

# COMO EVITAR HORAS DE TRABALHO E CRIAR ALV'S EM 10 MINUTOS COM 3 PASSOS SIMPLES

Aumentando sua performance em 8,5x.

## Resumo

Utilizando a classe CL\_SALV\_TABLE e um pouco de organização você irá produzir ALV's em minutos investindo seu tempo onde realmente importa, criando soluções.

Leandro Amancio Nunes  
[openABAP.com](http://openABAP.com)

## Sumário

Introdução.....	2
O resultado.....	3
Como funciona? .....	4
Descrevendo o passo 1: <i>O início de tudo.</i> ....	4
Descrevendo o passo 2: <i>Enfim um ALV</i> .....	5
Descrevendo o passo 3: <i>O estado da arte</i> .....	6
Criando um repositório de funcionalidades passo-a-passo. ....	7
Pausa para alguns possíveis esclarecimentos. ....	8
E agora, como eu uso o repositório? .....	9
Bônus.....	11
BÔNUS 1 – Como mostrar o ALV em um pop-up; .....	12
BÔNUS 2 – Como alterar a descrição de uma coluna específica; .....	13
BÔNUS 3 – Como alterar a cor de uma célula específica; .....	14
Considerações finais.....	16

## Introdução.

Olá caro consultor!

Há 5 anos descobri e venho utilizando esse atalho, economizando preciosas horas de trabalho tedioso e investindo-as onde realmente importa, nas regras de negócio e inteligência das soluções que ajudo a criar.

Não importa se você é um consultor experiente ou um ABAP recém-saído da academia, se deseja aumentar sua performance em mais de 8x no desenvolvimento e manutenção de seus relatórios ALV, tem que conhecer a técnica de 3 passos utilizando a classe CL\_SALV\_TABLE que utilizo. É extremamente simples e vou compartilhar contigo, apenas permita que eu me apresente.

Me chamo Leandro Amancio Nunes e sou idealizador do blog [openABAP.com](http://openABAP.com).

Trabalho com TI desde 2003 e com desenvolvimento de sistemas ERP desde 2005. Dos mais de 10 anos de experiência que possuo, 7 são dedicados a essa exclusiva e apaixonante linguagem, o **ABAP** (Advanced Business Application Programming).

Embora já há algum tempo trabalhando com SAP, estou longe de ser o profissional que gostaria pois tenho muito ainda a aprender, esse é um dos itens que me motivou a criar o blog openABAP, para que com as dúvidas e problemas vivenciados por mim e por outras pessoas eu pudesse aprender mais e assim também, ajuda-las.

Não sou escritor nem nada do gênero, mas resolvi criar esse e-book por que me lembro o quanto eu tive de penar copiando e colando código para o preenchimento do famoso *FieldCatalog*... muitas e muitas vezes eu fiz isso, até conhecer esse atalho que estou prestes a lhe ensinar.

Aliás, essa é a **missão do blog**, transformar a vida do maior número de pessoas possível com o ABAP, ele me proporcionou muitas conquistas e realizações e quero que outras pessoas também possam sentir essa sensação e quem sabe ajudar mais pessoas.

Pintou uma dúvida ou curiosidade? Pergunta lá no [blog](http://openABAP.com)!

E agora, sem mais enrolação, a mágica...

## O resultado.

Bem, se você está lendo esse e-book suponho que conheça o exclusivo mundo do ABAP e esteja interessado em aumentar sua produtividade com a técnica que vou te ensinar, mas, caso seja iniciante leia até o final pois quando chegar ao ponto de criar relatórios ALV saberá a maneira prática de construí-los.

Enfim, ao final desse e-book, você será capaz de criar um relatório ALV utilizando a classe CL\_SALV\_TABLE efetuando os seguintes 3 passos.

\*"Pense no caso hipotético em que temos a necessidade de criar um ALV para mostrar alguns dados do mestre de materiais.

```
* "Tabela interna onde armazenaremos os dados do ALV
DATA ti_mara TYPE TABLE OF mara.

* "Nome da tabela interna que contem os dados do ALV
CONSTANTS c_nm_tab_interna TYPE c LENGTH 07 VALUE 'TI_MARA'.

*      "Passo 1
PERFORM: zf_seleciona_mara USING ti_mara,
*      "Passo 2
        zf_criar_alv      USING c_nm_tab_interna,
*      "Passo 3
        zf_mostrar_salv   USING abap_true "-->Determina a utilização do STATUS GUI padrão
                                abap_true."-->Otimiza a largura das colunas em função do conteúdo
```

Voilà, veja o resultado abaixo, simples assim, sem suor e um amontoado de código e textos fixos!

Relatório criado em 10 minutos com os 3 passos.											
Manda...	Material	Criado em	Última modific.	StatAt...	Status	Nív.ma...	TpMaterial	Set.ind...	Grp.me...	NºMaterial	Unida
100	301288	14.12.2001	12.07.2014	KEDBPX	KEDBP		HIBE	B	4208	14020002	UN
100	301289	14.12.2001	22.02.2002	KEDBPX	KEDBP		HIBE	B	4208	2941	UN
100	301290	14.12.2001	22.02.2002	KEDBPX	KEDBP		HIBE	B	4208	1405	UN
100	301291	14.12.2001	22.02.2002	KEDBPX	KEDBP		HIBE	B	4208	1406	UN
100	301449	14.12.2001	22.02.2002	KEBP	KEBP		HIBE	B	4501	9892	UN
100	301183	14.12.2001	22.02.2002	KEDBPX	KEDBP		HIBE	B	4309	8694	UN
100	301184	14.12.2001	22.02.2002	KEBPD	KEBPD		HIBE	B	4309	1046	UN
100	301185	14.12.2001	22.02.2002	KEDBPX	KEDBP		HIBE	B	4111	1131	UN
100	301186	14.12.2001	22.02.2002	KEDBPX	KEDBP		HIBE	B	4116	9171	UN
100	301187	14.12.2001	22.02.2002	KEDBPX	KEDBP		HIBE	B	4116	9172	UN

Note que o ALV possui a estrutura da tabela interna utilizada na sua criação (TI\_MARA), mas é possível modificar esse e outros itens, porém, falaremos sobre isso mais adiante.

## Como funciona?

Abaixo descreverei cada um dos 3 passos para que compreenda o funcionamento, porém, você somente terá de implementar esses passos uma única vez, é muitíssimo simples, e mesmo que você não compreenda exatamente, seguindo todos os passos até o final você terá uma “maleta de ferramentas” para utilizar no seu dia-a-dia.

Mostrarei como organizar os *performs* de uma maneira que possa reutiliza-los em todos os seus próximos desenvolvimentos.

Descrevendo o passo 1: *O início de tudo.*

```
*          "Passo 1
PERFORM: zf_seleciona_mara USING c_nm_tab_interna.

*&-----*
*&      Form  ZF_SELECIONA_MARA
*&-----*
*      Seleciona os dados da tabela MARA e armazena na tabela interna
*      passada como parâmetro
*-----*
*      -->P_NM_TAB_INTERNA --> Armazenará o nome da tabela interna que
*armazena o resultado da seleção, neste caso TI_MARA.
*-----*
FORM zf_seleciona_mara USING p_nm_tab_interna TYPE dbobj_name.

* "Fiz declarações locais neste caso para ficar mais simples a compreensão
  FIELD-SYMBOLS <fs_tab_mara> TYPE ANY TABLE.

* "Atribuimos a area de memória da tabela TI_MARA ao <fs_tab_mara>
  ASSIGN (p_nm_tab_interna) TO <fs_tab_mara>.

* "Caso a atribuição tenha sido bem sucedida
  IF <fs_tab_mara> IS ASSIGNED.

*      "Selecionamos os dados tabela transparente MARA e armazenamos os dados
dentro de TI_MARA representada nesse caso por <FS_TAB_MARA>
  REFRESH <fs_tab_mara>.
  SELECT * UP TO 10 ROWS "Limitamos a 10 registros
    INTO CORRESPONDING FIELDS OF TABLE <fs_tab_mara>
    FROM mara.
  ENDIF.

ENDFORM.
```

Obs<sup>1</sup>. O “passo 1” foi criado apenas para ilustrar a necessidade de se ter preenchida a tabela interna que será utilizada no ALV. Faça a seleção e o preenchimento como achar melhor, os dados da sua tabela serão refletidos no ALV.

Obs<sup>2</sup>. Declaramos os objetos dentro da rotina para melhor compreensão, falaremos sobre isso mais adiante.

## Descrevendo o passo 2: *Enfim um ALV*

```
*          "Passo 2
PERFORM: zf_criar_alv USING c_nm_tab_interna.

*&-----*
*&      Form  ZF_CRIAR_ALV
*&-----*
*      Instancia objeto do tipo CL_SALV_TABLE e atribui tabela interna
*-----*
*      -->P_NM_TAB_INTERNA --> Armazena os dados que serão mostrados no ALV
*-----*
FORM zf_criar_alv USING p_nm_tab_interna TYPE dbobj_name.

*"Deixei as declarações locais neste caso para ficar mais clara a compreensão
DATA r_salv TYPE REF TO cl_salv_table.
FIELD-SYMBOLS <fs_tabela> TYPE ANY TABLE.

UNASSIGN <fs_tabela>.
FREE r_salv.

* "Atribuimos a area de memória da tabela TI_MARA ao <fs_tab_mara>
ASSIGN (p_nm_tab_interna) TO <fs_tabela>.
CHECK <fs_tabela> IS ASSIGNED.

* "Criamos objeto R_SALV com referência classe ALV utilizando dados da TI_MARA
CALL METHOD cl_salv_table=>factory
IMPORTING
    r_salv_table = r_salv
CHANGING
    t_table      = <fs_tabela>.

ENDFORM.
```

Pronto!

Neste momento já temos nosso ALV em memória, porém, ainda não é mostrado em tela pois existe uma enorme quantidade de customizações possíveis de serem feitas.

Neste caso não iremos alterar nenhuma propriedade ou comportamento do GRID, mas no **capítulo bônus** iremos alterar a cor de uma célula específica, a descrição de uma das colunas, mostrar o ALV em um pop-up muito profissional, além é claro de falar sobre as infinitas possibilidades que essa classe nos oferece.

### Descrevendo o passo 3: *O estado da arte*

```
*           "Passo 3
PERFORM: zf_mostrar_salv USING abap_true "Determina utilização STATUS GUI padrão
                                         abap_true."Otimiza a largura das colunas

*&-----*
*&      Form  ZF_MOSTRAR_SALV
*&-----*
*      Mostra efetivamente o ALV em tela com todas modificações
* implementadas
*-----*
*p_status_padrao      --> Determina a utilização do STATUS GUI padrão
*p_otimizar_largura_colunas --> Largura das colunas em função do conteúdo
*-----*
FORM zf_mostrar_salv USING p_status_padrao          TYPE abap_bool
                           p_otimizar_largura_colunas TYPE abap_bool.

DATA r_funcoes TYPE REF TO cl_salv_functions_list.

* "Instanciamos um objeto que irá tratar das funcionalidades padrões do ALV
* , 'dizemos' a ele que não queremos nenhum STATUS GUI personalizado e
* que também deverá utilizar um status GUI padrão (BCALV_TEST_FULLSCREEN)

FREE r_funcoes.
r_funcoes = r_salv->get_functions( ).
r_funcoes->set_all( p_status_padrao ).

* "Se for necessário deixamos as colunas do tamanho de seu conteúdo, ou seja,
* terá a largura do maior conjunto de dados contida em cada coluna
IF p_otimizar_largura_colunas EQ abap_true.
  FREE r_columns.
  r_columns = r_salv->get_columns( ).
  r_columns->set_optimize( abap_true ).
ENDIF.

* "Mostramos em tela
r_salv->display( ).
ENDFORM.
```

Após esse momento não poderemos interferir no comportamento nem aparência do ALV, somente interagir com ele em tela.

É como eu disse meu caro futuro amigo *expert* em ALVs, é assim que ficará conhecido por onde passar, não tem segredo, é muito simples de se utilizar essa facilidade disponibilizada pela própria SAP que fez isso exatamente para otimizar o trabalho dos ABAPers.

Com pouco tempo de estudo e dedicação sobre o assunto, você sequer se lembrará das outras maneiras de se fazer relatórios.

Veremos a seguir como organizar essas rotinas uma única vez para todos os seus desenvolvimentos.

## ***Criando um repositório de funcionalidades passo-a-passo.***

Penso que, a organização é um dos principais fatores para o sucesso de qualquer projeto, falaremos muito sobre esse tópico no blog, mas para que esse método de criação de ALV's em 3 passos funcione bem, a organização é a chave, é ela que nos permitirá ser muito mais produtivos com muito menos trabalho, esforço e código.

No capítulo bônus iremos falar sobre itens que poderão ser facilmente incorporados ao nosso repositório aumentando ainda mais sua produtividade.

Chega de enrolação, passemos ao que interessa.

Para conceber nosso repositório será necessário criar um include que conterá 2 dos 3 passos.

Faremos isso de maneira que seja possível reutiliza-los em todos os programas, novos ou não, de agora até o final dos tempos sem nunca ser necessário uma única manutenção.

Para economizar tempo segue abaixo um link para você efetuar o *download* do código, copie e cole dentro do seu include, ative-o e seguimos com o que realmente interessa pois somos movidos a resultados.

Baixe agora o [Repositório de soluções e produtividade openABAP.com](https://openABAP.com)



Pausa para alguns possíveis esclarecimentos.

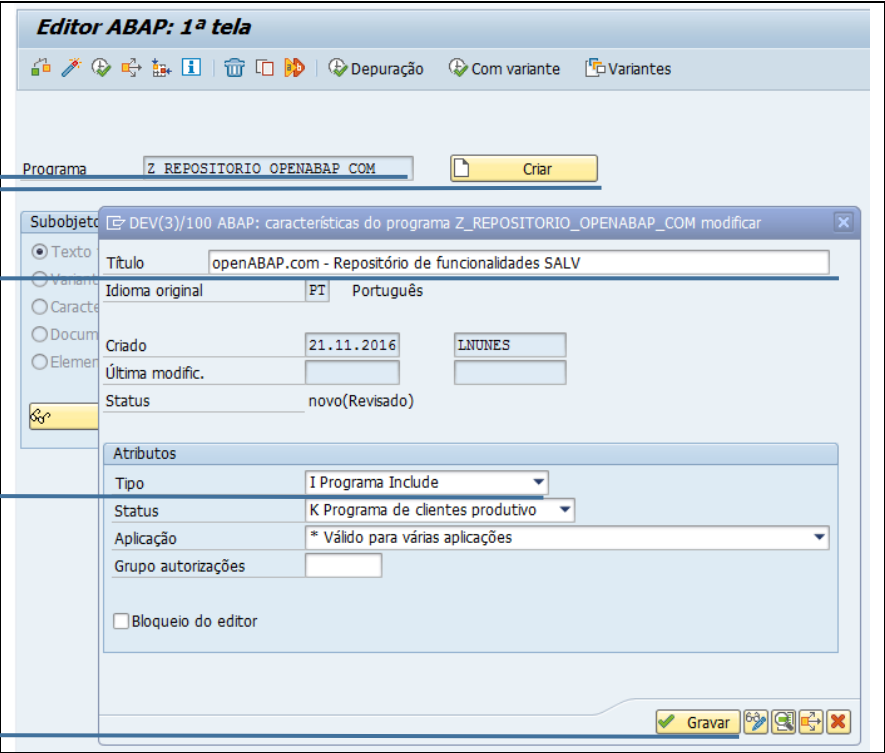
Caso você esteja se perguntando, “O que é um repositório?”.

É um lugar onde se guarda, arquiva ou coleciona alguma coisa. No nosso caso iremos guardar soluções e produtividade para reutilizar em vários programas.

Caso você esteja se perguntando, “Mas, o que é um include e como posso cria-lo?”.

É um programa fonte que contém codificações e que não poderá ser executado sozinho. Esta parte de programa deverá ser referenciada em nossos programas/relatórios por meio da instrução INCLUDE.

Para criar um programa *include* acesse a transação SE38 e faça o seguinte...

Informe o nome;	
Clique em criar;	
Informe o título;	
Informe o tipo “I include”;	
Clique em gravar;	

\*Não falaremos sobre os demais atributos possíveis, discutiremos isso no blog no momento adequado. Se tiver quaisquer dúvidas, me pergunte, terei prazer em ajuda-lo.

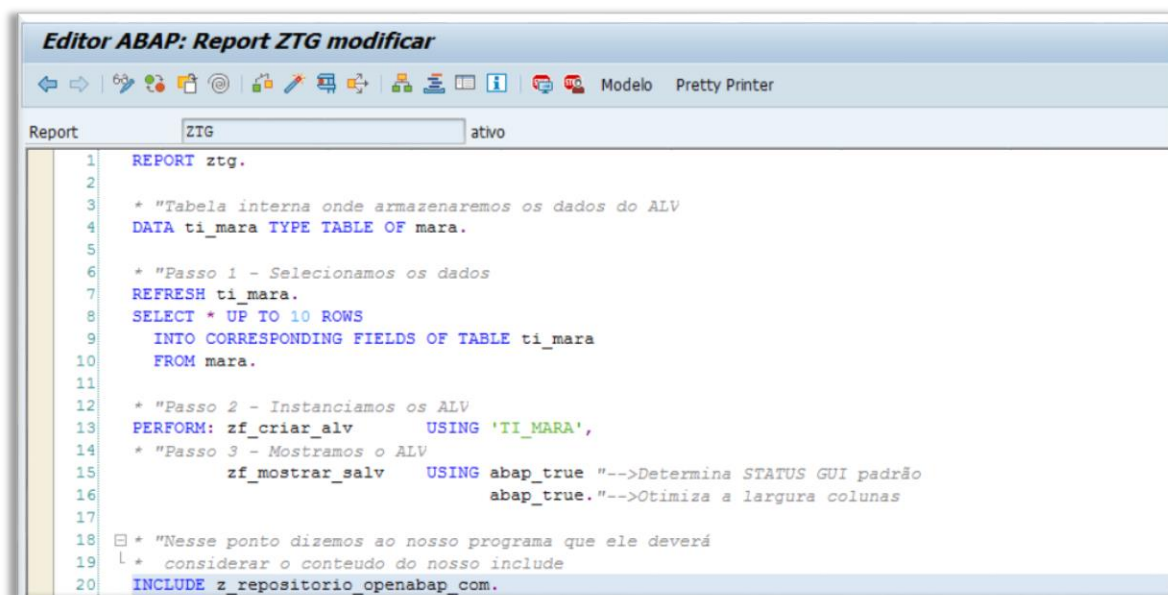
Informe o pacote e a request caso esteja criando de fato seu repositório ou selecione “objeto local” caso esteja apenas estudando.

Pronto, você acaba de criar um *include*.

E agora, como eu uso o repositório?

Não importa qual é a solução em que esteja trabalhando, basta “declarar” o include criado e poderá utilizar as rotinas que estão armazenadas dentro dele.

Veja no exemplo abaixo, criei o report ZTG (“Z” Teste Geral... pouco criativo, mas muito útil!!! rsrs) onde é feita a seleção de 10 materiais da tabela MARA e é mostrado em tela utilizando nosso repositório.



```
Editor ABAP: Report ZTG modificar
Report ZTG ativo
1 REPORT ztg.
2
3 * "Tabela interna onde armazenaremos os dados do ALV
4 DATA ti_mara TYPE TABLE OF mara.
5
6 * "Passo 1 - Selecionamos os dados
7 REFRESH ti_mara.
8 SELECT * UP TO 10 ROWS
9 INTO CORRESPONDING FIELDS OF TABLE ti_mara
10 FROM mara.
11
12 * "Passo 2 - Instanciamos os ALV
13 PERFORM: zf_criar_alv USING 'TI_MARA',
14 * "Passo 3 - Mostramos o ALV
15 zf_mostrar_salv USING abap_true "-->Determina STATUS GUI padrão
16 abap_true."-->Otimiza a largura colunas
17
18 * "Nesse ponto dizemos ao nosso programa que ele deverá
19 * considerar o conteudo do nosso include
20 INCLUDE z_repositorio_openabap_com.
```

Resultado...



Manda...	Material	Criado em	Última modif.	StatAt...	Status	Nív.ma...	TpMaterial	Set.ind...	Grp.me...	NºMaterial
100	301288	14.12.2001	12.07.2014	KEDBPX	KEDBP		HIBE	B	4208	14020002
100	301289	14.12.2001	22.02.2002	KEDBPX	KEDBP		HIBE	B	4208	2941
100	301290	14.12.2001	22.02.2002	KEDBPX	KEDBP		HIBE	B	4208	1405
100	301291	14.12.2001	22.02.2002	KEDBPX	KEDBP		HIBE	B	4208	1406
100	301449	14.12.2001	22.02.2002	KEBP	KEBP		HIBE	B	4501	9892
100	301183	14.12.2001	22.02.2002	KEDBPX	KEDBP		HIBE	B	4309	8694
100	301184	14.12.2001	22.02.2002	KEBPD	KEBPD		HIBE	B	4309	1046
100	301185	14.12.2001	22.02.2002	KEDBPX	KEDBP		HIBE	B	4111	1131
100	301186	14.12.2001	22.02.2002	KEDBPX	KEDBP		HIBE	B	4116	9171
100	301187	14.12.2001	22.02.2002	KEDBPX	KEDBP		HIBE	B	4116	9172

Deixei o passo número 1 fora do repositório justamente para frisar que esse passo será sempre diferente dependendo da aplicação, todos os tratamentos nos dados deverão ser feitos nesse passo antes de utilizarmos nossas funcionalidades. Esse será o item mais trabalhoso pois será sempre diferente.

Deixei, mas não deveria, o nome da tabela interna 'TI\_MARA' *hardcode* para ilustrar que o parâmetro deve ser o nome da tabela interna que contém os dados, mas deve-se criar uma constante ou utilizar dados dinâmicos.

E com 10 linhas de código temos nosso relatório ALV, **quanto de código você escreveria para ter o mesmo resultado?**

```
1.    REPORT ztg.

2.    DATA ti_mara TYPE TABLE OF mara.

3.    REFRESH ti_mara.
4.    SELECT * UP TO 10 ROWS
5.        INTO CORRESPONDING FIELDS OF TABLE ti_mara
6.        FROM mara.

7.    PERFORM: zf_criar_alv          USING 'TI_MARA',
8.            zf_mostrar_salv       USING abap_true
9.            abap_true.

10.   INCLUDE z_repositorio_openabap_com.
```

É isso que você fará em todos os seus relatórios, irá declarar o repositório e utilizar suas facilidades.

Inicialmente teremos apenas algumas, mas, se vocês quiserem e dividirem comigo suas dúvidas e sugestões, podemos implementar novas funcionalidades.

## Bônus.

Conforme mencionei irei presentear-lo com 3 funcionalidades dentre muitas, que a classe CL\_SALV\_TABLE nos oferece.

- 1 - Como mostrar o ALV em um pop-up profissional;
- 2 - Como alterar a descrição de uma coluna específica;
- 3 - Como alterar a cor de uma célula específica;

Além dessas três seguem mais algumas funcionalidades que poderíamos **implementar somente uma vez para reutiliza-las em todos os desenvolvimentos de sua empresa.**

- ✓ Tornar o ALV editável; **Na minha opinião é a mais poderosa das funcionalidades disponíveis, pois eliminará muitas telas de manutenção e as famosas *view's* de atualização de tabela. Se você tem interesse nesse item envie um e-mail para [lnunes@openabap.com](mailto:lnunes@openabap.com) ou deixe um contato no blog que estou desenvolvendo algo neste sentido.**
- ✓ Utilizar um status GUI "Z";
- ✓ Utilizar um Título customizado;
- ✓ Acrescentar *hotspot*, *link* e *check box* em determinadas células ou colunas;
- ✓ Controlar a forma como o usuário pode selecionar registros e colunas;
- ✓ Verificar quais registros, colunas e/ou células estão selecionados;
- ✓ Rotinas de interação de clique, duplo clique, *hotspot*, *link* e etc., famoso USER-COMMAND;
- ✓ Imagens no cabeçalho e rodapé, arrastar e soltar, impressão, ordenação, filtros, exportação pra Excel, controlar eventos, xml e é muita coisa... chega... não vou falar todas... ufa! rsrs;
- ✓ Essas são as que mais uso, mas tem muita facilidade nessa classe, muita mesmo;

Essa classe permite que se faça tudo e mais um pouco que o "ALV tradicional" faz, mas de forma muito mais simples e rápida. Ela nos permite organizar tudo de uma forma que economizaremos código, tempo e esforço desnecessários a essa atividade mecânica.

Vejamos a seguir, os bônus.

## BÔNUS 1 – Como mostrar o ALV em um pop-up;

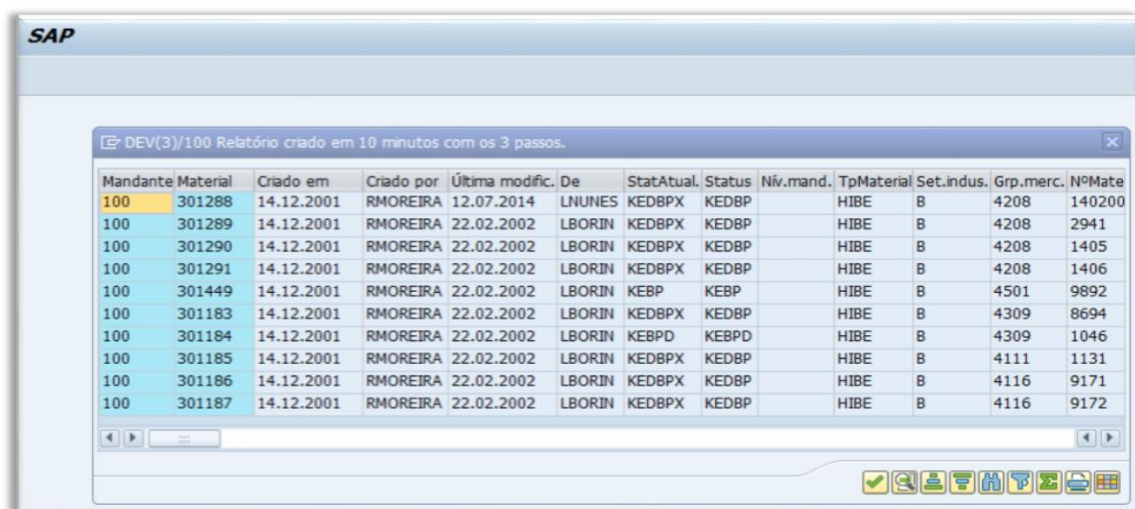
Caso esteja criando um pequeno ALV para mostrar uma lista de erros ou uma lista de atividades por exemplo, criar um pop-up é uma excelente saída. Para isso deveremos acrescentar ao nosso repositório o seguinte código:

```
*&-----*
*&      Form  ZF_MOSTRAR_SALV_EM_POPUP
*&-----*
*      Determina que o ALV será mostrado em tela em uma janela popup
*-----*
*      -->p_linha_ini   -->Linha onde será iniciada a janela do pop-up
*      -->p_coluna_ini  -->Coluna onde será iniciada a janela do pop-up
*      -->p_linha_fim   -->Linha onde será finalizada a janela do pop-up
*      -->p_coluna_fim  -->Coluna onde será finalizada a janela do pop-up
*-----*
FORM zf_mostrar_salv_em_popup USING p_linha_ini   TYPE i
                                p_coluna_ini   TYPE i
                                p_linha_fim    TYPE i
                                p_coluna_fim    TYPE i.

CALL METHOD r_salv->set_screen_popup
EXPORTING
    start_line      = p_linha_ini
    start_column    = p_coluna_ini
    end_line        = p_linha_fim
    end_column      = p_coluna_fim.
ENDFORM.                    " ZF_MOSTRAR_SALV_EM_POPUP
```

Informar em nosso programa a chamada desse *perform* depois de *instanciar-lo* e antes de *mostrá-lo* em tela para ter o resultado a seguir.

```
zf_mostrar_salv_em_popup USING 02  "p_linha_ini
                              10  "p_coluna_ini
                              12  "p_linha_fim
                              120, "p_coluna_fim.
```



Mandante	Material	Criado em	Criado por	Última modif.	De	StatAtual	Status	Nív.mand.	TpMaterial	Set.indus.	Grp.merc.	NºMate
100	301288	14.12.2001	RMOREIRA	12.07.2014	LNUNES	KEDBPX	KEDBP		HIBE	B	4208	140200
100	301289	14.12.2001	RMOREIRA	22.02.2002	LBORIN	KEDBPX	KEDBP		HIBE	B	4208	2941
100	301290	14.12.2001	RMOREIRA	22.02.2002	LBORIN	KEDBPX	KEDBP		HIBE	B	4208	1405
100	301291	14.12.2001	RMOREIRA	22.02.2002	LBORIN	KEDBPX	KEDBP		HIBE	B	4208	1406
100	301449	14.12.2001	RMOREIRA	22.02.2002	LBORIN	KEBP	KEBP		HIBE	B	4501	9892
100	301183	14.12.2001	RMOREIRA	22.02.2002	LBORIN	KEDBPX	KEDBP		HIBE	B	4309	8694
100	301184	14.12.2001	RMOREIRA	22.02.2002	LBORIN	KEBPD	KEBPD		HIBE	B	4309	1046
100	301185	14.12.2001	RMOREIRA	22.02.2002	LBORIN	KEDBPX	KEDBP		HIBE	B	4111	1131
100	301186	14.12.2001	RMOREIRA	22.02.2002	LBORIN	KEDBPX	KEDBP		HIBE	B	4116	9171
100	301187	14.12.2001	RMOREIRA	22.02.2002	LBORIN	KEDBPX	KEDBP		HIBE	B	4116	9172

## BÔNUS 2 – Como alterar a descrição de uma coluna específica;

Caso a descrição de uma coluna em particular não seja muito intuitiva podemos altera-la, nesse caso iremos alterar a descrição da coluna MATNR de “Material” para “Produto”. Para isso deveremos acrescentar ao nosso repositório o seguinte código:

```
*&-----*
*&      Form  ZF_ATRIBUIR_DESC_COLUNAS
*&-----*
*      Atribui descrições a uma coluna específica
*-----*
*      -->p_nome_coluna -->Nome da coluna que iremos usar
*      -->p_desc_curta  -->Descrição curta
*      -->p_desc_media  -->Descrição média
*      -->p_desc_longa  -->Descrição longa
*-----*
FORM zf_atribuir_desc_colunas USING p_nome_coluna TYPE lvc_fname
                                   p_desc_curta  TYPE scrtext_s
                                   p_desc_media  TYPE scrtext_m
                                   p_desc_longa  TYPE scrtext_l.

IF p_desc_curta IS INITIAL.
  MOVE: 'NO_S_DESC' TO p_desc_curta.
ELSEIF p_desc_media IS INITIAL.
  MOVE: 'NO_M_DESC' TO p_desc_media.
ELSEIF p_desc_longa IS INITIAL.
  MOVE: 'NO_LONG_DESCRIPTION' TO p_desc_longa.
ENDIF.

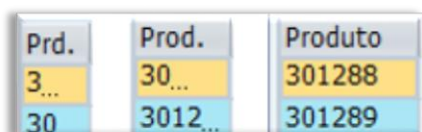
FREE: r_columns, r_column.

r_columns = r_salv->get_columns( ).
r_column = r_columns->get_column( p_nome_coluna ).
r_column->set_short_text( p_desc_curta ).
r_column->set_medium_text( p_desc_media ).
r_column->set_long_text( p_desc_longa ).
ENDFORM.                    " ZF_ATRIBUIR_DESC_COLUNAS
```

Informar em nosso programa a chamada desse *perform* depois de instancia-lo e antes de mostra-lo em tela para ter o resultado a seguir.

```
zf_atribuir_desc_colunas USING 'MATNR'      "p_nome_coluna
                              'Prd.'        "p_desc_curta 1
                              'Prod.'       "p_desc_media 2
                              'Produto',    "p_desc_longa 3
```

Temos 3 tamanhos de descrição que se adequam ao tamanho da coluna automaticamente caso o usuário modifique sua largura.



Prd.	Prod.	Produto
3...	30...	301288
30...	3012...	301289

### BÔNUS 3 – Como alterar a cor de uma célula específica;

Caso seja necessário marcar uma célula que contém uma data específica em vermelho por exemplo. Para isso deveremos agregar a seguinte rotina ao nosso repositório:

```
*&-----*
*&      Form  ZF_ADD_COR
*&-----*
*      Indica qual é a coluna que contém informações das cores
*-----*
*      -->p_nome_coluna_cor --
> Nome da coluna NO ALV que contém a configuração das cores
*-----*
FORM zf_add_cor USING p_nome_coluna_cor.

FREE: r_color.
r_color = r_salv->get_columns( ).
r_color->set_color_column( p_nome_coluna_cor ).

ENDFORM.                    " ZF_COLORIR
*&-----*
*&      Form  ZF_PINTA_CELULA
*&-----*
*      Usado para preenchimento de dados para colorir celulas do SALV
*-----*
*      -->p_nome_coluna "Nome coluna
*      -->p_cor         "Cor
*      <--
p_estrutura "Estrutura/coluna do ALV que armazena informações de cores
*-----*
FORM zf_pinta_celula USING p_nome_coluna TYPE lvc_fname
                        p_cor         TYPE lvc_col
                        CHANGING p_estrutura TYPE lvc_t_scol.
DATA wa_cor TYPE lvc_s_scol.

CLEAR wa_cor.
MOVE: p_nome_coluna TO wa_cor-fname,
      p_cor         TO wa_cor-color-col.
APPEND wa_cor TO p_estrutura.

ENDFORM.                    " ZF_PINTA_CELULA
```

Informar em nosso programa a chamada desse *perform* depois de instancia-lo e antes de mostra-lo em tela para ter o resultado a seguir.

```
zf_add_cor USING 'COR',
```



Mandante	Produto	Criado em	Criado por	Última modif.	De	StatAtual	Status	Nív.mand.	TpMaterial	Set.indus.	Grp.merc.	NºMate
100	301288	14.12.2001	RMOREIRA	12.07.2014	LNUNES	KEDBPX	KEDBP		HIBE	B	4208	140200
100	301289	14.12.2001	RMOREIRA	22.02.2002	LBORIN	KEDBPX	KEDBP		HIBE	B	4208	2941
100	301290	14.12.2001	RMOREIRA	22.02.2002	LBORIN	KEDBPX	KEDBP		HIBE	B	4208	1405
100	301291	14.12.2001	RMOREIRA	22.02.2002	LBORIN	KEDBPX	KEDBP		HIBE	B	4208	1406
100	301449	14.12.2001	RMOREIRA	22.02.2002	LBORIN	KEBP	KEBP		HIBE	B	4501	9892
100	301183	14.12.2001	RMOREIRA	22.02.2002	LBORIN	KEDBPX	KEDBP		HIBE	B	4309	8694



E finalmente após agregar as novas funcionalidades ao nosso programa ZTG, temos uma nova versão com mais linhas de código, porém, ainda são menos do que se não utilizássemos a técnica em 3 passos de que trata esse material.

Pergunto novamente, **quanto de código você escreveria para ter o mesmo resultado?**

```
1.  REPORT ztg.

2.  TYPES: BEGIN OF ty_mara.
3.  INCLUDE STRUCTURE mara.
4.  TYPES: cor TYPE lvc_t_scol.
5.  TYPES: END OF ty_mara.

6.  DATA ti_mara TYPE TABLE OF ty_mara.

7.  FIELD-SYMBOLS <fs_mara> LIKE LINE OF ti_mara.

8.  REFRESH ti_mara.
9.  SELECT * UP TO 10 ROWS
10.     INTO CORRESPONDING FIELDS OF TABLE ti_mara
11.     FROM mara.

12.  UNASSIGN <fs_mara>.
13.  LOOP AT ti_mara ASSIGNING <fs_mara> WHERE laeda EQ '20140712'.
14.     PERFORM zf_pinta_celula: USING 'LAEDA' 6 CHANGING <fs_mara>-cor.
15.  ENDLOOP.

16.  PERFORM: zf_criar_alv                USING 'TI_MARA',
17.           zf_mostrar_salv_em_popup USING 02  "p_linha_ini
18.                                           10  "p_coluna_ini
19.                                           12  "p_linha_fim
20.                                           120, "p_coluna_fim.
21.           zf_atribuir_desc_colunas USING 'MATNR'      "p_nome_coluna
22.                                           'Prd.'        "p_desc_curta
23.                                           'Prod.'       "p_desc_media
24.                                           'Produto',    "p_desc_longa
25.           zf_add_cor                USING 'COR',
26.           zf_mostrar_salv            USING abap_true
27.                                           abap_true.

28.  INCLUDE: z_repositorio_openabap_com.
```



## Considerações finais.

Somente para ficar claro, utilizamos nos exemplos “um código” simples, sem tratamento de exceções (famoso TRY, CATCH) e declarações locais, tudo para facilitar a compreensão mas sugiro que faça a declaração agrupada no início do include e que utilize sempre o tratamento de exceções.

A ideia desse e-book é mostrar-lhe que existe sempre uma maneira mais produtiva de fazer as mesmas coisas que fazemos diariamente para nos tornar mais produtivos e assertivos. Pense fora da caixa!

Usamos muito *hardcode* nos exemplos e quero deixar claro que **eu penso**, que você deve sempre utilizar parametrização, ou se não for necessário utilizar constantes

De agora em diante, tente colocar em prática o que vimos e quando estiver confortável, compartilhe seu progresso, facilite o dia de alguém, compartilhe. Acredito que quando você ensina algo novo a recíproca é verdadeira. Seja um replicador e envie suas descobertas para o [Blog](#).

Essa classe foi o primeiro de muitos assuntos que faremos e-books, e espero sinceramente que tenha gostado da leitura, mas, caso seja novo no assunto não deixe de aprender outras maneiras de se construir um ALV pois terá de dar manutenção em muuuitos *reports* antigos.

No momento em que escrevo esse e-book estou desenvolvendo uma série de *posts* para esclarecer como utilizar cada método da classe CL\_SALV\_TABLE e dependendo em que momento do tempo na linha da vida você está, o trabalho já deve estar concluído. Dê uma olhada [aqui](#), é um *post* novo toda semana.

Como já disse anteriormente não sou escritor e gostaria muito de ouvir sua opinião sobre o conteúdo, a maneira que passei a mensagem, organização da ideia e sugestões, envie suas descobertas, dúvidas, sugestões ou críticas para [lnunes@openABAP.com](mailto:lnunes@openABAP.com). Só assim eu **posso melhorar a qualidade deste e dos próximos materiais que estão por vir, fique ligado no blog e confira as novidades.**

Parabéns meu caro *expert*, nos vemos no blog.

Ps. Siga o Instagram do blog [@openABAP](#).

Ctrl+F3.

LNUNES