



Cursos de A a Z feitos pra você!

Apostila Debug para Funcionais

**Por: Alexandre Aparecido
Passarelli**

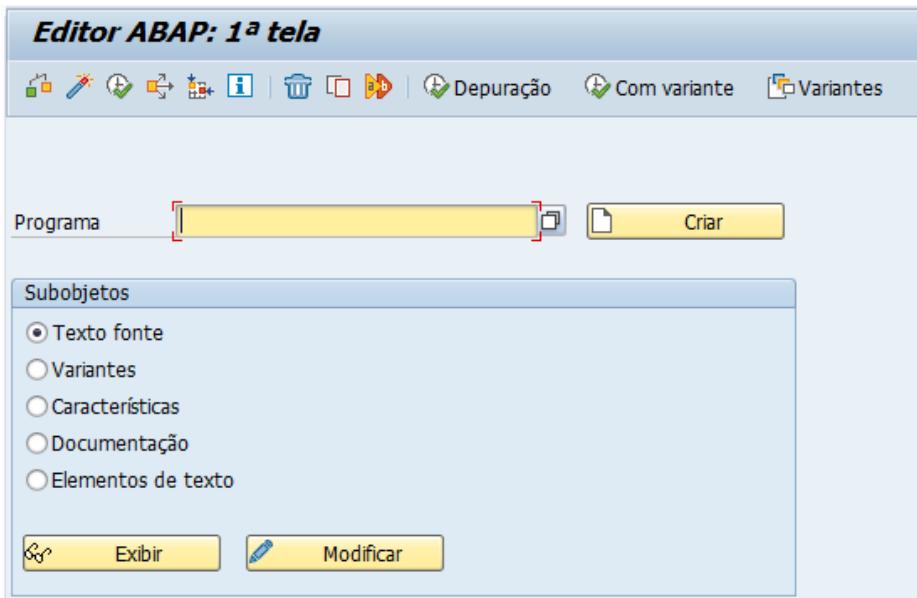
Sumário

1.0 - Introdução ao código ABAP (1ª parte)	3
2.0 - Objetos ABAP	6
● Variáveis	6
● Tela de seleção	7
● Variáveis	7
● Tabelas Internas	8
● Estruturas	8
● Ranges	9
● Constantes	9
● Blocos (Modularização de código)	10
● Condições (IFs)	10
● Operadores	11
● Seleções	11
● Leituras	11
3.0 - Introdução ao Debugger Antigo	13
Ferramentas Básicas	14
● Botão “Campos”:	15
● Botão “Tabela”:	16
● Botão “Ptos.Parada”:	17
● Botão “Chamadas”:	18
● Botão “Síntese”:	19
● Botão “Watchpoint”:	20
4.0 - Introdução ao Debugger Novo	22
Ferramentas Básicas	23
● Abas “Desktop 1, Desktop 2 e Desktop3”	27
● Aba Standard	28
● Aba Estruturas	30
● Aba Tabela	31
● Aba Exibir Detalhadamente	32
● Aba Pontos de parada/Watchpoints	33
5.0 – Dicionário de Dados (SE11)	34
● Tabela de Banco de Dados	35
● Categoria De Dados	36
● - Elemento de Dados	36
● - Domínio	37
● Grupo de Tipos	39
● Ajuda de Pesquisa	40
● Objetos de Bloqueio	42
● Índices	43
6.0 - Dicas e Truques (Debug Novo)	45
● Ponteiros em ABAP	45
● Como criar um BTE	48
● Como preencher mapeamentos de Batch Input de modo mais prático e rápido	49
● Como abrir opções secundárias na transação SE16N	51
● SM50 – Monitoramento de processos	53
● SM04 – Processos abertos, como encerrar	54
● ST05 (Trace ABAP)	56
● SLIN (Inspeção em Códigos)	61
● SCID (Análise do código)	63
● Pular travas de autorização	63
● Via Authority-Check	63
● Via função	64
● Encontrando EXITS	66
● BADI	66
● Enhancement	68
● User-Exit	69
7.0 - LSMW	73
8.0 - Query	73
9.0 - Configurações do SAP Gui	74

1.0 - Introdução ao código ABAP (1ª parte)

Antes de iniciarmos o entendimento de objetos que podemos criar no ABAP, vamos aprender como iniciar um novo desenvolvimento.

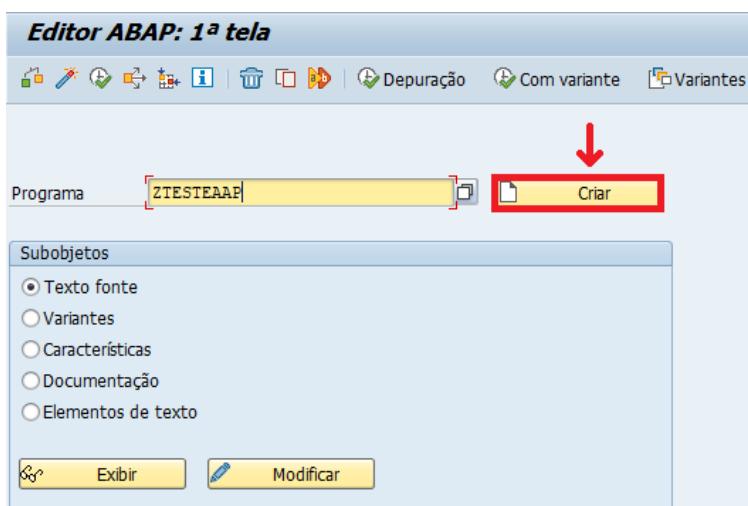
Acesse a transação: **SE38**



Para criar códigos em ABAP é necessário entender que antes os objetos que não são Standard SAP devem ser criados com suas letras iniciais sendo Z ou Y, sendo assim, ao tentarmos criar um programa com qualquer letra que não sejam as mencionadas anteriormente, veremos uma mensagem de que só é possível criar objetos SAP com chave de acesso, ou seja, para alterar objetos Standard é preciso de uma autorização da SAP para fazê-lo.

Informe no campo do Programa o nome do desenvolvimento que deseja criar, vamos usar como exemplo ZTESTE (Iniciais):

(Informe o nome do programa iniciando com Z ou Y, usei como exemplo AAP que são as iniciais do meu nome completo), em seguida clique em criar:





Em seguida, informe as opções descritas abaixo:

The screenshot shows the SAP ABAP dialog for modifying program characteristics. Key fields highlighted with red boxes are: 'Descrição do programa que será criado' (Description of the program to be created), 'Criado' (Created) with date '04.12.2017', 'Status' (Status) set to 'Programa de teste' (Test Program), and 'Tipo' (Type) set to 'Programa executável' (Executable Program). Arrows point from these fields to their respective descriptions: 'Descrição do programa' (Program description), 'Data de criação e usuário que criou' (Creation date and user who created), 'Tipo de programa, para testes use sempre o que está informado' (Type of program, always use what is indicated for tests), and 'Classificação do programa que será criado' (Classification of the program to be created). A 'Gravar' (Save) button is at the bottom left, and a 'Clique para confirmar' (Click to confirm) button is at the bottom right.

Na próxima tela, para não associar o objeto a nenhuma request sempre usamos a opção Objeto Local, pois para programas de teste é a mais recomendada e evita que um programa de teste seja associado a uma request que seguirá para produção:

The screenshot shows the SAP dialog for creating a catalog object entry. The 'Objeto' field contains 'R3TR PROG ZTESTE AAP'. In the 'Atributos' section, the 'Pacote' (Package) field is highlighted with a yellow background. At the bottom, there are two buttons: 'Objeto local' (Object local) and 'Síntese bloqueio' (Synthesis lock). A red arrow points to the 'Objeto local' button, which is highlighted with a red border.



Editor de Código ABAP:

The screenshot shows the SAP ABAP editor interface. The title bar reads "Editor ABAP: Report ZTESTEAAAP modificar". The toolbar includes standard file operations like Open, Save, Print, and Undo/Redo. Below the toolbar, there's a menu bar with "Report" selected. The main area displays the ABAP code for report ZTESTEAAAP. The code is as follows:

```
1 REPORT ztesteaap.
2
3   * -----
4   * Autor: Alexandre Aparecido Passarelli
5   * Empresa: AZ Treinamentos
6   * E-mail: alexandrepassarelli@hotmail.com
7   * Descrição: Demonstração Debug ABAP
8   *
```

Precisaremos sempre lembrar de duas coisas ao codificar em ABAP:

1. Sempre salvar o seu código, clicando no botão (Salvar), isso evita a perda de código caso a conexão sofra alguma instabilidade, pode acontecer em qualquer cliente.
2. Sempre ativar seu código, para ativar basta clicar no botão (Ativar) e confirmar a ativação, um programa só está pronto para ser executado após sua ativação, a ativação já salva automaticamente o código, sendo assim, caso a sintaxe não tenha erros, você poderá sempre ativar seu código e isso já garante seu salvamento.

2.0 - Objetos ABAP

- **Variáveis**

As variáveis são caixas de memória que guardam valores dentro de um programa, elas podem carregar qualquer tipo de valor, dependendo da sua declaração, vejamos um exemplo abaixo:

```
DATA: v_variavel(11) TYPE c.
```

Usando o exemplo da aula, imaginamos uma planilha do Excel, para representarmos mais graficamente o que uma variável é em uma planilha. variável seria um Campo, conforme demonstrado abaixo:

Planilha Excel			
	A	B	C
1			
2			
3			

Este campo pode assumir um tamanho específico ou um tipo específico, mas pode conter apenas uma informação por vez, da mesma forma que só poderíamos ter um valor no campo do Excel.

Abaixo seguem os tipos mais comuns de variáveis que podemos ter no sistema, bem como exemplos de codificação das mesmas:

Tipo	Estrutura	Exceções	Tamanho
C	Aceita todos os caracteres	Nenhuma	Configurado pelo programador
D	Aceita formatos de data	Não aceita sequências numéricas que não sejam datas	Tamanho padrão (8)
T	Aceita formatos de hora	Não aceita sequências numéricas que não sejam horas	Tamanho padrão (8)
N	Aceita apenas números de 0 a 9, seu valor inicial é 0	Não aceita Letras ou símbolos	Configurado pelo programador
I	Aceita números inteiros positivos e negativos	Não aceita valores fracionados	Tamanho String gerado pelo sistema
P	Aceita valores decimais positivos ou negativos	Não aceita letras ou símbolos	Tamanho gerado pelo programador
String	Aceita todos caracteres	Nenhuma	Tamanho Infinito
F	Aceita números exponenciais positivos ou negativos	Não aceita letras ou símbolos	Tamanho gerado pelo sistema

Exemplos no código:

```
DATA: v_teste(10) TYPE c,
      v_teste1     TYPE d,
      v_teste2     TYPE i,
      v_teste3     TYPE f,
      v_teste4     TYPE t,
      v_teste5(10) TYPE p DECIMALS 2,
      v_teste6(10) TYPE n.
```

O comando DATA serve para criar um objeto, ele é utilizado para várias declarações de objetos, a diferença sempre está no meio “TYPE”, que é o que define qual será o objeto a ser criado, o nome v_variavel é definido pelo usuário, assim como seu tamanho e tipo.

O comando DATA com “:” indica que você usará o mesmo comando para criar vários objetos, a outra forma de utilizar o comando DATA é usando ele em cada linha de declaração, para isso, teríamos pontos finais em cada linha e a chamada do comando no começo de cada linha e não vírgula como o exemplo acima

```
DATA v_teste(10) TYPE c.
DATA v_teste1 TYPE d.
DATA v_teste2 TYPE i.
DATA v_teste3 TYPE f.
DATA v_teste4 TYPE t.
DATA v_teste5(10) TYPE p DECIMALS 2.
DATA v_teste6(10) TYPE n.
```

- **Tela de seleção**

Na tela de seleção podemos ter dois tipos de códigos para a declaração de botões, o Parameter e o Select-options.

Parameter: é usado para declarar um botão com seleção única na tela de seleção, podem haver inúmeros parameters, todos são codificados conforme abaixo:

```
PARAMETERS: p_parameter(10) TYPE c.
```

O comando PARAMETERS é fixo, faz referência à chamada da função que cria o botão, já o nome p_parameter é o nome fornecido pelo usuário, acompanhado de seu tamanho e referência, o comando TYPE é o que indica ao que ele irá fazer referência, no caso, pode se fazer referência direta a um campo da tabela, por exemplo, TYPE mara-matnr.

Select-options: Tem a mesma função do parameter, porém permite fazer a seleção múltipla e com condições na tela de seleção, todos são codificados conforme abaixo:

```
SELECT-OPTIONS: s_select FOR mara-matnr.
```

O comando SELECT-OPTIONS é fixo, faz referência à chamada da função que cria o botão, já o nome s_select é fornecido pelo usuário, acompanhado de sua referência “FOR” que indica a qual campo ele será estruturado para a seleção múltipla.

- **Variáveis**

As variáveis são caixas de memória que guardam valores dentro de um programa, elas podem carregar qualquer tipo de valor, dependendo da sua declaração, vejamos um exemplo abaixo:

```
DATA: v_variavel(11) TYPE c.
```

O comando DATA serve para criar um objeto, ele é utilizado para várias declarações de objetos, a diferença sempre está no meio “TYPE”, que é o que define qual será o objeto a ser criado, o nome v_variavel é definido pelo usuário, assim como seu tamanho e tipo.

- **Tabelas Internas**

As tabelas internas são tabelas de memória que guardam valores de uma seleção ou de uma execução no programa, elas contêm uma estrutura parecida com as transações SE16, porém são exibidos apenas os nomes técnicos dentro das mesmas, vejamos um exemplo da codificação de uma tabela interna abaixo:

```
DATA: t_mara TYPE TABLE OF mara.
```

Utilizando o exemplo do Excel, a tabela seria igual a seleção abaixo:

	A	B	C	D	E	F	G	H
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								

Como dito anteriormente, o comando DATA indica a criação de um objeto no programa, o nome “t_mara” é definido pelo usuário para dar nome à tabela, o comando TYPE TABLE OF é o que define que esse objeto é uma tabela referenciada a tabela que se encontra a frente “mara”.

É possível criar uma tabela baseada em uma estrutura dentro do programa, usando o comando TYPES, vejamos um exemplo de como é realizada abaixo:

```
TYPES: BEGIN OF ty_mara,
      matnr TYPE mara-matnr,
      wrkst TYPE mara-wrkst,
    END OF ty_mara.
```

```
DATA: t_mara TYPE TABLE OF ty_mara.
```

- **Estruturas**

Uma estrutura é uma tabela de apenas uma linha, pode ser utilizada para a leitura das linhas de uma tabela interna, ou apenas para o preenchimento de vários campos, pode ser comparada também a variáveis, porém todas dentro de um mesmo objeto, se definirmos objetos no TYPES e em seguida criarmos uma estrutura, teremos um objeto com várias variáveis que podem guardar valores para a execução no programa, vejamos o exemplo de sua codificação abaixo:

```
DATA: e_mara TYPE mara.
```

Utilizando o exemplo do Excel, a tabela seria igual a seleção abaixo:

	A	B	C	D	E	F	G	H
1								
2								
3								
4								
5								
6								
7								
8								

O comando TYPE faz referência à tabela Mara, criando uma estrutura com todos os seus campos, ou seja, todos os campos são como variáveis dentro de um único objeto, fazendo referência a uma tabela.

Também é possível criar uma estrutura baseada em uma tabela criada no programa, seguindo o mesmo exemplo da tabela interna, veja seu código abaixo:

```

TYPES: BEGIN OF ty_mara,
       matnr TYPE mara-matnr,
       wrkst TYPE mara-wrkst,
     END OF ty_mara.

DATA: e_mara TYPE ty_mara.

```

- **Ranges**

Os ranges são intervalos de valores usados dentro de um programa, eles contém uma estrutura parecida com o SELECT-OPTIONS e permitem que você carregue intervalos com diferentes condições dentro de um código, vejamos abaixo um exemplo da codificação de um range:

```
DATA: r_range TYPE RANGE OF mara-matnr.
```

O comando TYPE RANGE OF indica que o objeto será um range baseado em uma referência "mara-matnr".

O range contém valores operacionais, tais como:

EQ, NE, BT, etc., esses valores serão explicados em outro tópico.

- **Constantes**

Uma constante é um valor fixo dentro do programa, porém facilita a edição de valores fixos, diferenciando-a de um hard code, que seria a inserção de um código diretamente em uma variável, tabela, etc.

Por exemplo:

```
CONSTANTS: c_constante(11) TYPE c VALUE 'Hello Word'.
```

Tendo a constante declarada, podemos passá-la quantas vezes precisarmos a vários pontos do programa:

```
v_variavel = c_constante.
```

Quando usamos um hard code, temos que digitar para cada utilização todo o seu valor, também implicando na troca de todos os valores, no caso da constante, bastaria mudar apenas o valor depois do comando “VALUE” e todos os objetos que fazem referência a constante seriam afetados.

```
v_variavel = 'Hello World'.
```

- **Blocos (Modularização de código)**

Sempre que passamos a tela de seleção, definimos o programa em blocos, o que facilita a organização dos códigos dentro de um código fonte, esses blocos são chamados de PERFORMS.

Um PERFORM quando codificado gera um FORM, que é a sua subsequência, a chamada PERFORM pode se fazer referência a diferentes FORMS, já que o código PERFORM é apenas o código que dá função a sequência de blocos, vejamos um exemplo abaixo:

```
PERFORM: f_form1,
          f_form2.

FORM f_form1.
ENDFORM.

FORM f_form2.
ENDFORM.
```

O comando PERFORM está chamando dois FORMS, que estão criados logo abaixo dos mesmos, sempre que o ponteiro do debug passar por uma dessas chamadas, ele cairá dentro de um FORM, e assim que terminar, seguirá a sequência para o programa ou para o próximo FORM.

- **Condições (IFs)**

Um IF é uma pergunta que podemos fazer ao programa em determinada condição, sendo assim, podemos verificar se um comando executou algo com sucesso = “0” ou erro = “4”. Para o SAP todo retorno de sucesso assume o valor de sistema inicial (zero), caso algum erro aconteça o valor assumido sempre é diferente de zero, normalmente é 4. Abaixo vejam alguns exemplos de IF e como utilizá-los.

Exemplos:

```
DATA: v_teste(100) TYPE c.

IF v_teste IS NOT INITIAL.
  MESSAGE: 'Variável está preenchida' TYPE 'S'.
ENDIF.

IF v_teste IS INITIAL.
  MESSAGE: 'Variável está vazia' TYPE 'S'.
ENDIF.
```

• Operadores

Um operador define qual será a condição aplicada a uma lógica, eles podem ser usados de forma simples, usando os operadores tradicionais ou por comandos ABAP, ambos com a mesma função, são eles:

- = ou EQ: - Igual a.
- <> ou NE - Diferente de.
- <, LT - Menor que.
- >, GT - Maior que.
- <=, LE - Menor ou igual a.
- >=, GE - Maior ou igual a.

• Seleções

Quase todos os programas do SAP são criados com base em seleções em tabelas transparentes, dentro de um código, podemos encontrar diversas seleções de diferentes formas, porém vejamos o exemplo e os detalhes de uma seleção simplificada:

```
SELECT *
FROM mara
INTO TABLE t_mara
WHERE matnr EQ v_matnr.
```

O comando SELECT dá forma ao código iniciando uma seleção, você pode informar * para todos os campos ou informar quais os campos que deseja selecionar, o comando FROM indica de qual tabela serão selecionados os campos, o comando INTO TABLE define em qual tabela interna serão aplicados os campos assim como o comando WHERE diz quais serão as condições para a seleção, podem haver mais de uma condição, elas também podem ser diferentes, baseadas nos operadores.

• Leituras

As leituras são realizadas para vermos todos os valores ou um único valor de uma tabela, temos dois tipos principais de leituras, o LOOP e o READ TABLE, vejamos os detalhes de cada um:

LOOP - É utilizado para ler os valores de uma tabela interna, passando por todos os valores baseados ou não em uma condição (WHERE), vejamos abaixo um exemplo do código de um LOOP:

```
LOOP AT t_mara INTO e_mara.
ENDLOOP.
```

Entre o LOOP e ENDLOOP será aplicada a codificação que se faz necessária para toda as linhas da leitura, o comando INTO indica em que estrutura será lida cada linha do LOOP, no caso “e_mara”, o LOOP realizado desta maneira irá ler todas as linhas preenchidas.

READ TABLE - É utilizado para ler uma única linha de uma tabela, dentro ou fora de um LOOP, é extremamente essencial quando temos uma tabela transparente que tem cabeçalho e item, onde lemos o cabeçalho no LOOP e cada item dentro do READ TABLE, vejamos abaixo um exemplo do código de um READ TABLE:

```
READ TABLE t_mara INTO e_mara WITH KEY matnr = v_matnr.
```

Se o valor da condição WITH KEY for encontrado, a estrutura “e_mara” terá a linha selecionada pela condição, se o valor não for atendido, a estrutura ficará vazia.

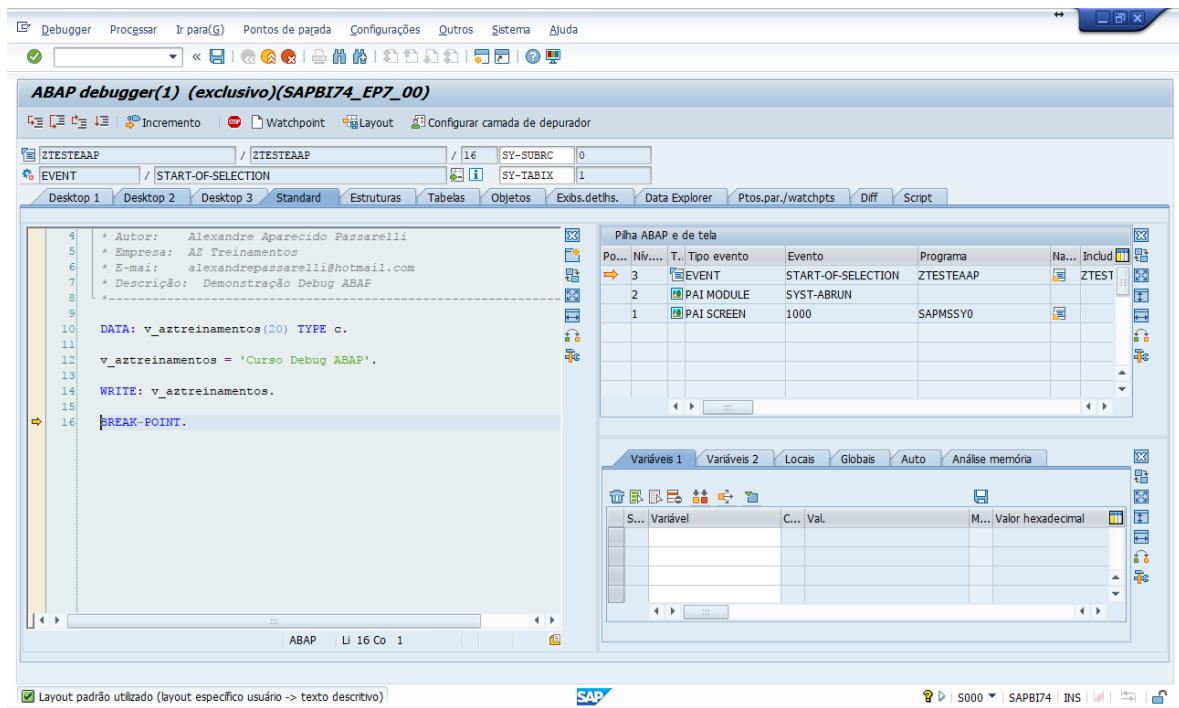


3.0 - Introdução ao Debugger Antigo

O online debugger é uma ferramenta para diagnosticar problemas com o código do programa.

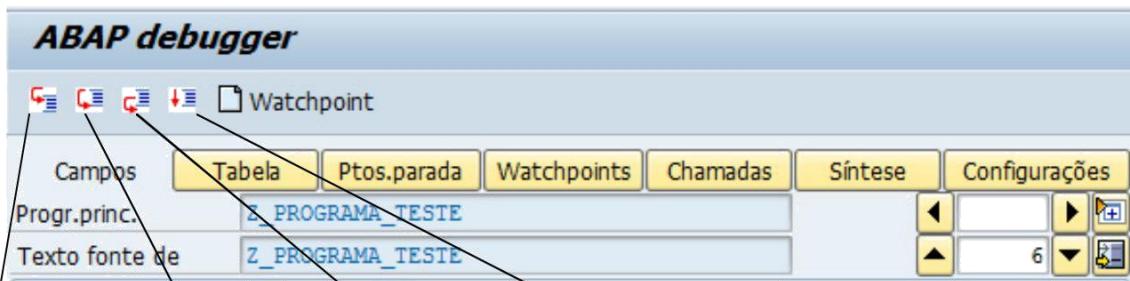
Ele permite a execução passo a passo de cada etapa que o programa executa, permitindo entender como um programa funciona, a encontrar problemas ou até mesmo alterar os dados de execução.

É necessário ter uma leve noção de código ABAP para que se possa entender como o Debug funciona, vamos visualizar este ponto no capítulo 2, a fim de entender o básico para melhor gerenciamento da ferramenta.



Ferramentas Básicas

Vejamos abaixo algumas ferramentas da tela inicial do Debug:



(Avança uma etapa), ou seja, lê a próxima linha do código (F5).

(Executar) faz com que o processo vá para a próxima linha mesmo que a forçando (F6).

(Avançar) executa o processo até que seja encontrada uma nova parada ou o fim da execução (F8).

(Retorno) avança até a saída de um bloco, retornando para o programa principal (F7).

Nome do programa

Nome do Include

Indicador de onde o ponteiro do debugger se encontra.

Move a visualização para os lados.

Número da linha da execução.

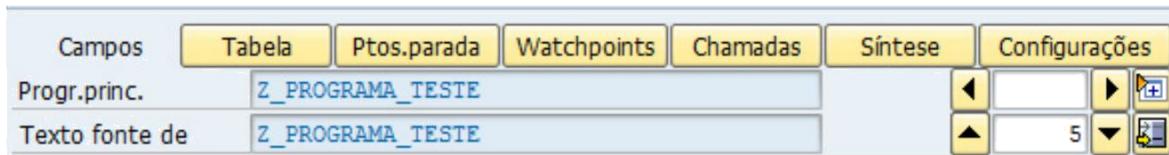
```
Campos Tabela Ptos.parada Watchpoints Chamadas Síntese Configurações
Progr.princ. Z_PROGRAMA_TESTE
Texto fonte de Z_PROGRAMA_TESTE
EVENT START-OF-SELECTION
*-----*
DATA: v_variavel(11) TYPE c.

*-----*
* Passar valores para a variável *
*-----*
*-----*
```

```
v_variavel = 'Hello World'.
```

(Código Fonte) exibe o código ABAP desenvolvido.

O botão abaixo contém todas as outras funções do Debug que vamos ver passo a passo:



- **Botão “Campos”:**

Campos

Contém toda a estrutura dos campos que selecionamos para exibição e modificação durante o Debug, todos os campos estarão sendo exibidos abaixo do código fonte, conforme a tela abaixo:

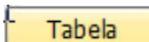
Nomes do campo		1 - 4	Conceúdo campo					
SY-SUBRC	0	1	SY-TABIX	1	SY-DBCNT	0	SY-DYNRR	1000
Valores fixos da tabela Sist, que contém informações da execução do debugger, tal como sy-subrc (retorno de uma função), sy-tabix (linha), dentre outros.								

Annotations pointing to specific elements:

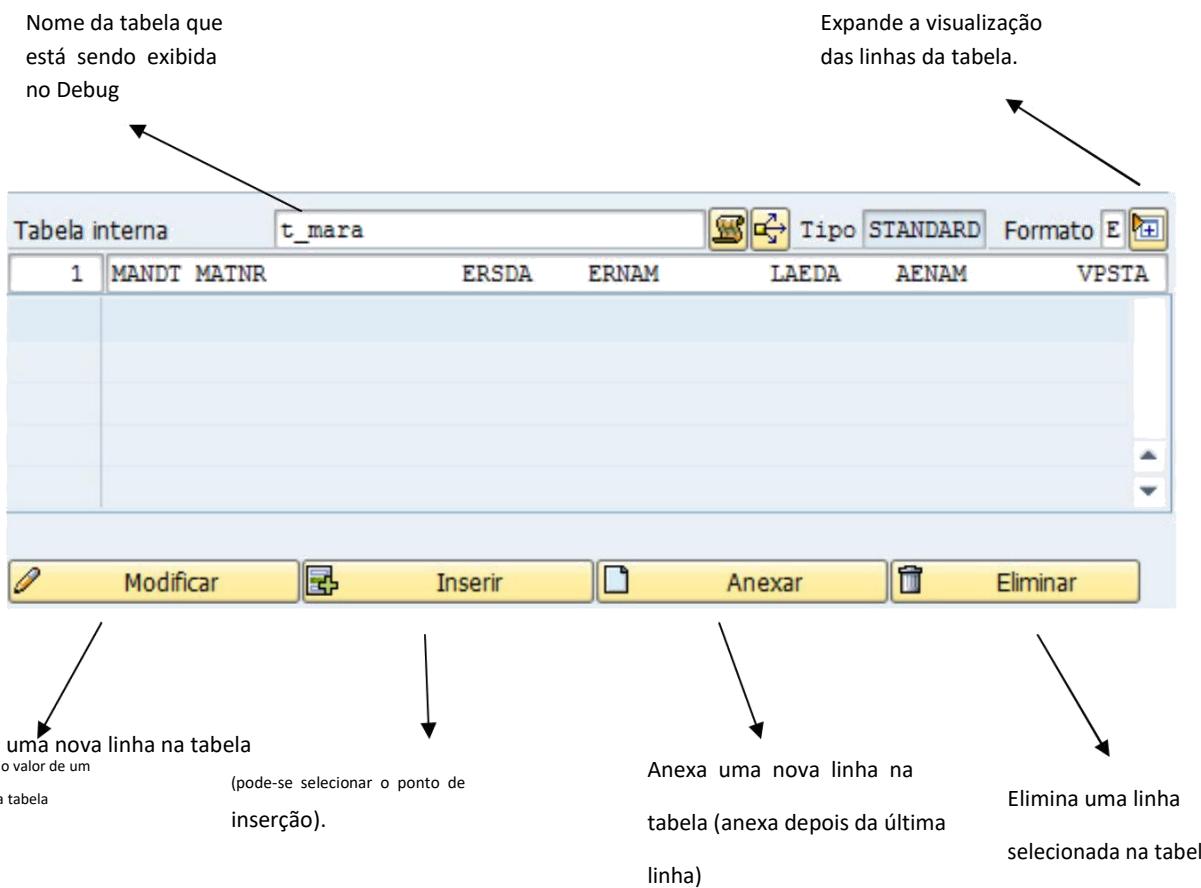
- Left arrow: Limpa todos os campos deixando a tela em branco.
- Middle left arrow: Clique duas vezes ou insira o nome de um campo para ser exibido.
- Middle right arrow: Página para baixo para poderem ser adicionados ou exibidos mais campos.
- Right arrow: Exibe o conteúdo do campo à esquerda ou permite a alterar o valor do mesmo.

Valores fixos da tabela Sist, que contém informações da execução do debugger, tal como sy-subrc (retorno de uma função), sy-tabix (linha), dentre outros.

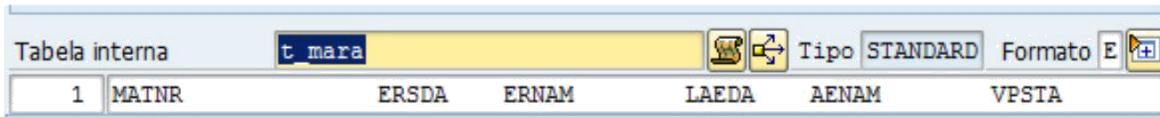
- Botão “Tabela”:



Contém a estrutura para a visualização e modificação dos campos das tabelas internas, nesta opção também é possível inserir e eliminar linhas, os detalhes de cada botão estão no exemplo abaixo:



Veja que os campos exibidos abaixo são editáveis, permitindo assim apagar o valor de um campo da tabela e inserir outro, conforme no exemplo abaixo:



Apague o valor ERSDA e pressione ENTER note que o valor do campo sumiu, porém ele ainda está na tabela, esse controle serve apenas para a exibição dos campos que você deseja, ou a ordem que preferir.

Tabela interna		t_mara								
1	MATNR	ERSDA	ERNAM	LAEDA	AENAM	VPSTA				
1		00000000<		00000000						

Tabela interna		t_mara								
1	MATNR		ERNAM	LAEDA	AENAM	VPSTA				
1				00000000						

- **Botão “Ptos.Parada”:**

Ptos.parada

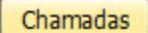
Exibe todos os pontos de paradas que estão ativos no programa, os detalhes estão no exemplo abaixo:

Exibe o ponto de parada, é possível
eliminá-lo dando um duplo clique.

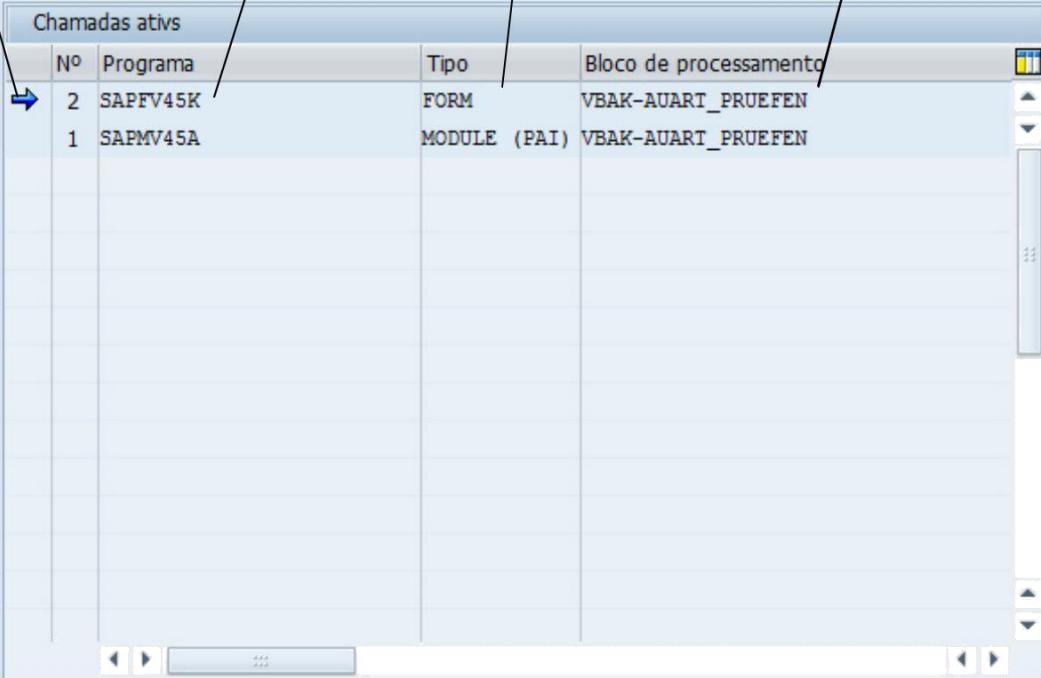
Exibe o nome do programa e a linha ao
qual o ponto de parada está anexado.

Ptos.parada			
Nº	Tipo ponto parada		Em (caminho absol.)
1	Ponto de programa	Z_PROGRAMA_TESTE (18)	
2	Ponto de programa	Z_PROGRAMA_TESTE (20)	
3	Ponto de programa	Z_PROGRAMA_TESTE (22)	
4	Ponto de programa	Z_PROGRAMA_TESTE (24)	
5	Ponto de programa	Z_PROGRAMA_TESTE (28)	
6	Ponto de programa	Z_PROGRAMA_TESTE (26)	
7	Ponto de programa	Z_PROGRAMA_TESTE (30)	
8	Ponto de programa	Z_PROGRAMA_TESTE (32)	

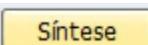
- Botão “Chamadas”:



O botão chamado tem como objetivo indicar o bloco ativo na execução, por exemplo, se o programa entrar em um FORM, esta opção exibirá que o processo está dentro de um Form, ou dentro de dois, indicando em quantos processos o programa se aprofundou, ótimo para quando existem várias chamadas de Perform dentro de um programa, facilitando para que o usuário não se perca de onde o ponteiro está passando, abaixo seguem os detalhes dessa ferramenta:

Indica em qual bloco está o indicador do debug	Nome do programa em que o bloco está ativo	Tipo do bloco, (Form, Include, Module).	Nome do bloco																
 <table border="1"> <thead> <tr> <th colspan="4">Chamadas ativos</th></tr> <tr> <th>Nº</th><th>Programa</th><th>Tipo</th><th>Bloco de processamento</th></tr> </thead> <tbody> <tr> <td>2</td><td>SAPFV45K</td><td>FORM</td><td>VBAK-AUART_PRUEFEN</td></tr> <tr> <td>1</td><td>SAPMV45A</td><td>MODULE (PAI)</td><td>VBAK-AUART_PRUEFEN</td></tr> </tbody> </table>				Chamadas ativos				Nº	Programa	Tipo	Bloco de processamento	2	SAPFV45K	FORM	VBAK-AUART_PRUEFEN	1	SAPMV45A	MODULE (PAI)	VBAK-AUART_PRUEFEN
Chamadas ativos																			
Nº	Programa	Tipo	Bloco de processamento																
2	SAPFV45K	FORM	VBAK-AUART_PRUEFEN																
1	SAPMV45A	MODULE (PAI)	VBAK-AUART_PRUEFEN																

- Botão “Síntese”:



Síntese

O botão síntese tem como objetivo demonstrar por quais programas o indicador do debugger passou, sendo assim, em uma transação standard, por exemplo, você saberá quais são os pontos e programas utilizados até então, facilitando a volta em passos anteriores para recuperar valores, abaixo os detalhes desta ferramenta:

Blocos de processamento do programa básico atual			
Programa	Bloco	Tipo do bloco, (Form, Include, Module).	Nome do bloco
FV45KFAK_VBAK_FUELLEN_TVAK	FORM	VBAK_FUELLEN_TVAK	
FV45KFAK_VBAK_FUELLEN_TVCOM	FORM	VBAK_FUELLEN_TVCOM	
FV45KFAK_VBAK_FUELLEN_TVKO	FORM	VBAK_FUELLEN_TVKO	
FV45KFAK_VBAK_FUELLEN_VBAKKOM	FORM	VBAK_FUELLEN_VBAKKOM	
FV45KFAK_VBAK_FUELLEN_VBKLA	FORM	VBAK_FUELLEN_VBKLA	
FV45KFAK_VBAK_FUELLEN_ZAKAKOM	FORM	VBAK_FUELLEN_ZAKAKOM	
FV45KFAK_VBAK_LOESCHEN	FORM	VBAK_LOESCHEN	
FV45KFAK_VBAK_PARTNER_AENDERN	FORM	VBAK_PARTNER_AENDERN	
FV45KFAK_VBAK_PRUEFEN	FORM	VBAK_PRUEFEN	
FV45KFAK_VBAK_PSTYV_NEUERMITTE	FORM	VBAK_PSTYV_NEUERMITTELN	
FV45KFAK_XVBAK_INITIALISIEREN	FORM	XVBAK_INITIALISIEREN	
FV45KFAK_XVBAK_LOESCHEN	FORM	XVBAK_LOESCHEN	
FV45KFAK_STATUS_VBAK_ANLEGEN_V	FORM	STATUS_VBAK_ANLEGEN_VBK	
FV45KFAK_VBAK_UPDATE_FROM_VEDA	FORM	VBAK_UPDATE_FROM_VEDA	
FV45KFAK_VBAK_FUELLEN_T683C	FORM	VBAK_FUELLEN_T683C	

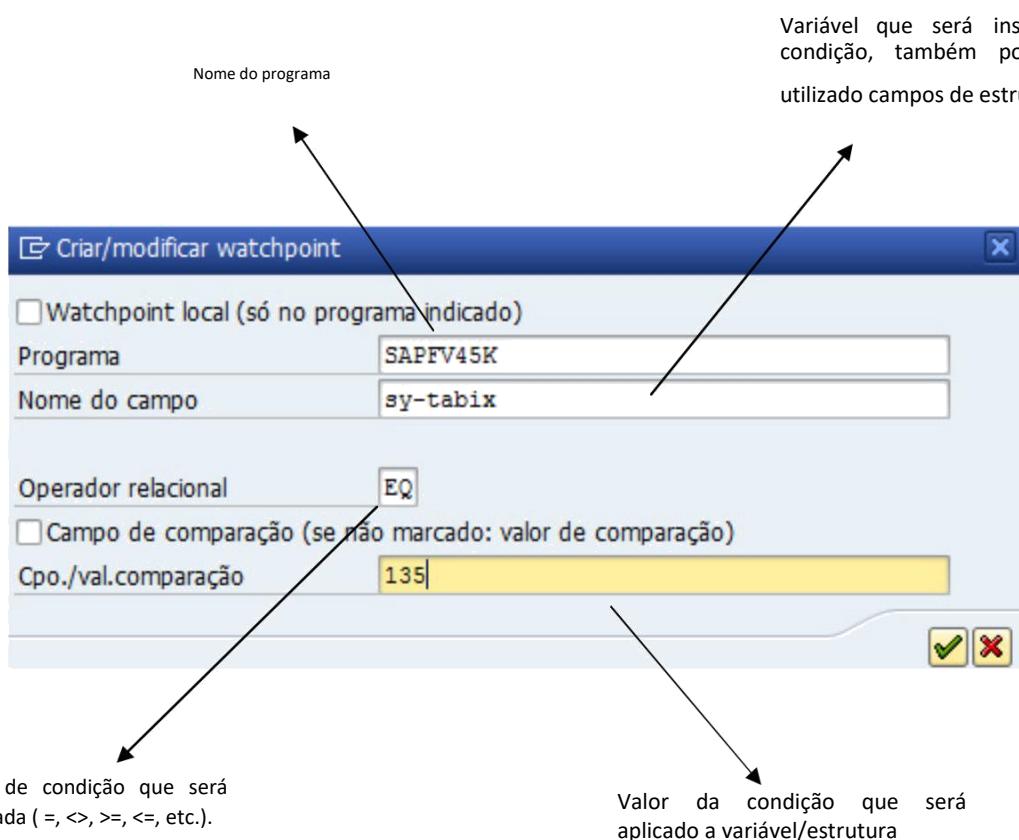
- Botão “Watchpoint”:

Watchpoint

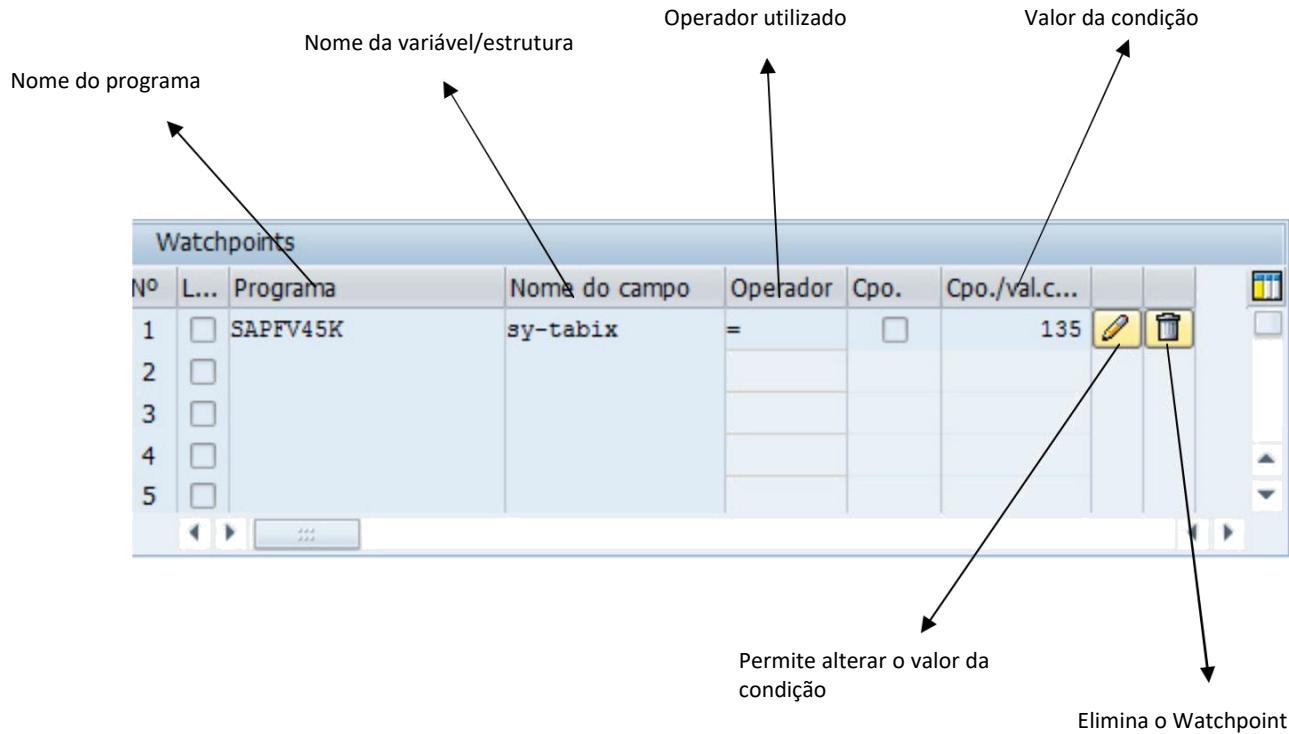
Watchpoints

O Watchpoint tem como objetivo inserir uma condição para um ponto de parada, por exemplo, se você quer que dentro de um Loop o programa pare exatamente na linha 135, esta será a opção que te ajudará, caso contrário você terá que passar linha a linha no Debug, perdendo grande tempo de análise, também é possível criar um ponto de parada baseado no preenchimento de uma variável, ou quando ela fica vazia, esta opção é muito importante para poupar tempo de análise e ser objetivo na condição que você deseja, abaixo seguem os detalhes dessa ferramenta:

Primeiro criamos o Watchpoint com o botão que fica acima da barra de botões >



O botão **Watchpoints** serve para monitorar todos os Watchpoints que foram criados até o momento, podendo eliminar ou editar algum que não se faça mais necessário, veja os detalhes no exemplo abaixo:





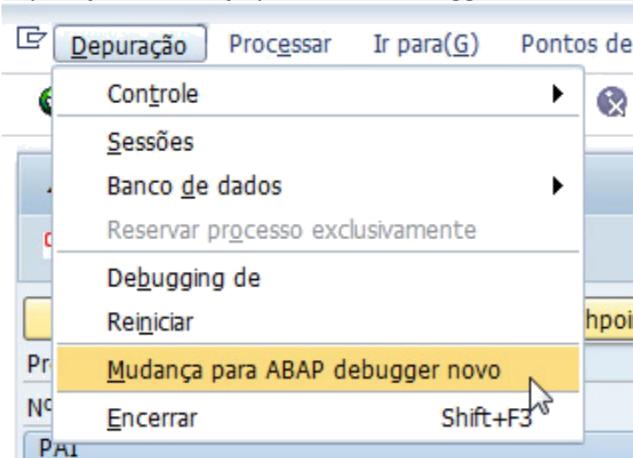
4.0 - Introdução ao Debugger Novo

O online debugger novo é uma ferramenta para diagnosticar problemas com o código do programa.

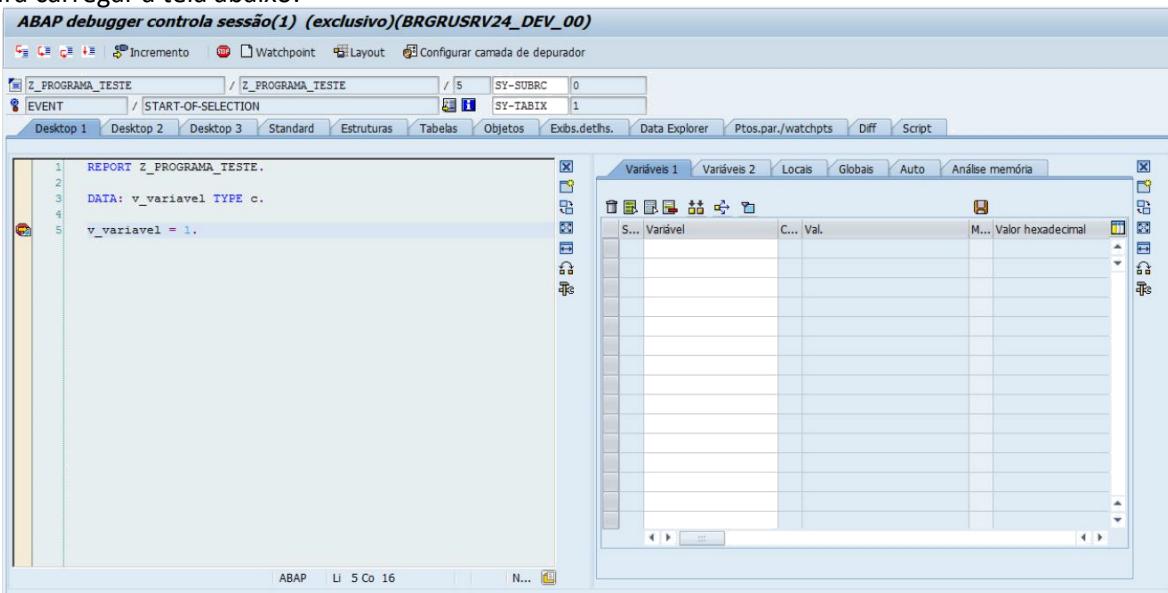
Com algumas opções a mais, ele facilita a depuração do programa, permitindo um ambiente mais visual e mais agradável, também temos funções específicas para este Debug.

Para abrir o Debug novo, basta seguir a imagem abaixo:

Depuração/Mudança para ABAP debugger novo

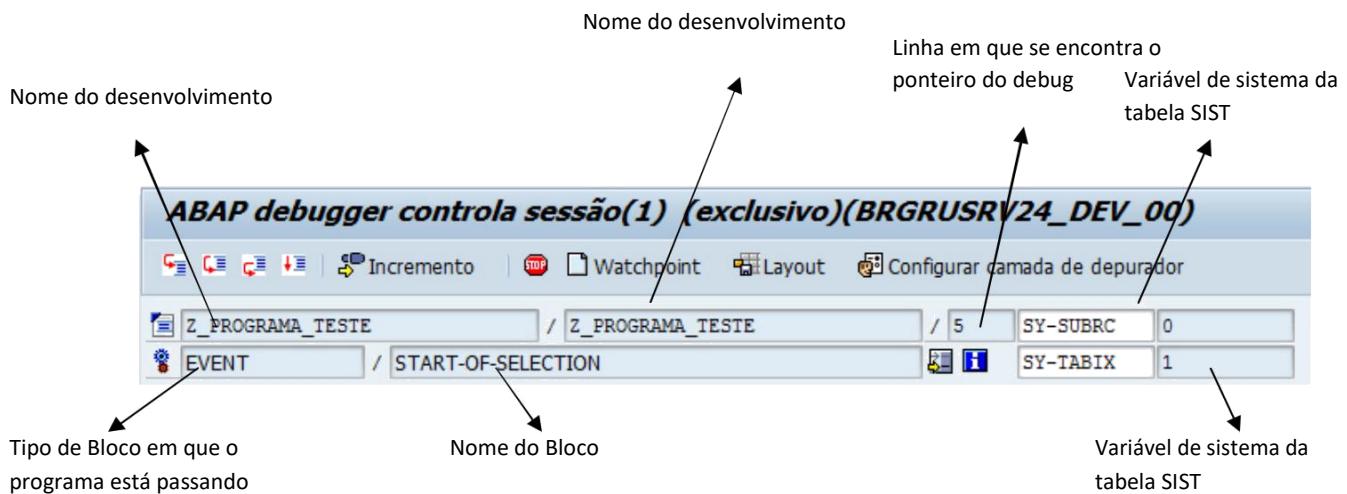


Irá carregar a tela abaixo:

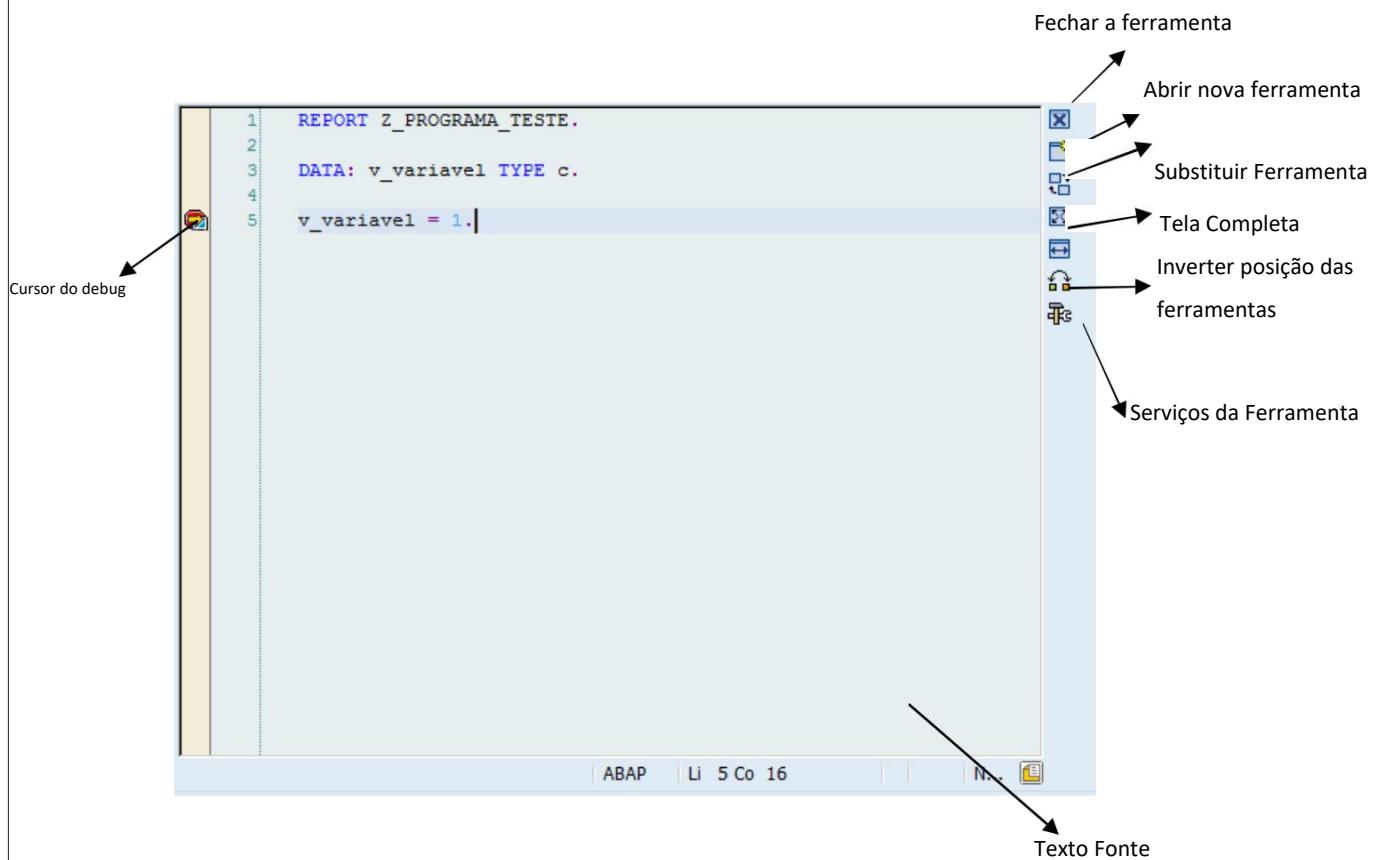


Ferramentas Básicas

Vejamos abaixo, algumas ferramentas do novo Debugger detalhadamente:

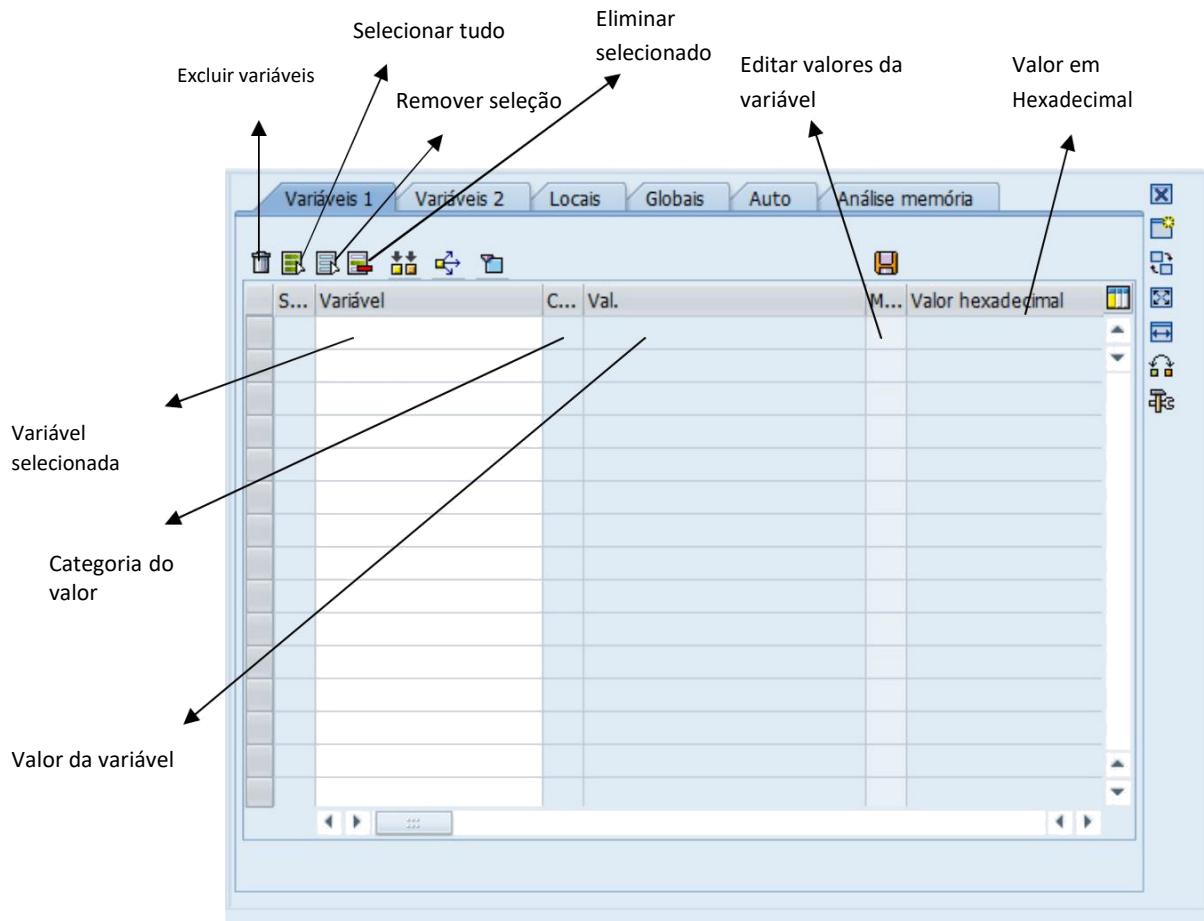


A visualização do texto fonte fica mais parecida com o desenvolvimento ABAP, destacando as palavras chaves para melhor entendimento, é possível também rolar a tela para cima e para baixo com o mouse.



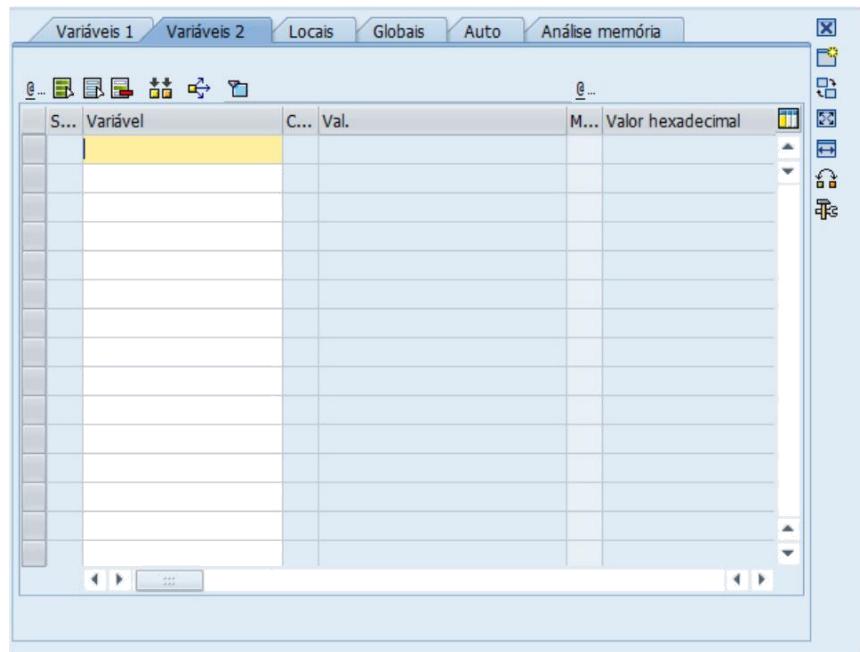


Esta é a tela inicial do Debug, similar a outra primeira tela do Debugger antigo, como ela é uma das telas principais, vamos explorar suas opções para depois darmos continuidade às outras ferramentas do Debug Novo:

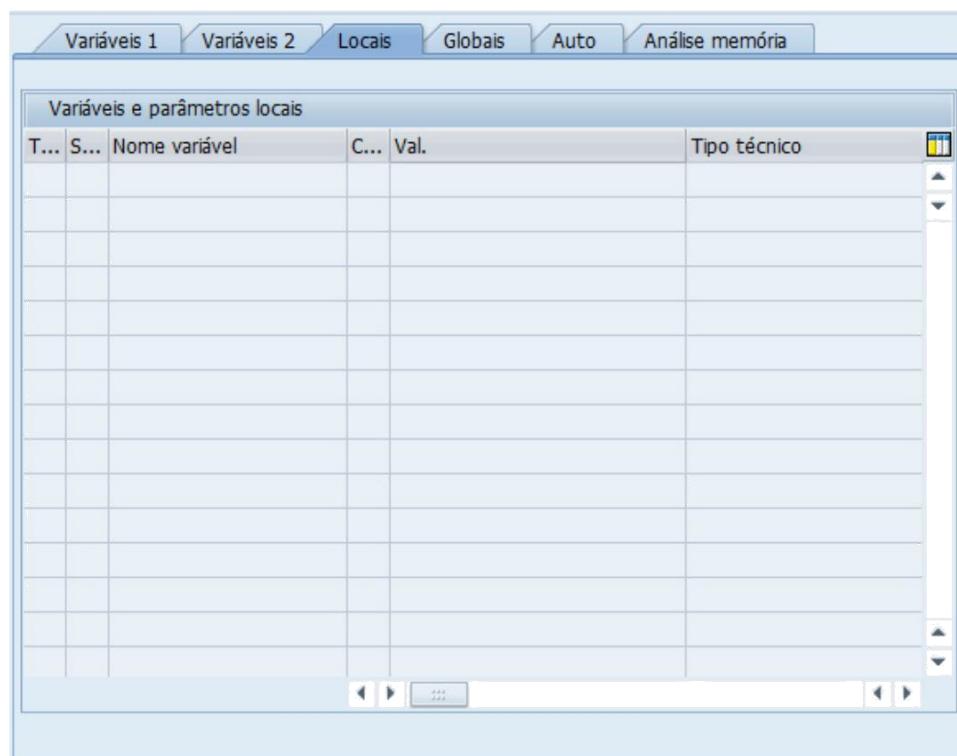




A Aba de variáveis dois permite que você selecione mais variáveis em um segundo plano, contém as mesmas funções da tela anterior:



A Aba Locais, mostra todos os objetos declarados dentro de um bloco, muito útil ao passar por um PERFORM desconhecido e ver quais são os objetos e os valores de seus objetos, pode ser usado principalmente com debug standard, para encontrar valores para Badis, enhancements, exits, etc.





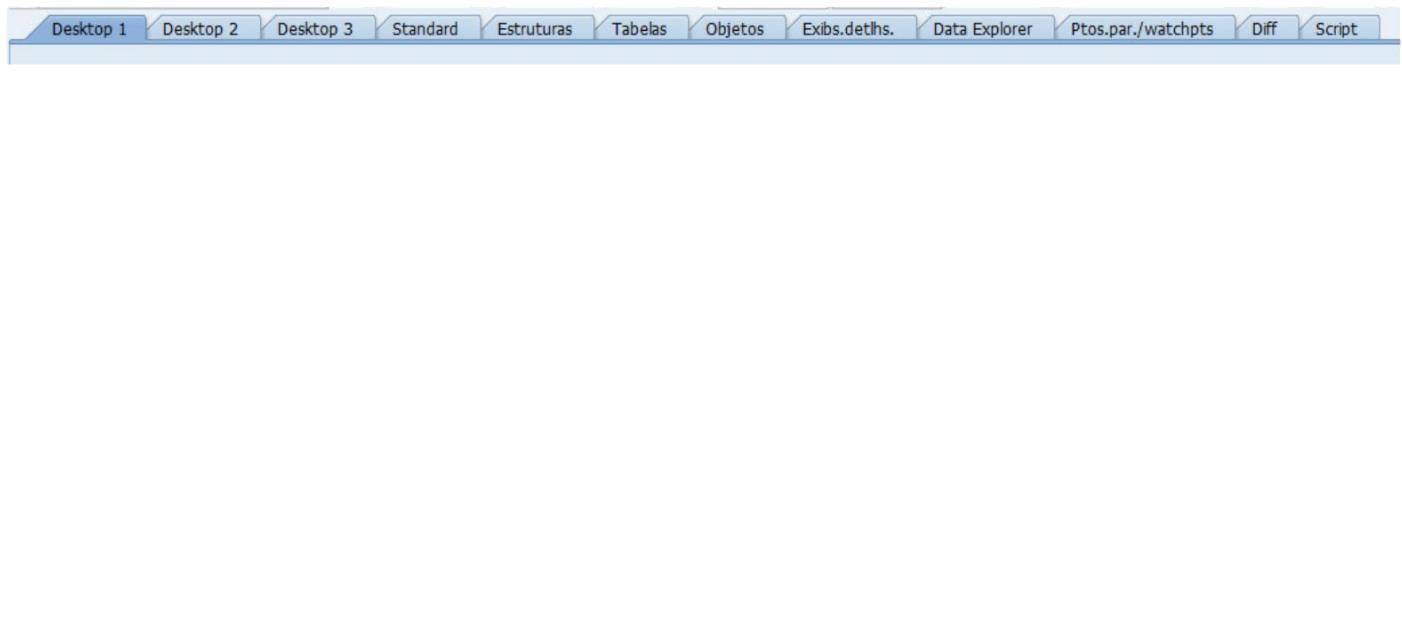
A Aba Globais tem a mesma função da locais, porém não somente para os objetos locais, mas sim para todos os objetos declarados no programa, também muito útil para encontrar exits, badis, etcs.

The screenshot shows the SAP Debugger interface with the 'Variáveis globais' tab selected. The main area displays a table of global variables with columns: T... (Type), S... (Size), Nome variável (Variable Name), C... (Value), Val. (Value), and Tipo técnico (Technical Type). The table lists various variables like CVBAK, CVBRK, VVBRK, etc., each with its technical type listed in the last column. A toolbar with icons for copy, paste, search, and refresh is visible on the right side of the table.

T...	S...	Nome variável	C...	Val.	Tipo técnico
		CVBAK		Structure: flat, not cha...	Structure: flat, not charli...
		CVBRK		Structure: flat, not cha...	Structure: flat, not charli...
		VVBRK		Structure: flat, not cha...	Structure: flat, not charli...
		CVBAP		Structure: flat, not cha...	Structure: flat, not charli...
		CFPLA		Structure: flat, not cha...	Structure: flat, not charli...
		CFPLT		Structure: flat, not cha...	Structure: flat, not charli...
		CVBRP		Structure: flat, not cha...	Structure: flat, not charli...
		VVBRP		Structure: flat, not cha...	Structure: flat, not charli...
		CVBEP		Structure: flat, not cha...	Structure: flat, not charli...
		CVBFA		Structure: flat, not cha...	Structure: flat, not charli...
		CVBKD		Structure: flat, not cha...	Structure: flat, not charli...
		CVBPA		000000	Structure: flat, charlike(5...
		CVBSN		Structure: flat, not cha...	Structure: flat, not charli...
		CVBUP		000000	Structure: flat, charlike(6...
		CVBUK			Structure: flat, charlike(1...

As Abas não mencionadas são desnecessárias para uma visualização básica do Debug, iremos seguir apenas no que se faz mais necessário para o entendimento básico do ABAP Debugger Novo.

A primeira tela que vimos contém algumas ferramentas, porém, temos uma sequência de abas logo acima que também contém novas opções com mais ferramentas, vamos explorar passo a passo cada uma delas:

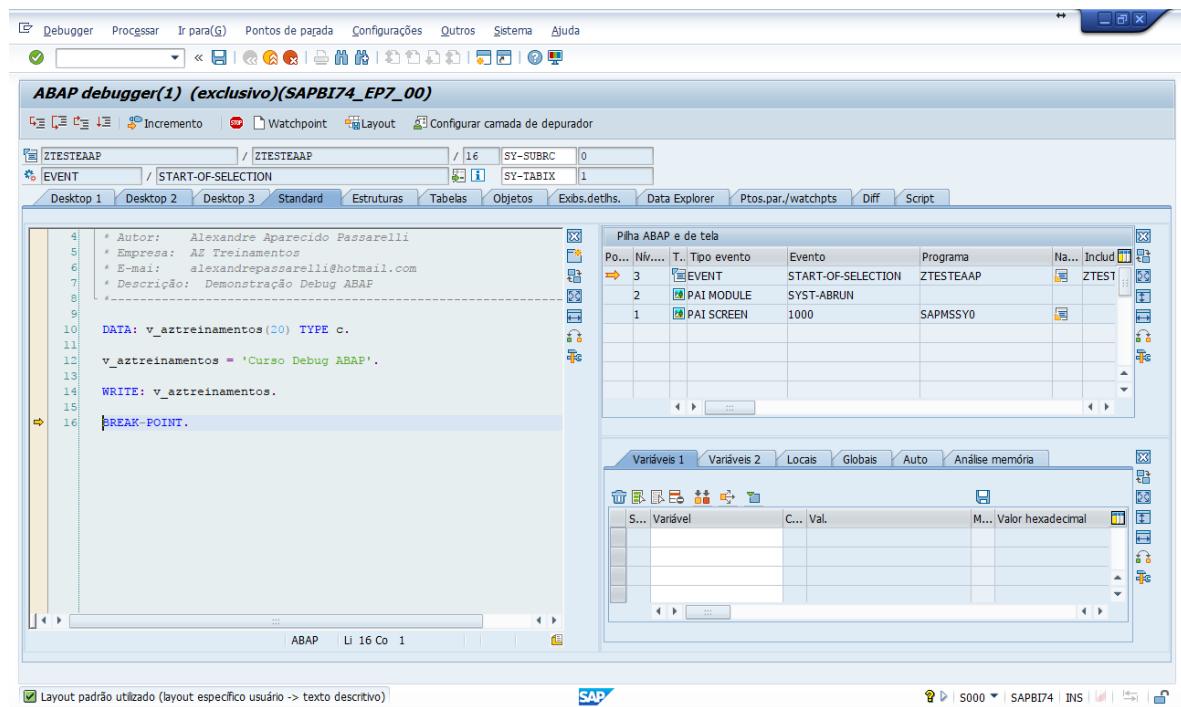




- **Abas “Desktop 1, Desktop 2 e Desktop3”**

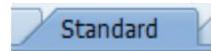
As abas Desktop 1, Desktop 2 e Desktop 3 são editáveis, você pode configurar cada uma das da forma que achar melhor, usando a opção ferramenta nova:

Com essas opções, você pode incluir ou remover novas ferramentas ao debugger, as que já estão visíveis normalmente ao iniciar o Debug Novo são: Texto Fonte e Exibição rápida das variáveis, como exploramos logo acima.

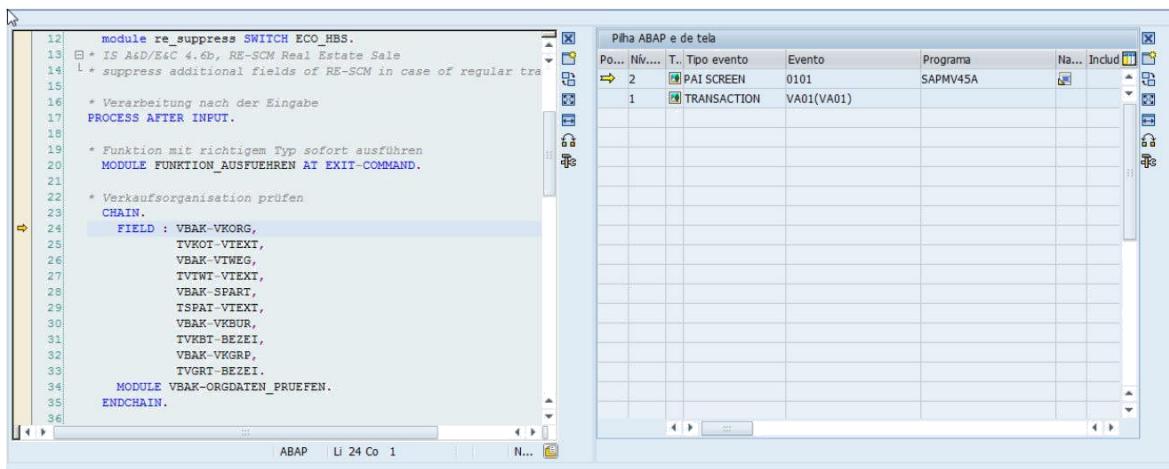




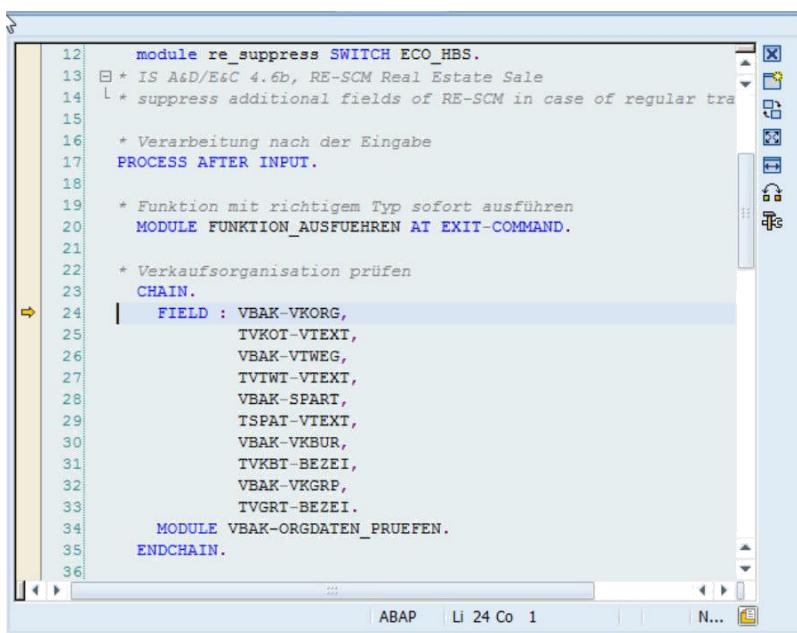
- **Aba Standard**



A Aba standard vem com as ferramentas texto fonte e pilha ABAP e de tela preenchidas, também é possível editá-la, mas ela é muito útil na identificação dos passos por onde o programa passou, muito similar a opção “síntese” do debug antigo.



O editor do texto fonte mostra onde o ponteiro do debug se encontra, caso selecionado um outro momento o ponteiro ficará logo na chamada desse bloco, facilitando a visualização de outros pontos de um programa:



A ferramenta Pilha ABAP e de Tela é um componente default da aba Standard, como dito anteriormente, serve para visualizarmos os pontos em que o programa já passou, permitindo voltar a esses pontos para a exibição dos valores, ou para a análise do código, vejamos esta ferramenta detalhadamente abaixo:

Po...	Nív....	T..	Tipo evento	Evento	Programa	Na...	Inclui
	2		PAI SCREEN	0101			
	1		TRANSACTION	VVA01(VA01)	SAPMV45A		

Indica em que bloco está o ponteiro

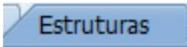
Nível do bloco

Tipos de bloco

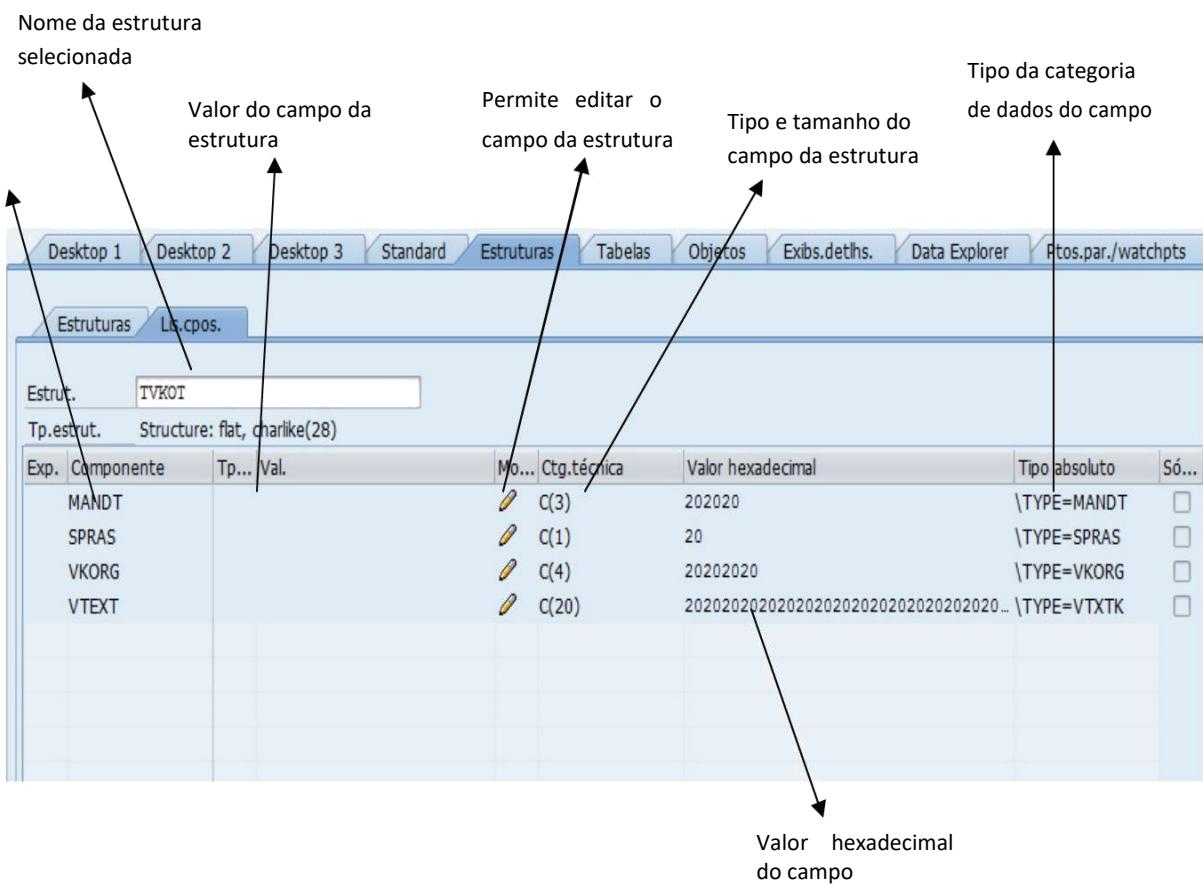
Nome do bloco

Nome do programa

- **Aba Estruturas**



A aba estrutura mostra em uma visualização melhor as estruturas que você selecionar no programa, basta dar um duplo clique na estrutura e seleciona-la posteriormente que automaticamente esta ferramenta será exibida, permitindo realizar algumas operações, conforme veremos abaixo:



- **Aba Tabelas**



A aba tabelas permite realizar operações específicas nas tabelas internas do programa, da mesma forma da opção anterior (Estruturas), porém mais completa, vejamos abaixo essas ferramentas detalhadamente:

Nome da tabela selecionada → Permite inserir campos da tabela → Layout das colunas exibidas → Exibe o cabeçalho da tabela → Permite avançar numericamente as colunas → Controle de visualização

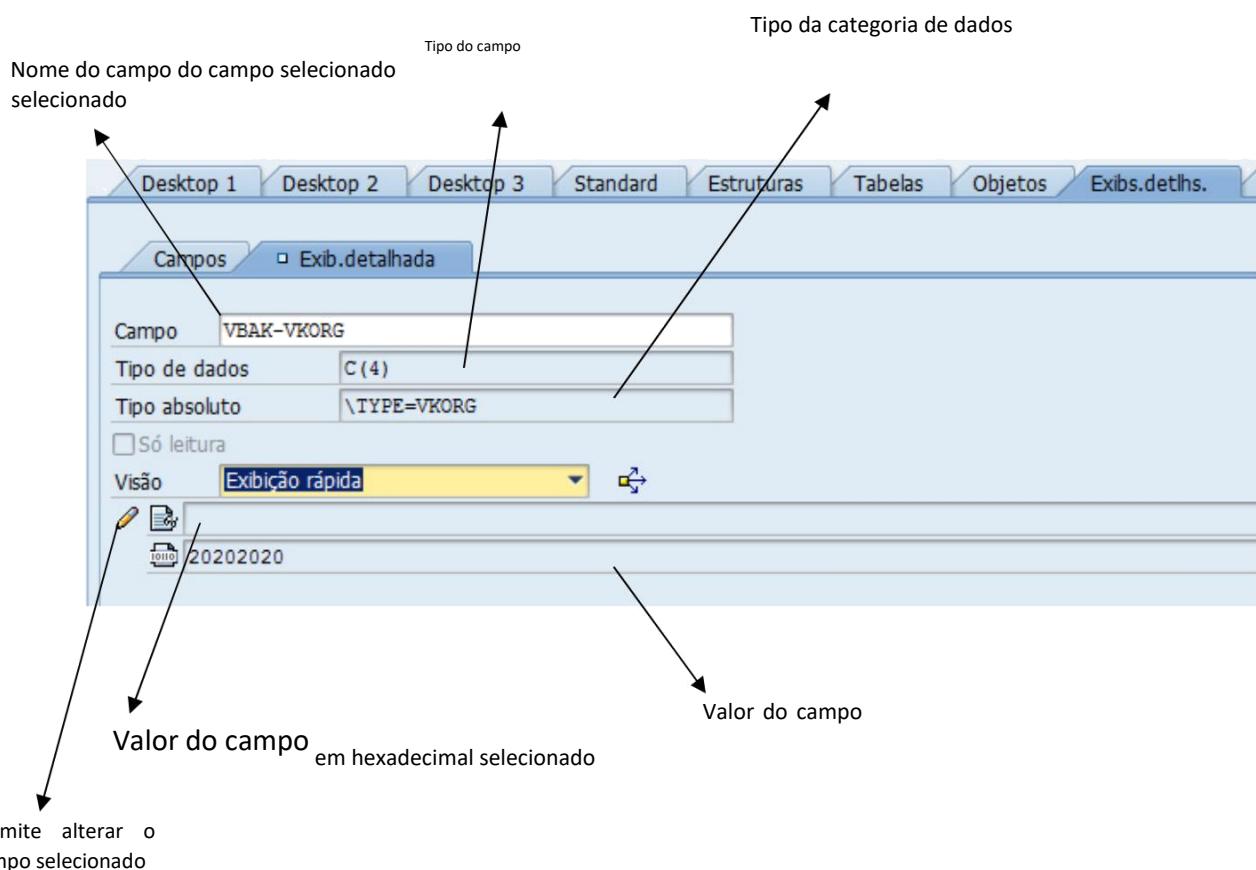
Número da linha → Permite selecionar uma ou mais linhas → Campos da tabela, é possível editar os campos clicando duas vezes sobre o campo desejado.

The screenshot shows the SAP Debugger interface with the 'Tabelas' tab selected. A table named 'CVBRK' is displayed with columns like MANDT, VBELN, FKART, FKTYP, VBTYP, WAERK, VKORG, VTWEG, KALSM, and KNU. Annotations point to various elements: arrows from the top row point to the table name, insertion buttons, column layout, header display, and numeric navigation; an arrow from the left edge points to the row number; an arrow from the bottom right points to the selection handles; and a large arrow at the bottom points to the table body.

- **Aba Exibir Detalhadamente**

Exibs.detlhs.

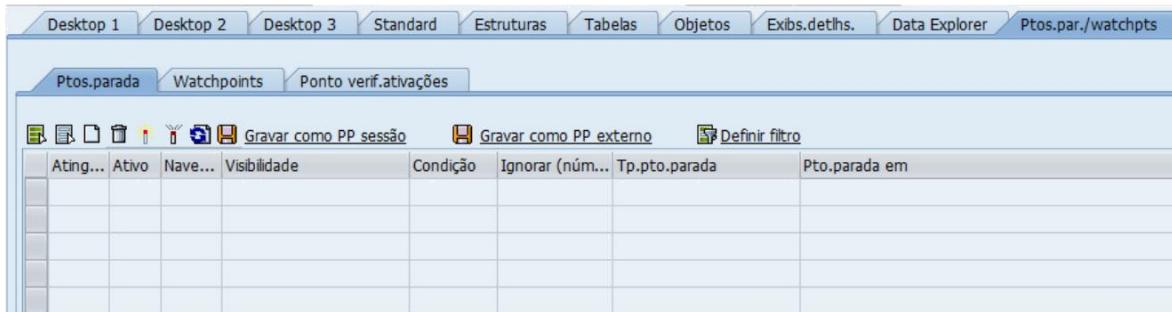
Toda vez que clicarmos duas vezes sobre um campo, essa ferramenta será automaticamente aberta, exibindo detalhes do campo e permitindo sua alteração, ela contém informações detalhadas de cada campo selecionado unicamente, vejamos abaixo algumas opções detalhadas dessa ferramenta.



- **Aba Pontos de parada/Watchpoints**

Ptos.par./watchpts

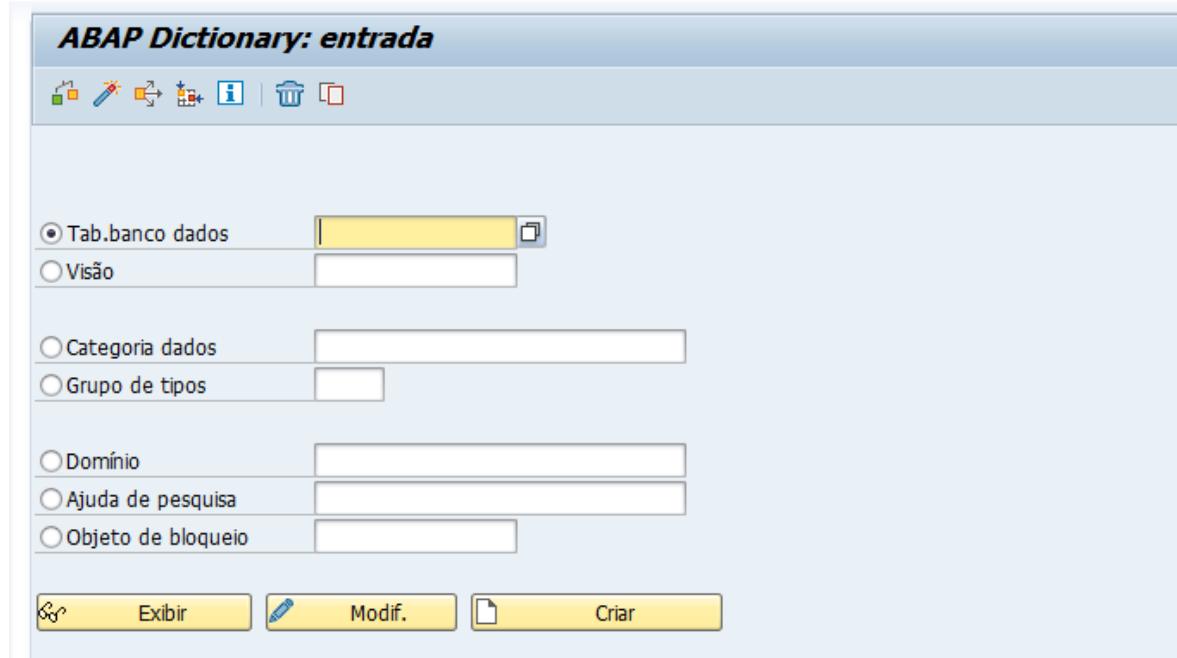
Permite visualizar, alterar, adicionar ou remover pontos de parada e Watchpoints, muito similar a ferramenta do debug antigo:



5.0 – Dicionário de Dados (SE11)

O SAP é uma interface gráfica que tem como base principal o banco de dados, o mesmo antes da versão atual do Hana, funciona com base de dados SQL, como já vimos nos exemplos de código.

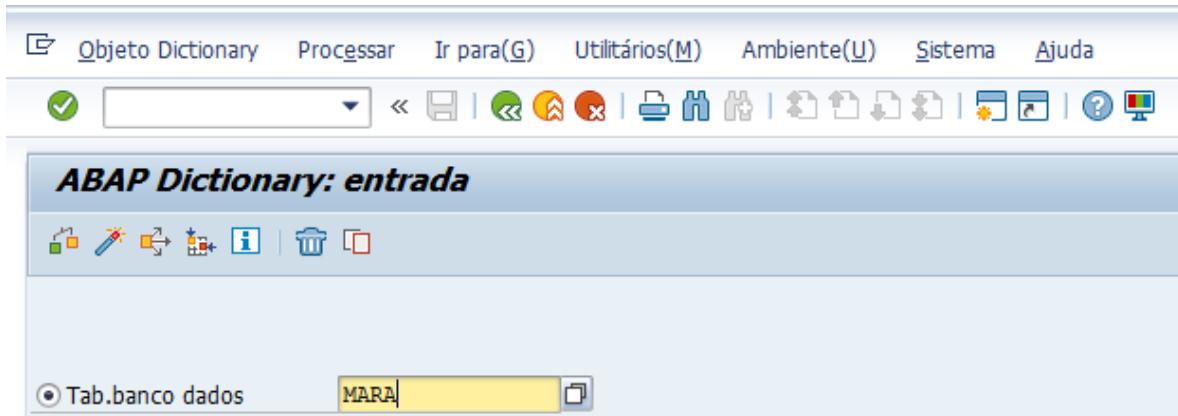
Podemos criar muitos objetos de dicionário de dados no SAP, abaixo veremos os principais e como eles funcionam, este entendimento também é essencial na hora de especificar um documento, pois algumas opções são desconhecidas, causando lógicas menos eficazes.





• Tabela de Banco de Dados

Neste opção, podemos criar, modificar ou deletar uma tabela de banco de dados, que chamamos de tabela transparente, esta tabela é a tabela de banco de dados, esta é a forma que vemos o banco de dados SAP através da interface gráfica, para facilitar a visualização

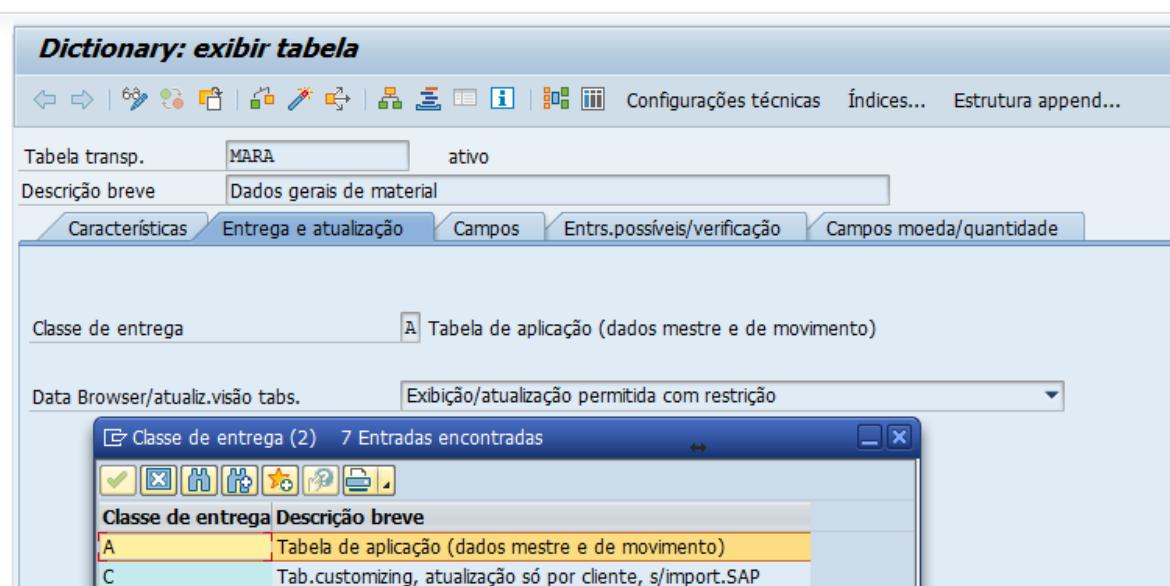


Na ana Entrega e atualização, especificamos o tipo de tabela que vamos criar, ou podemos visualizar seu tipo, as duas mais comuns e mais conhecidas são os tipos:

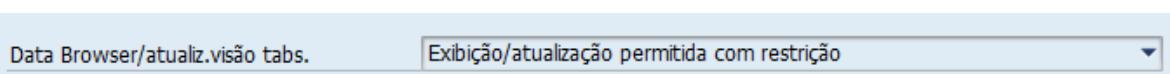
A – Tabela de Aplicação: Normalmente todas as tabelas transparentes são criadas assim

C – Tabela de Customizing: Esta é a tabela que gera request para que seus dados sejam transportados para outro ambiente.

Os demais tipos são pouco usados ou usados em casos muito específicos



Caso ao tentar autalizar o valor de uma tabela você veja a mensagem que a tabela está bloqueada com restrições, para este caso deve ser verificada a opção: **Data Browser/Atualiz. Visão tabs.**, para que seja permitida a alteração, sempre deve estar a opção abaixo, caso contrário, a tabela terá bloqueios para alterações, bloqueios estes referentes a objetos de autorização:





• *Categoria De Dados*

Nesta opção, podemos criar, modificar ou deletar as três opções abaixo:

• - Elemento de Dados

O elemento de dados é objeto que atribui a categoria de um campo da tabela, ou seja, ele diz qual será o nome do campo, qual será o domínio utilizado para indicar o tipo e o tamanho, bem como outras opções mais complexas.

ABAP Dictionary: entrada

This screenshot shows the 'ABAP Dictionary: entrada' interface. It includes a toolbar with icons for search, insert, update, delete, and help. Below the toolbar, there are several radio buttons and input fields:

- Tab.banco dados: An empty input field.
- Visão: An empty input field.
- Categoria dados: A selected radio button with the value 'MATNR' in an input field.
- Grupo de tipos: An empty input field.
- Domínio: An empty input field.
- Ajuda de pesquisa: An empty input field.
- Objeto de bloqueio: An empty input field.

At the bottom are buttons for 'Exibir' (View), 'Modif.' (Modify), and 'Criar' (Create).

Ao exibir um elemento de dados, podemos ver algumas opções abaixo, que indicam seu tamanho e tipo, associadas ao domínio, porém, nesta mesma aba pode ser configurado o tamanho e tipo, independente de estar associado a um domínio, em seguida veremos o conceito de domínio que está diretamente associado ao elemento de dados.

Dictionary: exibir elemento de dados

This screenshot shows the 'Dictionary: exibir elemento de dados' interface for the data element 'MATNR'. It includes a toolbar with various icons and tabs for documentation and additional documentation.

Elemento de dados: MATNR

Descrição breve: Nº do material

Caracts. Ctg.dds. Caracts.adicionais Denomin.campo

Caracts tab selected, showing:

- Categoria elementar: Selected radio button.
- Domínio: Selected radio button with value 'MATNR'. Sub-fields: Ctg.dds. (CHAR), Compr (18). Description: Nº material (campo C18).
- Tipo incorporado: Unselected radio button.
- Tipo de referência: Unselected radio button.
- Referência a tipo instalado: Unselected radio button.



O Elemento de dados também tem a função de atribuir o nome ao campo, quando criamos um relatório ALV, este mesmo pode chamar diretamente o nome do campo que foi definido no elemento de dados, você ainda pode definir 3 tamanhos de visualização para condensar o nome do campo, caso o usuário redimensione a coluna que tem este campo.

Dictionary: exibir elemento de dados

Elemento de dados	MATNR	vo
Descrição breve	Nº do material	
Caracts. Ctg.dds. Caracts.adicionais Denomin.campo		
breve	Compr 10	Denomin.campo Material
Médio	15	Material
longo	20	Material
Título	18	Material

• - Domínio

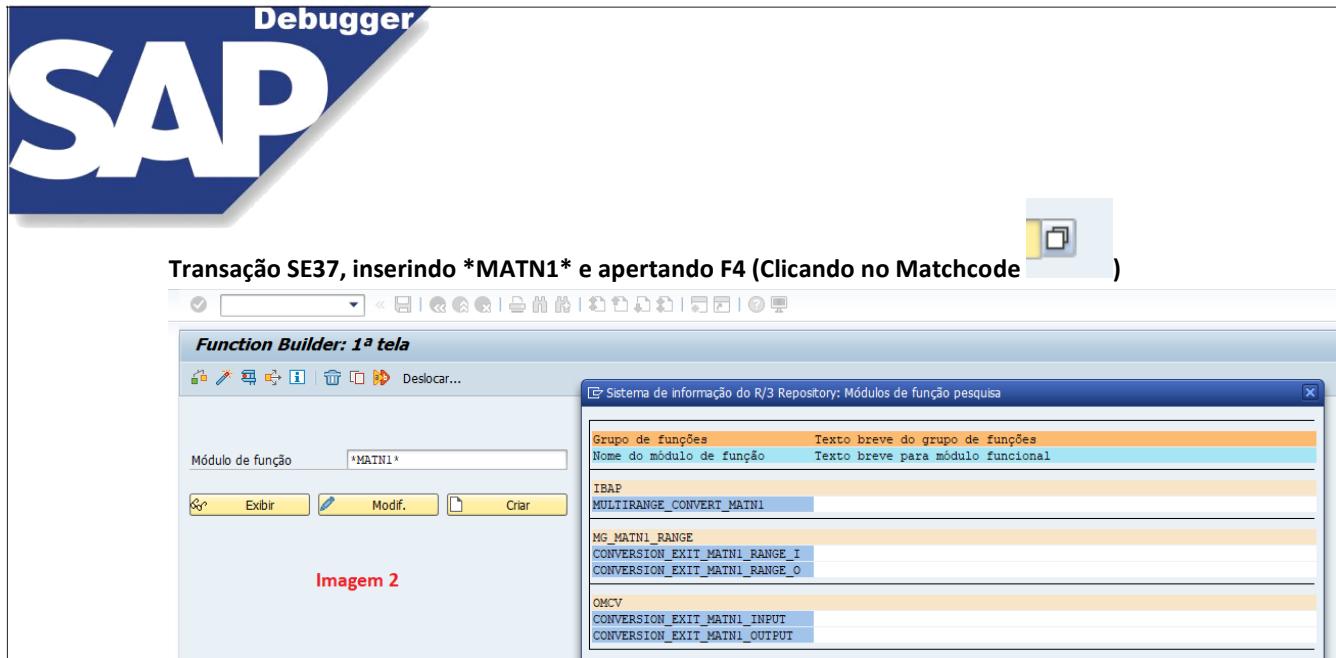
O domínio é o objeto que grava as configurações características de um campo, ou seja, seu tamanho e tipo, da mesma forma que falamos das variáveis no começo da apostila, cada variável possui um tipo e um tamanho que podem ser definidos pelo programador ABAP, o domínio é o objeto físico de banco de dados que guarda a mesma informação, só que estes mesmos podem ser usados apenas dentro de elemento de dados, que por sua vez são associados aos campos de tabela.

É importante saber que normalmente os Elementos de Dados, Domínios e Campos de Tabela tem os mesmos nome, estamos usando o exemplo do campo MATNR, pois o mesmo se encaixa nesta configuração, existem algumas exceções, mas são bem poucas:

Quando se fala em EXIT de Conversão, aquelas que conseguem pegar o valor de um campo e atribuir uma máscara para que o campo seja exibido de outra forma, é no domínio também que configuramos isso, veja abaixo na imagem o campo onde está o valor **MATN1** que indica que existe ativa a **EXIT DE CONVERSÃO** (Imagem 2)

Dictionary: exibir domínio

Domínio	MATNR	ativo
Descrição breve	Nº material (campo C18)	
Caracts. Definição Interv.vals.		
Formato		
Tipo de dados	CHAR	Cadeia de caracteres
Nº de posições	18	
Casas decimais	0	
Características da saída		
Comprim.saída	18	
Rotina conversão	MATN1	
<input type="checkbox"/> Sinal +/-		
<input type="checkbox"/> Letr.minúsculas		



No teste abaixo utilizei a função **CONVERSION_EXIT_MATN1_OUTPUT** que tira os zeros a esquerda do material para exibição, é este conceito que é usado para aplicar uma **EXIT DE CONVERSÃO** a um campo.

Testar módulo de função: tela de resultado					
Teste p/grupo de funções	OMCV				
Módulo de função	CONVERSION_EXIT_MATN1_OUTPUT				
Maiúsculas/minúsculas	<input type="checkbox"/>				
Tmp.exec.: 336 Microsegundos					
<table border="1"> <thead> <tr> <th>Parâmetro importação</th><th>Valor</th></tr> </thead> <tbody> <tr> <td>INPUT</td><td>0000000000000000023</td></tr> </tbody> </table>		Parâmetro importação	Valor	INPUT	0000000000000000023
Parâmetro importação	Valor				
INPUT	0000000000000000023				
<table border="1"> <thead> <tr> <th>Parâms.export.</th><th>Valor</th></tr> </thead> <tbody> <tr> <td>OUTPUT</td><td>23</td></tr> </tbody> </table>		Parâms.export.	Valor	OUTPUT	23
Parâms.export.	Valor				
OUTPUT	23				

- **Grupo de Tipos**

Os grupos de tipos são igualmente aplicados quando precisamos definir a estrutura de uma WORK ÁRREA (Estrutura) explicada na parte de introdução ao código ABAP, porém esta estrutura que chamamos de tipo pode ser usada em qualquer objeto ABAP, quando criamos um Tipo de Estrutura dentro de um programa ABAP, a mesma só pode ser utilizada dentro daquele programa, se a mesma estrutura for necessária em outro programa ABAP, será preciso criar o código ABAP em ambas. Quando definimos um Grupo de Tipos nesta opção, criamos um objeto que pode ser usado em objetos gerais do SAP, um exemplo disso é o ALV que usa o grupo de tipos SLIS:

The screenshot shows the SAP Dictionary interface for displaying a type group. The title bar says "Dictionary: exibir grupo de tipos". The toolbar includes standard icons for copy, paste, search, and print. Below the toolbar, there are two input fields: "Grupo de tipos" containing "SLIS" and "ativo", and "Descrição breve" containing "Globale Typen für generische Listbausteine". There are two tabs at the bottom: "Caracts." and "Txt.fonte", with "Txt.fonte" being active. The main area displays the ABAP code for the SLIS type pool:

```

1 type-pool slis .
2
3 types: slis_list_type(1) type n,
4       slis_char_1(1) type c,
5       slis_text40(40) type c.
6
7 types: slis_tabname(30) type c,
8       slis_fieldname(30) type c,
9       slis_sel_tab_field(60) type c,
10      slis_formname(30) type c,
11      slis_entry(60) type c,
12      slis_edit_mask(60) type c,
13      slis_coldesc(4) type c.
14
15 *Accessibility
16 types: slis_qinfo_alv type alv_s_qinf,
17        slis_t_qinfo_alv type slis_qinfo_alv occurs 0.
18
19  types: begin of slis_filtered_entries,
20    [* index type i,
21    [* end of slis_filtered_entries.
22  types: slis_t_filtered_entries type i occurs 0.
23
24  ---- Structure for additional fieldcat
25  types: begin of slis_add_fieldcat,
26    |   fieldname type tvbe slis fieldname.

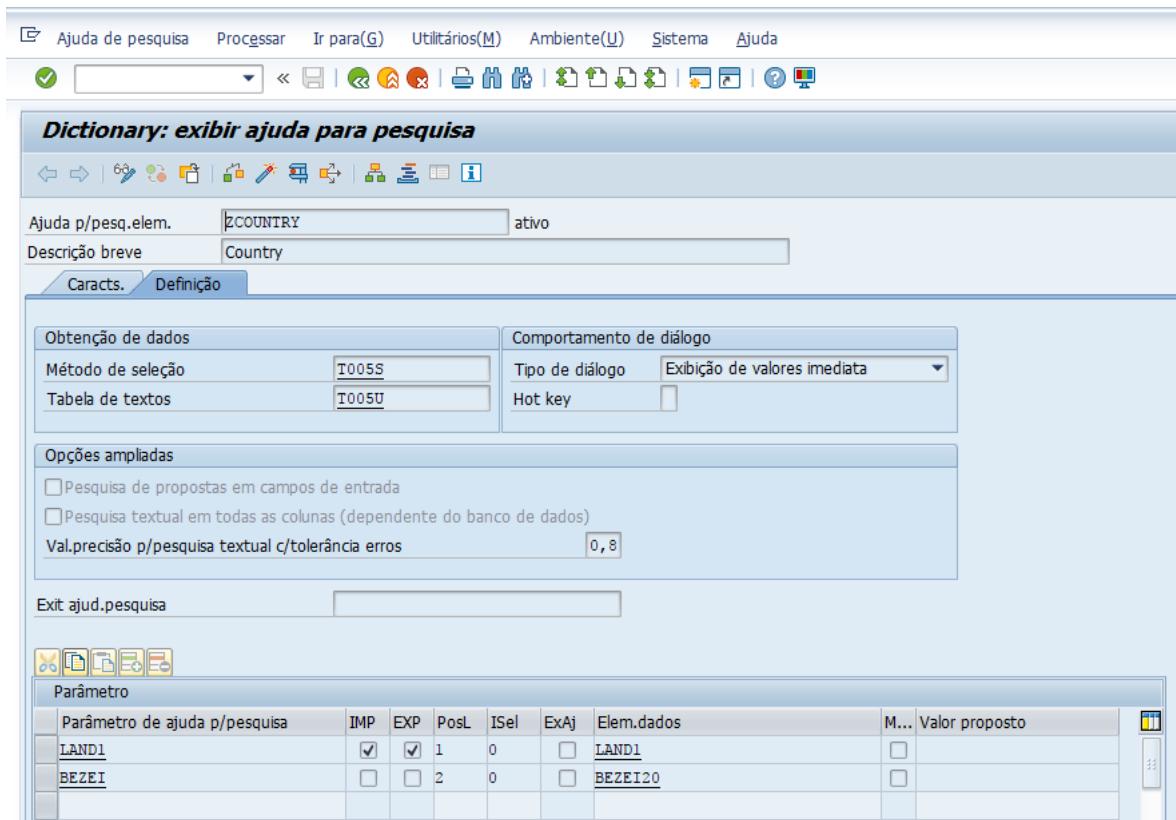
```

Utilizando o grupo de tipos criado na SE11, sempre que um ABAP for codificar e construir um ALV ele utilizará uma declaração em seu programa indicando que definirá objetos do tipo SLIS, assim ele não precisará digitar todo esse código acima dentro do seu programa, é uma forma de reaproveitar o código ABAP que utilizaremos diversas vezes, o mesmo é atribuído basicamente ao conceito de classe.

- **Ajuda de Pesquisa**

Quando falamos em Ajuda de Pesquisas, estamos falando do mesmo que chamamos de Match Code nos termos técnicos de SAP, ou até mesmo Search Help, que é o botão F4 do teclado que pressionamos sempre que precisamos pesquisar algo no SAP.

O SAP contém Ajudas de Pesquisa por todos os lados, na maioria das transações, porém, ao nos depararmos com os cenários do dia a dia vemos que algumas transações ou tabelas não oferecem o objeto de ajuda de pesquisa disponível, o que nos faz não ter a certeza de que o dado que esta sendo inserido naquele campo vá existir naquela seleção, e é por isso que esta opção é importante, é aqui que é possível criar uma ajuda de pesquisa para associar a qualquer campo do SAP:

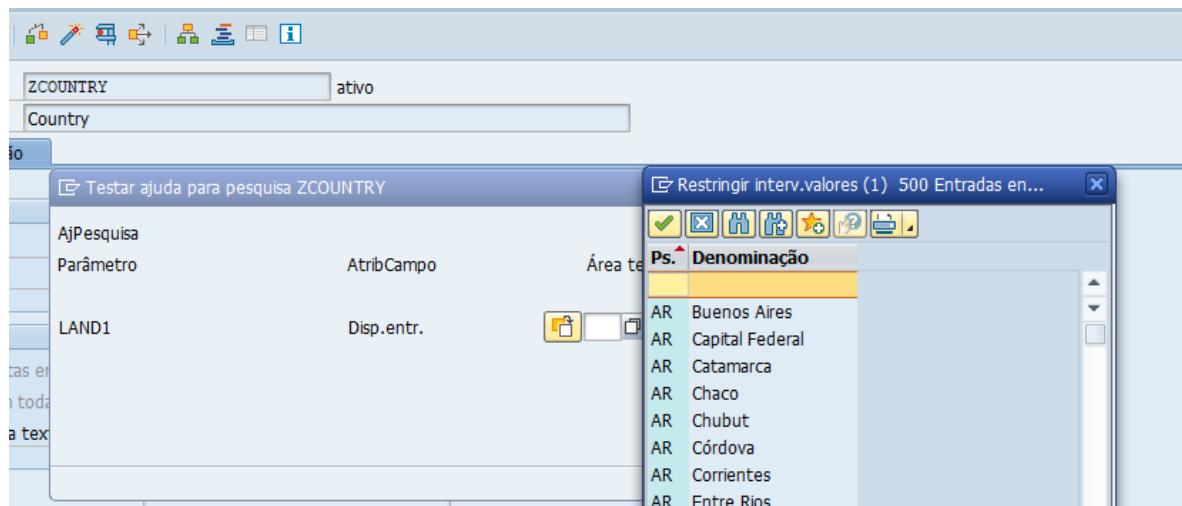


É possível definir a ordem que os campos serão exibidos, bem como qual campo será o selecionado e preencherá o conteúdo do campo do qual foi associado.

Existem formas de codificar dentro de ajudas de pesquisa, para torná-las mais complexas se necessário, no exemplo acima estamos vendo uma ajuda de pesquisa Z, ou seja, criada por algum ABAP, ao executarmos essa ajuda de pesquisa (F8), vemos o resultado abaixo:



O campo selecionado será o AR, o mesmo irá preencher o campo a qual essa ajuda de pesquisa for associada, essa configuração pode ser feito dentro da criação de tabelas ou dentro do código ABAP.



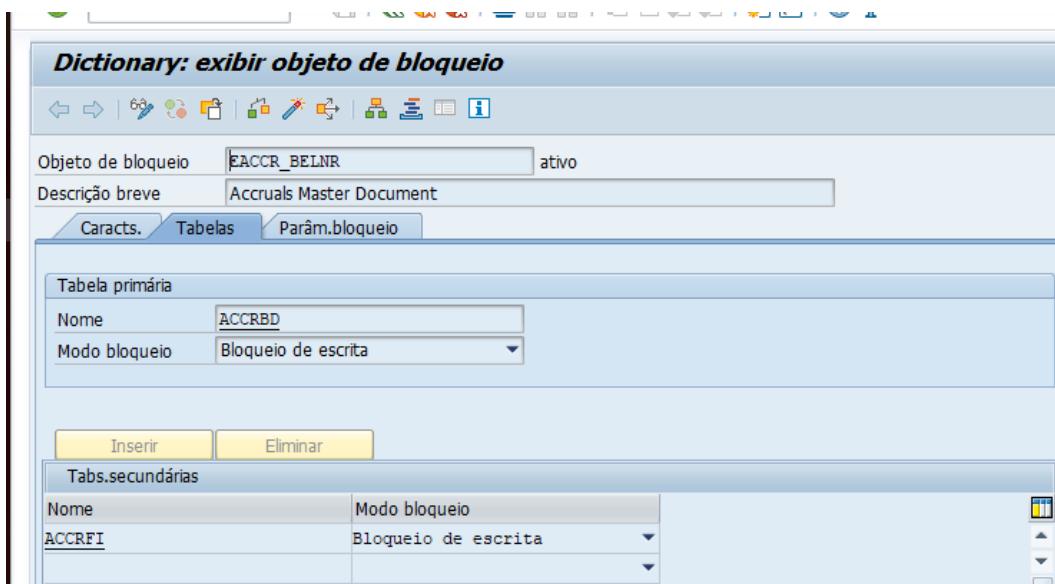
• Objetos de Bloqueio

Ainda que muitos não conheçam, a opção que exibe a mensagem abaixo, pode ser criada por um ABAP e associada em qualquer nova transação criada no SAP, este objeto garante que um usuário irá bloquear o acesso a um documento de vendas (como no exemplo) para que outra pessoa ao entrar no mesmo documento não consiga fazer modificações também.



Objetos de bloqueio são normalmente criados com a letra E no início, mesmo quando são objetos “Zs”, caso o ABAP não conheça esta opção, peça para o mesmo pesquisar **por Enqueue/Dequeue no SCN da SAP**, abaixo segue um link junto da imagem de como funciona o objeto internamente:

<https://archive.sap.com/discussions/thread/710214>



Como podemos ver na imagem acima, é possível escolher a opção da qual será o bloqueio (**escrita/gravação/leitura**)

- Índices

Os índices em ABAP são formas que temos de buscar conteúdos de tabela através de campos que não são chave, ou seja, que causam lentidão nas proezas.

Normalmente temos programas que trabalham com poucos dados e precisamos associar uma seleção a este programa, porém com poucos campos disponíveis as seleções podem causar muita lentidão no processamento e até mesmo causar TIMEOUT de sistema, para isso existe a opção de índices que permite que se crie uma forma mais rápida de buscar dados na tabela com uma configuração específica de seleção.

Transação SE11, informando a tabela e clicando no botão índices

The screenshot shows the SAP Dictionary: exhibir tabela interface for the MARA table. The 'Índices...' button in the toolbar is highlighted with a red arrow. The table view below lists various fields with their properties and descriptions.

Campo	Chv Val... Elemento dados	Tipo de d...	Compr.	Casas...	Descrição breve	Grupo
MANDT	<input checked="" type="checkbox"/> MANDT	CHAR	3		0 Mandante	
MATNR	<input checked="" type="checkbox"/> MATNR	CHAR	18		0 Nº do material	
.INCLUDE	<input type="checkbox"/> MARARA	STRO	0		0 Divisão de dados MARA	
ERSDA	<input type="checkbox"/> ERSDA	DATS	8		0 Data de criação	
ERINAM	<input type="checkbox"/> ERINAM	CHAR	12		0 Nome do responsável que adicionou o objeto	
LAFDA	<input type="checkbox"/> LAFDA	DATS	8		0 Data da última modificação	
AENAM	<input type="checkbox"/> AENAM	CHAR	12		0 Nome do responsável pela modificação do objeto	
VFSTA	<input type="checkbox"/> VFSTA	CHAR	15		0 Status de atualização do material completo	
PSTAT	<input type="checkbox"/> PSTAT_D	CHAR	15		0 Status de atualização	
LVORM	<input type="checkbox"/> LVORMA	CHAR	1		0 Marcar mat.para eliminação a nível de mandante	
MITART	<input type="checkbox"/> MITART	CHAR	4		0 Tipo de material	
MBSRH	<input type="checkbox"/> MBSRH	CHAR	1		0 Setor industrial	
MATKL	<input type="checkbox"/> MATKL	CHAR	9		0 Grupo de mercadorias	
BISMT	<input type="checkbox"/> BISMT	CHAR	18		0 Nº material antigo	
MEINS	<input type="checkbox"/> MEINS	UNIT	3		0 Unidade de medida básica	
BSTME	<input type="checkbox"/> BSTME	UNIT	3		0 Unidade de medida do pedido	
ZEINR	<input type="checkbox"/> ZEINR	CHAR	22		0 Nº documento (sem sistema de administração de documentos)	
ZEIAR	<input type="checkbox"/> ZEIAR	CHAR	3		0 Tipo de documento (s/sistema administração documentos)	

Existem índices Standard que podem ser usados em qualquer programa ABAP e também é possível criar um índice Z com o campo que é necessário da tabela em questão, para esta segunda opção é sempre necessário verificar se sua empresa permite a criação, pois isso pode causar impactos no banco de dados caso não seja feito da forma correta, é sempre bom alinhar com a equipe de Basis da empresa sobre a criação de novos índices, mas caso o índice já exista na tabela e tenha os campos que você precisa é possível usar sem preocupações futuras, a única coisa obrigatória é a utilização de todos os campos, não podemos usar o índice se nossa seleção não tiver todos os campos sendo passados conforme o objeto de índice (Imagem 2)

The screenshot shows the 'Índices de tabela MARA' configuration screen. It lists several existing indices (A, B.., H61, HSS, L) and allows creating new ones (M01, M02, M03, M05). The 'L' index is selected, showing its details: Name: MARA~L, Status: ativ., Author: DDIC, Date: 01.08.2014, and a list of fields: S, IE, SBD, SBD, SBD, SBD, Schalter, Status. The 'Status' column contains values like ISM_MEDIA_BASIS_MO... and Off.

SAP Debugger

Dictionary: exibir índice de extensão

Ident.índice: MARA, M01 desativado, Schalter: ISM_MEDIA_BASIS_MODIF

Descrição breve: IS-M: Hierarchielevel

Última modificação: SAP, 22.06.2005

Status: novo, Gravado

Idioma original: DE Alemão

Pacote: JCOREMOD

Imagen 2

Índice não único (radio button selected)

Índice em todos os sistemas BD (radio button selected)

em sistemas banco de dados selecionados (radio button)

Nenhum índice banco de dados (radio button)

Índice único (índice banco de dados obrigatório) (radio button)

Cpos.índice

Nome campo	Descrição breve	TpD.	Compr
MANDT	ndante	CLNT	3
ISMHIERARCHLEVL			

Exemplo no código ABAP em uma seleção que está sendo aplicado um índice:

```

SELECT b~tabname b~as4local b~as4vers b~sqltab
      a~ddllanguage a~ddtext
      into table gt_join
      from DD02T as a inner join DD02L as b
      on a~tabname = b~tabname
      and a~as4local = b~as4local
      where a~tabname like gv_table
            and a~ddllanguage = 'EN'
%_HINTS ORACLE 'INDEX("T_00" "DD03L~5")'.

```

Exemplo do comando Hints aplicado ao código ABAP que indica que aquele índice indicado entre "" será priorizado nesta seleção.

6.0 - Dicas e Truques (Debug Novo)

Podemos usar as dicas tanto no Debug novo quanto no antigo, muitas pessoas ainda preferem o Debug antigo, embora sua interface não é tão amigável, muitas pessoas por costume sempre mudam para ele, porém, vamos trabalhar apenas com as dicas no Debug novo, visto que a SAP não da mais suporte a versão 4.6 do SAP (Que foi a última a ter o Debug Antigo)

- **Ponteiros em ABAP**

Através dos ponteiros em ABAP, podemos recuperar um valor de um programa mesmo que este já tenha sido executado anteriormente.

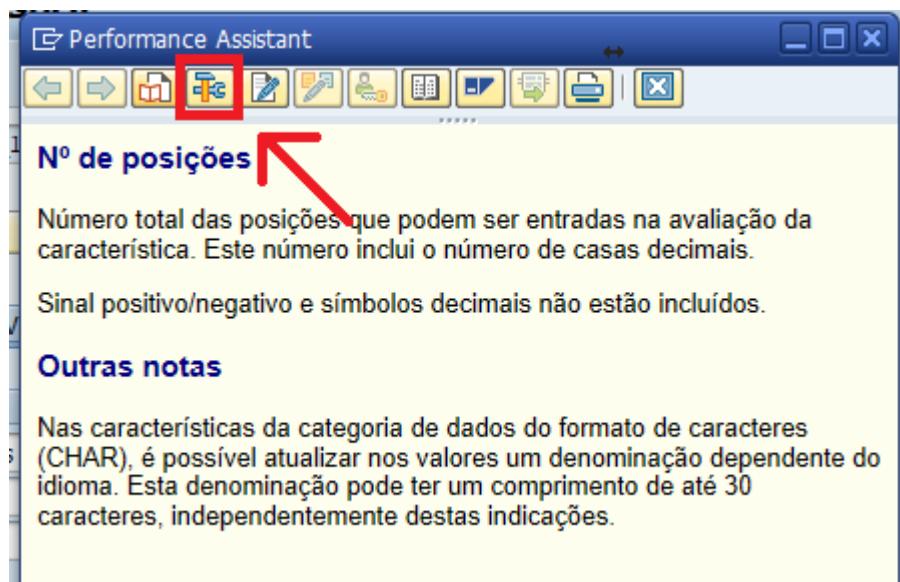
Vejamos um exemplo abaixo para entender melhor.

Transação CT04, informando uma característica:

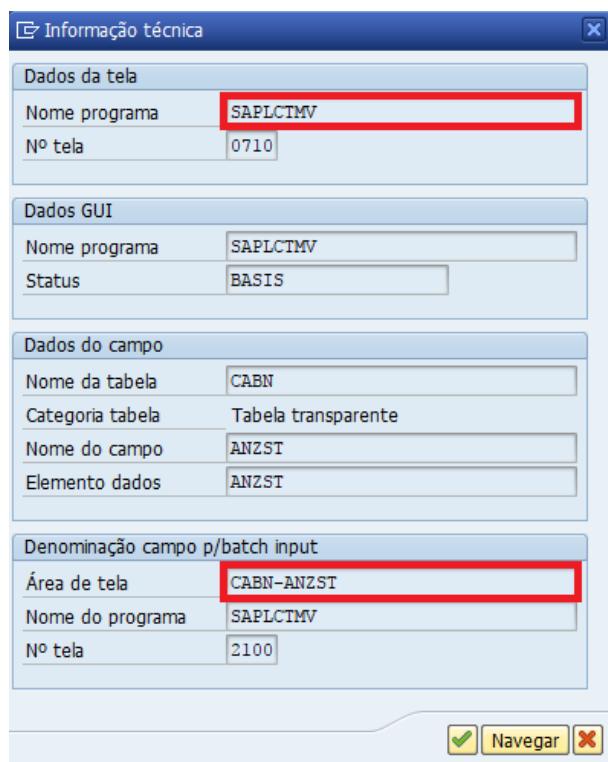
Nesta tela temos focado o campo Nº de posições, e em um exemplo prático precisamos deste valor em outra tela desta transação, mas para ter certeza de que este campo pode ser usado em outras abas, temos primeiramente que verificar seu ponteiro, para isso, vamos utilizar a configuração técnica para pegar as informações necessárias para verificação de ponteiro:



Clicar no campo Nº de posições e em seguida pressionar F1 no teclado e clicar em Configurações Técnicas, conforme a imagem abaixo:



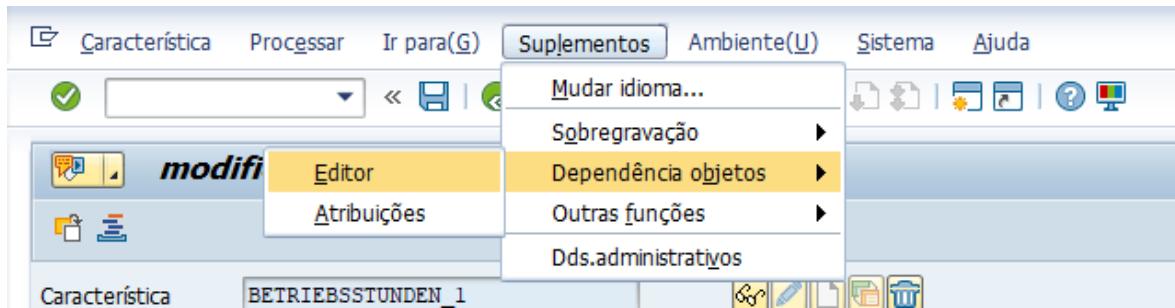
Vamos guardar as informações abaixo, antes de verificarmos se o campo está disponível em outra aba ou parte do programa:



SAPLCTMV - CABN-ANZST



Após guardarmos as informações, vamos para a próxima aba ou parte do programa, é importante que seja do mesmo processo para isto funcionar:



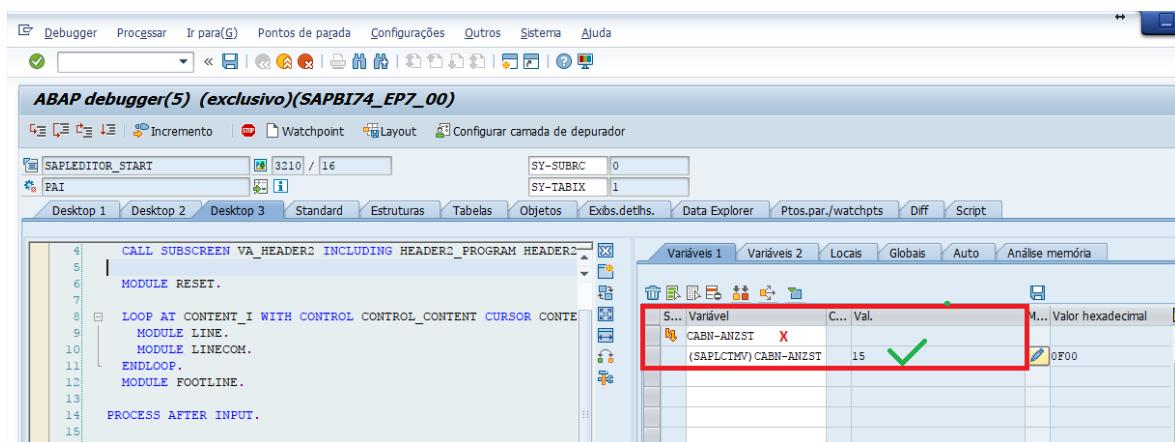
Ao entrar na tela abaixo, vamos abrir o Debug (/H no TCODE conforme a imagem abaixo) pressionar ENTER no teclado.



Conforme demonstrado na imagem abaixo, ao informar na aba de variáveis do debug apenas o nome do campo que precisamos “CABN-ANZST”, o Debug demonstra que este campo não está disponível.

Porém, ao informarmos “(SAPLCTMV)CABN-ANZST” o valor 15 foi exibido e assim sabemos que é possível recuperar este campo em outra tela via código ABAP (normalmente via EXIT), o ABAP utilizará esta opção via FIELD SYMBOL ABAP, abaixo segue um exemplo de como ficaria no código ABAP este mesmo exemplo:

```
ASSIGN ('(SAPLCTMV)CABN-ANZST') TO <fs_anzst>.
```





- **Como criar um BTE**

(Texto retirado do site)

BTE - Business Transaction Events, lembra um pouco o famoso enhancements. Muito comum o seu uso no módulo FI, representam componentes de vendas e de distribuição a receber e a pagar. As BTE's não são utilizadas apenas pela SAP, mas também por clientes e terceiros. A BTE tem uma interface pré-definida e permite anexar funcionalidade adicional na forma de um Function Module.

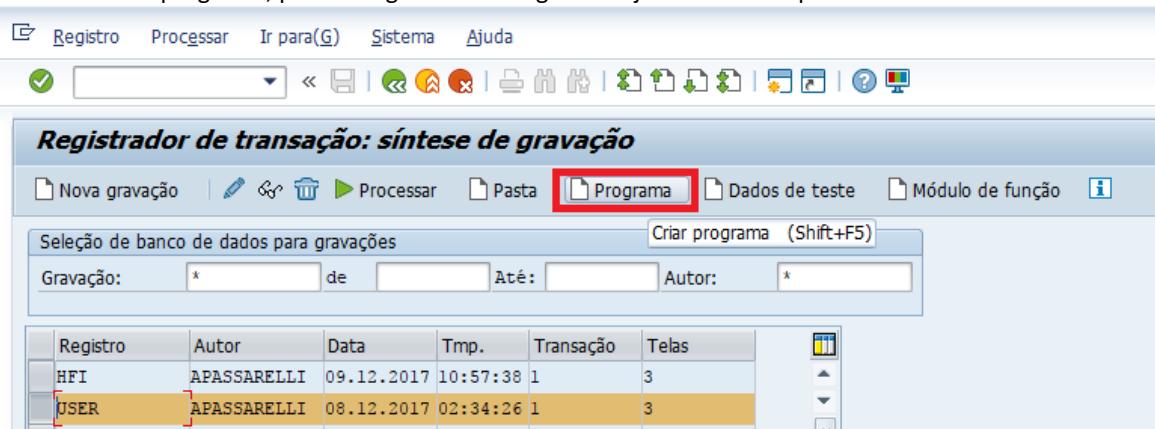
Existe um site muito legal, fazendo um trabalho muito bacana explicando passo a passo cada opção de várias coisas no SAP, mesmo não conhecendo o autor pessoalmente já vi que é uma pessoa muito bacana que gosta de passar seu conhecimento adiante, por isso, vou deixar o link do próprio site explicando como se criar uma BTE:

<http://abapfox.blogspot.com.br/2016/08/conhecendo-o-modulo-de-funcao-bte.html>

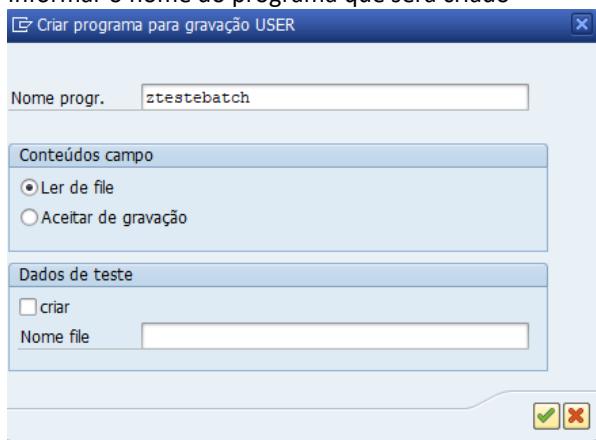
- Como preencher mapeamentos de Batch Input de modo mais prático e rápido

Neste tópico não explicaremos como um Batch Input é criado, porém a maioria dos funcionais sabe criar um mapeamento de Batch Input através da transação SHDB, porém, muitos nunca viram que é possível mapear um passo e já criar um código ABAP que cria automaticamente aquele código, facilitando na hora de passar para o ABAP o mapeamento para que o programa seja criado, a única obrigatoriedade para esta dica é ter um usuário ABAP que permita a criação de programa ABAP, caso seja no QA, tem que ser permitido também a criação de programas no ambiente, pois é necessário gerar um programa ABAP para pegar o código, conforme abaixo:

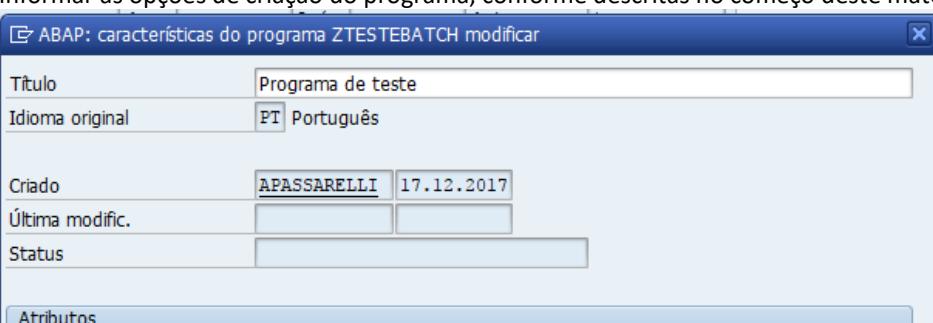
O mapeamento abaixo tira uma cópia de um usuário ABAP e cria outro, inserindo uma senha padrão, ao clicar no botão programa, podemos gerar um código ABAP já com este mapeamento:



Informar o nome do programa que será criado



Informar as opções de criação do programa, conforme descritas no começo deste material.



O programa gerado já tem o código que o ABAP precisa para gerar o Batch Input, normalmente este código é bem trabalhoso de se construir baseado no mapeamento que normalmente é enviado via Excel (**não que não ajude, rs**).

Enviar este programa para o ABAP com o mapeamento é um atalho para a construção de algo mais rápido e para os funcionais mais “ABAPs”, cada ponto em verde não deve ser alterado, pois é o mapeamento da tela como ela consta no sistema, cada parte que está em preto com por exemplo “record-BNAME_001” são os campos que são alterados na tela durante a gravação, neste exemplo é o usuário que é informado ao tentarmos copiar.

The screenshot shows the SAP ABAP editor interface with the title "Editor ABAP: Report ZTESTEBATCH modificar". The code listed is:

```

Report ZTESTEBATCH ativo
62 do.
63   read dataset dataset into record.
64   if sy-subrc <> 0. exit. endif.
65
66   perform bdc_dynpro      using 'SAPLSUID_MAINTENANCE' '1050'.
67   perform bdc_field       using 'BDC_CURSOR'
68   perform bdc_field       using 'SUID_ST_BNAME-BNAME'.
69   perform bdc_field       using 'BDC_OKCODE'
70   perform bdc_field       using '=COPY'.
71   perform bdc_field       using 'SUID_ST_BNAME-BNAME'
72   perform bdc_field       using record-BNAME_001.
73   perform bdc_dynpro      using 'SAPLSUID_MAINTENANCE' '1200'.
74   perform bdc_field       using 'BDC_CURSOR'
75   perform bdc_field       using 'GV_COPY_UNAME_DST'.
76   perform bdc_field       using 'BDC_OKCODE'
77   perform bdc_field       using '=COPY'.
78   perform bdc_field       using 'GV_COPY_UNAME_SRC'
79   perform bdc_field       using record-GV_COPY_UNAME_SRC_002.
80   perform bdc_field       using 'GV_COPY_UNAME_DST'
81   perform bdc_field       using record-GV_COPY_UNAME_DST_003.
82   perform bdc_field       using 'GS_COPY_OPTIONS-DEFAULTS'
83   perform bdc_field       using record-DEFAULTS_004.
84   perform bdc_field       using 'GS_COPY_OPTIONS-PARAMETERS'
85   perform bdc_field       using record-PARAMETERS_005.
86   perform bdc_field       using 'GS_COPY_OPTIONS-REFUSER'
87   perform bdc_field       using record-REFUSER_006.
88   perform bdc_field       using 'GS_COPY_OPTIONS-ROLES'
89   perform bdc_field       using record-ROLES_007.
90   perform bdc_field       using 'GS_COPY_OPTIONS-PROFILES'
91
92

```



- Como abrir opções secundárias na transação SE16N

A transação SE16N pode ter muitas opções que ficam escondidas para a maioria dos usuários, mas com apenas um debug com opção editável podemos abrir novas opções que facilitam nossas correções urgentes que surgem em cenários caóticos.

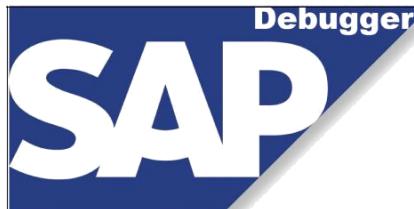
Vejamos abaixo como explorá-las:

NomeCpo.	O.	De valor	Até val.	Mais	Saída	Nome técnico
Nome da tabela					<input checked="" type="checkbox"/>	TABNAME
Stand de ativação					<input checked="" type="checkbox"/>	AS4LOCAL
Versão					<input checked="" type="checkbox"/>	AS4VERS
Categoria tabela					<input checked="" type="checkbox"/>	TABCLASS
...						

Ao executar a transação, veremos os dados normalmente, de uma forma mais limpa e prática, mas sem nenhuma opção secundária:

Material	Criado em	Criado por	Última modif.	Modif.por	Status atualização	Status atualiz.	Mt	TMat	S	Grp.merc.	NºMaterial	UMB	UMP
23	23.01.2004	BOHNSTEDT			K	K	X	ROH	1			CDA	
38	04.09.1995	CADCPIC	15.02.1999	MUELLERJ	KDEVG	KDEVG		HALB	M	00107		UN	
43	23.01.2004	BOHNSTEDT	01.03.2004	BOHNSTEDT	KBV	KBV		HAWA	1			HOR	
58	05.01.1996	DIEHL	23.01.2003	I021066	KLBX	KLB		HIBE	M			UN	
59	05.01.1996	DIEHL	23.01.2003	I021066	KLBX	KLB		HIBE	M			UN	
68	12.01.1996	PANACEK	23.01.2003	I021066	KEDPLQXZ	KEDPL		FHMI	A	013		UN	
78	10.06.1996	DIEHL	23.01.2003	I021066	KVX	KV	X	DIEN	M			UN	
88	27.05.1997	MORLEY	22.01.2003	I021066	KVB	KVB	X	FERT	M	02004		UN	
89	27.05.1997	MORLEY	22.01.2003	I021066	KV	KV	Y	FERT	M	02004		UN	

Via degug, temos uma estrutura de sistema chamada DB, esta estrutura abre novas opções para a execução dos dados na SE16N, vejamos como fazer abaixo:



Preenchemos os dados necessários para a seleção e antes de executar, vamos dar um /H seguido de enter e pressionar F8 no teclado:

Exibição geral de tabela

Tabela: MARA

Tab.textos: MAKT

Variante exib.:

Nº máx.ocorrências: 500

Critérios seleção

NomeCpo.	O.	De valor	Até val.	Mais	Saída	Nome técnico
Mandante						MANDT
Material					<input checked="" type="checkbox"/>	MATNR
Criado em					<input checked="" type="checkbox"/>	ERSDA
Criado por					<input checked="" type="checkbox"/>	ERNAM
Última modif.					<input checked="" type="checkbox"/>	LAEDA

A estrutura GD contém diversos campos que podem personalizar a forma de exibição dos dados na SE16N, é difícil numerar todas as opções possíveis, é preciso testar cada uma delas e verificar de acordo com a necessidade maior, infelizmente não existe uma forma de passar X para todos os campos desta estrutura, a única forma de descobrir novas opções é marcando X em algum dos campos e pressionando F8 no teclado para ver o que resultou, no exemplo abaixo vemos como abrir a opção de Adicionar, Modificar e Eliminar registros, para isso usamos os campos EDIT e SAPEDIT passando X através do Debug para eles e em seguida pressione F8 no teclado:

ABAP debugger(3) (exclusivo)(SAPBIT14_EPZ_00)

SAPSELEIN 0100 / 19 SY-SUBRC 0
FAI SY-TABIX 1

Varáveis 1 Varáveis 2 Locais Globais Auto Análise memória

S... Variável	C... Val.	M... Valor hexadecimal
GD-SAPEDIT	X	2000
GD-EDIT	X	2000

Foram adicionados os campos para a edição do conteúdo, mesmo se tratando de uma tabela Standard

MARA: exibição das entradas encontradas

Tabela de pesquisa: MARA Dados gerais de material

Nº ocorrências: 500

TmpExec.: 00:00:01 Nº máx.ocorrências: 500

Material	Criado em	Criado por	Ultima modif., Modif.por	Status atualização	Status atualiz.	Mt	TMat	S	Grp.merc.	NºMaterial	UMB	UMP
23	23.01.2004	BOHNSTEDT		K	K	ROH	1			CDA		
38	04.09.1995	CADCPIC	15.02.1999	MUELLERJ	KDEVG	X	HALB	1	00107	UN		
43	23.01.2004	BOHNSTEDT	01.03.2004	BOHNSTEDT	KBV		HAWA	1		HOR		
58	05.01.1996	DIEHL	23.01.2003	I021066	KLBX	KLB	HIBE	M		UN		
59	05.01.1996	DIEHL	23.01.2003	I021066	KLBX	KLB	HIBE	M		UN		
68	12.01.1996	PANACEK	23.01.2003	I021066	KEDPLQXZ	KEDPL	FHMI	A	013	UN		
78	10.06.1996	DIEHL	23.01.2003	I021066	KVX	KV	X	DIEN	M	UN		

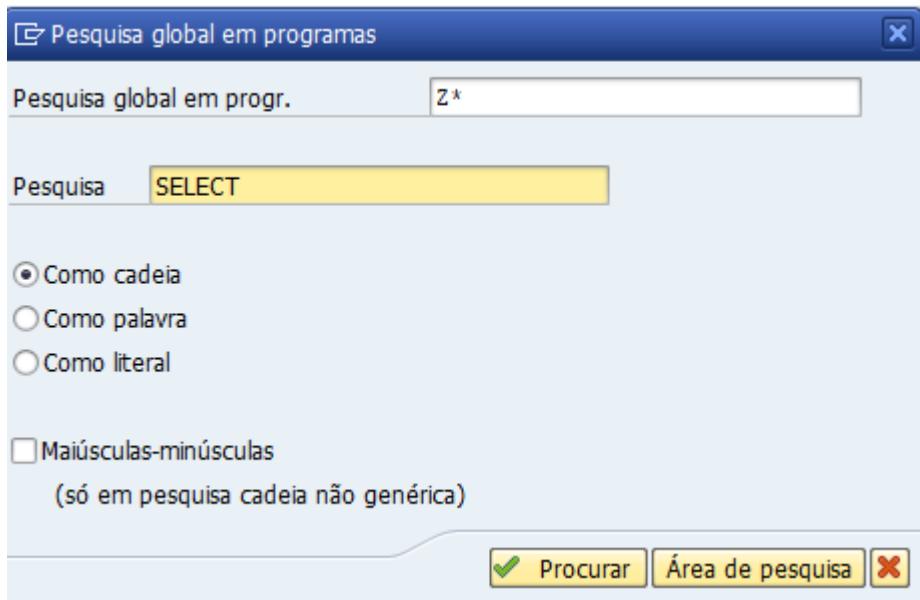


• SM50 – Monitoramento de processos

Podemos monitorar os processos que estão rodando no SAP e debugar a partir destes processos que já estão em andamento, mesmo que sejam executados via Job ou estejam “travados na tela”, para isso usamos a transação SM50

No exemplo em questão vou utilizar a transação EWK1 para pesquisar uma sintaxe ABAP em todos os programas “Zs” do SAP:

A transação EWK1 irá procurar a instrução SELECT em todos programas “Zs*” do SAP, este processo costuma demorar.



Através da SM50 consegui visualizar o processamento que fiz anteriormente e consigo ver inclusive em que momento do programa ele está, no caso abaixo ele está fazendo uma leitura de uma tabela “REPOSRC”, para debugar o programa, basta marcar, a linha e ir na opção conforme a imagem 2:

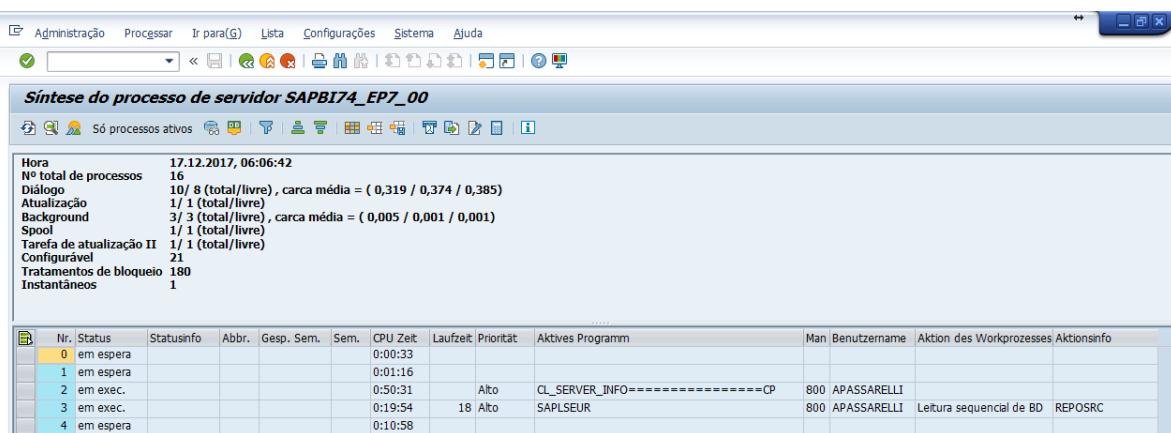




Imagen 2

Processo									
Programa Depuração 74_EP7_00									
Finalizar modo Shift+F2 Cancelar(O)									
Gerar instantâneo									
Finalizar Shift+F3 17, 06:07:58									
Nº total de processos 10 Dialogo 10/ 4 (total/livre) , carca média = (1,126 / 0,572 / 0,449) Atualização 1/ 1 (total/livre) Background 3/ 3 (total/livre) Spool 1/ 1 (total/livre) , carca média = (0,001 / 0,000 / 0,000) Tarefa de atualização II 1/ 1 (total/livre) Configurável 21 Tratamentos de bloqueio 180 Instantâneos 1									
.....									
Nr.	Status	Statusinfo	Abbr.	Gesp. Sem.	Sem.	CPU Zeit	Laufzeit	Priorität	Aktives Programm
0	em espera					0:00:33			
1	em exec.					0:01:16		Médio	SAPMSSY1
2	em exec.					0:50:32	6	Alto	SSFALRTEXP
3	em exec.					0:20:25	94	Alto	SAPLSEUR
4	parado	Resposta RFC				0:10:58		Médio	CL_BGRFC_SUPERVISOR=====CP
									800 BGRFC_SUPER

A clicar em depuração, o indicador irá debuggar o programa exatamente onde o programa está processando no momento, caso a tela não abra pode ser que o programa esteja travado fazendo algum processo, mas ao liberar a tela de debug será exibida, conforme o exemplo abaixo:

(/hs)ABAP debugger(6) (exclusivo) - ATTACHED -(SAPBI74_EP7_00)

INCREMENTO Watchpoint Layout Configurar camada de depurador

SAPLSEWORKINGAREA / LSEWORKINGAREAU04 / 179 SY-SUBRC 4

FUNCTION / RS_OBJECT_IN_WORKING_AREA SY-TABIX 0

Desktop 1 Desktop 2 Desktop 3 Standard Estruturas Tabelas Objetos Exibs.dethis Data Explorer Ptos.par./watchpts Diff Script

```

167   ELSE.
168   " Doch nachfragen, um Status anzuseigen
169   SELECT * FROM dwinactive WHERE object = object
170   AND obj_name = obj_name.
171   IF sy-subrc = 0.
172   IF dwinactive-uname = sy-uname.
173   object_is_work_item = 'X'.
174   ELSE.
175   object_inactive_version = 'X'.
176   ENDIF.
177   EXIT.
178   ENDIF.
179   ENDSELECT.
180   ENDIF.
181   ELSE.
182   SELECT SINGLE * FROM dwinactive
183   WHERE object = object
184   AND obj_name = obj_name .
185   IF sy-subrc = 0.
186   object_is_work_item = 'X'.
187   ELSEIF sy-subrc <> 0 AND obj_name(1) = 'L' OR obj_name(1)
188   l_include = obj_name.
189   l_length = STRLEN( l_include ) - 3.
190   IF l_length < 0.
191   EXIT.

```

Pilha ABAP e de tela

Po...	Nív...	T...	Evento	Programa	N...	Inclu...
8		FUNCTION	RS_OBJECT_IN_WORKIN...	SAPLSEUR	LSEW	
7		FORM	OBJ_STATE_MARK	SAPLSEUR	LSEUR	
6		FORM	FIND_GLOBAL	SAPLSEUR	LSEUR	
5		FUNCTION	RS_EDTR_SEARCH	SAPLSEUR	LSEUR	
4		EVENT	START-OF-SELECTION	EWUCU02B	EWUC	
3		PAI MODULE	SYST-ABRUN	SAPMSSY0		
2		PAI SCREEN	1000	SAPMSSY0		

Variáveis 1 Variáveis 2 Locais Globais Auto Análise memória

S...	Variável	C...	Val.	M...	Valor hexadecimal

Abrangência: \FUNCTION rs_object_in_workin... ABAP U 170 Co 30

Layout padrão utilizado (layout específico usuário -> texto descritivo)

• SM04 – Processos abertos, como encerrar

Esta é uma dica rápida de como encerrar seu processo caso o mesmo fique preso por algum motivo (normalmente quando o SAP cai)

(Mensagem exibida ao tentar editar um programa de uma sessão que fico presa)

Usuário APASSARELLI já está processando ZTESTEAAP .



Na transação SM04 podemos ver o nome do usuário logado e o nome de sua maquina, essa opção vai exibir todos os usuários então tome cuidado para não encerrar um usuário de outra pessoa, depois de selecionar seu usuário que esta com a sessão presa (identifiquei, pois a transação é a SE38 de edição de código)

Mand.	Benutzername	Logon Lokation	Anwendung	Hora de diálogo	Sess..	Prioridade	Session-Typ	Speicher	RFC Hdl	Anwendungsinfo
000	SAP_WSRT	SAPBI74.skimg.ah		17.12.2017 09:13:22	1	Médio	RFC	3.117	40768738	
000	SAP_WSRT	SAPBI74.skimg.ah		17.12.2017 09:13:23	1	Médio	RFC	3.117	60829935	
800	APASSARELLI	MWXP3_USR01	SE38	17.12.2017 09:12:32	1	Alto	GUI	19.519		
800	APASSARELLI	MWXP3_USR01		17.12.2017 09:13:23	6	Alto	GUI	80.503		
800	BGRFC_SUPER	SAPBI74.skimg.ah		17.12.2017 09:13:03	1	Médio	RFC	3.090	10486445	
800	BGRFC_SUPER	SAPBI74.skimg.ah		17.12.2017 09:13:21	1	Alto	Escalonador BGRFC	3.083	60815731	
800	BGRFC_SUPER	SAPBI74.skimg.ah		17.12.2017 09:13:20	1	Alto	Escalonador BGRFC	3.079	60814628	

Clicar no menu Usuário e ir na opção demonstrada abaixo:

Mand.	Benutzername	Logon Lokation	Anwendung	Hora de diálogo	Sess..	Prioridade	Session-Typ	Speicher	RFC Hdl	Anwendungsinfo
000	SAP_WSRT	SAPBI74.skimg.ah		17.12.2017 09:13:22	1	Médio	RFC	3.117	40768738	
000	SAP_WSRT	SAPBI74.skimg.ah		17.12.2017 09:13:23	1	Médio	RFC	3.117	60829935	
800	APASSARELLI	MWXP3_USR01	SE38	17.12.2017 09:12:32	1	Alto	GUI	19.519		
800	APASSARELLI	MWXP3_USR01		17.12.2017 09:13:23	6	Alto	GUI	80.503		
800	BGRFC_SUPER	SAPBI74.skimg.ah		17.12.2017 09:13:03	1	Médio	RFC	3.090	10486445	
800	BGRFC_SUPER	SAPBI74.skimg.ah		17.12.2017 09:13:21	1	Alto	Escalonador BGRFC	3.083	60815731	
800	BGRFC_SUPER	SAPBI74.skimg.ah		17.12.2017 09:13:20	1	Alto	Escalonador BGRFC	3.079	60814628	

Aí clicar em local o usuário será encerrado apenas naquela máquina da qual ficou preso, mas caso o usuário seja comum para todos como um usuário ABAP a opção Em todo o sistema irá efetuar logoff de todos que estão logados neste usuário, então cuidado rs.

Ao encerrar localmente, o usuário bloqueado irá sumir da lista:

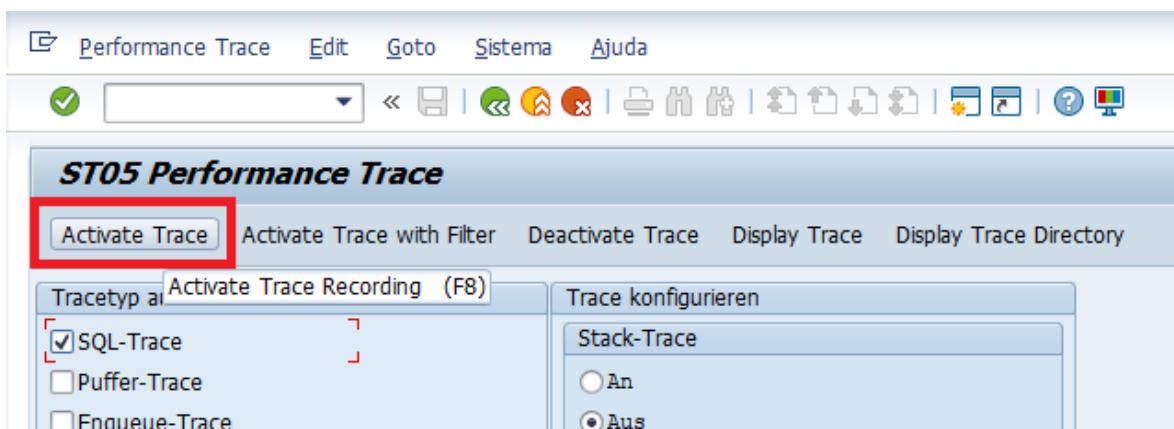
Mand.	Benutzername	Logon Lokation	Anwendung	Hora de diálogo	Sess..	Prioridade	Session-Typ	Speicher	RFC Hdl	Anwendungsinfo
000	SAP_WSRT	SAPBI74.skimg.ah		17.12.2017 09:17:21	1	Médio	RFC	3.117	40768738	
000	SAP_WSRT	SAPBI74.skimg.ah		17.12.2017 09:17:14	1	Médio	RFC	3.117	60829935	
800	APASSARELLI	MWXP3_USR01	SM04	17.12.2017 09:17:21	6	Alto	GUI	86.405		
800	BGRFC_SUPER	SAPBI74.skimg.ah		17.12.2017 09:13:21	1	Alto	Escalonador BGRFC	3.083	60815731	
800	BGRFC_SUPER	SAPBI74.skimg.ah		17.12.2017 09:13:20	1	Alto	Escalonador BGRFC	3.079	60814628	

- **ST05 (Trace ABAP)**

Utilizamos o TRACE para monitorar um processo que está em andamento e ver quais seleções que este processo passou, é ideal que o trace seja ativado apenas durante a execução do cenário que deve ser analisado, pois além de consumir muita memória do sistema, ele irá mapear todos os SELECTs que serão executados naquele usuário, ou seja, pode ter inúmeras linhas não necessárias na análise.

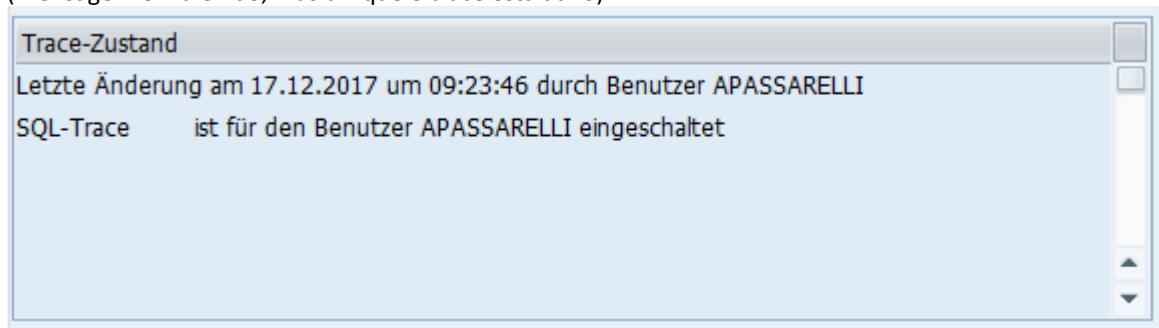
Para ativar o trace e monitorar um processo, faça como nas informações abaixo:

Transação ST05, vamos monitorar a entrada na transação VA01, par ativar o TRACE clique no botão conforme a imagem abaixo:



A partir deste ponto Trace estará ativo e você pode fazer normalmente o processo que precisa mapear.

(Mensagem em alemão, mas diz que o trace está ativo)





Processo na VA01 a ser monitorado:

The screenshot shows the SAP VA01 transaction interface. The title bar reads "Criar ordem de cliente: 1ª tela". Below the title bar, there are three tabs: "Criar com referência" (selected), "Venda", "Síntese de itens", and "Emissor do pedido". The main area is titled "Dados organizacionais" and contains fields for "Organização vendas", "Canal distribuição", "Setor de atividade", "Escritório de vendas", and "Equipe de vendas". A "Tipo de ordem" field is set to "ORB".

Entrei na transação e vou sair em seguida e desativar o trace, para ver de onde que o SAP faz a seleção do Tipo de Ordem de venda ORB no SAP

The screenshot shows the SAP ST05 transaction interface. The title bar reads "criar Ordem padrão BR: Síntese". Below the title bar, there are tabs for "Ordens" and "Documento". The main area contains fields for "Ordem padrão BR" (highlighted with a yellow box), "Valor líquido" (0,00), "Emissor da ordem" (highlighted with a yellow box), "Receb.mercad.", "Nº do pedido", "Data pedido", and "Data deseja.rem." (17.12.2017). Below these fields is a grid with columns for "Centro fornec.", "Peso total" (0,000), "Volume" (0,000), and "DtFixPreço" (17.12.2017). Other tabs visible include "Síntese de itens", "Detalhe de item", "Emissor do pedido", "Recrutamento", "Expedição", and "Motivo de recusa".

Desative o Trace na transação ST05 e clique em Display Trace ao lado da opção desativar



Debugger

Performance Trace Edit Goto Sistema Ajuda

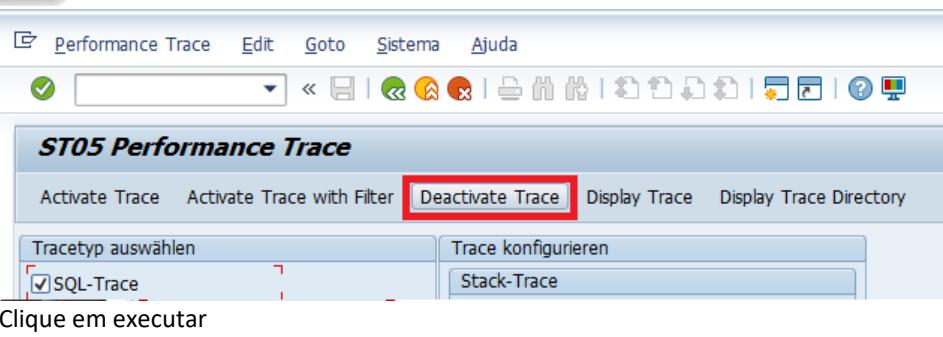
ST05 Performance Trace

Activate Trace Activate Trace with Filter Deactivate Trace Display Trace Display Trace Directory

Tracetypr auswählen Trace konfigurieren

SQL-Trace Stack-Trace

Clique em executar



Programa Processar Ir para(G) Sistema Ajuda

ST05 Filter-Bedingungen für Trace-Sätze

Trace-Typen

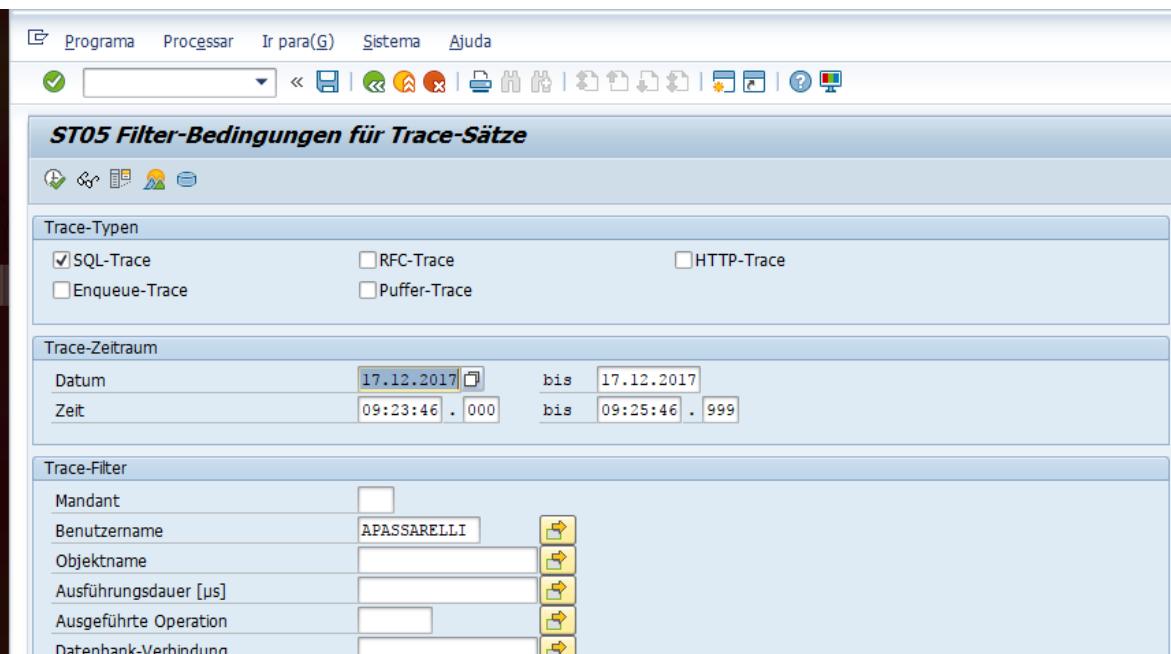
SQL-Trace RFC-Trace HTTP-Trace
 Enqueue-Trace Puffer-Trace

Trace-Zeitraum

Datum 17.12.2017 bis 17.12.2017
Zeit 09:23:46 . 000 bis 09:25:46 . 999

Trace-Filter

Mandant
Benutzername APASSARELLI
Objektname
Ausführungszeit [us]
Ausgeführte Operation
Datenbank-Verbindung



Clique em pesquisar e informe a opção que deseja encontrar, no nosso caso é o tipo de ordem ORB, porém este caso não retornou registros, é bastante comum, pois o SAP não trabalha dessa forma, mas a opção de pesquisa pode ajudar, por isso foi colocada neste exemplo e como quase nunca funciona da forma que queremos, na segunda imagem abaixo vejamos como podemos explorar as linhas (no olho) para entender melhor

Trace Edit Goto Sistema Ajuda

Performance Analysis: Trace Display (Main Records)

hh:mm:ss...	Σ Dauer	Σ Sätze	Programm-Name	Objektname	Anweisung
	3.857.862	2.363			
09:23:46.219	318	1	CL_ST05_TRACE_MAIN_M=====CP	DDNTT	SELECT WHERE
	13.131	1	CL_ST05_TRACE_MAIN_M=====CP	DDNTF	SELECT WHERE
09:23:46.283	168.725	1	CL_ST05_TRACE_MAIN_M=====CP	ST05_TRACE	SELECT WHERE
09:23:46.402	201	0	D_ST05_TRACE_MAIN		COMMIT WORK
09:24:23.629					UPDATE SET
09:24:23.630					DELETE WHERE
09:24:23.687					COMMIT WORK
09:24:23.709					ACT SELECT WHERE
09:24:23.735					SELECT WHERE
09:24:23.780					SELECT WHERE
09:24:23.827					COMMIT WORK
09:24:49.765					SELECT WHERE
09:24:49.891	77.863	1	SAPMV45A	USRBFZ	SELECT WHERE
09:24:49.891	500	1	SAPMV45A	USRREFZ	SELECT WHERE

Procurar

Term.pesq.: ORB

Direç.pesq.:

Só procurar palavra completa ou valor total

Exibir nûm.ocorrências

✓ ✘



Análise visual:

A coluna selecionada em laranja mostra as condições das seleções que foram executadas em todos os processos que rodaram para aquele usuário durante o período que o Trace ficou ativo, quando precisamos de algo específico normalmente essa coluna ajuda a identificar a tabela que estamos procurando, para termos certeza do que houve naquela linha, basta dar um duplo clique sobre ela.

O select será exibido como é feito no código ABAP e suas condições de seleção serão exibidas em variáveis A0, A1, A2 e em seguida descritas na legenda, isso pode facilitar muito a análise de um problema no SAP.

The screenshot shows the SAP SE11 interface with the following details:

- Toolbar:** Statement Details, Edit, Goto, Sistema(Y), Ajuda.
- Header:** SQL Trace Record
- Section:** Details zum ausgewählten SQL-Trace-Satz
- Query:**

```
SELECT
    "PROGNAME", "SPRSL", "OBJ_CODE", "TEXT"
FROM
    "D347T"
WHERE
    "PROGNAME" = ? AND "SPRSL" = ? AND "OBJ_CODE" = ?
ORDER BY
    "PROGNAME", "SPRSL",
    "OBJ_CODE" WITH CS OPTLEVEL( 5 ) QUERY_DEGREE( 1 ) LOCATION( SAPLSLVC
    _FULLSCREEN , 4180 ) SYSTEM( EP7 , SAPEP7 )
```
- Section:** Variablen
- Variables:**

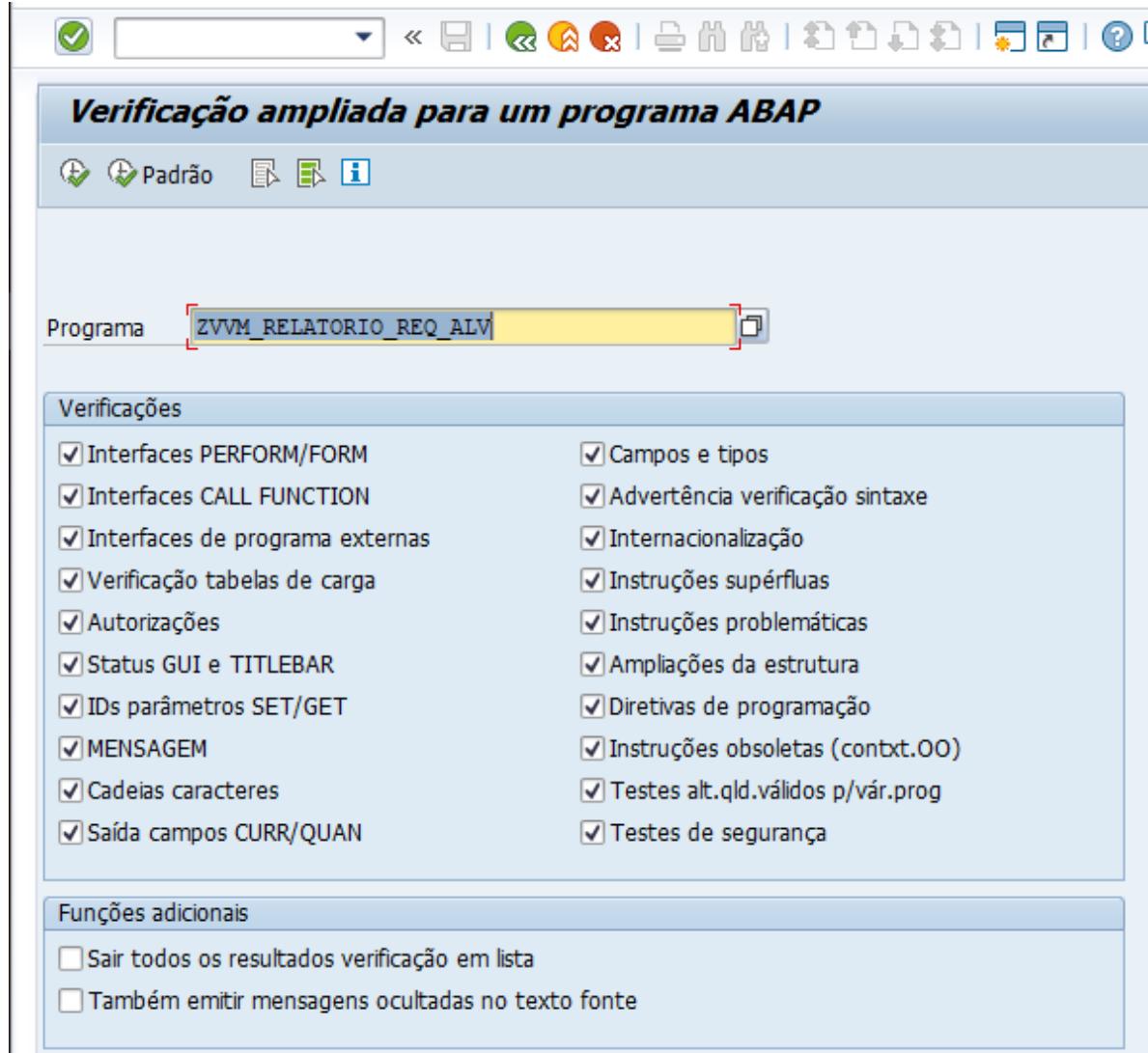
A0(CH,40)	= 'SAPMV45B'
A1(CH,1)	= 'P'
A2(CH,20)	= 'A01'

- **SLIN (Inspeção em Códigos)**

A transação SLIN faz uma análise do programa ABAP que for informado, informando possíveis erros que devem ser verificados pelo programador, é uma análise standard que pode ajudar caso um programa esteja apresentando qualquer comportamento estranho, principalmente se for performance.

Transação SLIN

Marcando todas as opções e informando o nome do programa ABAP, clique em executar



Resultado:

A lista de erros será exibida e pode ser analisada clicando duas vezes em cima do erro:

Síntese SLIN				
	Exibir resultados	Exibir todos os resultados	Exibição do teste individual	
	Verif.p/progr.ZVVM_RELATORIO_REQ_ALV	Erro	Adverts.	Mensagens
Ambiente de teste		0	0	0
Interfaces PERFORM/FORM		0	0	5
Interfaces CALL FUNCTION		0	0	0
Interfaces de programa externas		0	0	0
Autorizações		0	0	0
Status GUI e TITLEBAR		0	0	0
IDs parâmetros SET/GET		0	0	0
MESSAGE		3	0	0
Cadeias de caracteres		3	0	0
Saída campos CURR/QUAN		0	0	0
Características de campo		0	6	0
Instruções supérfluas		0	0	0
Advertência da verificação de sintaxe		0	1	0
Verif.tamanhos carregamento		0	0	0
Internacionalização		0	0	0
Instruções problemáticas		0	0	0
Ampliações da estrutura		0	0	0
Diretivas de programação		0	20	0
Instruções obsoletas		0	7	0
Testes vál.p/vários progs.		0	0	0
Testes de segurança		0	0	0
Erros e advertências ocultos		0	0	0

Normalmente um programa ABAP de uma consultoria que segue padrões mais rígidos não pode apresentar erros no SLIN, apenas Warnings e informações, que são a segunda e terceira informação que o programa retorna.

Exemplo da demonstração do erro:

Síntese SLIN	
Exibir	Modificar
Mensagens para MESSAGE(Erro)	
Programa: ZVVM_RELATORIO_REQ_ALV Linha: 22 [Prio 2]	
O ID MESSAGE ZMM010 não existe	
Pode ser oculto pelo pragma ##MG_MISSING.	
Programa: ZVVM_RELATORIO_REQ_ALV Linha: 111 [Prio 3]	
Nº de campos WITH de MESSAGE 398 para ID 00: 1	
Nº de caractere de preenchimento na mensagem: 4	
Mensagem: & &	
Nº max.caractere preenchim.no texto descr.: 4	
Texto descriptivo:	
&CAUSE& Platzhalter für Batch-Input-Fehlertext, diese Meldung wird nicht ausgegeben. &V1& &V2&	
&V3& &V4& &SYSTEM_RESPONSE& &WHAT_TO_DO&	
Pode ser oculto pelo pragma ##MG_MISSING.	
Programa: ZVVM_RELATORIO_REQ_ALV Linha: 170 [Prio 3]	
Nº de campos WITH de MESSAGE 398 para ID 00: 1	
Nº de caractere de preenchimento na mensagem: 4	
Mensagem: & &	
Nº max.caractere preenchim.no texto descr.: 4	
Texto descriptivo:	
&CAUSE& Platzhalter für Batch-Input-Fehlertext, diese Meldung wird nicht ausgegeben. &V1& &V2&	
&V3& &V4& &SYSTEM_RESPONSE& &WHAT_TO_DO&	
Pode ser oculto pelo pragma ##MG_MISSING.	



- **SCID (Análise do código)**

Com a mesma utilidade que a transação SLIN, o Code Inspector pode avaliar situações do código de forma mais ampla e pode ser configurado para funcionar apenas para um problema em questão, as possibilidades de uso do Code Inspector são inúmeras e abaixo segue um link de como utilizar bem esta ferramenta, de um blog muito bom que sempre acompanho:

<https://abapinho.com/2014/03/o-detective-do-abap/>

- **Pular travas de autorização**

Uma das coisas que mais atrapalha a vida de um funcional e até mesmo ABAP é a falta de autorização para conseguir fazer algum procedimento no SAP, mesmo que ninguém irá te punir se você entrar.

Para isso, temos três formas bem práticas de pular qualquer trava que o sistema coloque.

- **Via Authority-Check**

Ao ser travado em qualquer processo, um momento antes da mensagem de falta de autorização digite /H no TCODE e pressione enter.

Aperte a tecla F9 no teclado e digite a sintaxe AUTHORITY-CHECK

The screenshot shows the SAP ABAP debugger interface. The main window displays an ABAP module with the following code:

```
12 module re_suppress SWITCH ECO_HBS.  
13   * IS A&D/E&C 4.6b, RE-SCM Real Estate Sale  
14   L * suppress additional fields of RE-SCM in case of regular tra  
15  
16   * Verarbeitung nach der Eingabe  
17   PROCESS AFTER INPUT.  
18  
19
```

The code editor has a cursor at line 19, specifically on the word "CREATE". Below the code editor, a "Criar pontos de parada" (Create Breakpoints) dialog box is open, showing the command "authority-check" entered in the "ComandosABAP" tab. In the top right corner of the SAP interface, there is a "Pilha ABAP e de tela" (ABAP and screen stack) window showing the current stack state:

Po...	Nív....	T...	Tipo evento	Evento
2		PAI SCREEN	0101	
1		TRANSACTION	VA01(VA01)	



Ao confirmar o programa criará pontos de parada em todos os objetos de autorização que aquele processo passar, ao pressionar F8 no teclado o programa sempre irá para o próximo objeto de bloqueio, para conseguir pular a etapa, é preciso editar o resultado da variável de retorno SY-SUBRC após o authority-check, conforme a imagem abaixo:

The screenshot shows the SAP ABAP debugger interface. The code editor displays a portion of the program MV45AF0B_BERECHTIGUNG_PRUEFEN. A break point is set at line 84. The variable editor on the right shows the variable SY-SUBRC with a value of 4. The status bar indicates the current transaction is SAPBI74_EP7_00.

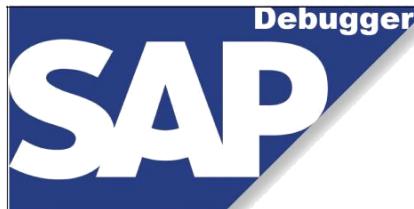
O Debug ficará em cima da linha do authority-check, ao pressionar F5 ou F6 para passar pela instrução o valor de SY-SUBRC será 0, pois você não tem autorização para ir além deste ponto, ao editar o valor da variável para 0 e seguir fazendo isso em todos os outros “authoritys-checks” todas as travas serão liberadas para a trasação que você precisa.

This screenshot shows the SAP variable editor with the variable SY-SUBRC set to 0. The status bar indicates the current transaction is SAPBI74_EP7_00.

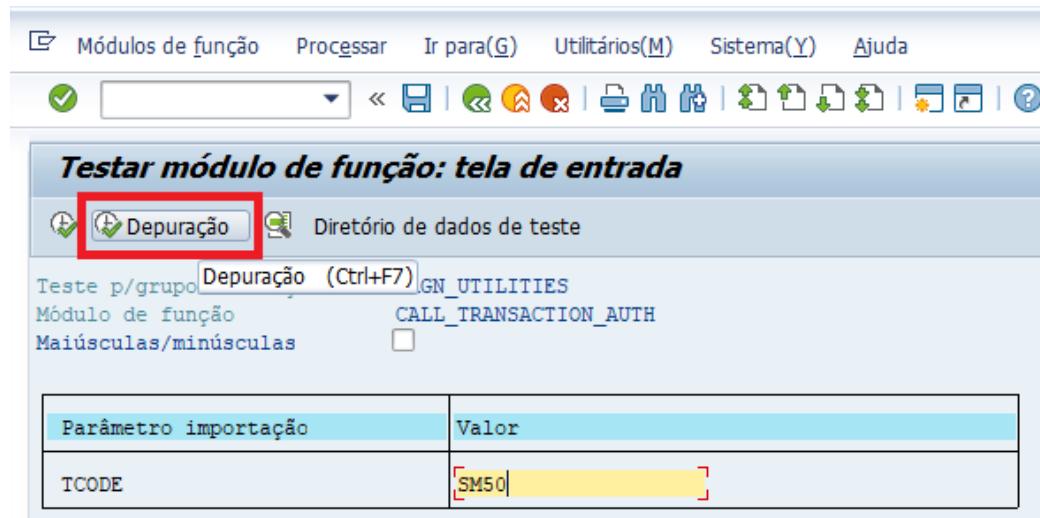
- **Via função**

Acessar a transação SE37 e informar a função “call_transaction_auth” conforme abaixo e executar:

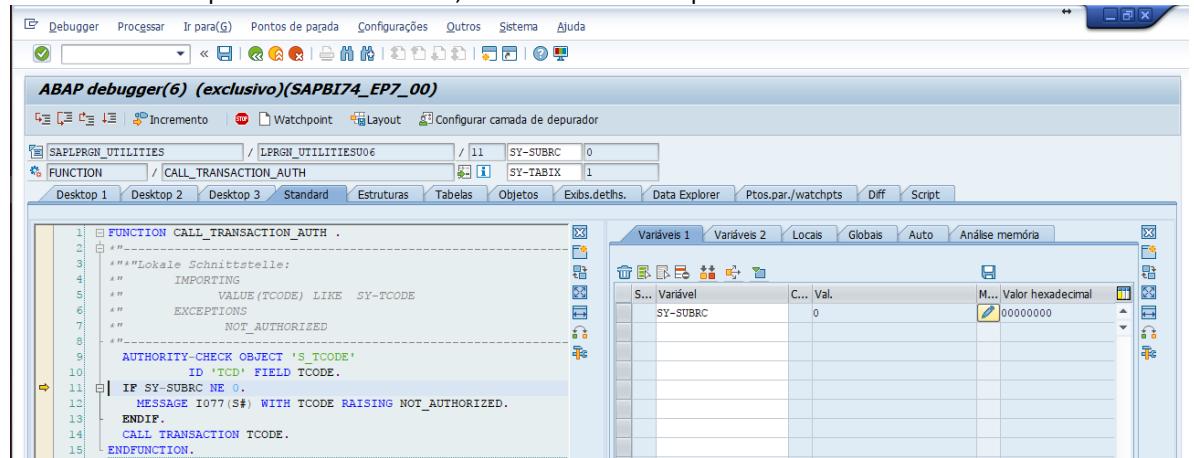
The screenshot shows the SAP Function Builder interface. The search bar contains "call_transaction_auth". Below it, there are buttons for "Exibir" (Display), "Modif." (Modify), and "Criar" (Create). The status bar indicates the current transaction is SE37.



Informar a transação da qual você não tem acesso e clicar em Depuração, conforme a imagem abaixo:



Realizar o mesmo procedimento anterior, mudando SY-SUBRC para 0



- **Encontrando EXITS**

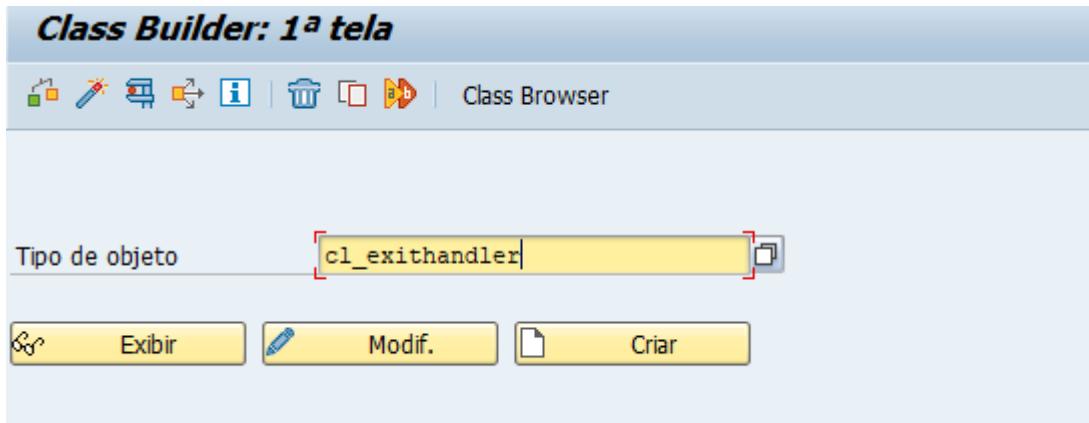
A forma mais rápida que temos de resolver um problema é através de uma exit em um standard, temos os 3 tipos de exits mais comuns, são eles User-Exit, BADI e Enhancement, os detalhes sobre cada um e suas diferenças são detalhes do curso ABAP, porém a forma de encontrar cada um desses recursos segue abaixo:

- **BADI**

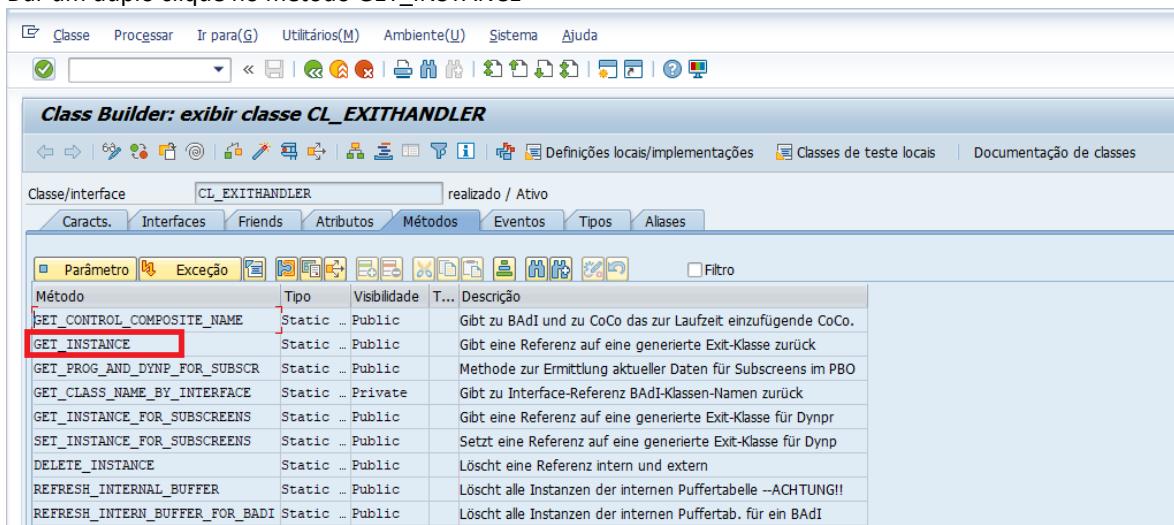
Para encontrar uma BADI, podemos usar inúmeros programas que podemos encontrar no SCN, os famosos “Z_FIND_BADI”, segue abaixo um exemplo do código deste programa, mas para quem gosta de caçar “na raça” segue o tutorial de como fazer:

Link para o Z_FIND_BADI: <https://archive.sap.com/discussions/thread/1733065>

Acessar a transação SE24 informando a classe CL_EXITHANDLER e clicar em exibir



Dar um duplo clique no método GET_INSTANCE





Marcar um BREAK-POINT na linha 14

Class Builder: Classe CL_EXITHANDLER exibir

Método: GET_INSTANCE

Tp.	Parâmetro	Atrib.tipo	Descrição
no	value(EXIT_NAME)	TYPE EXIT_DEF OPTIONAL	BADI-Def.
no	value(NULL_INSTANCE_ACCEPTED)	TYPE SEEX_BOOLEAN DEFAULT SEEX_FALSE	Nul-Instanz akzeptabel, wenn es keine aktiven Imp. gibt?
no	value(ACT_IMP_EXISTING)	TYPE SXTR_BOOLEAN	Gibt es eine aktive Implementierung
no	INSTANCE	TYPE ANY	Instanz
no	NO_REFERENCE		Keine Referenz
no	NO_INTERFACE_REFERENCE		Keine Interface-Referenz
no	NO_EXIT_INTERFACE		Kein Exit-Interface vorhanden
no	CLASS_NOT_IMPLEMENT_INTERFACE		Die Exitklasse implementiert nicht das Exit-Interface
no	SINGLE_EXIT_MULTIPLY_ACTIVE		Mehrfache Aktivierung bei singulärem Exit
no	CAST_ERROR		Fehler beim Cast
no	EXIT_NOT_EXISTING		Exit existiert nicht
no	DATA_INCONS_IN_EXIT_MANAGEM		Dateninkonsistenz in der Exit-Verwaltung

```
10      mig_enhsproname TYPE enhspotname,
11      isImpl TYPE enhboolean,
12      mig_badi_name TYPE enhbadiid.
13
14 | CALL METHOD cl_exithandler->get_class_name_by_interface
15   EXPORTING
16     instance          = instance
17   IMPORTING
18     class_name        = class_name
19   CHANGING
20     exit_name         = exit_name
21   EXCEPTIONS
22     no_reference      = 1
23     no_interface_reference = 2
24     no_exit_interface = 3
25     data_incons_in_exit_managem = 4
26     class_not_implement_interface = 5
```

Abrangência: \METHOD get_instance

Ponto de parada da sessão definido

A partir deste momento, em qualquer processo que você passar pelo SAP, seu usuário irá exibir as BADIS que passam por aquele ponto, para saber identificar a BADI, basta abrir o campo EXIT_NAME, este campo contém o nome das BADIs daquele ponto, no exemplo abaixo é a WB_PROCESS_FCODE

ABAP debugger(6) (exclusivo)(SAPB174_EP7_00)

METHOD / GET_INSTANCE(CL_EXITHANDLER)

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14 | CALL METHOD cl_exithandler->get_class_name_by_interface
15   EXPORTING
16     instance          = instance
17   IMPORTING
18     class_name        = class_name
19   CHANGING
20     exit_name         = exit_name
21   EXCEPTIONS
22     no_reference      = 1
23     no_interface_reference = 2
24     no_exit_interface = 3
25     data_incons_in_exit_managem = 4
26     class_not_implement_interface = 5
```

Variáveis 1 Variáveis 2 Locais Globais Auto Análise memória

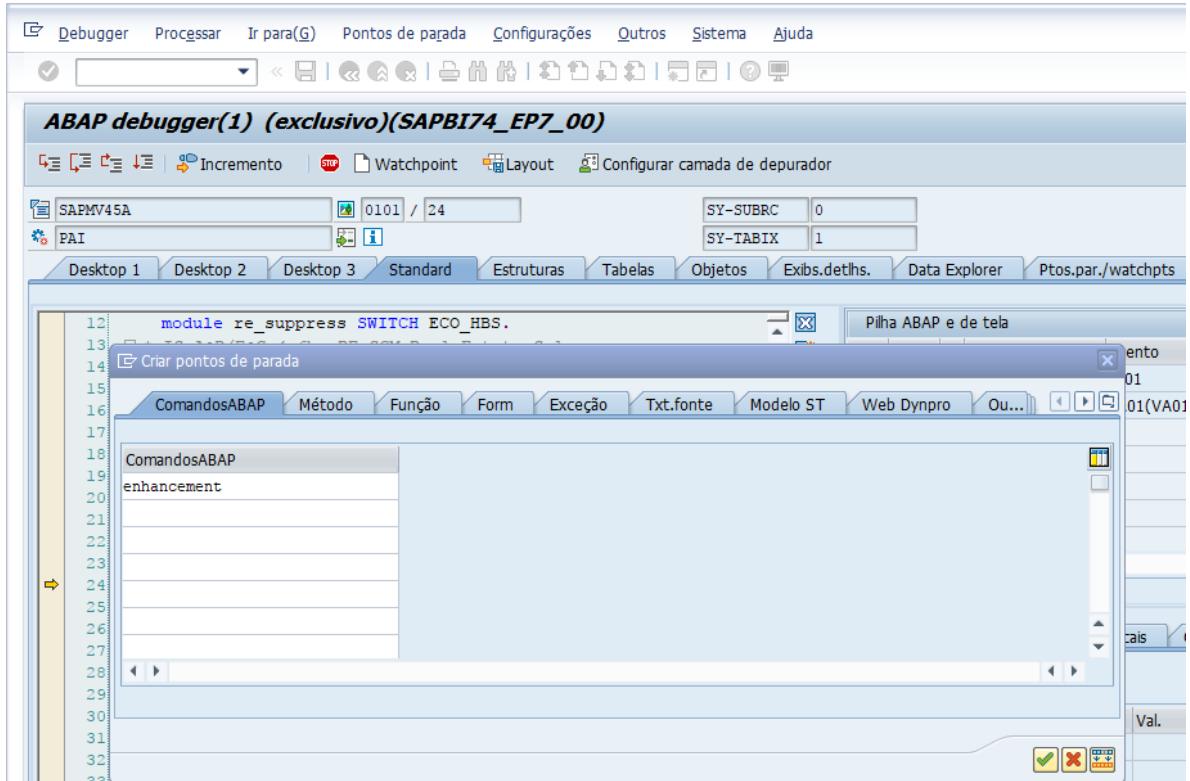
S...	Variável	C...	Val.	M...	Valor hexadecimal
	EXIT_NAME			WB_PROCESS_FCODE	570042000F00500052



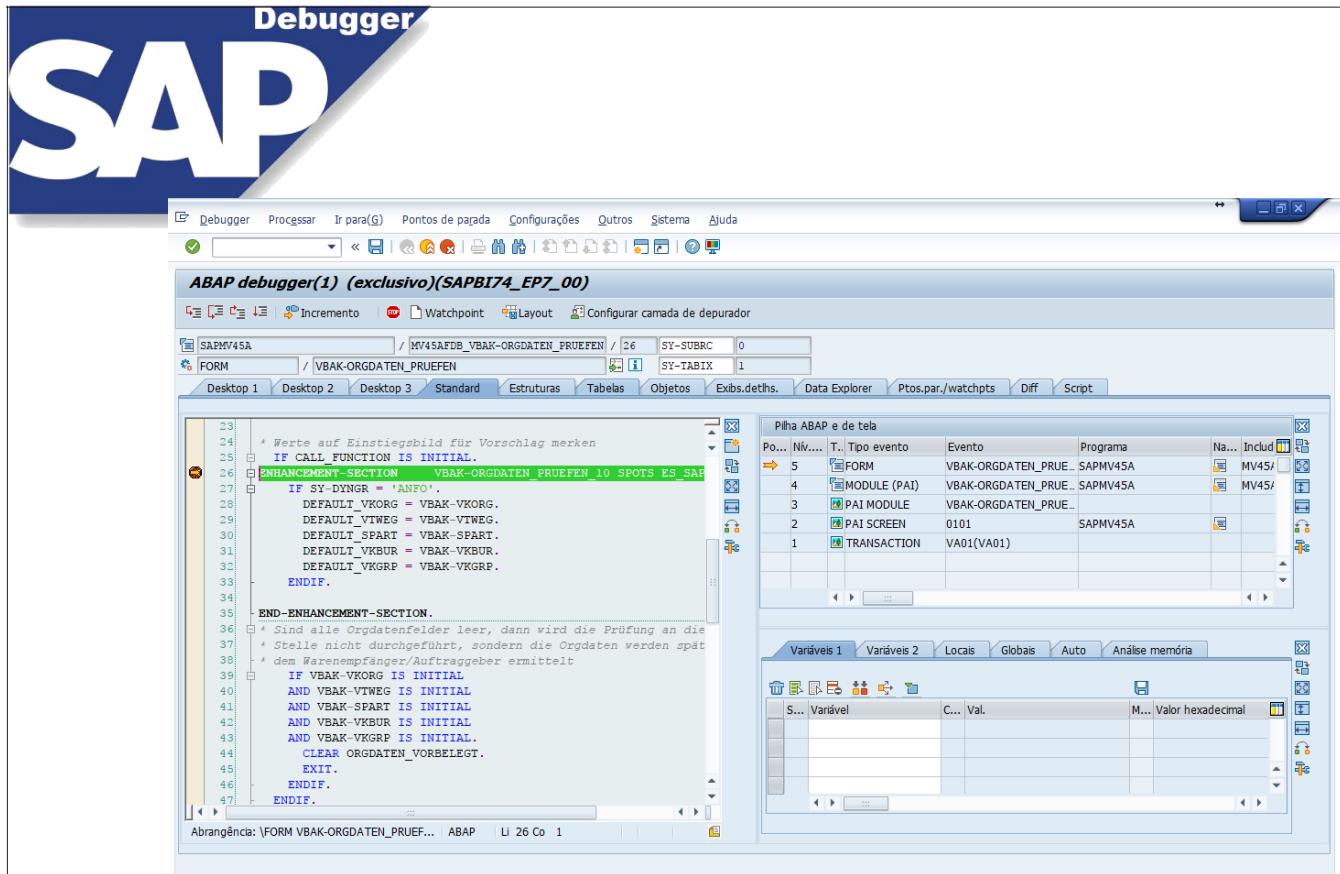
- Enhancement

Para encontrar um Enhancement utilizamos um processo parecido com o de Authority-Check, colocamos o programa no ponto antes de onde queremos descobrir se existe um Enhancement, inserimos o /H no TOCDE para abrir o Debug e em seguida pressionamos F9 no teclado:

Ao informar a sintaxe Enhancement vamos fazer o programa parar em todos que existem naquele programa:



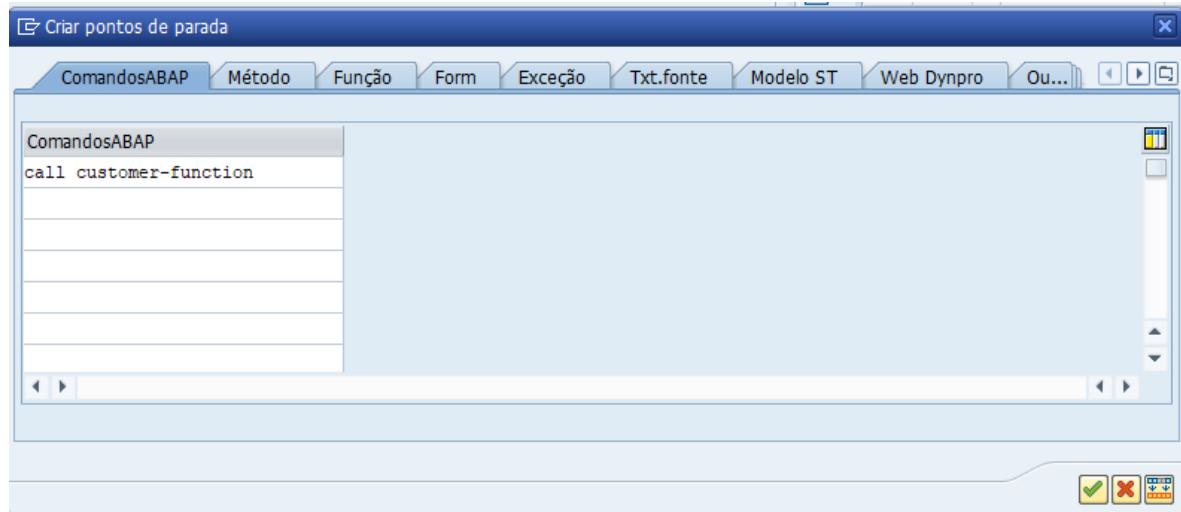
Exemplo:



- **User-Exit**

Ainda seguindo o mesmo padrão de do Enhancement e Authority-Check, para encontrar uma User-Exit, vamos até o ponto do processo que queremos encontrar uma exit, inserimos o /H no TCODE para abrir o debug e pressionamos F9:

Ao informar a sintaxe “call customer-function” a partir daquele ponto o programa passará por todas as exits possíveis naquele processo:



Exemplo:

SAP Debugger

The screenshot shows the SAP ABAP debugger interface. The code editor displays a portion of the MV45AOOF_FELDAUSWAHL program, specifically lines 168 to 192. The code includes several ENHANCEMENT-POINT statements and a Userexit block. The call stack (Pilha ABAP e de tela) shows the execution path from a TRANSACTION VA01 down to the PBO module FELDAUSWAHL. The variables view (Variáveis 1) is currently empty.

```

168:      endtry.
169:      endif.
170:      endif.
171:
172:      call customer-function '004'
173:          exporting
174:              i_screen_name      = screen-name
175:              i_vbap             = vbap
176:              i_vbup             = xvbp
177:              i_screen_group4   = screen-group4
178:              i_t180_aktyp       = t180-aktyp
179:          changing
180:              c_screen_active    = screen-active
181:              c_screen_invisible = screen-invisible
182:              c_screen_input     = screen-input.
183:
184:
185:      ENHANCEMENT-POINT MV45AOOF_FELDAUSWAHL_17 SPOTS ES_SAPMV4
186:
187:      ENHANCEMENT-POINT MV45AOOF_FELDAUSWAHL_16 SPOTS ES_SAPMV4
188:
189: * Userexit
190:     perform userexit_field_modification.
191:
192:     modify screen.

```

Po...	Nív...	T...	Tipo evento	Evento	Programa	Na...	Inclui
6			MODULE (PBO)	FELDAUSWAHL	SAPMV45A		MV45/
5			PBO MODULE	FELDAUSWAHL			
4			PBO SUBSCREEN	4701	SAPMV45A		
3			PBO SUBSCREEN	4021	SAPMV45A		
2			PBO SCREEN	4001	SAPMV45A		
1			TRANSACTION	VA01(VA01)			

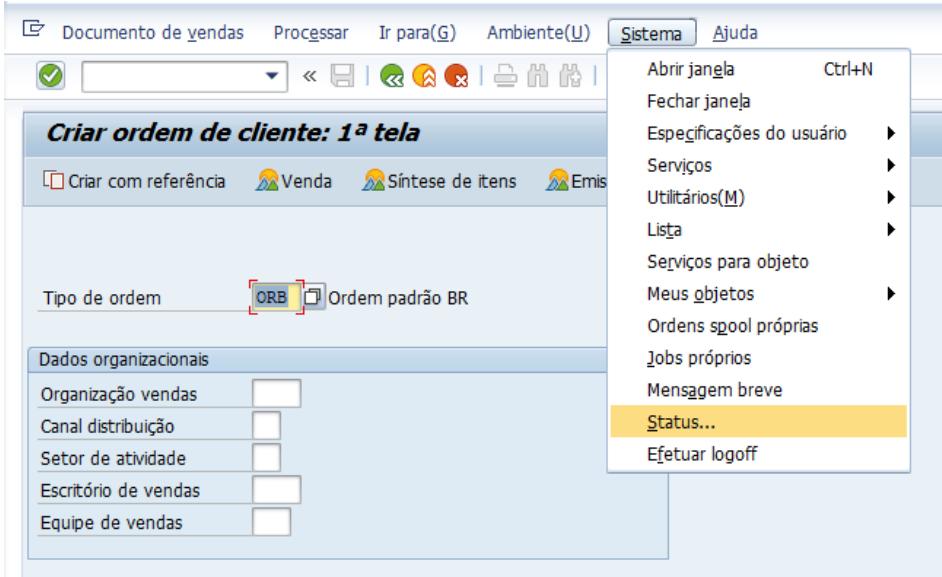
Variáveis 1 Variáveis 2 Locais Globais Auto Análise memória

Podemos encontrar USER-EXITS também através do nome do programa de uma transação Standard, no caso vamos usar a da VA01:

Para pegar o nome do programa de uma transação:



Acessar a transação desejada e seguir no menu:

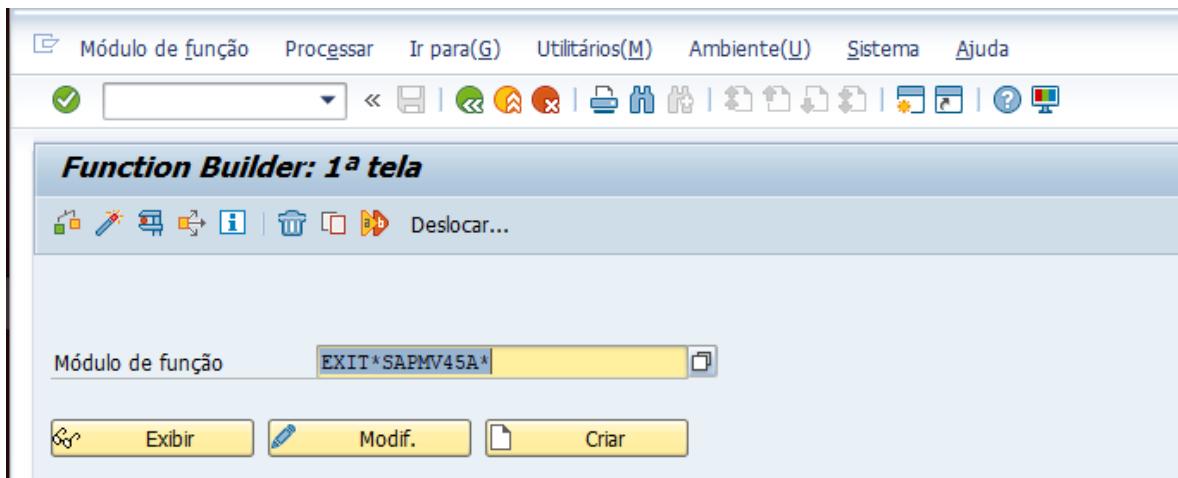


Nome do programa

The screenshot shows the 'Sistema: status' dialog box. It has three main sections: 'Dados de utilização', 'Dados SAP', and 'Dados host'. The 'Dados SAP' section is currently active and expanded. It contains two tables: 'Dados Repository' and 'Dados do sistema SAP'. In the 'Dados Repository' table, the 'Transação' field is set to 'VA01' and the 'Programa (tela)' field is set to 'SAPMV45A', both of which are highlighted with a red box. A red arrow points from the text above to the 'Programa (tela)' field. In the 'Dados do sistema SAP' table, fields include 'Versão componentes' (EHP7 para SAP E...), 'Nº instalação' (0000000001), 'Dt.vencim.licença' (31.12.9999), and 'Sist.unicode' (sim). The 'Dados host' section contains information about the operating system ('Windows NT'), machine category ('4x X86_64'), server name ('SAPBI74_EP7_00'), and platform ID ('562'). The 'Dados do banco de dados' section contains details about the database ('Datenbank-System' DB6, 'Release' 10.05.0003, 'Nome' EP7, 'Host' SAPBI74, 'Titular' SAPEP7). At the bottom right of the dialog box are buttons for 'Salvar' (checkmark), 'Navegar' (arrow), and 'Cancelar' (cross).

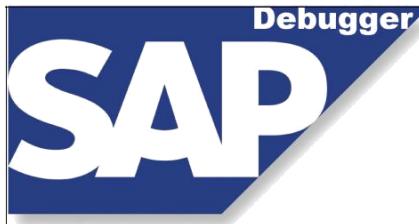


Preencher de acordo com a configuração abaixo e pressionar F4:



Todas as User-Exits daquela transação (VA01) serão exibidas:

Sistema de informação do R/3 Repository: Módulos de função pesquisa	
Grupo de funções	Texto breve do grupo de funções
Nome do módulo de função	Texto breve para módulo funcional
XOIK	
EXIT_SAPMV45A_910	
EXIT_SAPMV45A_911	
EXIT_SAPMV45A_920	
EXIT_SAPMV45A_930	
XVVA	
EXIT_SAPMV45A_001	
EXIT_SAPMV45A_002	
EXIT_SAPMV45A_003	
EXIT_SAPMV45A_004	
EXIT_SAPMV45A_005	



7.0 - LSMW

O material que segue no link abaixo é uma colaboração de um aluno e grande amigo que quero deixar registrado aqui como forma de agradecimento, quero tornar esse material cada vez melhor e mais completo, caso tenha interesse em contribuir dessa forma, basta me comunicar.

Manuais de LSMW por [Gelton Sobreira Costa](https://aztreinamentos.com/blog/manual-de-lsmw-sap): <https://aztreinamentos.com/blog/manual-de-lsmw-sap>

8.0 - Query

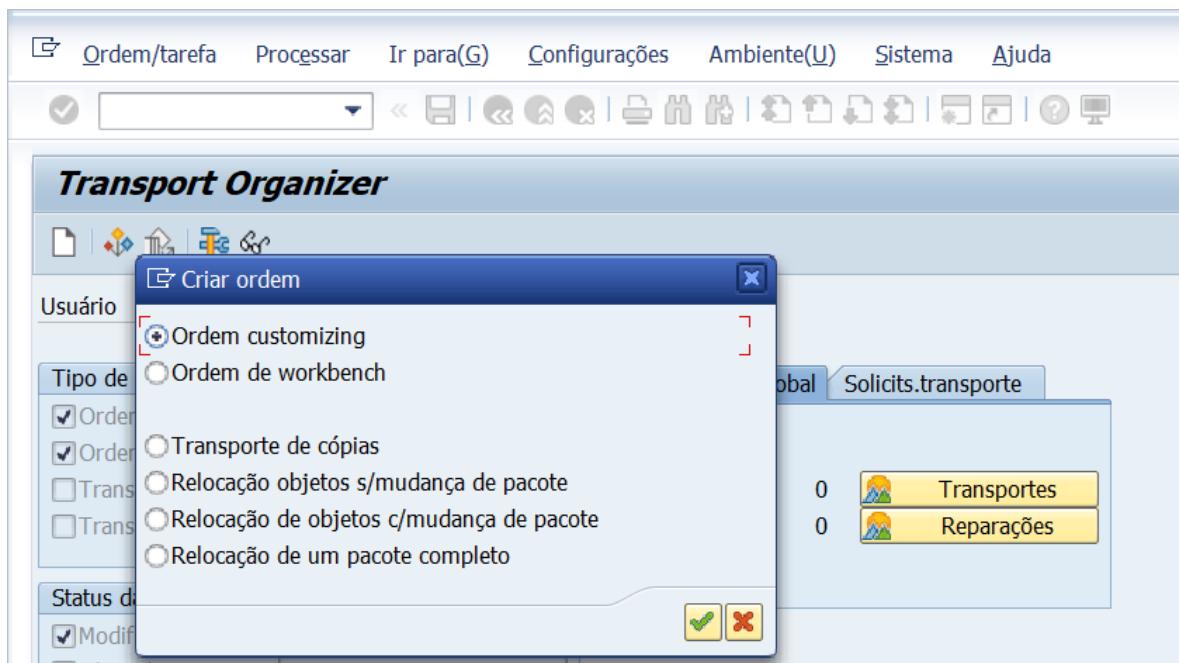
Mesmo sendo de uma versão mais antiga, esse manual pode ajudar muito na hora de construir uma query, com a query você pode criar relacionamentos de tabelas para exibi-las em relatórios sem a necessidade de um programador ABAP, e ainda poderá personalizar a tela se seleção e os campos que serão exibidos nesses relatórios, segue ao lado o link para o manual em PDF, o material é bastante auto explicativo, em breve teremos uma nova versão: <https://aztreinamentos.com/blog/manual-de-infoset-query>

9.0 – Configurações do SAP Gui

Para esta explicação, irei usar a versão 7.5 do SAP, caso você ainda não tenha, por favor, peça o link de instalação dessa nova versão pelo e-mail: [contato@aztreinamentos.com](mailto: contato@aztreinamentos.com), o mesmo será enviado à todos que solicitarem.

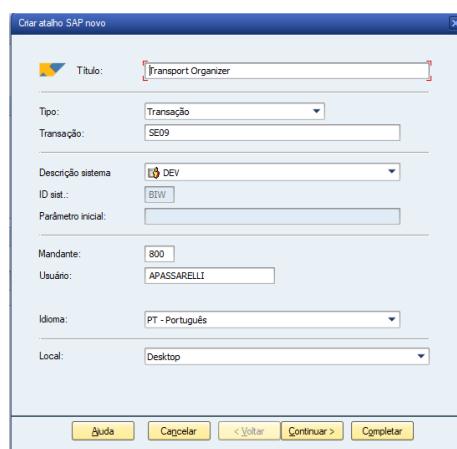
- Habilitar Debug em Popup

Neste caso, usarei o exemplo da criação de uma request, ao clicar em criar a nova request na transação SE09, será aberto um popup com as opções de requests existentes, se precisarmos debugar exatamente deste ponto o campo do TCODE que permite inserir o /H estará na tela de trás, não permitindo assim que você possa debugar a partir do ponto da popup.

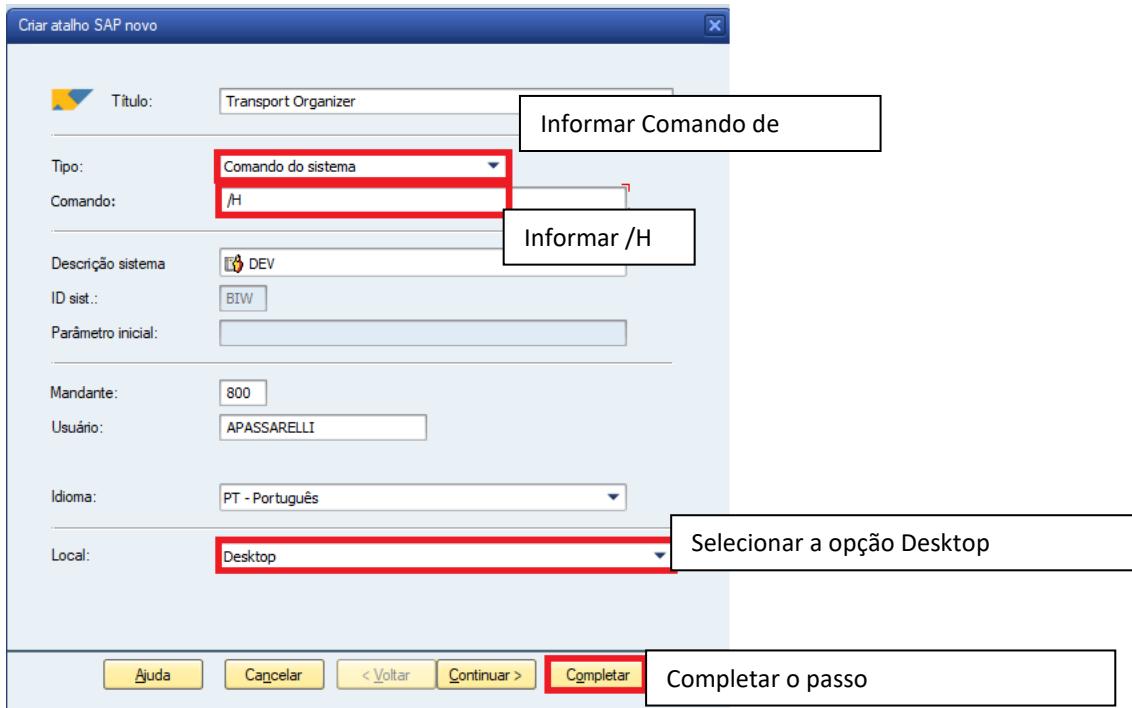


Vamos criar um /H físico em arquivo, para que ele possa ser arrastado em cima de uma janela popup e iniciar o Debug a partir desse ponto.

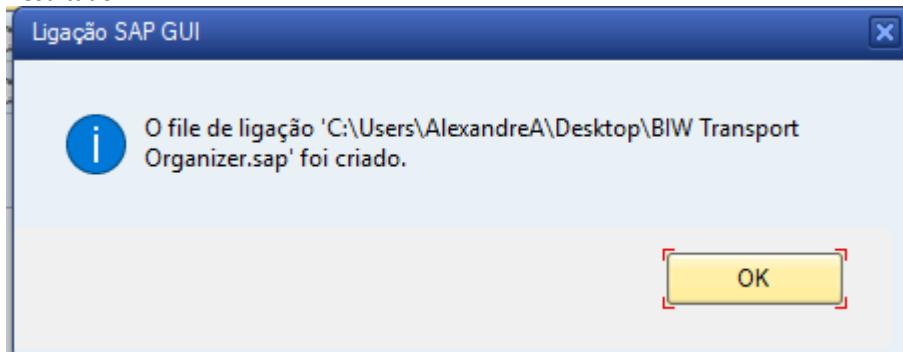
Para isso, na barra de ferramentas do SAP, encontre o botão (criar ligação) , será exibida uma tela conforme abaixo:



Preencha as informações conforme na imagem abaixo para conseguirmos criar o /H físico.

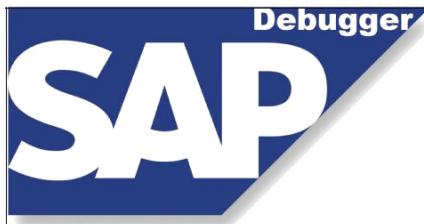


Resultado:

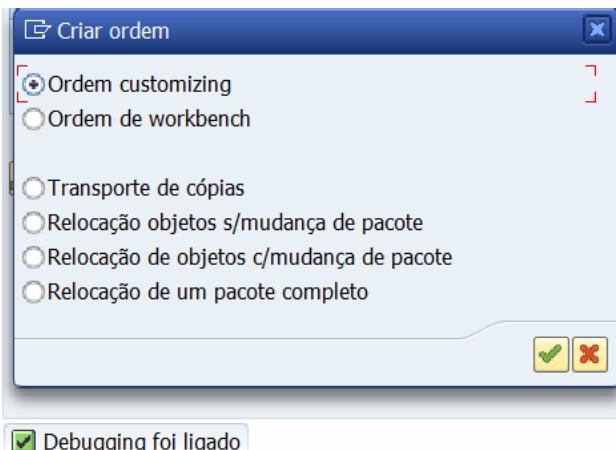


Arquivo na área de trabalho





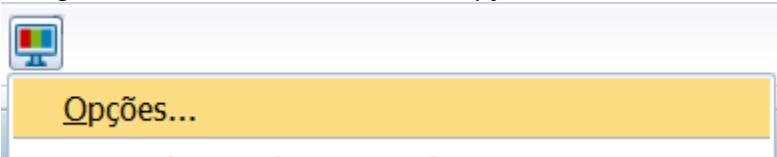
Para debugar um popup, basta selecionar o arquivo acima, manter pressionado o clique (para mover) e arrastar em cima da janela do SAP que você pretende debugar, isso funciona para qualquer tipo de Popup:



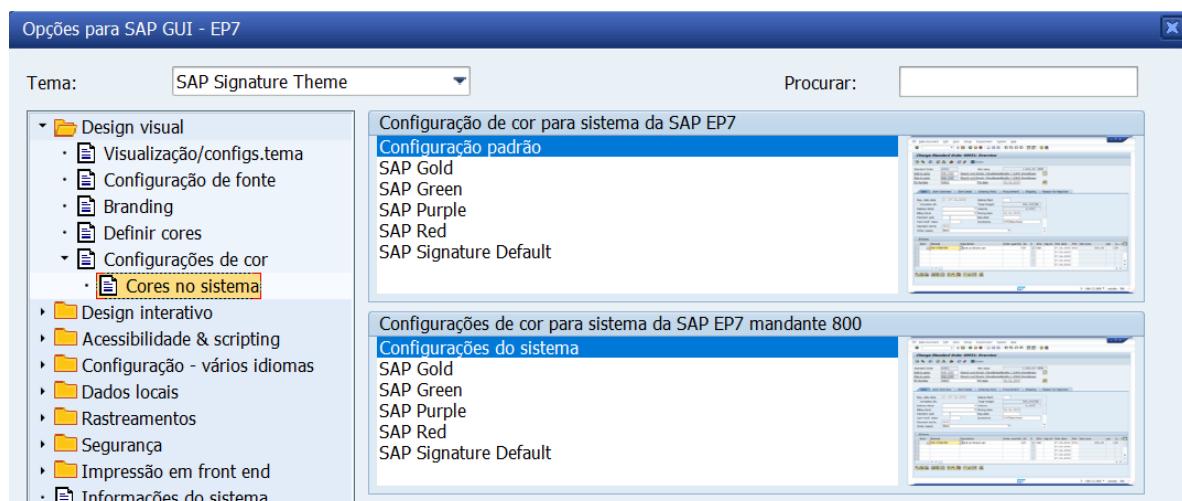
- Mudança de cores nos mandantes

O SAP permite que você tenha uma cor diferente para cada mandante que entrar, isso é essencial para separar DEV, QAS e PRD quando é preciso ficar nos três ao mesmo tempo, para evitar que alguma confusão aconteça no ambiente errado, podemos usar as cores definidas para cada mandante, conforme o tutorial abaixo:

Navegar até o botão abaixo e selecionar ``Opções``:



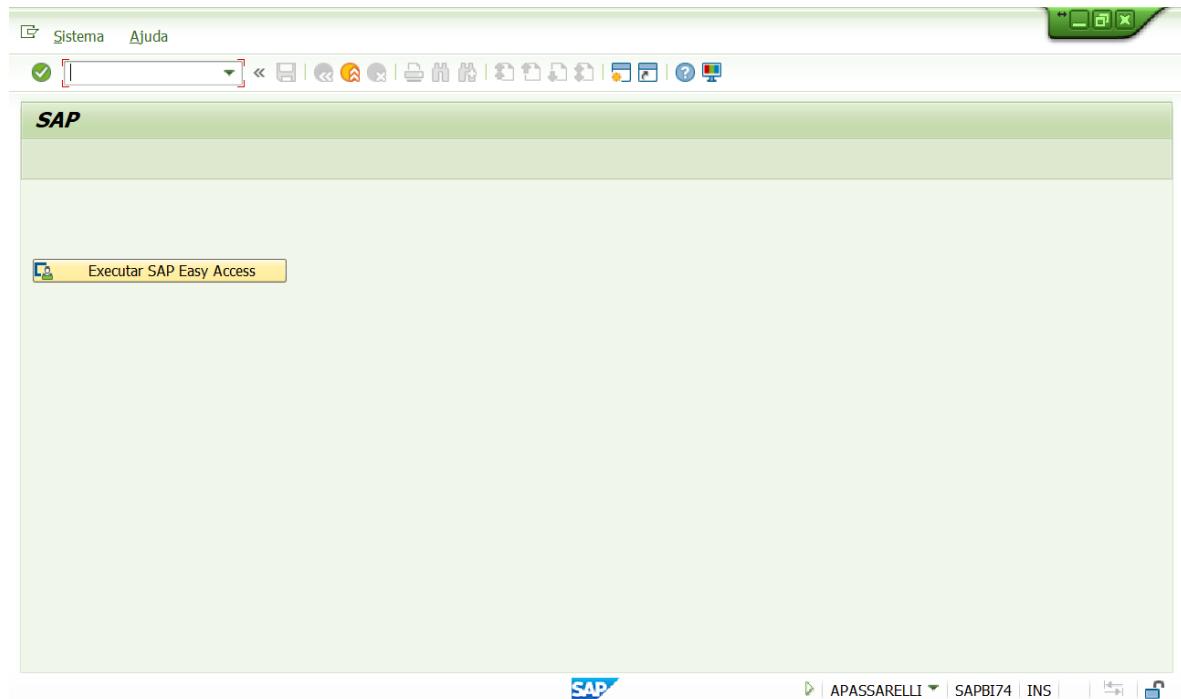
Veremos no decorrer deste tópico algumas informações adicionais, porém, para iniciarmos o entendimento das opções, vamos primeiramente trabalhar com as cores:





Para trocar a cor da sua entrada de Logon, basta estar logado em qual mandante você deseja atualizar e selecionar nas opções da imagem acima as cores que você quer ver em cada uma, lembrando que a mudança de cores só ocorre quando o mandante for diferente, caso sejam ambos mandantes 100, o sistema irá identificar a mesma cor.

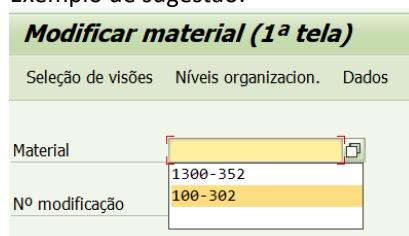
Exemplo (Cor: SAP Green)



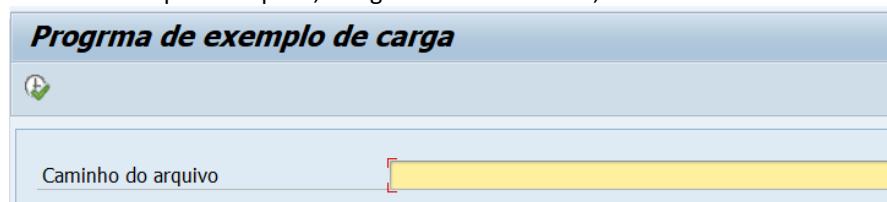
- Aumentar tamanho do campo de sugestão de variáveis

O SAP disponibiliza uma memória dos últimos valores digitados no campo daquele computador, isso nos ajuda a preencher mais rapidamente informações, principalmente na hora de testes, porém, essa opção não está aberta para todos os campos e o problema é que o SAP tem um tamanho Default do que ele pode gravar, vamos aumentar esse tamanho para o máximo, para que mesmo um caminho de arquivo fique armazenado nas sugestões.

Exemplo de sugestão:

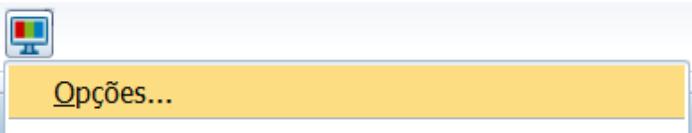


Já em um campo de arquivo, a sugestão não acontece, devido ao tamanho do campo:

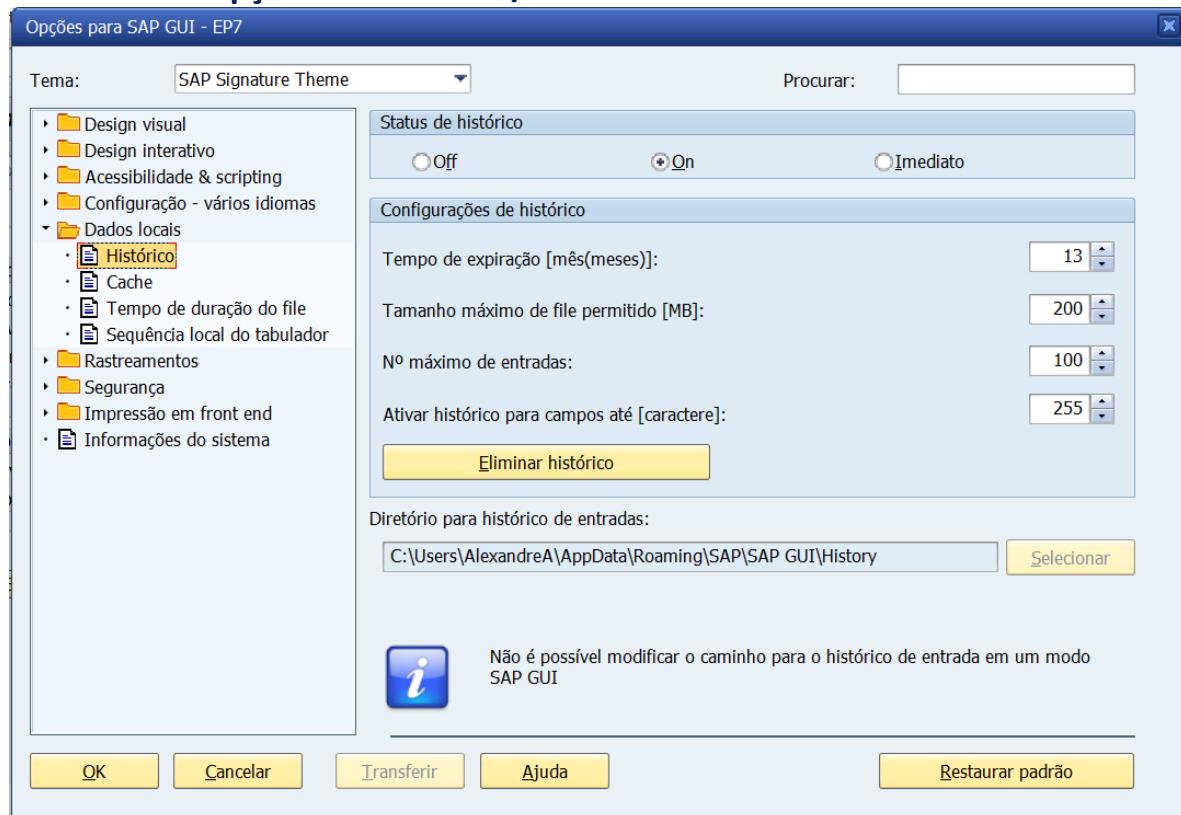




Ainda nas opções do menu abaixo:



Vamos até a opção Dados Locais/Histórico:



A opção abaixo marcada em vermelho é a que define qual o tamanho máximo que será guardado o histórico, no caso 255 é o limite máximo do SAP, deixando assim, caso você já tenha usado um arquivo em um programa de carga (como no exemplo acima), o SAP irá sugerir mesmo um caminho do windows, veja na segunda imagem abaixo o exemplo:

Configurações de histórico

Tempo de expiração [mês(meses)]:	13
Tamanho máximo de file permitido [MB]:	200
Nº máximo de entradas:	100
Ativar histórico para campos até [caractere]:	255

Exemplo:

Programa de exemplo de carga

Caminho do arquivo: C:\TEMP\teste.txt



- Mudar a visualização do editor ABAP

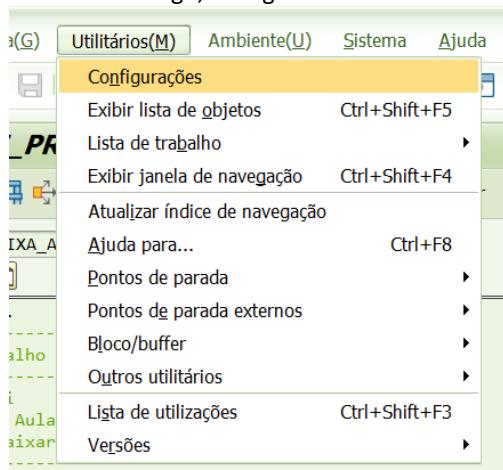
Abaixo podemos ver um código ABAP de exemplo sendo exibido no editor de código antigo, ou seja, alguns padrões de cores e visualização podem ficar mais difíceis de serem entendidos, é possível mudar para a versão mais visual, para isso, basta seguir o tutorial depois da imagem de exemplo abaixo:

Editor ABAP: Report Z_PROGRAMABAIXA_AULA2 exibir

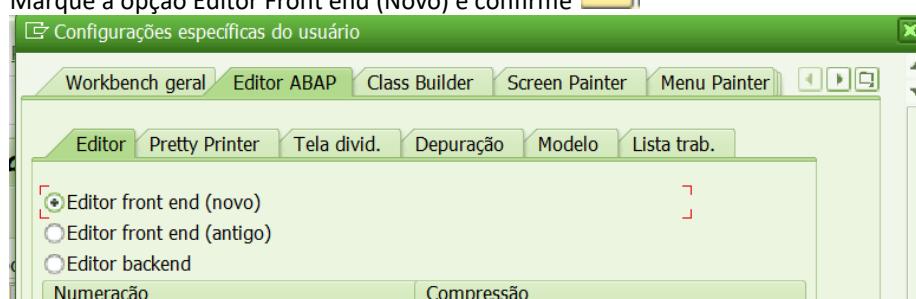
Report Z_PROGRAMABAIXA_AULA2 ativo

```
REPORT z_programabaixa_aula2.  
*-----*  
*          Cabeçalho do programa  
*-----*  
* Autor: Alexandre Passarelli  
* Módulo: Treinamento ABAP - Aula 2 (Curso Particular)  
* Descrição: Programa para baixar dados de uma tabela para Arquivo  
*-----*  
* Modificação  
*-----*  
* DD/MM/AAAA - Chamado XXXXXX - Request XXXXXXXX - Versão: 001  
*-----*  
*-----*  
* Tipos  
*-----*  
TYPES: BEGIN OF ty_arquivo,  
       line(100) TYPE c,  
       END OF ty_arquivo.  
*-----*  
* Tabelas Internas  
*-----*
```

Dentro do código, navegue até o menu abaixo:



Marque a opção Editor Front end (Novo) e confirme





Resultado:

Abaixo temos um exemplo do que pode ser alterado na visualização do código, normalmente as telas de edição da SE38 são brancas, com as fontes mais coloridas para indicar o que é sintaxe, o que é um objeto ou comentário no código, esse estilo abaixo se chama ABAP Dark, é possível instalar em seu ambiente, porém é necessário ter alguns arquivos, caso queira deixar seu editor de códigos da maneira abaixo, basta usar o link abaixo da imagem para baixar o tutorial e arquivos.

Editor ABAP: Report Z_PROGRAMABAIXA_AULA2 exibir

Report Z_PROGRAMABAIXA_AULA2 ativo

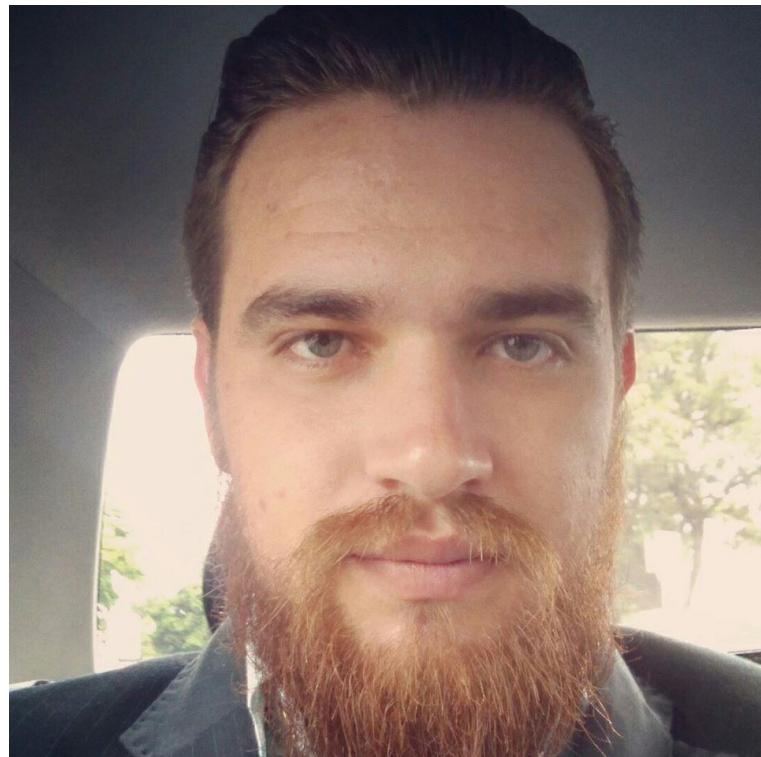
```
1 REPORT z_programabaixa_aula2.
2 *
3 *                               Cabeçalho do programa
4 *
5 * Autor: Alexandre Passarelli
6 * Módulo: Treinamento ABAP - Aula 2 (Curso Particular)
7 * Descrição: Programa para baixar dados de uma tabela para Arquivo
8 *
9 * Modificação
10 *
11 * DD/MM/AAAA - Chamado XXXXXX - Request XXXXXXXXX - Versão: 001
12 *
13 *
14 * Tipos
15 *
16 TYPES: BEGIN OF ty_arquivo,
17   line(100) TYPE c,
18   END OF ty_arquivo.
19 *
20 * Tabelas Internas
21 *
22 DATA: t_arquivo      TYPE TABLE OF ty_arquivo,
23       t_ztbazcliente TYPE TABLE OF ztbazcliente.
24 *
25 * Estruturas
26 *
27 DATA: wa_arquivo      TYPE ty_arquivo,
28       wa_ztbazcliente TYPE ztbazcliente.
```

[Link para Download do ABAP Dark \(Clique aqui\).](#)



Cursos de A a Z feitos pra você!

Material desenvolvido por: Alexandre Aparecido Passarelli



Seguem meus contatos:

Celular: **019-99199-5759**

Skype: **Xandelz**

E-mail: alexandrepassarelli@hotmail.com

LinkedIn: <https://www.linkedin.com/in/alexandre-aparecido-passarelli-aa64a529/>