S);
  end;
```

Novos Processamentos Específicos:

TEK-> CANCELAR COMANDO SQL EM EXECUÇÃO NO FIREBIRD

TEK-> DESCONECTAR USUÁRIO FIREBIRD

Novos Indicadres Chave de Desempenho

FAT: STATUS PEDIDOS DE VENDA EM ABERTO

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, 2016 e 2018)
- 3ª lugar (2009)

"O que eu faço não o sabes agora, mas depois o entenderás." Jo 13.7