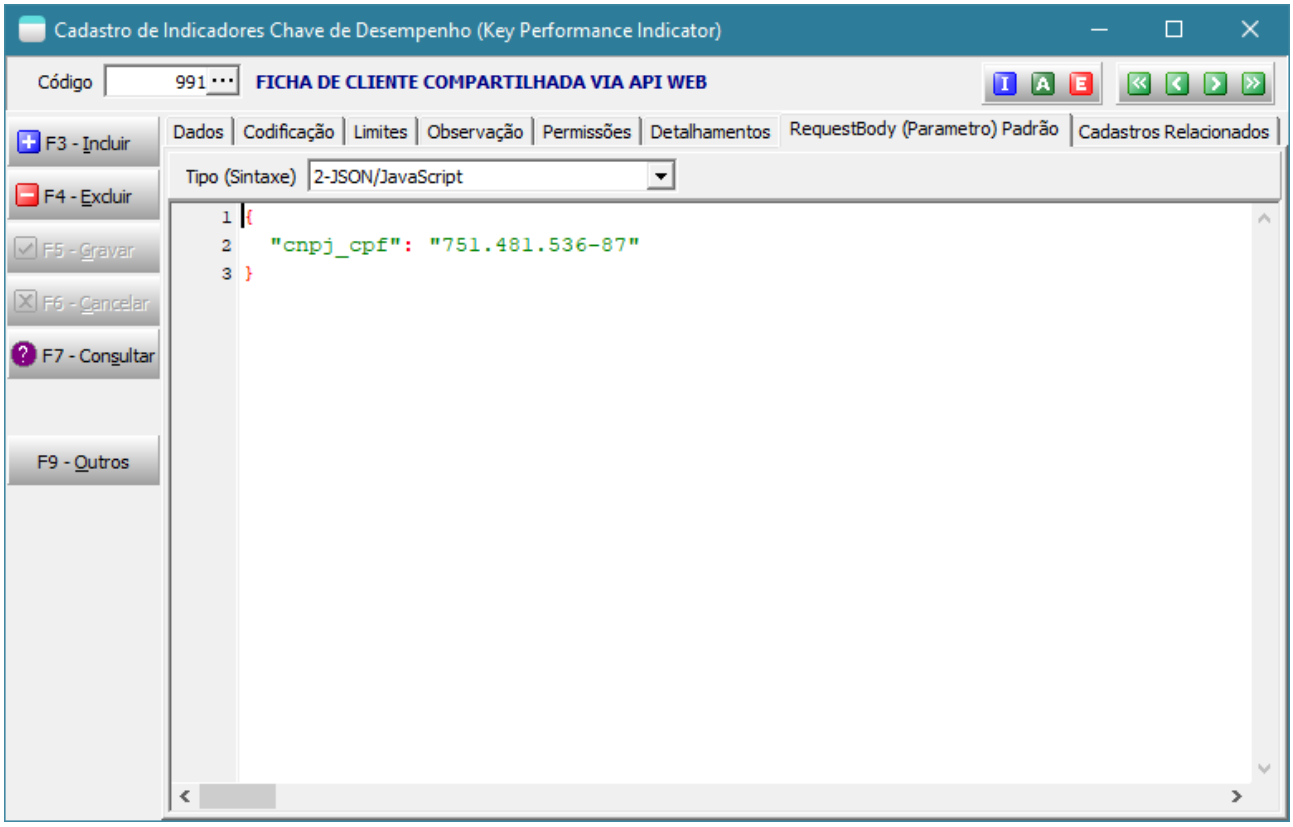
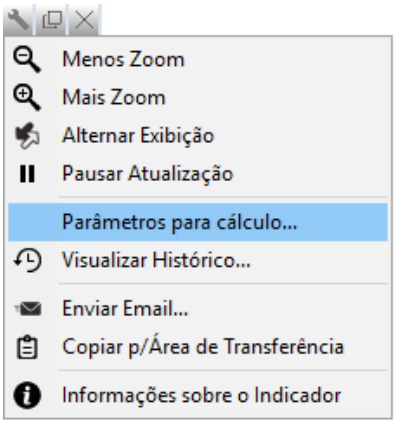
Novidades no módulo BI (Inteligência de Negócios e Gerador de Relatórios) da Tek-System

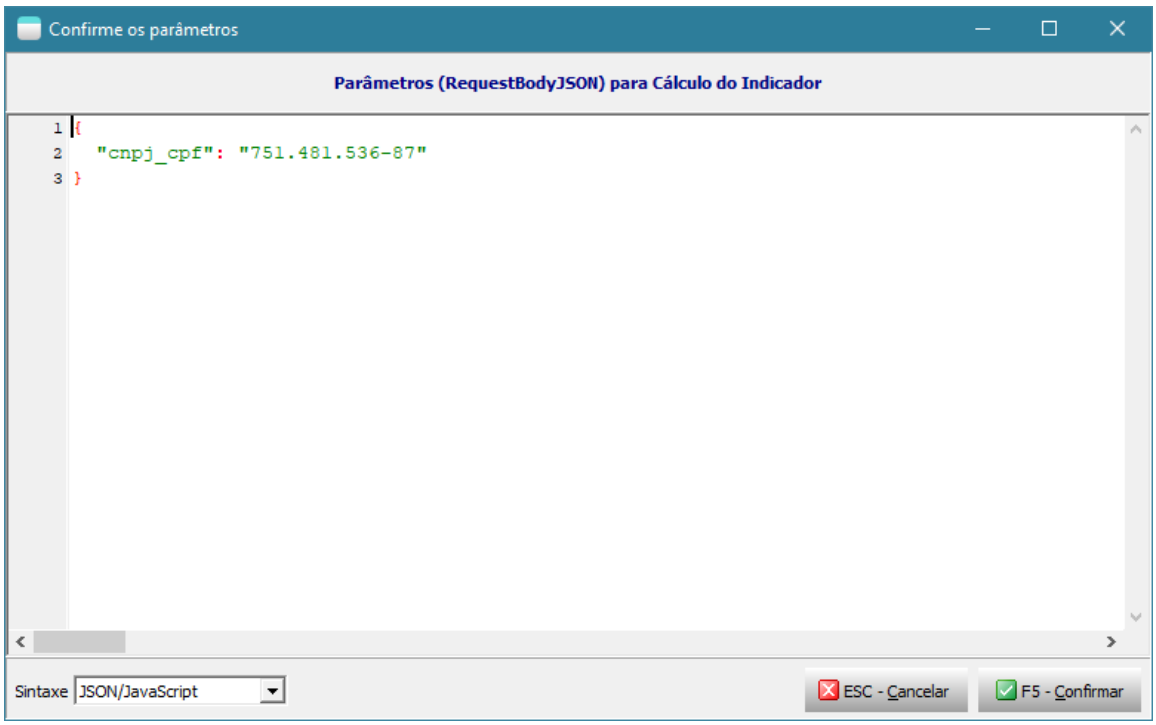
Inovações

- No **cadastro de indicadores passa a ser permitido informar um parâmetro padrão para o seu cálculo**, facilitando a codificação/amplitude do mesmo. Este conteúdo estará acessível, na codificação, através da constante RequestBodyJSON. Também é possível informar o seu tipo, facilitando a visualização diferenciada através das cores do editor.



- Na barra de ferramentas do indicador, agora é possível acessar os parâmetros para cálculo do indicador e editá-los.





- ✓ O valor atribuído modificará automaticamente o cálculo do indicador em execução, bem como o cálculo dos seus detalhamentos.
- ✓ Quando isto é feito, o cache do indicador deixará de ser considerado/armazenado até que se retorne para o parâmetro padrão.
- ✓ Não é aconselhado que se faça o registro histórico do log do indicador quando seu parâmetro estiver modificado.
- ✓ Deve-se ter cuidado ao alterar este parâmetro e o mais breve possível retorná-lo para o padrão.
- ✓ No caso de passar parâmetro vazio, o sistema utilizará o padrão para seus cálculos.
- ✓ Este recurso é muito útil quando se tem um indicador baseado nos últimos X dias. Pode-se colocar o número de dias sendo o parâmetro de cálculo. Ex: Vendas nos últimos 30 dias. Desta forma, o indicador estará mostrando as vendas nos últimos 30 dias, mas se o usuário quiser ampliar a análise temporariamente para 7, 15, 45, 60, 90 etc, faria sem problemas.

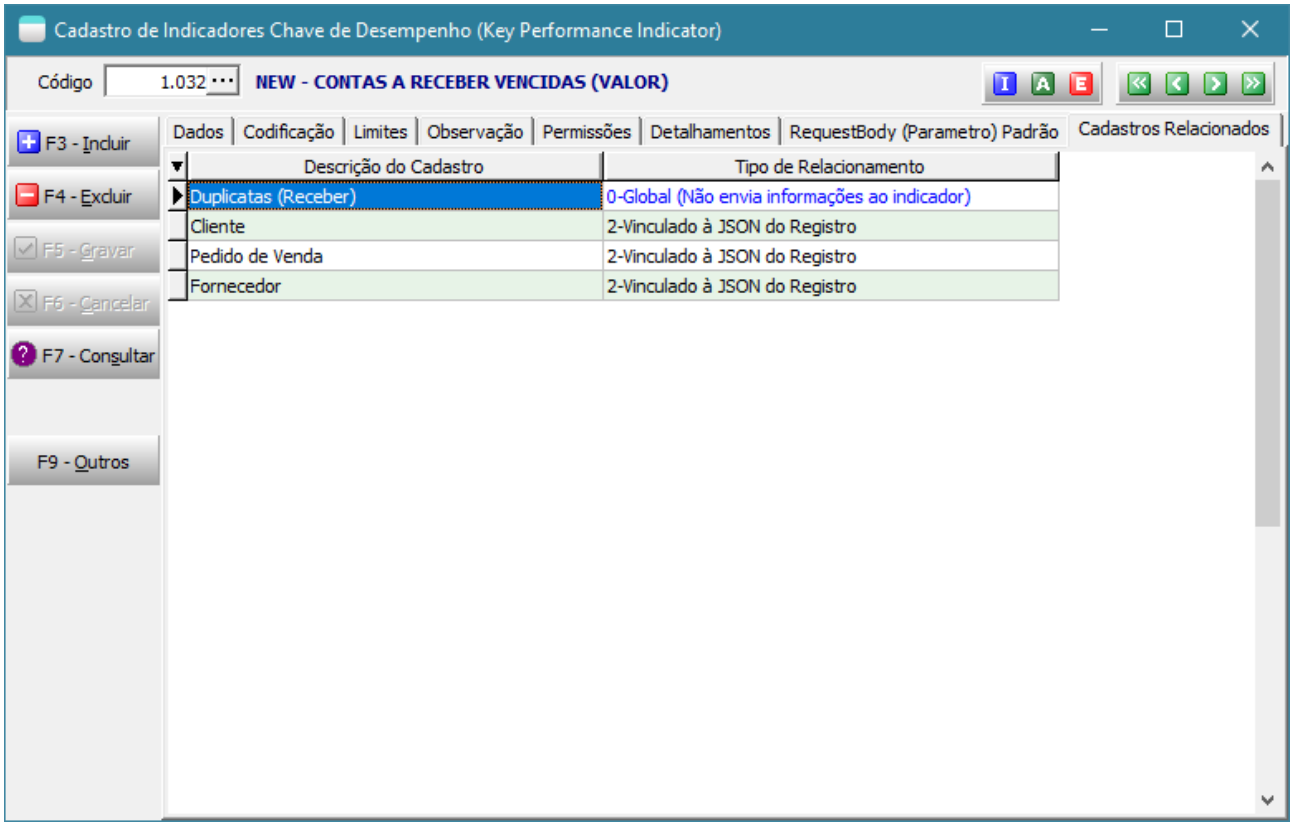
- Exemplos codificação de indicadores usando o parâmetro RequesBodyJSON:

```
1) StrToInt(Trim(RequestBodyJson)); // O parâmetro seria um número.
2) ExecuteReader(RequestBodyJSON); // O parâmetro seria uma SQL completa.
3) RequestBodyJSON; // O indicador mostraria apenas o que fosse o parâmetro.
   Este tipo de codificação ajudará na identificação dos parâmetros
   vinculados a cadastros, que será explicado adiante neste documento.
4) ExecuteReader('select * from banco where banco.codigo_banco = ' +
   RequestBodyJson); // O parâmetro seria uma string que seria utilizada como
   filtro em uma SQL.
```

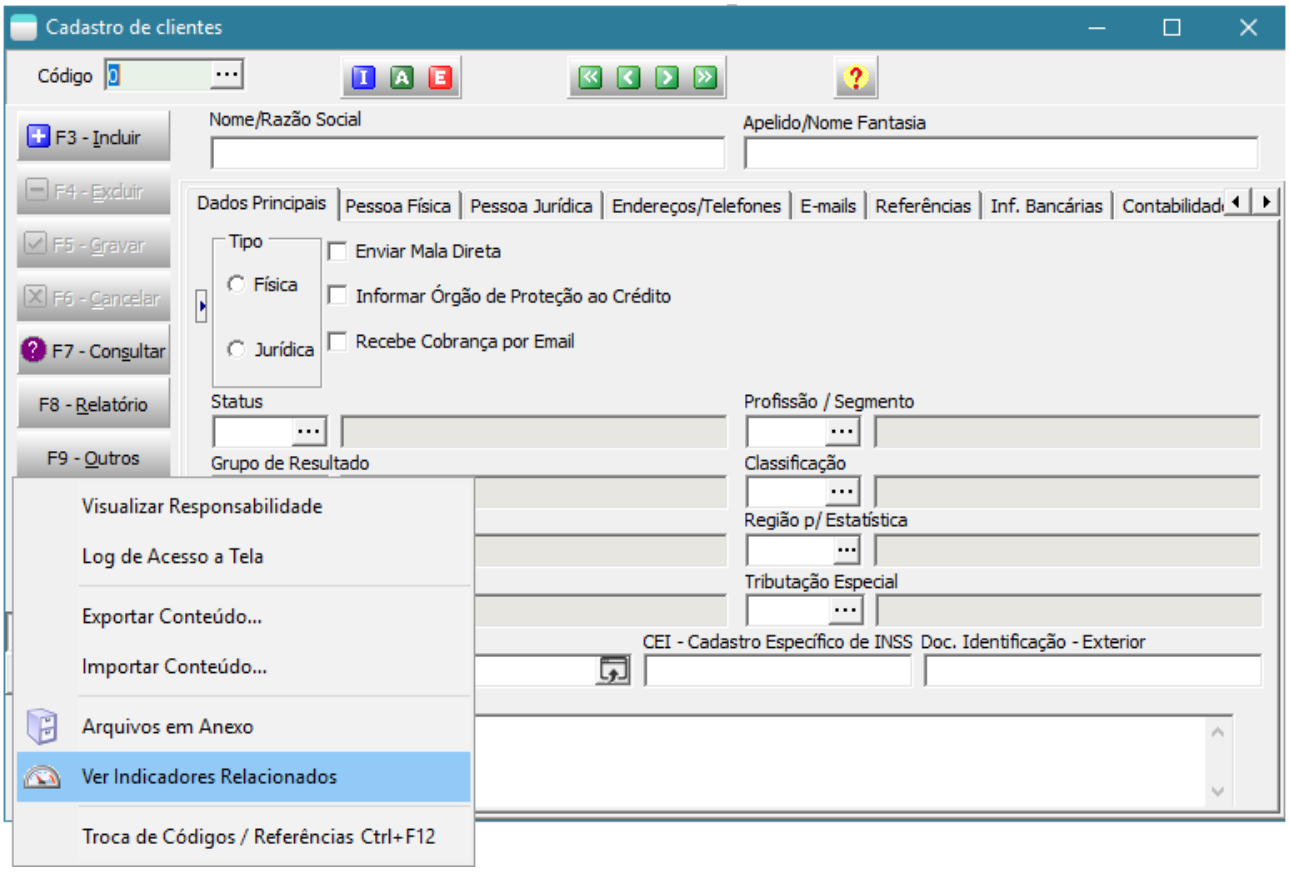
- Exemplos de chamada recursiva de indicadores, passando parâmetros de execução.

```
1) Indicador_Valor(991, '{"cnpj_cpf":"751.481.536-87"}')
2) IndicadorTEK_Valor('BI_MeusIndicadoresNumericos', 'QUALQUER COISA');
3) Indicador_Detalhamento(962, 'Mês atual', '{"CodigoDesenvolvedor": 1}');
4) Indicador_Detalhamento(991, 'Contas a Receber Vencidas',
   '{"cnpj_cpf":"751.481.536-87"}');
5) IndicadorTEK_Detalhamento('PercDispersaoKMViagensUltimos365Dias', 'Por
   Motorista', 'XXXXXX');
```

- Criada a possibilidade de **relacionar os indicadores aos cadastros do sistema**.



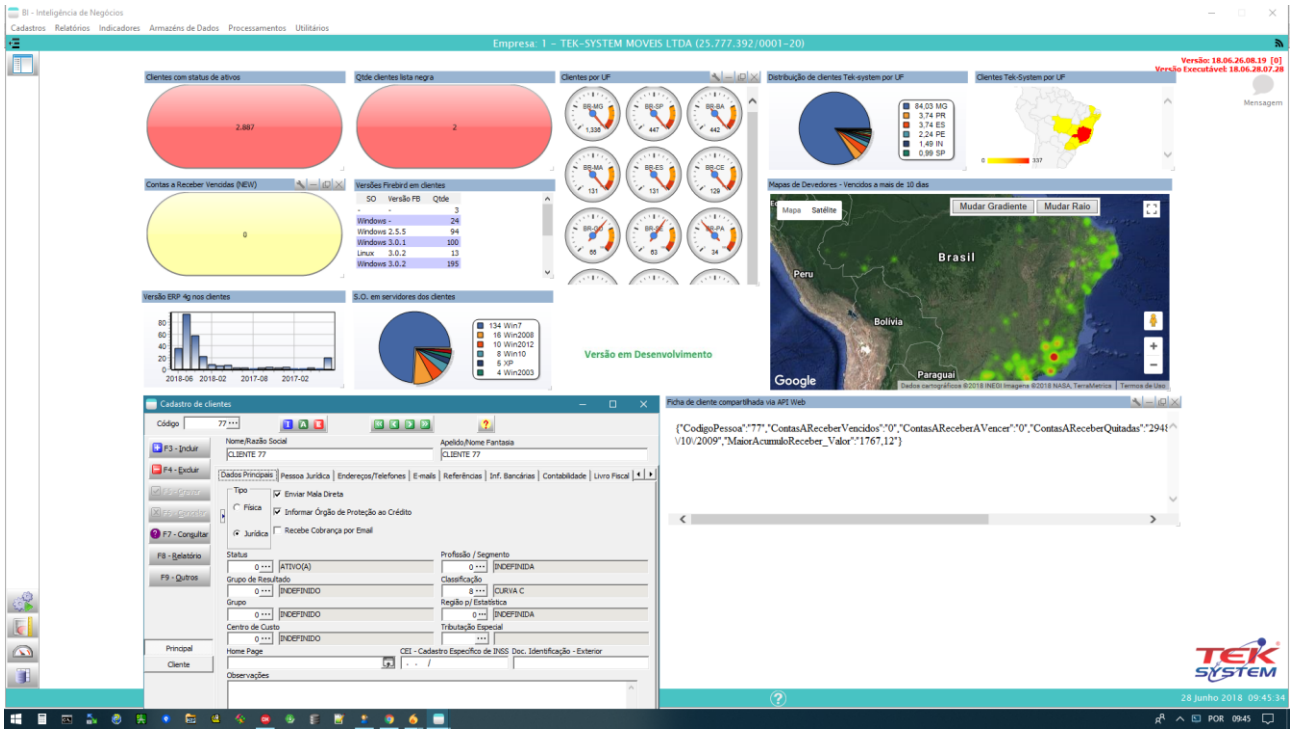
Os indicadores associados ao cadastro podem ser acessados através do botão F9-Outros, opção Ver Indicadores Relacionados.



Uma lista com os indicadores ativos, que o usuário tem acesso, associados ao cadastro será apresentada. O usuário deverá escolher quais indicadores deseja visualizar naquele momento.

Código	Descrição Amigável do Indicador	Descrição Oficial do Indicador	Tipo de Relacionamento
1	Clientes com status de ativos	CLIENTES COM STATUS DE ATIVOS	0-Global (Não envia informações ao indicador)
2	Clientes com status de ativos (MG)	CLIENTES COM STATUS DE ATIVOS (MG)	0-Global (Não envia informações ao indicador)
548	Clientes por UF	CLIENTES POR UF	0-Global (Não envia informações ao indicador)
348	Clientes que mais usaram o suporte (ult30d)	CLIENTES QUE MAIS USARAM O SUPORTE (ULT30	0-Global (Não envia informações ao indicador)
376	Clientes Tek-System por UF	CLIENTES TEK-SYSTEM POR UF	0-Global (Não envia informações ao indicador)
1.032	Contas a Receber Vencidas (NEW)	NEW - CONTAS A RECEBER VENCIDAS (VALOR)	2-Vinculado à JSON do Registro
644	Distribuição de clientes Tek-system por UF	DISTRIBUIÇÃO DE CLIENTES TEK-SYSTEM POR UF	0-Global (Não envia informações ao indicador)
991	Ficha de cliente compartilhada via API Web	FICHA DE CLIENTE COMPARTILHADA VIA API WEI	1-Vinculado à Chave (id) do Registro
660	Mapas de Devedores - Vencidos a mais de 10 dias	MAPAS DE DEVEDORES - VENCIDOS A MAIS DE 10	0-Global (Não envia informações ao indicador)
1.028	Número Informado via Parâmetro RequestBodyJsc	NÚMERO INFORMADO VIA PARÂMETRO REQUEST	1-Vinculado à Chave (id) do Registro
563	Qtde aniversariantes - mês atual	QTDE ANIVERSARIANTES - MES ATUAL	0-Global (Não envia informações ao indicador)
383	Qtde clientes c/mais de 1 servidor de aplicação	QTDE CLIENTES C/MAIS DE 1 SERVIDOR DE APLIC	0-Global (Não envia informações ao indicador)
142	Qtde clientes lista negra	QTDE CLIENTES LISTA NEGRA	0-Global (Não envia informações ao indicador)
1.024	RequestBodyJSON	REQUEST_BODY_JSON	2-Vinculado à JSON do Registro
382	S.O. em servidores dos clientes	S.O. EM SERVIDORES DOS CLIENTES	0-Global (Não envia informações ao indicador)
695	Serviços cobrados por clientes	SERVIÇOS COBRADOS POR CLIENTES	0-Global (Não envia informações ao indicador)
479	Versão ERP 4g nos clientes	VERSÃO ERP4G NOS CLIENTES	0-Global (Não envia informações ao indicador)
381	Versões Firebird em clientes	VERSÕES FIREBIRD EM CLIENTES	0-Global (Não envia informações ao indicador)

Uma janela de tamanho razoável será exibida para cada indicador selecionado. Cabe ao usuário redimensioná-las para visualizar melhor as informações.



O tipo de relacionamento do indicador com o cadastro indica como o indicador deverá se comportar quando no cadastro houver a rolagem para outro registro:

0) **Global (Não envia informações ao indicador)** -> O indicador é relacionado ao cadastro como um todo, não receberá parâmetros ao rolar registros, portanto não requererá o recálculo. Este é o relacionamento mais fácil de ser feito pois não requer modificação na codificação. Ex.: Indicador: *Contas a receber em aberto*. Pode ser relacionado com o cadastro de duplicatas a receber, com o cadastro de clientes... O funcionamento é como se cada cadastro fosse agora um painel de bordo.

1) **Vinculado à Chave (id) do Registro** -> O indicador depende do registro selecionado para que seu cálculo seja efetuado. Receberá o código chave (sequência) do cadastro. Também é simples de implementar pois o parâmetro é apenas um código inteiro. Normalmente é usado quando o indicador é vinculado a apenas um cadastro. Ex.: Indicador: *Contas a receber do cliente*. Relacionado apenas ao cadastro de clientes.

2) **Vinculado à JSON do Registro** -> O indicador depende do registro selecionado para que seu cálculo seja efetuado. Receberá um arquivo do tipo JSON com todos os campos da tabela principal do cadastro e mais algumas informações de controle. Requererá uma codificação um pouco mais elaborada, mas permite, por exemplo, que um mesmo indicador seja relacionado a diversos tipos de cadastros. Ex.: Indicador: *Contas a receber da pessoa*. Pode ser relacionado com o cadastro de clientes, com o cadastro de pedidos (se relacionando com a pessoa do pedido), com o cadastro de assistência (se relacionando com a pessoa da assistência), com o cadastro de duplicatas (se relacionando com a pessoa da duplicata) ... Veja exemplo de codificação ao final deste documento.

- Criada a possibilidade de **alternar a exibição de indicadores do tipo WebBrowser**. Esta exibição alternativa mostrará a codificação da página HTML resultante da interpretação da codificação do indicador com cores de destaque. Em algumas situações, por exemplo em indicadores que usam a API do Google Maps, que utilizam toda a área útil do indicador, é difícil a abertura do menu de contexto da página em execução para selecionar a opção “Exibir código fonte”.

- Criada a **possibilidade de visualizar um indicador a partir de um processamento específico**. Veja codificação de exemplo ao final deste documento.

- **Recompatibilização com API Google Maps** para visualizar endereços no mapa. Devido a mudanças na API pela Google, de vez em quando torna-se necessário esta atualização.

Exemplos de relacionamentos que podem ser feitos entre indicadores e cadastros:

- A partir do cadastro de duplicatas:
  - → Vlr. Total de Duplicatas Vencidas
  - → Vlr. Total de Duplicatas a Vencer
  - → Vlr. Duplicatas Vencidas do Cliente
  - → Últimas compras do cliente da duplicata
  - → Mapa de devedores

- A partir do cadastro de projetos ou contas a pagar:
  - → % limite gastos por projetos
  - → Detalhamento de gastos por projetos
  - → Projetos em andamento
  - → Proporção de contas a pagar para contas a receber
  - → Contas a pagar por grupo de resultado
  - → Contas a pagar por UF ou Banco

- A partir do cadastro de clientes:
  - → Vlr. Duplicatas Vencidas do Cliente
  - → Compras nos últimos 6 meses
  - → Qtde Cliente Inativos
  - → Distribuição de Clientes por UF
  - → Últimos 10 produtos adquiridos

- A partir do cadastro de consultores:
  - → Clientes inativos de cada consultor
  - → Meta atingida no mês do consultor
  - → Vlr. Inadimplência do consultor

- A partir do cadastro de assistência:
  - → % assistência sobre faturamento nos últimos 30 dias
  - → Itens que mais deram assistência
  - → Qtde assistência do cliente nos últimos 6 meses

- A partir do cadastro de carga:
  - → Cargas liberadas para faturamento
  - → Pedidos que estão bloqueados

- A partir do cadastro de cheques:
  - → Cheques não depositados

- A partir do cadastro de cidades/UF/região:
  - → Preço de combustíveis praticado na cidade
  - → Vendas realizadas para aquela UF/Cidade
  - → Duplicatas vencidas naquela cidade
  - → Recolhimento de assistências a serem feitos naquela região

- A partir do cadastro de contas ou movimento do caixa:
  - → Saldo Realizado
  - → Saldo a realizar
  - → Distribuição dos valores por conta

- A partir do cadastro de produtos/volumes/peças ou matérias primas:
  - → Custo do Estoque Total
  - → M^3 do Estoque Total
  - → Últimas compras deste item

- A partir do cadastro de veículo:
  - → Veículos em Viagem
  - → Veículos disponíveis
  - → Média da Idade da Frota de Veículos ativos
  - → últimas viagens daquele veículo.

- A partir do cadastro de moedas:
  - → Cotação atual da moeda
  - → Outros indicadores econômicos

- A partir do cadastro de ordens de pagamento:
  - → Vlr. Ordens Pagamento Não Identificadas
  - → Vlr. Ordens Pagamento Não Identificadas - por Conta

- A partir de Ordem de Produção:
  - → Exibir informações do pedido relacionado.

- A partir do cadastro de pedidos de venda:
  - → % assistência / faturamento
  - → % pedidos bloqueados - com opção de detalhar
  - → % atingimento meta venda geral
  - → % atingimento meta venda consultor do pedido
  - → % atingimento meta venda supervisor do pedido
  - → Duplicatas em aberto do cliente

- A partir do cadastro de status
  - → Qtde Pedidos naquele status - com opção de detalhar
  - → Qtde Clientes naquele status

Novas classes e funções disponibilizadas para interpretação

CriarFormPeloNome: Cria um formulário (tela) a partir de classes registradas do sistema.

PropriedadesDeClasse: Lista todas as propriedades de uma classe registrada no sistema.

PropriedadesDeObjeto: Lista todas as propriedades de um objeto criado em memória.

CamposDeClasse: Lista todos os campos (fields) de uma classe registrada no sistema.

CamposDeObjeto: Lista todos os campos (fields) de um objeto criado em memória.

MetodosDeClasse: Lista todos os métodos (procedures e functions) de uma classe registrada no sistema.

MetodosDeObjeto: Lista todos os métodos (procedures e functions) de um objeto criado em memória.

AtribuirValorPropriedadeDeObjeto: Atribui (SET) um valor a uma propriedade de um objeto criado em memória.

ValorPropriedadeDeObjeto: Recupera (GET) o valor de uma propriedade de um objeto criado em memória.

ExecutarMetodoDeObjeto: Executa um método (procedure ou function) de um objeto criado em memória.

FormCriadoPeloNome: Devolve a instancia de um formulário (tela) criado em memória.

DMCriadoPeloNome: Devolve a instancia de um Data Module criado em memória.

* A maioria destas funções faz uso de RTTI em suas execuções.

Novos modelos de Armazéns de dados

TEK-> PCP: APONTAMENTO DE PRODUÇÃO E RETRABALHO

Novos modelos de indicadores padrões disponibilizados

FIN-> VLR.CONTAS A RECEBER - POR SITUAÇÃO

FIN-> VLR.CONTAS A RECEBER VENCIDAS - POR SITUAÇÃO

FIN-> VLR.CONTAS A RECEBER A VENCER - POR SITUAÇÃO

ODBC -> INTELIPOST: DASHBOARD PEDIDOS (COMERCIO)

Novos modelos de Processamentos Específicos

TEK-> FACILITADOR DE MANUTENÇÃO DE CEP

Integrações Homologadas / Atualizadas

Tek-System - API ECOMMERCES

Arquivos JSON c/ Lay-out proprietário da Tek-System

Intelipost (Comercial)

Madeira Madeira

FBits

EZ-Commerce

Dafiti

SkyHub

Integracommerce

XCommerce

Magento

NEOGRID

GAZIN

Audaces

Codificações para testes das classes/funções

- **Exemplo de codificação de indicador relacionado a cadastro com o tipo de relacionamento "2-vinculado à JSON do Registro":**

```
unit ContasAReceberVencidas;

var
  ClasseCadastro: string;
  RegistroVazio: Boolean;
  CodigoPessoa: Integer;

function Main: Currency;
begin
  CarregarParametros;
  Result := ExecuteScalar(MontaSQL(False));
end;

procedure CarregarParametros;
begin
  CodigoPessoa   := -1;
  ClasseCadastro := GetValueJson(RequestBodyJSON, 'ClasseCadastro');
  RegistroVazio  := GetValueJson(RequestBodyJSON, 'RegistroVazio');

  if (ClasseCadastro <> '') then
    begin
      if (ClasseCadastro = 'TClassPessoaCad_Cliente') then
        CodigoPessoa := GetValueJson(RequestBodyJSON, 'CODIGO_PESSOA') else
      if (ClasseCadastro = 'TClassPedidoVenda') then
        CodigoPessoa := GetValueJson(RequestBodyJSON, 'CLIENTE_DOCFAT')
      else
        raise Exception.Create(MensagemPersonalizada + 'Indicador não preparado
para realizar o cálculo a partir do cadastro cuja classe é ' +
QuotedStr(ClasseCadastro));
    end;
end;

function MontaSQL(Detalhada: Boolean): String;
var SQL, Campos: string;
begin
  SQL := ' select ' + #13;

  if Detalhada then
    SQL := SQL +
      '   DUPLICATA.DOCUMENTO_DUP                "Documento",'  + #13 +
      '   cast(DUPLICATA.VENCIMENTO_DUP as Date) "Vencimento",' + #13 +
      '   DUPLICATA.VALORABERTO_DUP              "Vlr.Aberto"'
  else
    SQL := SQL + '   sum(DUPLICATA.VALORABERTO_DUP)';

  SQL := SQL + #13 +
    ' from DUPLICATA' + #13 +
    ' where DUPLICATA.TIPO_DUP           = 1' + #13 +
    '   and DUPLICATA.VENCIMENTO_DUP     < ' + DataSQL(HOJE, 0) + #13 +
    '   and DUPLICATA.VALORABERTO_DUP    > 0' + #13 +
    '   and DUPLICATA.SUBSTITUIDA_DUP    = ' + QuotedStr('N') + #13 +
    '   and DUPLICATA.DESCONSIDERADA_DUP = ' + QuotedStr('N') + #13;

  if (ClasseCadastro <> '') then // Se está relacionado a algum cadastro e não é
de forma global então fará algum filtro
    begin
      if RegistroVazio then // Não tem informações para identificar, então
mostrará vazio
```

```
        SQL := SQL + ' and DUPLICATA.AUTOINC_DUP = -1'
      else // Deverá relacionamento com a pessoa
        SQL := SQL + ' and DUPLICATA.PESSOA_DUP = ' + IntToStr(CodigoPessoa);
    end;

  Result := SQL;
end;

end.
```

- **Exemplo de codificação usando as funções de RTTI. Criando um indicador a partir da execução de um processamento específico.**

```
procedure Main;
var FormIndicador: TFIndicador;
begin
  MostrarLogTexto(PropriedadesDeClasse('UIndicador', 'TFIndicador'),
'Propriedades da Classe', False);
  MostrarLogTexto(CamposDeClasse('UIndicador', 'TFIndicador'), 'Campos da
Classe', False);
  MostrarLogTexto(MetodosDeClasse('UIndicador', 'TFIndicador'), 'Métodos da
Classe', False);

  FormIndicador := CriarFormPeloNome('FIndicador');

  AtribuirValorPropriedadeDeObjeto(FormIndicador, 'CodigoIndicador',    651);
  AtribuirValorPropriedadeDeObjeto(FormIndicador, 'TipoIndicador',      7);
  AtribuirValorPropriedadeDeObjeto(FormIndicador, 'DescricaoIndicador', 'Teste
Indicador');

  ExecutarMetodoDeObjeto(FormIndicador, 'Configurar', [150, 400, 8]);
//  FormIndicador.BorderWidth := 1;
//  FormIndicador.Color := clBlack;
  FormIndicador.Show;

  Application.ProcessMessages;

  MostrarLogTexto(PropriedadesDeObjeto(FormIndicador), 'Propriedades do Objeto',
False);
  MostrarLogTexto(CamposDeObjeto(FormIndicador), 'Campos do Objeto', False);
  MostrarLogTexto(MetodosDeObjeto(FormIndicador), 'Métodos do Objeto', False);

  ShowMessage('Descrição do Indicador: ' +
ValorPropriedadeDeObjeto(FormIndicador, 'DescricaoIndicador'));
end;
```

- **Manipulando 2 indicadores chamados em processamentos específicos, através de seus parâmetros.**

```
unit ProcessamentoEspecifico;

var sRequestBodyJSON: string;

procedure Main;
var FormIndicador1, FormIndicador2: TFIndicador;
begin
  sRequestBodyJSON := '';

  FormIndicador1 := CriarFormPeloNome('FIndicador');
  FormIndicador2 := CriarFormPeloNome('FIndicador');
  try
    // Configuração do primeiro indicador
        FormIndicador1.Caption := 'Ficha de cliente: ';
      FormIndicador1.Top  := 100;
        FormIndicador1.Left := 100;

        AtribuirValorPropriedadeDeObjeto(FormIndicador1, 'CodigoIndicador',
991);
        AtribuirValorPropriedadeDeObjeto(FormIndicador1, 'TipoIndicador',
9);
        AtribuirValorPropriedadeDeObjeto(FormIndicador1, 'DescricaoIndicador',
FormIndicador1.Caption);
        AtribuirValorPropriedadeDeObjeto(FormIndicador1, 'RequestBodyJSON',
sRequestBodyJSON);

      ExecutarMetodoDeObjeto(FormIndicador1, 'Configurar', [100, 450, 8]);

        FormIndicador1.Show;

    // Configuração do segundo indicador
        FormIndicador2.Caption := 'Parametro: ';
      FormIndicador2.Top  := 300;
        FormIndicador2.Left := 100;

        AtribuirValorPropriedadeDeObjeto(FormIndicador2, 'CodigoIndicador',
1024);
        AtribuirValorPropriedadeDeObjeto(FormIndicador2, 'TipoIndicador',
9);
        AtribuirValorPropriedadeDeObjeto(FormIndicador2, 'DescricaoIndicador',
FormIndicador2.Caption);
        AtribuirValorPropriedadeDeObjeto(FormIndicador2, 'RequestBodyJSON',
sRequestBodyJSON);

      ExecutarMetodoDeObjeto(FormIndicador1, 'Configurar', [100, 450, 8]);

        FormIndicador2.Show;

        Application.ProcessMessages;

    sRequestBodyJSON := '{"cnpj_cpf":"751.481.536-87"}';
        while ConfirmarFiltros do
          begin
        sRequestBodyJSON := Filtro(1);

        // Ajustando a exibição do primeiro indicador de acordo com novo
parâmetro
        AtribuirValorPropriedadeDeObjeto(FormIndicador1, 'DescricaoIndicador',
FormIndicador1.Caption + sRequestBodyJSON);
        AtribuirValorPropriedadeDeObjeto(FormIndicador1, 'RequestBodyJSON',
sRequestBodyJSON);
        ExecutarMetodoDeObjeto(FormIndicador1, 'AtualizarValorIndicador');
```

```
        // Ajustando a exibição do segundo indicador de acordo com novo
parâmetro
        AtribuirValorPropriedadeDeObjeto(FormIndicador2, 'DescricaoIndicador',
FormIndicador2.Caption + sRequestBodyJSON);
        AtribuirValorPropriedadeDeObjeto(FormIndicador2, 'RequestBodyJSON',
sRequestBodyJSON);
        ExecutarMetodoDeObjeto(FormIndicador2, 'AtualizarValorIndicador');
      end;
  finally
    FormIndicador1.Close;
    FormIndicador2.Close;
  end;
end;

function ConfirmarFiltros: Boolean;
var CDSFiltros: TClientDataSet;
begin
  CDSFiltros := TClientDataSet.Create;
  try
    CDSFiltros.Data := EstruturaDeFiltrosDinamicos;
    {01} IncluirFiltroDinamico(CDSFiltros, 'RequestBodyJson', cTipoFiltro_Memo,
'', '', '', sRequestBodyJSON);
    CDSFiltros.Data := ExecutarFiltroDinamico(CDSFiltros.Data, 'Informe os
parâmetros para os indicadores');

    Result := (not CDSFiltros.IsEmpty);
  finally
    CDSFiltros.Free;
  end;
end;

end.
```


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)

"O seu Deus o ensina e o instrui acerca do que há de fazer" Is 28.26