

REPORT A FUNDO

Sumario:

Este tutorial demonstrar componente usado em um REPORT ABAP, para cada componente é descrito algumas de suas funcionalidades, eventos, sintaxes, que podem ocorrer. Não são demonstradas neste tutorial todas as formas de se utilizar os componentes, mais aqui estão as mais usadas é mais “Eficientes”.

Sobre o Autor:

Uderson Luis Fermino, formado em Ciências da Computação pela Faculdade de Pesquisa e Ensino IPEP, atua no mercado a 2 anos como desenvolvedor Java nas plataformas: (J2SE, J2EE e J2ME), com participação em grandes projetos envolvendo estas tecnologias. É consultor ABAP com experiências em REPORT, ALV (GRID, LIST, BLOCK, OO, TREE, HIERARQUICK), IDOC, ALE, ONLINE, SAPSCRIPT, SMARTFORM, NETWEAVER (JCO, BSP, WebDynpro).

Email:

uderson@gmail.com

“tudo posso naquele que me fortalece” (Filipenses 4:13).

Todos os programas ABAP, no fundo no fundo são programas feitos como os elementos do MODULE POOL, tendo como funções, e processos idênticos ao module pool, esta sessão será explicados alguns componentes visuais gerados em um REPORT, e alguns eventos que podem ocorrer durante o ciclo de vida de cada componente.

Todos os componentes que serão explicados estão na seguinte tela:

Exemplos de SCREENS

The screenshot displays a typical SAP ABAP report interface. At the top, there is a navigation bar with icons for back, forward, and search, along with a 'Cadastrar' button. Below this, the screen is divided into several sections:

- Desenja receber noticias:** A section with two radio buttons, 'Sim' (checked) and 'Não'.
- Botões Dentro do FRAME:** A section containing four buttons: 'Cadastrar', 'Consultar', 'Apagar', and 'Sair'.
- Material Com Intervalo:** A section with a text input field, a yellow range selector, and a search icon.
- Material Sem Intervalo e com titulo:** A section with a text input field and a search icon.
- Login:** A section with three input fields: 'Usuario', 'Senha' (masked with asterisks), and 'Tipo'.
- Sexo:** A section with two radio buttons: 'Feminino' (selected) and 'Masculino'.
- Tabbed Interface:** A section with five tabs: 'Dados Pessoais', 'Endereço', 'Escolaridade', 'Experiencias Profiss', and 'Idiomas'. The 'Dados Pessoais' tab is active, showing a list of fields: 'Nome' (with a checkmark), 'Data de Nascimento', 'Email', 'RG', and 'CPF', each with a corresponding input field.

Será explicado de cima para baixo onde cada componente terá uma ligação.

“tudo posso naquele que me fortalece” (Filipenses 4:13).

Barra de botões



Cada botão está associado a uma tabela chamada de SSCRFIELDS, existe outras tabelas que fazem a mesma funcionalidade do que esta tabela, porem será explicado está tabela para exemplificar. Está tabela contem a seguinte ordem:

UCOMM	CHAR70	Telas, código de função que acionou o PAI
FROM_TEXT	CHAR12	Texto com 12 dígitos
TO_TEXT	CHAR120	Texto com 12 dígitos
FUNCTXT_01	CHAR144	Tela de seleção: texto para botões
FUNCTXT_02	CHAR144	Tela de seleção: texto para botões
FUNCTXT_03	CHAR144	Tela de seleção: texto para botões
FUNCTXT_04	CHAR144	Tela de seleção: texto para botões
FUNCTXT_05	CHAR144	Tela de seleção: texto para botões
SEARCH_BTN	CHAR40	Telas de sel: botão para conjunto de valores

Por esta tabela somente é possível inserir 5 botões, os campos FUNCTXT_0N, é o campo responsável pela inserção de um ícone ou texto, está tabela deverá ser carregada no EVENTO INITIALIZATION, do REPORT, este evento é acionado quando existe uma instancia do REPORT, antes de criar as telas os fazer qualquer lógica, ou acionar qualquer outro EVENTO, este evento é acionado, ele é utilizado para justamente carregar dados que precisam ser carregados e mostrados antes de mostrar qualquer outros dados.

Para criar os botões usamos a seguinte sintaxe:

SELECTION-SCREEN: FUNCTION KEY N.

Onde N varia pelo numero do botão, usando uma numeração de 1 a te a quantidade requerida, como a tabela que está sendo exemplificada usa somente 5 botões então N varia de 1 a 5.

Ficando:

“tudo posso naquele que me fortalece” (Filipenses 4:13).

```
SELECTION-SCREEN: FUNCTION KEY 1,  
                  FUNCTION KEY 2,  
                  FUNCTION KEY 3,  
                  FUNCTION KEY 4,  
                  FUNCTION KEY 5.
```

No EVENTO INITIALIZATION, usamos a seguinte sintaxe:

```
INITIALIZATION.  
  SSCRFIELDS-FUNCTXT_01 = '@0De'.  
  SSCRFIELDS-FUNCTXT_02 = '@0Ee'.  
  SSCRFIELDS-FUNCTXT_03 = '@0De'.  
  SSCRFIELDS-FUNCTXT_04 = '@0Ee'.  
  SSCRFIELDS-FUNCTXT_05 = 'Cadastrar'.
```

Observe que estamos inserido ícones pelos código ASCII relativo do ABAP, e no quinto botão estamos inserindo um texto "CADASTRAR", até aqui já temos os botões na tela e com as LABELS que queremos, porem não adiantar termos botões se não fizermos os tratamentos.

Para tratar os eventos dos botões, basta utilizarem o campo SSCRFIELDS-UCOMM, que recebe o código do evento acionado pela SAIDA PAI (PROCESS AFTER INPUT), o evento PAI gravara os seguintes dados: "FC0N", onde N é relativo ao numero do botão acionado. O evento AT SELECTION-SCREEN é acionado quando um determinado componente é acionado (quando um evento de um componente é gerado), como existe diversos componentes na tela será necessário verificar o código de cada componente para para gerar alguma ação relativo a cada botão. Para exemplificar tempos o botão 5, gerado pela sintaxe:

```
SELECTION-SCREEN FUNCTION KEY 5,
```

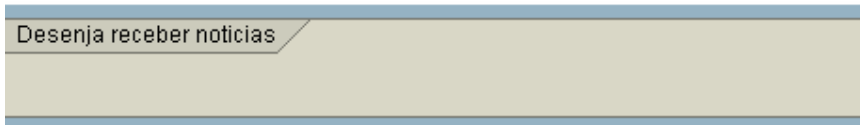
Quando este botão for acionado ele o PAI do REPORT, gerará o numero FC05, e fará a inserção deste valor no campo SSCRFIELDS-UCOMM, com este valor inserido, basta verificar se existe alguma evento FC05, e fazer alguma lógica relativo ao evento:

"tudo posso naquele que me fortalece" (Filipenses 4:13).

AT SELECTION-SCREEN.

```
CASE SSCRFIELDS-UCOMM.  
  WHEN 'FC05'.  
    PERFORM CADASTRAR_DADOS.  
  ENDCASE.
```

SELECTION-SCREEN BLOCK



Todos os componentes devem que desejamos que estejam organizados podem estar dentro de um bloco, um bloco pode conter FRAME, como a figura acima, pode conter um TITULO, ou não, mais é obrigatório que um SELECTION-SCREEN BLOCK contenha no mínimo um componente, na figura acima foi retirado os componente, por um editor de imagens somente para exemplificar.

Sintaxe básica:

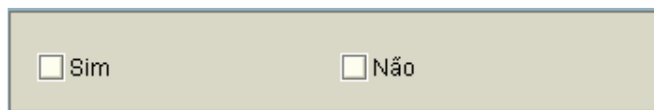
```
<Inicio do bloco>  
SELECTION-SCREEN BEGIN OF BLOCK nome_do_bloco.
```



```
SELECTION-SCREEN END OF BLOCK nome_do_bloco.  
<Fim do bloco>
```

Sintaxe com frame:

```
<Inicio do bloco>  
SELECTION-SCREEN BEGIN OF BLOCK nome_do_bloco WITH FRAME.
```



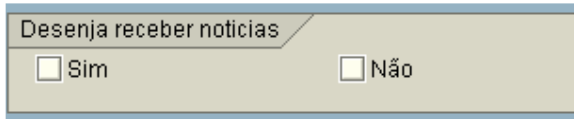
```
SELECTION-SCREEN END OF BLOCK nome_do_bloco.  
<Fim do bloco>
```

“tudo posso naquele que me fortalece” (Filipenses 4:13).

Sintaxe com frame e titulo

<Inicio do bloco>

```
SELECTION-SCREEN BEGIN OF BLOCK nome_do_bloco WITH FRAME
TITLE TEXT-100.
```



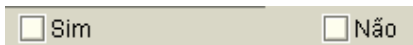
Desenja receber noticias

Sim Não

```
SELECTION-SCREEN END OF BLOCK nome_do_bloco.
```

<Fim do bloco>

CHECKBOX.

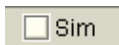


Sim Não

Os ckeckbox são components que são usados para multipla escolha, onde podemos escolher diversar opções, como os checkbox de qualquer outra linguagem. Os checkbos são componentes associados ao tipo de dados C, ele apenas contem um caracter onde X representa que o componente está checado e SPACE está deschechado.

Sintaxe básica:

```
PARAMETERS CB_SIM AS CHECKBOX.
```



Sim

Sintaxe que inicializa o componente acionado:

```
PARAMETERS C_SIM AS CHECKBOX DEFAULT 'X'.
```



Sim


Para verificar os CHECKBOX que foram acionados basta usar um simples IF, verificando o componente se está marcado ou não:

```
IF C_SIM = 'X'
WRITE / 'Você escolheu a opção SIM'.
ENDIF.
```

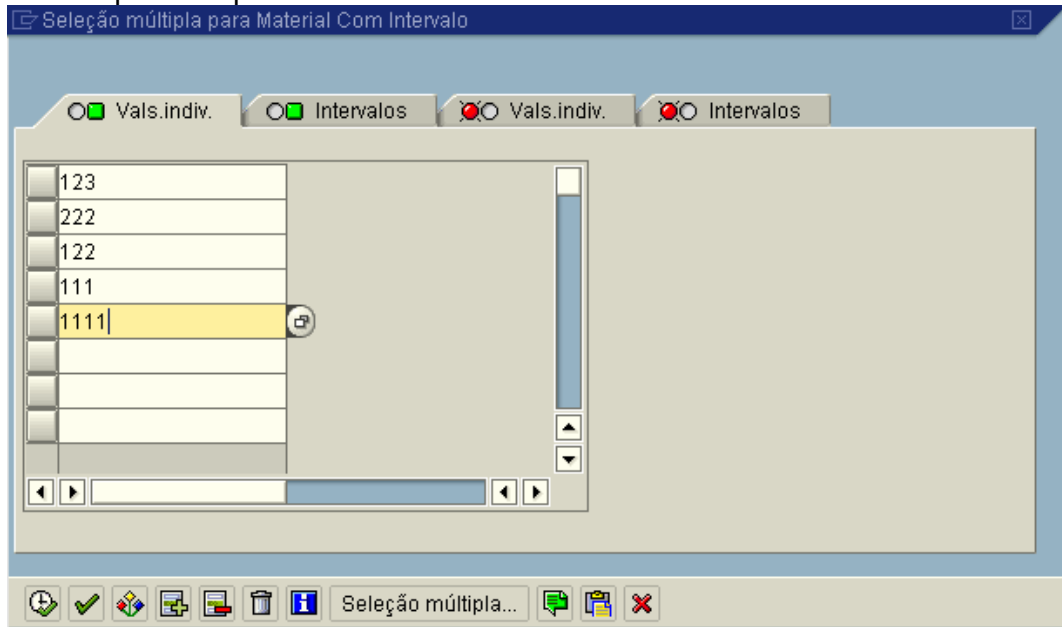
“tudo posso naquele que me fortalece” (Filipenses 4:13).

SELECT-OPTIONS :



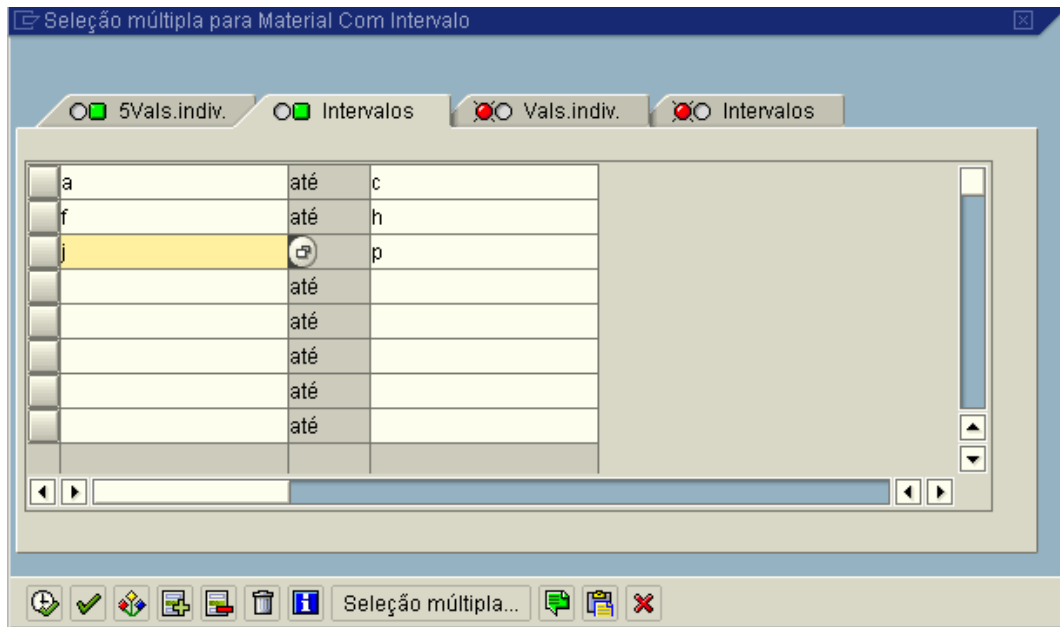
Como o nome já próprio diz, OPTIONS, de opções, significa, MAIS DE UM, este componente é usado para capturar valores entre intervalos, conhecidos como BETWEEN, onde capturamos um valor inicial e um valor final, este componente nos dá a condição de escolhermos determinados intervalos de valores, quando acionamos ação MATCHCODE: .

Onde é possível passar os valores INDIVIDUAIS:



Ou passar valores de até múltiplos:

“tudo posso naquele que me fortalece” (Filipenses 4:13).



Sintaxe basica:

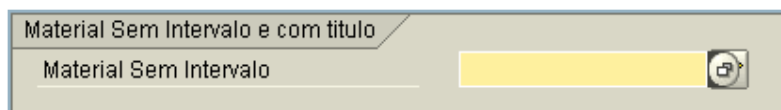
SELECT-OPTIONS nome_do_componente **FOR** tabela-campo

Ex.:

SELECT-OPTIONS S_MATNR **FOR** MARA-MATNR.

Este componente deve fazer referencia a uma tabela interna ou tabela Z ou tabela STANDARD conforme o exemplo.

É possível ter um SELECT-OPTIONS se um de ATÉ, usando a sintaxe NO INTERVALS, do SELECTION-SCREEN BLOCK, ficando:



Sitaxe:

SELECTION-SCREEN BEGIN OF BLOCK MSI WITH FRAME TITLE TEXT-MSI **NO INTERVALS**.

SELECT-OPTIONS S_MSI **FOR** MARA-MATNR.
SELECTION-SCREEN END OF BLOCK MSI.

“tudo posso naquele que me fortalece” (Filipenses 4:13).

Observe que o FRAME é menor, é do tamanho do SELECT-OPTIONS

PARAMETER:

Dados Pessoais	
Nome Meio	<input type="text"/>
Nome Final	<input type="text"/>

Os parameter são componentes de ENTRADA (INPUT), que podem ser utilizados, para entradas de dados de qualquer tipo de dado, diferente de outras linguagem que os componentes de entradas são sempre do TIPO (TEXT ou TEXTO), o parameter de ABAP, pode conter entrada direta de TEXTO (C), Números (I, N, P, D), Datas. Este componentes pode conter MASCARAS de entradas como de SENHAS (****), datas(__/__/__), Horas (__:__:__), uma variedade de opções.

Sintaxe básica:

PARAMETER nome_do_componente tipo_de_dado

O tipo de dado pode variar como todos os tipo de dados do ABAP (N, I, P, D, C...), cada um com as suas características, para o tipo de dados C é necessário passar o tamanho do componente, exemplo:

PARAMETERS: P_NOMEM(30) TYPE C,

Um parameter do tipo de dados de entrada C (Caractere) do tamanho 30.

Os tipos de dados podem ser referenciados como Elementos de dados, campos de tabelas.

“tudo posso naquele que me fortalece” (Filipenses 4:13).

Valor Obrigatório


Pode acontecer que queremos que uma determinada entrada seja obrigatória, basta usar a sintaxe:

```
PARAMETERS nome_do_componenter tipo_de_dado OBLIGATORY,
```

Exemplo:

```
PARAMETERS: P_NOME(30) TYPE C OBLIGATORY,
```

Caso o usuário tente fazer algum cadastro, e este componente estiver vazio, será gerado uma mensagem informando o usuário que ele deverá preencher o PARAMETER, que é obrigatório. A mensagem de AVISO(ERRO) aparece na barra de STATUS.

A screenshot of a red message box with a white 'X' icon in the top-left corner. The text inside the box reads "Preencher todos os campos obrigatórios".

Um parameter declarado do tipo de DADO "D, P, I", não aceita dados que não sejam numéricos, valores que ALFA, não são aceitos, o próprio parameter bloqueia a entrada.

Ex.:

```
PARAMETER: P_TI TYPE I,  
            P_TD TYPE D,  
            P_TP TYPE P.
```

Valor Default

Existem casos que são necessário inicializar alguns componentes com valores, para isto bastam utilizar a sintaxe DEFAULT, juntamente do parameter.

Ficando:

“tudo posso naquele que me fortalece” (Filipenses 4:13).

```
PARAMETER nome_do_paramenter TYPE tipo_de_dados DEFAULT  
'Valor_default.
```

Exe.:

```
PARAMETER P_VALOR_PADRAO TYPE I DEFAULT '2000',
```

Tamanho Visual

O componente pode ter uma tamanho Maximo de entrada que é definido pelo tipo de dado, que o parameter fará referencia, porem é possível aumentar o tamanho do componente visivelmente, este aumento ou diminuição não interfere no tamanho Maximo de dados de entrada, usar a sintaxe `VISIBLE LENGTH`, para definir o tamanho visível do camponente.

Sintaxe:

```
PARAMETER nome_do_paramenter TYPE/LIKE tipo_de_dado VISIBLE  
LENGTH N.
```

Onde N é o tamanho do componetente está sintaxe, somenete é aceita para tipos de dados: C, N, X ou P.

Exemplo:

```
PARAMETER P_RG(20) TYPE C VISIBLE LENGTH 3.
```

Observe que este parameter aceita 20 caracteres de entrada, porem o tamanho visual do paramenter é de 3 CARACTERES.



RG

SENSIVE CASE

“tudo posso naquele que me fortalece” (Filipenses 4:13).

Em ABAP, os caracteres não são tratados como SENSITIVE CASE, não importa de são maiúsculos ou minúsculos, é como no sistema operacional Windows, mais existe caso que pode ocorrer que é necessário diferenciar estes caracteres, por exemplo, suponha que iremos coletar uma arquivo TEXTO que está no disco rígido, para coletar existe diversas funções que fazem esta operação, porem todas exigem a entrada do caminho (endereço) que está arquivado o arquivo, se estivermos utilizando um Sistema Operacional Windows, não é necessário se preocupar com caminho do arquivo (se está em maiúsculo ou minúsculo), mais se estivermos trabalhando em um sistema operacional com o Kernel voltado ao UNIX, por exemplo um LINUX, teremos que nos preocupar quanto ao caminho se está em Maiúsculo ou Minúsculo cada caractere, por default o ABAP, converte tudo para maiúsculo, para ver basta entrar com algum dado em um parâmetro e dar um enter que veremos que ele converte tudo para maiúsculo, para que possamos ter os caracteres diferenciados, que não sejam convertidos pelo ABAP, basta usar a sintaxe: SENSITIVE CASE.

Sintaxe.:

```
PARAMETERS: P_NOME(30) TYPE C LOWER CASE.
```

Com esta sintaxe basta escrever em maiúsculo ou minúsculo e dar um enter que os caracteres serão conservados conforme o escrito.

Evento AT SELECTION-SCREEN ON VALUE-REQUEST FOR

Este evento é acionado quando o usuário clica sobre o ícone:



do parameter. Quando é clicado neste componente é acionado um evento este evento é chamado de função F4_REQUEST_HELP.

“tudo posso naquele que me fortalece” (Filipenses 4:13).

Sintaxe:

AT SELECTION-SCREEN ON VALUE-REQUEST FOR nome_do_parameter.

Exemplo: quando este evento for acionado, será mostrado uma tela:

```

/*
* Tela que terá diverso parameter um dele (P_NOME)
*acionará uma evento
*/

SELECTION-SCREEN BEGIN OF SCREEN 100 AS SUBSCREEN.
SELECTION-SCREEN BEGIN OF BLOCK B1 WITH FRAME.
PARAMETERS: P_NOME(30)   TYPE C OBLIGATORY,
             P_DTNASC    TYPE D,
             P_EMAIL(100) TYPE C,
             P_RG(20)    TYPE C,
             P_TTT       TYPE I,
             P_CPF(20)   TYPE C.
SELECTION-SCREEN END OF BLOCK B1.
SELECTION-SCREEN END OF SCREEN 100.

/*
* Evento do Parameter P_NOME
* este evento chama o perform ESCOLHAS.
*/

AT SELECTION-SCREEN ON VALUE-REQUEST FOR P_NOME.
  PERFORM ESCOLHAS.

/*
* Perform Escolha
*
*/

*&-----*
*&      Form  ESCOLHAS
*&-----*
*       Performe que captura dados de entrada da relativo ao numero
*       de materiais.
*&-----*
*&-----*

FORM ESCOLHAS .
  DATA: V_NUMERO LIKE SVAL OCCURS 0 WITH HEADER LINE.

  DATA: BEGIN OF T_TESTE OCCURS 0,
         FI TYPE C,
         FS TYPE C,
         END OF T_TESTE.

  V_NUMERO-TABNAME = 'MARA'.

```

“tudo posso naquele que me fortalece” (Filipenses 4:13).

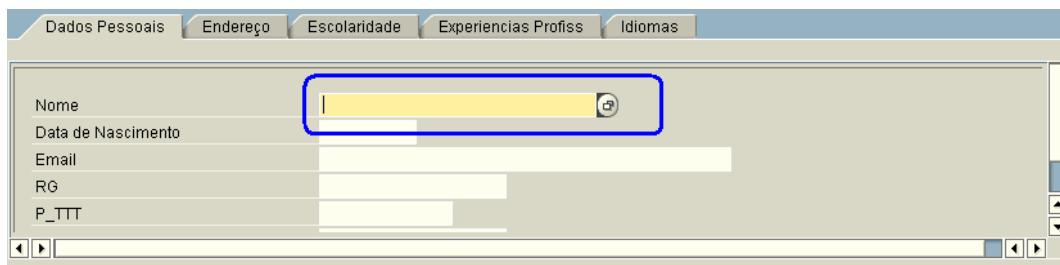
```

V_NUMERO-FIELDNAME = 'MATNR'.
APPEND V_NUMERO.

CALL FUNCTION 'POPUP_GET_VALUES'
  EXPORTING
    POPUP_TITLE           = 'Valores da mara'
    START_COLUMN          = '5'
    START_ROW             = '5'
  TABLES
    FIELDS                = V_NUMERO.
  EXCEPTIONS
    ERROR_IN_FIELDS      = 1
    OTHERS                = 2
ENDFORM.                " ESCOLHAS

```

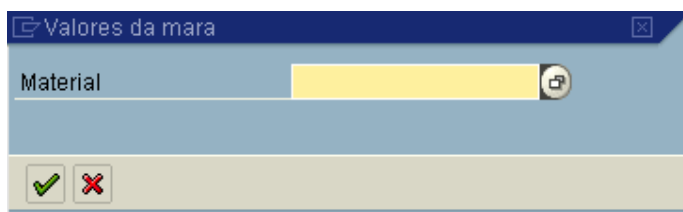
Exemplo visual:



Quando for clicado neste componente (F4_REQUEST_HELP), será gerado o evento:

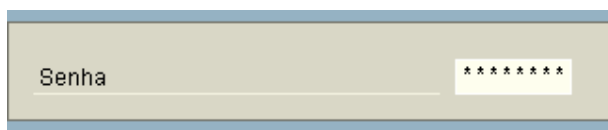
```
AT SELECTION-SCREEN ON VALUE-REQUEST FOR P_NOME.
```

Dentro do perform existe uma função que chamará a tela contendo um select-options de dados da Tabela de materiais MARA.



Mascara no Parameter

Criando uma mascara na caixa de entrada INPUT para senha, com asteristico, para o componente PARAMETER:



“tudo posso naquele que me fortalece” (Filipenses 4:13).

No evento AT SELECTION-SCREEN OUTPUT, que é um evento acionado todas as vezes que é necessário criar (apresentar a tela), todas as vezes que a tela for desenhada este evento será acionado, mesmo que não exista fluxo lógico para ser realizado, dentro do evento, dar um LOOP AT SCREEN, que irar capturar todos os componentes SCREEN (da tela) e andarà um a uma, para identificar o componente atual basta verificar o nome do componente com a sintaxe:

```
SCREEN-NAME = 'Nome do componente'.
```

Caso o componente seja o requerido, o componente terá uma das opções chamadas INVISIBLE que assume 0, para mostrar os valores digitados e 1, para não ocultar os valores digitados. No exemplo usaremos um parameter referente a senha, que é um dado que deve digitado e não pode ser visto por alguma usuário que não seja o usuário que está fazendo a entrada dos dados.

Criando o parameter:

```
SELECTION-SCREEN BEGIN OF BLOCK M1 WITH FRAME.  
  PARAMETERS P_PWD(8) TYPE C.  
SELECTION-SCREEN END OF BLOCK M1.
```

No evento AT SELECTION-SCREEN OUTPUT, dar um loop no SCREEN (TELA) é verificar se o ponteiro está apontando para o PARAMETER P_PWD, caso esteja SETAR o parâmetro INVISIBLE para 1., após alterar o status dar um MODIFY SCREEN, para realizar as modificações

```
AT SELECTION-SCREEN OUTPUT.
```

```
  LOOP AT SCREEN.  
    IF SCREEN-NAME = 'P_PWD'.  
      SCREEN-INVISIBLE = '1'.  
      MODIFY SCREEN.  
    ENDIF.  
  ENDLOOP.
```

Para aprendizado EXISTEM os seguintes parâmetros em um SCREEN:

“tudo posso naquele que me fortalece” (Filipenses 4:13).

INPUT que desabilita o componente: não permite entrada, quando estiver marcado com 1, e fica desabilitado quando estiver marcado com 0, por default todos componentes são inicializados como 1, para serem editados



Exemplo para desabilitar:

```
LOOP AT SCREEN.
  IF SCREEN-NAME = 'P_TIPO'.
    SCREEN-INPUT = '0'.
    MODIFY SCREEN.
  ENDIF.
ENDLOOP.
```

ACTIVE este parâmetro retira o componente da tela quando é marcado com 0, por default ele é marcado com 1, para ser mostrado na tela.

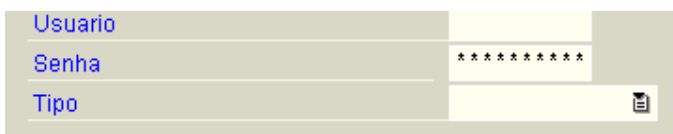
Exemplo para ocultar o componente:

```
LOOP AT SCREEN.
  IF SCREEN-NAME = 'P_USER'.
    SCREEN-ACTIVE = '0'.
    MODIFY SCREEN.
  ENDIF.
ENDLOOP.
```



Observe que o Usuário não está aparecendo o componente PARAMETER.

INTENSIFIED, este parâmetro, faz com que a cor do componente mude para cor AZUL, ficando por exemplo:



Ex.:

“tudo posso naquele que me fortalece” (Filipenses 4:13).

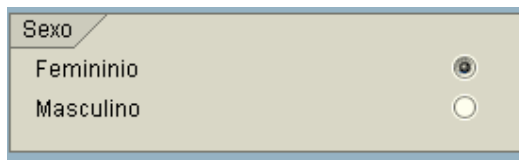
```

AT SELECTION-SCREEN OUTPUT.
  LOOP AT SCREEN.
    IF SCREEN-GROUP1 = 'USE'.
      SCREEN-INTENSIFIED = '1'.
      MODIFY SCREEN.
    ENDIF.

```

Neste exemplo estamos modificando o grupo USE que foi definido nos componentes como o comando “componente MODIFY ID xyz”.

RADIOBUTTON:



Diferente do CHECKBOX o RADIOBUTTON, é um componente de escolha única, como no exemplo acima uma pessoa somente (teoricamente sabemos disso, mais como o mundo esta evoluído, existe pessoas que acham que tem os dois sexos), pode conter somente um sexo ou FEMININO ou MASCULINO, como em qualquer outra linguagem este componente é associado a alguma marcação, se estivéssemos utilizando HTML, ter componentes como os mesmos nome já deixaria esta amarração feita, no ABAP usa-se o conceito de GRUPO, todos os rádios que estiverem em um grupo, somente um poderá ser acionado.

Sintaxe:

```

PARAMETER      nome_do_componente      RADIOBUTTON      GROUP
nome_do_grupo

```

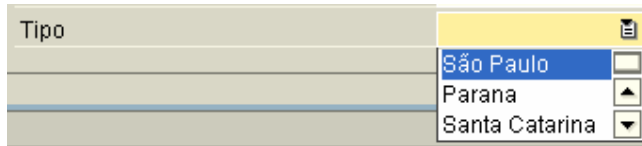
Ex.:

```

SELECTION-SCREEN BEGIN OF BLOCK BZ1 WITH FRAME TITLE TEXT-
300.
PARAMETERS: R_FEN RADIOBUTTON GROUP RADI,
             P_MAS RADIOBUTTON GROUP RADI.
SELECTION-SCREEN END OF BLOCK BZ1.

```

“tudo posso naquele que me fortalece” (Filipenses 4:13).

COMBOBOX:

O componente COMBOBOX é um componente usado para facilitar, a captura de dados do usuário, ele é carregado automaticamente como os valores predefinidos, para exemplificar, temos um cadastro de funcionários este funcionário, contem dados pessoais como endereços, neste endereço terá a cidade e o estado que ele mora, para que o usuário tenha que digitar o nome do estado, podemos deixar um combo carregado como os nome dos estados brasileiros.

Para carregar um combobox é necessário algumas regras:

1º Declarar uma WorkArea e uma estrutura do tipo VRM_VALUES:

```
DATA: IT_LIST TYPE VRM_VALUES,  
      LIST_VALUE TYPE VRM_VALUE.
```

2º Declarar o combobox:

```
SELECTION-SCREEN BEGIN OF BLOCK B2 WITH FRAME.  
      PARAMETERS P_LIST(20) TYPE C AS LISTBOX  
      VISIBLE LENGTH 15.  
SELECTION-SCREEN END OF BLOCK B2.
```

3º Carregar o combobox:

É necessário carregar o combobox antes de ser mostrado na tela, para carregar antes de mostrar, é necessário utilizar o EVENTO INITIALIZATION, como já explicado este evento é acionado antes de qualquer outro evento, antes de qualquer ação.

“tudo posso naquele que me fortalece” (Filipenses 4:13).

```
INITIALIZATION.  
  PERFORM CARREGA_COMBO.
```

Observe que passamos um o nome de performs para ser chamado, este perform conterà a lógica de carregamento para carregar os combobox.

4º Lógica do PERFORM

Aqui estará à lógica, para carregar os combobox.

```
FORM CARREGA_COMBO .  
  
  LIST_VALUE-KEY = '01'.  
  LIST_VALUE-TEXT = 'São Paulo'.  
  APPEND LIST_VALUE TO IT_LIST.  
  
  LIST_VALUE-KEY = '02'.  
  LIST_VALUE-TEXT = 'Parana'.  
  APPEND LIST_VALUE TO IT_LIST.  
  
  LIST_VALUE-KEY = '03'.  
  LIST_VALUE-TEXT = 'Santa Catarina'.  
  APPEND LIST_VALUE TO IT_LIST.  
  
  LIST_VALUE-KEY = '04'.  
  LIST_VALUE-TEXT = 'Bahia'.  
  APPEND LIST_VALUE TO IT_LIST.  
  
  LIST_VALUE-KEY = '05'.  
  LIST_VALUE-TEXT = 'Pernambuco'.  
  APPEND LIST_VALUE TO IT_LIST.  
  
  LIST_VALUE-KEY = '06'.  
  LIST_VALUE-TEXT = 'Amazonas'.  
  APPEND LIST_VALUE TO IT_LIST.  
  
  LIST_VALUE-KEY = '07'.  
  LIST_VALUE-TEXT = 'Piaui'.  
  APPEND LIST_VALUE TO IT_LIST.  
  
  LIST_VALUE-KEY = '08'.  
  LIST_VALUE-TEXT = 'Maranhão'.  
  APPEND LIST_VALUE TO IT_LIST.  
  
  LIST_VALUE-KEY = '09'.
```

“tudo posso naquele que me fortalece” (Filipenses 4:13).

```
LIST_VALUE-TEXT = 'Para'.  
APPEND LIST_VALUE TO IT_LIST.  
  
LIST_VALUE-KEY = '10'.  
LIST_VALUE-TEXT = 'Tocantins'.  
APPEND LIST_VALUE TO IT_LIST.  
  
CALL FUNCTION 'VRM_SET_VALUES'  
  EXPORTING  
    ID      = 'P_LIST'  
    VALUES = IT_LIST.
```

Primeiro preenchemos a WORKAREA e depois anexamos a WORKAREA para a estrutura, tanto WORKAREA como a ESTRUTURA, contem dois campos:

KEY = Ordem que aparecera o valor.

TEXT = Texto que aparecerá no componente

Após ter carregado a ESTRUTURA com todos os valores é necessário chamar a função VRM_SET_VALUES, que recebe o ID do parameter (componente - combobox) e a estrutura contendo os valores da ESTRUTURA.

Para capturar as informações basta capturar o dados do parameter combobox,

Ex.: write p_combobox.

Desta forma o componente apenas passa o índice da chave (KEY), que foi escolhido para captura o valor é possível fazer uma lógica que por exemplo entro com um índice e retorna uma string.

Exemplo:

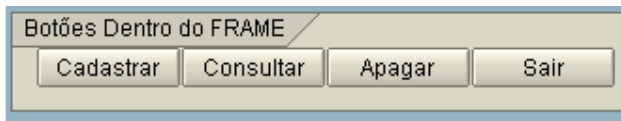
“tudo posso naquele que me fortalece” (Filipenses 4:13).

```
CASE key.  
  when '01'  
    imprime 'este é o campo 01'  
  
  when '02'  
    imprime 'este é o campo 02'  
ENDCASE.
```

Para acompanhar os dados de testes que estamos fazendo, basta criar um perform que entre com o índice do combobox escolhido e ele retorne uma string contendo o nome do estado associado ao índice.

```
DATA ESTADO(20) TYPE C.  
  
ESTADO = P_LIST.  
PERFORM ESTADO CHANGING ESTADO.  
WRITE / ' Estado:', ESTADO.  
  
FORM ESTADO CHANGING P_ESTADO.  
  
CASE P_ESTADO.  
  WHEN '01'.  
    P_ESTADO = 'São Paulo'.  
  WHEN '02'.  
    P_ESTADO = 'Parana'.  
  WHEN '03'.  
    P_ESTADO = 'Santa Catarina'.  
  WHEN '04'.  
    P_ESTADO = 'Bahia'.  
  WHEN '05'.  
    P_ESTADO = 'Pernambuco'.  
  WHEN '06'.  
    P_ESTADO = 'Amazonas'.  
  WHEN '07'.  
    P_ESTADO = 'Piaui'.  
  WHEN '08'.  
    P_ESTADO = 'Maranhão'.  
  WHEN '09'.  
    P_ESTADO = 'Para'.  
  ENDCASE.  
ENDFORM.          " ESTADO
```

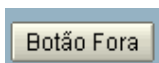
“tudo posso naquele que me fortalece” (Filipenses 4:13).

BOTÃO DENTRO DE UM SCREEN:

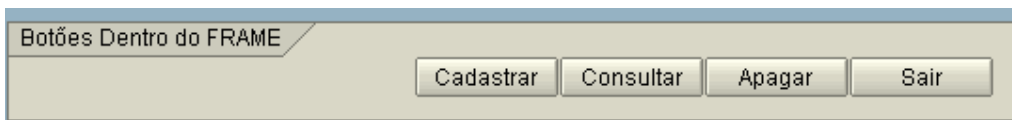
Os botões servem para criar interação e ações com o usuário, neste caso, temos quatro que fazem interação com o usuário, uma quando acionado faz o cadastro dos dados da tela, o outro faz uma determinada consulta, um outro que apaga um determinado registro, e um que sai da tela. Este são apenas para exemplificar, podemos inserir botões em SCREEN BLOCK, SCREEN BLOCK com FRAME, dentro de uma Subscreen, dentro de uma WINDOWS.

Ex. :

Fora de um FRAME:



Dentro de um FRAME:



Sintaxe Basica:

SELECTION-SCREEN BEGIN OF LINE.

SELECTION-SCREEN PUSHBUTTON posição_inicial(tamanho do botão)
nome_do_botão **USER-COMMAND** nome_do
evento_usado_no_user_commnad.

SELECTION-SCREEN END OF LINE.

SELECTION-SCREEN BEGIN OF LINE, faz com o botão fique em uma linha. Para que os botões estejam um ao lado do outro deve este dentro da sintaxe:

“tudo posso naquele que me fortalece” (Filipenses 4:13).

SELECTION-SCREEN BEGIN OF LINE.**SELECTION-SCREEN END OF LINE.**

SELECTION-SCREEN PUSHBUTTON é o comando para a criação do botão, é necessário passar uma posição inicial (caso não seja passada será iniciado na posição 1), esta posição varia de até 100(podendo varia de maquina para maquina), entre parenteses é determinado o tamanho que o botão terá (este tamanho é relativo Caracteres) após é especificado o nome do botão, e após este USER-COMMAND, é identificado o nome do comando, para ser usado no tratamento de evento do botão.

Ex.:

SELECTION-SCREEN BEGIN OF LINE.

```
SELECTION-SCREEN PUSHBUTTON 30(10) BT1 USER-COMMAND BT01.  
SELECTION-SCREEN END OF LINE.
```

Botão que inicia na posição 30 tem o tamanho de 10 caracteres, tem o nome de BT1, seu comando de EVENTO USER-COMMAND é BT01.

Para carregar o rotulo (label) que irá aparecer no BOTÃO, é necessário como todos os outros componentes ser inicializado no EVENTO INITIALIZATION, ficando

INITIALIZATION.

```
nome_do_botão = 'rotulo'.
```

Ex.:

INITIALIZATION.

```
BT1 = 'Cadastrar'.
```

Para verificar o evento do botão, quando o botão é acionado, usa-se o EVENTO AT SELECTION-SCREEN, verificando em uma estrutura de

“tudo posso naquele que me fortalece” (Filipenses 4:13).

condição(geralmente um CASE), qual o botão acionado, verificando pelo código que foi passado na criação, no exemplo BT01.

Ficando:

AT SELECTION-SCREEN.

```
CASE SSCRFIELDS-UCOMM.  
  WHEN 'BT01' .  
    PERFORM CADASTRAR_DADOS.  
  ENDCASE.
```

Observe que estamos usando a mesma tabela SSCRFIELDS, que usamos para os botões da barra de botões, onde o campo UCOMM desta tabela é preenchido todas as vezes que é acionado um evento é o PAI(Process After Input) da tela é processado

TABSTRIP:

The screenshot shows a SAP Tabstrip with five tabs: 'Dados Pessoais', 'Endereço', 'Escolaridade', 'Experiencias Profiss', and 'Idiomas'. The 'Dados Pessoais' tab is selected and highlighted with a red box. Below the tabs, a form contains the following fields:

Nome	<input type="text"/>
Data de Nascimento	<input type="text"/>
Email	<input type="text"/>
RG	<input type="text"/>
CPF	<input type="text"/>

The screenshot shows a SAP Tabstrip with five tabs: 'Dados Pessoais', 'Endereço', 'Escolaridade', 'Experiencias Profiss', and 'Idiomas'. The 'Endereço' tab is selected and highlighted with a red box. Below the tabs, a form contains the following fields:

Rua	<input type="text"/>
CEP	<input type="text"/>
Numero	<input type="text"/>
Cidade	<input type="text"/>
Tipo	<input type="text"/>

The screenshot shows a SAP Tabstrip with five tabs: 'Dados Pessoais', 'Endereço', 'Escolaridade', 'Experiencias Profiss', and 'Idiomas'. The 'Escolaridade' tab is selected and highlighted with a red box. Below the tabs, a form contains the following fields:

Gratul	<input type="text"/>
Ano Concluinte	<input type="text"/>

“tudo posso naquele que me fortalece” (Filipenses 4:13).

The image shows two screenshots of a web application interface. The first screenshot displays a tabbed interface with five tabs: 'Dados Pessoais', 'Endereço', 'Escolaridade', 'Experiencias Profiss', and 'Idiomas'. The 'Experiencias Profiss' tab is active and contains four input fields: 'Empresa', 'Função', 'Data de Adminssão', and 'Data de Demissão'. The second screenshot shows the same interface with the 'Idiomas' tab active, containing two input fields: 'Idioma' and 'Nivel'. Both tabs are highlighted with a red border in the original image.

O TABSTRIP, é um tabulador que junta diversos SCREEN em UM SÓ, podendo conter diversos tipo de componentes.

Neste exemplo criaremos SUBSCREENS e JUNTA-LAS no TABSTRIP

Para criar uma SUBSCREEN, basta usar a sintaxe:

<Inicio da subscreen>

SELECTION-SCREEN BEGIN OF SCREEN numero_da_sub_screen **AS**
SUBSCREEN.

SELECTION-SCREEN END OF SCREEN numero_da_sub_screen.

<Fim da subscreen>

Criando 5 (Cinco subscreen) conte tendo FRAME, BLOCK, PARAMER(S).

```
SELECTION-SCREEN BEGIN OF SCREEN 100 AS SUBSCREEN.
  SELECTION-SCREEN BEGIN OF BLOCK B1 WITH FRAME.
  PARAMETERS: P_NOME(30)   TYPE C OBLIGATORY,
                P_DTNASC(10) TYPE C,
                P_EMAIL(100) TYPE C,
                P_RG(20)     TYPE C,
                P_CPF(20)    TYPE C.
  SELECTION-SCREEN END OF BLOCK B1.
SELECTION-SCREEN END OF SCREEN 100.
```

“tudo posso naquele que me fortalece” (Filipenses 4:13).

```
SELECTION-SCREEN BEGIN OF SCREEN 200 AS SUBSCREEN.  
  SELECTION-SCREEN BEGIN OF BLOCK B2 WITH FRAME.  
  PARAMETERS: P_RUA(30)    TYPE C,  
              P_CEP(30)    TYPE C,  
              P_NUMERO(5)   TYPE C,  
              P_CIDADE(30) TYPE C,  
              P_LIST(20)   TYPE C AS LISTBOX VISIBLE  
              LENGTH 15.  
  SELECTION-SCREEN END OF BLOCK B2.  
SELECTION-SCREEN END OF SCREEN 200.
```

```
SELECTION-SCREEN BEGIN OF SCREEN 300 AS SUBSCREEN.  
  SELECTION-SCREEN BEGIN OF BLOCK B3 WITH FRAME.  
  PARAMETERS: P_GRAL(10) TYPE C,  
              P_ANO_C(10) TYPE C.  
  SELECTION-SCREEN END OF BLOCK B3.  
SELECTION-SCREEN END OF SCREEN 300.
```

```
SELECTION-SCREEN BEGIN OF SCREEN 400 AS SUBSCREEN.  
  SELECTION-SCREEN BEGIN OF BLOCK B4 WITH FRAME.  
  PARAMETERS: P_EMP(20) TYPE C,  
              P_FUN(20) TYPE C,  
              P_DTA(10) TYPE C,  
              P_DTD(10) TYPE C.  
  SELECTION-SCREEN END OF BLOCK B4.  
SELECTION-SCREEN END OF SCREEN 400.
```

```
SELECTION-SCREEN BEGIN OF SCREEN 500 AS SUBSCREEN.  
  SELECTION-SCREEN BEGIN OF BLOCK B5 WITH FRAME.  
  PARAMETERS: P_IDIOMA(20) TYPE C AS LISTBOX VISIBLE  
              LENGTH 15,  
              P_NIVEL(20)  TYPE C AS LISTBOX VISIBLE  
              LENGTH 15.  
  SELECTION-SCREEN END OF BLOCK B5.  
SELECTION-SCREEN END OF SCREEN 500.
```

Criando o TABSTRIP, e jundando as SUBSCREENS:

“tudo posso naquele que me fortalece” (Filipenses 4:13).

SELECTION-SCREEN:

```

      BEGIN OF TABBED BLOCK MYTAB FOR 7 LINES,
            TAB (20) BUTTON1 USER-COMMAND PUSH1 DEFAULT
SCREEN 100,
            TAB (20) BUTTON2 USER-COMMAND PUSH2 DEFAULT
SCREEN 200,
            TAB (20) BUTTON3 USER-COMMAND PUSH3 DEFAULT
SCREEN 300,
            TAB (20) BUTTON4 USER-COMMAND PUSH4 DEFAULT
SCREEN 400,
            TAB (20) BUTTON5 USER-COMMAND PUSH5 DEFAULT
SCREEN 500,
      END OF BLOCK MYTAB.

```

Cria um TAB (ABA) contendo o tamanho 20 (20 caracteres, mais pode variar, quanto mais caracteres maior será a ABA), **BUTTONN** nome do TAB, **USER-COMMAND PUSHN** (comando a ser tratado quando quiser fazer tratamento de eventos) chamando a **SUBSCREEN NNN** com default.

```

TAB      tamango_em_caracter      nome_do_tab      USER-COMMAND
user_command DEFAULT SCREEN numero_da_screen.

```

Sintaxe básica para criar um TABSTRIP

SELECTION-SCREEN:

```

      BEGIN OF TABBED BLOCK nome_do_tabstrip FOR
numero_de_linha LINES,

```

Insere os subscreen

```

      END OF BLOCK nome_do_tabstrip.

```

O Numero de LINHA é a quantidade de linha que terá o TABSTRIP:

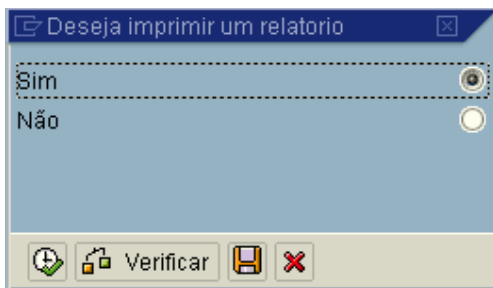
Podendo variar de 0 até N, quanto mais linhas maior é o TABSTRIP.

“tudo posso naquele que me fortalece” (Filipenses 4:13).

Para que as abas inicialize com os títulos será necessário informar no EVENTO INITIALIZATION, o nome do TAB e o texto que será inicializado:

BUTTON1 = 'Dados Pessoais'.
BUTTON2 = 'Endereço'.
BUTTON3 = 'Escolaridade'.
BUTTON4 = 'Experiências Profissionais'.
BUTTON5 = 'Idiomas'.

WINDOWS:



WINDOWS é uma SCREEN, que é chamada separada da tela do SCREEN, ela como o nome próprio já diz, ela é uma janela, que HERDA as mesmas características de uma SCREEN, como as SUBSCREEN, é possível inserir qualquer componente dentro desta WINDOWS(PARAMER, RADIOBUTTON, PUSHBUTTON, CCHECKBOX, SCREEN BLOCK (com ou sem FRAME, SELECT-OPTIONS), os dados podem ser capturados como sendo os dados da SCREEN, não existindo mágica, podendo atribuir o valor com componente em uma variável, utilizando em condições, imprimindo, gravando em BD.

Sintaxe Básica:

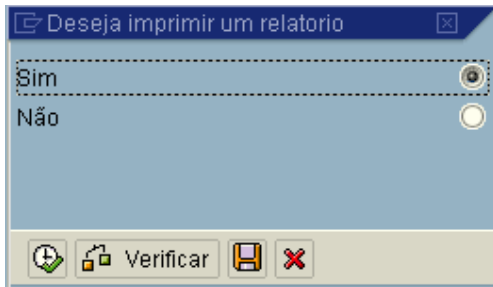
```
SELECTION-SCREEN BEGIN OF SCREEN numero_da_screen AS WINDOW  
TITLE txt_titulo.
```

<aqui insere os componentes>

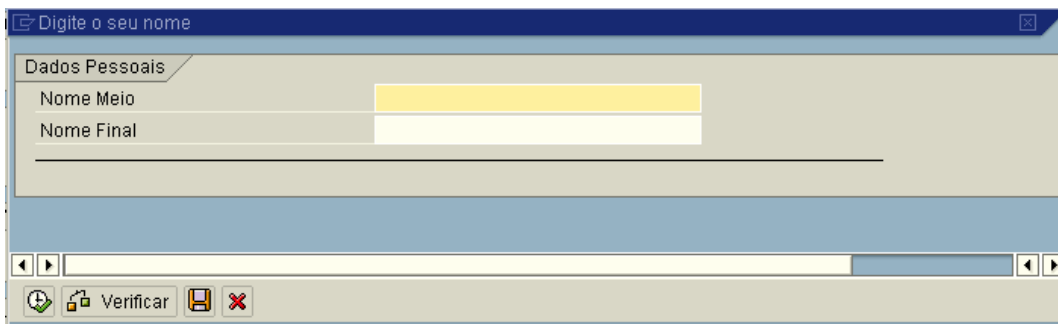
```
SELECTION-SCREEN END OF SCREEN 600.
```

“tudo posso naquele que me fortalece” (Filipenses 4:13).

Ex:



```
SELECTION-SCREEN BEGIN OF SCREEN 600 AS WINDOW TITLE TEXT-600.
PARAMETERS: BUTTON10 RADIOBUTTON GROUP GRP,
             BUTTON20 RADIOBUTTON GROUP GRP.
SELECTION-SCREEN END OF SCREEN 600.
```



```
SELECTION-SCREEN BEGIN OF SCREEN 700 AS WINDOW TITLE TEXT-700.
  SELECTION-SCREEN BEGIN OF BLOCK BL1 WITH FRAME TITLE TEXT-BL1.
    PARAMETERS: P_NOMEM(30) TYPE C,
               P_NOMEF(30) TYPE C.
    SELECTION-SCREEN ULINE.
  SELECTION-SCREEN END OF BLOCK BL1.
SELECTION-SCREEN END OF SCREEN 700.
```

Para chamar a tela usamos a sintaxe:

```
CALL SELECTION-SCREEN numero_da_screen STARTIND X Y ENDING X Y.
```

OU

```
CALL SCREEN numero_da_screen STARTIND X Y ENDING X Y.
```

“tudo posso naquele que me fortalece” (Filipenses 4:13).

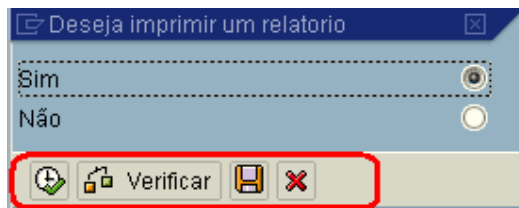
Onde chama uma determinada SCREEN pelo numero correspondente, passando as cordenadas de inicialização X Y, e passando as coordenadas de tamanho com relação a iniciação com os valores X Y.

Exe. :

```
CALL SELECTION-SCREEN 700 STARTING AT 3 4 ENDING AT 100  
10.
```

Este comando chamara a screen de numero 700, iniciando na posição X = 3 e Y = 4, relativo ao plano cartesiano da tela do R/3 (geralmente varia de uma faixa de 1 até 50 (“geralmente”)) e finalizando com uma tamanho X = 100 Y = 10, com base nos valores iniciais.

Uma chamada com **CALL SELECTION-SCREEN**, fica:



Uma chamada com **CALL SCREEN**, fica:



“tudo posso naquele que me fortalece” (Filipenses 4:13).

```

**&-----*
**& Report  ZSELECTION_SCREEN
**&
**&-----*
**&
**&
**&-----*

```

REPORT ZSELECTION_SCREEN.

TABLES: SSCRFIELDS, MARA, USR02.

TYPE-POOLS: VRM.

DATA: IT_LIST TYPE VRM_VALUES,
LIST_VALUE TYPE VRM_VALUE,
IT_IDIOMA TYPE VRM_VALUES,
IDIOMA_VALUE TYPE VRM_VALUE,
IT_NIVEL TYPE VRM_VALUES,
NIVEL_VALUE TYPE VRM_VALUE,
IT_TIPO TYPE VRM_VALUES,
TIPO_VALUE TYPE VRM_VALUE.

DATA: V_SEX0(20) TYPE C,
V_RECEBE(20) TYPE C,
V_CONT(2) TYPE C VALUE '1'.

DATA: ESTADO(20) TYPE C.

SELECTION-SCREEN BEGIN OF SCREEN 100 AS SUBSCREEN.

SELECTION-SCREEN BEGIN OF BLOCK B1 WITH FRAME.

PARAMETERS: P_NOME(30) TYPE C OBLIGATORY,
P_DTNASC(10) TYPE C,
P_EMAIL(100) TYPE C,
P_RG(20) TYPE C,
P_CPF(20) TYPE C.

SELECTION-SCREEN END OF BLOCK B1.

SELECTION-SCREEN END OF SCREEN 100.

SELECTION-SCREEN BEGIN OF SCREEN 200 AS SUBSCREEN.

SELECTION-SCREEN BEGIN OF BLOCK B2 WITH FRAME.

PARAMETERS: P_RUA(30) TYPE C,
P_CEP(30) TYPE C,
P_NUMERO(5) TYPE C,
P_CIDADE(30) TYPE C,
P_LIST(20) TYPE C AS LISTBOX VISIBLE LENGTH 15.

“tudo posso naquele que me fortalece” (Filipenses 4:13).

SELECTION-SCREEN END OF BLOCK B2.
SELECTION-SCREEN END OF SCREEN 200.

SELECTION-SCREEN BEGIN OF SCREEN 300 AS SUBSCREEN.
SELECTION-SCREEN BEGIN OF BLOCK B3 WITH FRAME.
PARAMETERS: P_GRAL(10) TYPE C,
 P_ANO_C(10) TYPE C.
SELECTION-SCREEN END OF BLOCK B3.
SELECTION-SCREEN END OF SCREEN 300.

SELECTION-SCREEN BEGIN OF SCREEN 400 AS SUBSCREEN.
SELECTION-SCREEN BEGIN OF BLOCK B4 WITH FRAME.
PARAMETERS: P_EMP(20) TYPE C,
 P_FUN(20) TYPE C,
 P_DTA(10) TYPE C,
 P_DTD(10) TYPE C.
SELECTION-SCREEN END OF BLOCK B4.
SELECTION-SCREEN END OF SCREEN 400.

SELECTION-SCREEN BEGIN OF SCREEN 500 AS SUBSCREEN.
SELECTION-SCREEN BEGIN OF BLOCK B5 WITH FRAME.
PARAMETERS: P_IDIOMA(20) TYPE C AS LISTBOX VISIBLE LENGTH
15,
 P_NIVEL(20) TYPE C AS LISTBOX VISIBLE LENGTH
15.
SELECTION-SCREEN END OF BLOCK B5.
SELECTION-SCREEN END OF SCREEN 500.

SELECTION-SCREEN BEGIN OF SCREEN 600 AS WINDOW TITLE TEXT-
600.
PARAMETERS: BUTTON10 RADIOBUTTON GROUP GRP,
 BUTTON20 RADIOBUTTON GROUP GRP.
SELECTION-SCREEN END OF SCREEN 600.

SELECTION-SCREEN BEGIN OF SCREEN 700 AS WINDOW TITLE TEXT-
700.
SET TITLEBAR '0700'.
SET PF-STATUS '0710'.
SELECTION-SCREEN BEGIN OF BLOCK BL1 WITH FRAME TITLE TEXT-
BL1.
PARAMETERS: P_NOMEM(30) TYPE C,
 P_NOMEF(30) TYPE C.

“tudo posso naquele que me fortalece” (Filipenses 4:13).

SELECTION-SCREEN ULINE.
SELECTION-SCREEN END OF BLOCK BL1.
SELECTION-SCREEN END OF SCREEN 700.

SELECTION-SCREEN BEGIN OF BLOCK BX1 WITH FRAME TITLE TEXT-100.
SELECTION-SCREEN BEGIN OF LINE.
PARAMETERS C_01 AS CHECKBOX DEFAULT 'X'.
SELECTION-SCREEN COMMENT 3(20) TEXT-001 FOR FIELD C_01.
PARAMETERS C_02 AS CHECKBOX.
SELECTION-SCREEN COMMENT 25(20) TEXT-002 FOR FIELD C_02.
SELECTION-SCREEN END OF LINE.
SELECTION-SCREEN END OF BLOCK BX1.

SELECTION-SCREEN BEGIN OF BLOCK BT WITH FRAME TITLE TEXT-BT1.
SELECTION-SCREEN BEGIN OF LINE.
SELECTION-SCREEN PUSHBUTTON 30(10) BT1 USER-COMMAND BT01.
SELECTION-SCREEN PUSHBUTTON (10) BT2 USER-COMMAND BT02.
SELECTION-SCREEN PUSHBUTTON (10) BT3 USER-COMMAND BT02.
SELECTION-SCREEN PUSHBUTTON (10) BT4 USER-COMMAND BT04.
SELECTION-SCREEN END OF LINE.
SELECTION-SCREEN END OF BLOCK BT.

SELECTION-SCREEN BEGIN OF BLOCK M WITH FRAME.
SELECT-OPTIONS S_MATNR FOR MARA-MATNR.
SELECTION-SCREEN END OF BLOCK M.

SELECTION-SCREEN BEGIN OF BLOCK MSI WITH FRAME TITLE TEXT-MSI NO INTERVALS.
SELECT-OPTIONS S_MSI FOR MARA-MATNR.
SELECTION-SCREEN END OF BLOCK MSI.

SELECTION-SCREEN BEGIN OF BLOCK BY1 WITH FRAME TITLE TEXT-200.
PARAMETERS: P_USER(10) TYPE C MODIF ID USE,
P_PWD(10) TYPE C MODIF ID USE,
P_TIPO(20) TYPE C AS LISTBOX VISIBLE LENGTH 15
MODIF ID USE.
SELECTION-SCREEN END OF BLOCK BY1.

SELECTION-SCREEN BEGIN OF BLOCK BZ1 WITH FRAME TITLE TEXT-300.
PARAMETERS: R_FEN RADIOBUTTON GROUP RADI,
P_MAS RADIOBUTTON GROUP RADI.
SELECTION-SCREEN END OF BLOCK BZ1.

SELECTION-SCREEN: FUNCTION KEY 1,

“tudo posso naquele que me fortalece” (Filipenses 4:13).

FUNCTION KEY 2,
 FUNCTION KEY 3,
 FUNCTION KEY 4,
 FUNCTION KEY 5.

SELECTION-SCREEN:

```

      BEGIN OF TABBED BLOCK MYTAB FOR 7 LINES,
        TAB (20) BUTTON1 USER-COMMAND PUSH1 DEFAULT
SCREEN 100,
        TAB (20) BUTTON2 USER-COMMAND PUSH2 DEFAULT
SCREEN 200,
        TAB (20) BUTTON3 USER-COMMAND PUSH3 DEFAULT
SCREEN 300,
        TAB (20) BUTTON4 USER-COMMAND PUSH4 DEFAULT
SCREEN 400,
        TAB (20) BUTTON5 USER-COMMAND PUSH5 DEFAULT
SCREEN 500,
      END OF BLOCK MYTAB.

```


* INICIALIZAÇÃO

INITIALIZATION.

```

  BUTTON1 = 'Dados Pessoais'.
  BUTTON2 = 'Endereço'.
  BUTTON3 = 'Escolaridade'.
  BUTTON4 = 'Experiencias Profissonais'.
  BUTTON5 = 'Idiomas'.
*  BUTTON6 = 'Botões'.

```

```

  BT1 = 'Cadastrar'.
  BT2 = 'Consultar'.
  BT3 = 'Apagar'.
  BT4 = 'Sair'.

```

```

  SSCRFIELDS-FUNCTXT_01 = '@0De'.
  SSCRFIELDS-FUNCTXT_02 = '@0Ee'.
  SSCRFIELDS-FUNCTXT_03 = '@0De'.
  SSCRFIELDS-FUNCTXT_04 = '@0Ee'.
  SSCRFIELDS-FUNCTXT_05 = 'Cadastrar'.

```

```

  MYTAB-PROG = SY-REPID.
  MYTAB-DYNNR = 100.
  MYTAB-ACTIVETAB = 'BUTTON2'.

```

“tudo posso naquele que me fortalece” (Filipenses 4:13).

```
PERFORM CARREGA_COMBO.

AT SELECTION-SCREEN ON VALUE-REQUEST FOR P_NOME.
  CALL SELECTION-SCREEN 700 STARTING AT 3 4 ENDING AT 100
  10.

*
* STARTING AT X (1 a 30) Y (1 a 30)
* posição de inicialização da tela
*
* ENDING AT X(1 150) Y (1 a 30)
* posição do tamanho da tela, relativo ao
* tamanho da tela
*

AT SELECTION-SCREEN OUTPUT.
  LOOP AT SCREEN.
    IF SCREEN-NAME = 'P_PWD'.
      SCREEN-INVISIBLE = '1'.
      MODIFY SCREEN.
    ENDIF.
    IF SCREEN-GROUP1 = 'USE'.
      SCREEN-intensified = '1'.
      IF SCREEN-NAME = 'P_USER'.
        SCREEN-ACTIVE = '1'.
        MODIFY SCREEN.
      ENDIF.
    ENDIF.
    IF SCREEN-NAME = 'P_TIPO'.
      SCREEN-INPUT = '1'.
      MODIFY SCREEN.
    ENDIF.
    MODIFY SCREEN.
  ENDIF.
ENDLOOP.

AT SELECTION-SCREEN.
  CASE SSCRFIELDS-UCOMM.
    WHEN 'FC05'.
      CALL SELECTION-SCREEN 600 STARTING AT 10 3 ENDING
      AT 42 7.
      PERFORM IMPRIME_DADOS.
    WHEN 'BT01'.
      PERFORM IMPRIME_DADOS.
    WHEN 'BT04'.
      MESSAGE i888(zselection_screen) WITH text-ER1
      sscrfields-ucomm.
```

“tudo posso naquele que me fortalece” (Filipenses 4:13).

ENDCASE.

START-OF-SELECTION.

CALL SELECTION-SCREEN 600 STARTING AT 10 3 ENDING AT 42
7.
PERFORM SELESIONA_DADOS.
PERFORM IMPRIME_DADOS.

END-OF-SELECTION.

```

*&-----*
*&      Form  SELESIONA_DADOS
*&-----*
*      text
*-----*
* --> p1      text
* <-- p2      text
*-----*
FORM SELESIONA_DADOS .
*  SELECT * FROM USR02 WHERE BNAME IN USERNAME
*                          AND ERDAT IN LASTLOGI
*                          AND CLASS IN CLASSTYP.
*  ENDSELECT.
ENDFORM.              " SELESIONA_DADOS
*&-----*
*&      Form  IMPRIME_DADOS
*&-----*
*      text
*-----*
* --> p1      text
* <-- p2      text
*-----*

FORM IMPRIME_DADOS.

*  WRITE: / 'Usuario ', USR02-BNAME,
*          'Last Login Date ', USR02-TRDAT,
*          'Last Login Time ', USR02-LTIME,
*          'CLASS ', USR02-CLASS.
*
*  IF BUTTON10 = 'X'.
*    ULINE.

```

“tudo posso naquele que me fortalece” (Filipenses 4:13).

```
WRITE: / 'Usuario: ', P_USER,
        / 'Senha: ', P_PWD,
        / 'tipo: ', P_TIPO.
SKIP.
ULINE.

ESTADO = P_LIST.
PERFORM ESTADO CHANGING ESTADO.
WRITE: / ' Rua: ', P_RUA,
        / ' Numero :', P_NUMERO,
        / ' Cep:', P_CEP,
        / ' Cidade:', P_CIDADE,
        / ' Estado:', ESTADO.

SKIP.
ULINE.

WRITE: / 'Nome: ', P_NOME,
        / 'Data de Nascimento:', P_DTNASC,
        / 'Email: ', P_EMAIL,
        / 'RG: ', P_RG,
        / 'CPF: ', P_CPF.

SKIP.
ULINE.

WRITE: / 'Escolaridade: ', P_GRAL,
        / 'Ano de conclusão: ', P_ANO_C.

SKIP.
ULINE.

WRITE: / 'Empresa: ', P_EMP,
        / 'Função: ', P_FUN,
        / 'Data de Admissão: ', P_DTA,
        / 'Data de Demissão: ', P_DTD.

SKIP.
ULINE.

*   IF R_FEN = 'X'.
*       WRITE: / 'Sexo: Feminino'.
*   ELSE.
*       WRITE: / 'Sexo: Masculino'.
*   ENDIF.

WRITE: / 'Sexo: ', V_SEX0.
```

“tudo posso naquele que me fortalece” (Filipenses 4:13).

```

SKIP.
ULINE.

WRITE: / 'Idioma: ', P_IDIOMA,
        / 'Nivel: ', P_NIVEL.

SKIP.
ULINE.

*   ENDIF.
ENDFORM.                                " IMPRIME_DADOS

*&-----*
*&      Form  CARREGA_COMBO              *
*&-----*
*      text                             *
*-----*
*  --> p1      text                     *
*  <-- p2      text                     *
*-----*
FORM CARREGA_COMBO .

LIST_VALUE-KEY = '01'.
LIST_VALUE-TEXT = 'São Paulo'.
APPEND LIST_VALUE TO IT_LIST.

LIST_VALUE-KEY = '02'.
LIST_VALUE-TEXT = 'Parana'.
APPEND LIST_VALUE TO IT_LIST.

LIST_VALUE-KEY = '03'.
LIST_VALUE-TEXT = 'Santa Catarina'.
APPEND LIST_VALUE TO IT_LIST.

LIST_VALUE-KEY = '04'.
LIST_VALUE-TEXT = 'Bahia'.
APPEND LIST_VALUE TO IT_LIST.

LIST_VALUE-KEY = '05'.
LIST_VALUE-TEXT = 'Pernambuco'.
APPEND LIST_VALUE TO IT_LIST.

LIST_VALUE-KEY = '06'.
LIST_VALUE-TEXT = 'Amazonas'.
APPEND LIST_VALUE TO IT_LIST.

LIST_VALUE-KEY = '07'.
LIST_VALUE-TEXT = 'Piaui'.

```

“tudo posso naquele que me fortalece” (Filipenses 4:13).

```
APPEND LIST_VALUE TO IT_LIST.
```

```
LIST_VALUE-KEY = '08'.  
LIST_VALUE-TEXT = 'Maranhão'.  
APPEND LIST_VALUE TO IT_LIST.
```

```
LIST_VALUE-KEY = '09'.  
LIST_VALUE-TEXT = 'Para'.  
APPEND LIST_VALUE TO IT_LIST.
```

```
LIST_VALUE-KEY = '10'.  
LIST_VALUE-TEXT = 'Tocantins'.  
APPEND LIST_VALUE TO IT_LIST.
```

```
CALL FUNCTION 'VRM_SET_VALUES'  
  EXPORTING  
    ID      = 'P_LIST'  
    VALUES = IT_LIST.
```

```
IDIOMA_VALUE-KEY = '01'.  
IDIOMA_VALUE-TEXT = 'Inglês'.  
APPEND IDIOMA_VALUE TO IT_IDIOMA.
```

```
IDIOMA_VALUE-KEY = '02'.  
IDIOMA_VALUE-TEXT = 'Espanhol'.  
APPEND IDIOMA_VALUE TO IT_IDIOMA.
```

```
IDIOMA_VALUE-KEY = '03'.  
IDIOMA_VALUE-TEXT = 'Francês'.  
APPEND IDIOMA_VALUE TO IT_IDIOMA.
```

```
IDIOMA_VALUE-KEY = '04'.  
IDIOMA_VALUE-TEXT = 'Italiano'.  
APPEND IDIOMA_VALUE TO IT_IDIOMA.
```

```
IDIOMA_VALUE-KEY = '05'.  
IDIOMA_VALUE-TEXT = 'Alemão'.  
APPEND IDIOMA_VALUE TO IT_IDIOMA.
```

```
CALL FUNCTION 'VRM_SET_VALUES'  
  EXPORTING  
    ID      = 'P_IDIOMA'  
    VALUES = IT_IDIOMA.
```

```
NIVEL_VALUE-KEY = '01'.  
NIVEL_VALUE-TEXT = 'basico'.  
APPEND NIVEL_VALUE TO IT_NIVEL.
```

```
NIVEL_VALUE-KEY = '02'.
```

“tudo posso naquele que me fortalece” (Filipenses 4:13).

```

NIVEL_VALUE-TEXT = 'intermediario'.
APPEND NIVEL_VALUE TO IT_NIVEL.

NIVEL_VALUE-KEY = '03'.
NIVEL_VALUE-TEXT = 'avancado'.
APPEND NIVEL_VALUE TO IT_NIVEL.

NIVEL_VALUE-KEY = '04'.
NIVEL_VALUE-TEXT = 'fluyente'.
APPEND NIVEL_VALUE TO IT_NIVEL.

CALL FUNCTION 'VRM_SET_VALUES'
  EXPORTING
    ID      = 'P_NIVEL'
    VALUES = IT_NIVEL.

TIPO_VALUE-KEY = '01'.
TIPO_VALUE-TEXT = 'consultor'.
APPEND TIPO_VALUE TO IT_TIPO.

TIPO_VALUE-KEY = '02'.
TIPO_VALUE-TEXT = 'gerente'.
APPEND TIPO_VALUE TO IT_TIPO.

CALL FUNCTION 'VRM_SET_VALUES'
  EXPORTING
    ID      = 'P_TIPO'
    VALUES = IT_TIPO.

* CLEAR: TIPO_VALUE, IT_TIPO,
*         NIVEL_VALUE, IT_NIVEL,
*         IDIOMA_VALUE, IT_IDIOMA.

ENDFORM.                " CARREGA_COMBO
*&-----*
*&      Form  ESTADO
*&-----*
*       text
*-----*
*       <--P_ESTADO  text
*-----*
FORM ESTADO  CHANGING P_ESTADO.

CASE P_ESTADO.
  WHEN '01'.
    P_ESTADO = 'São Paulo'.
  WHEN '02'.
    P_ESTADO = 'Parana'.

```

“tudo posso naquele que me fortalece” (Filipenses 4:13).

```
WHEN '03'.
  P_ESTADO = 'Santa Catarina'.
WHEN '04'.
  P_ESTADO = 'Bahia'.
WHEN '05'.
  P_ESTADO = 'Pernambuco'.
WHEN '06'.
  P_ESTADO = 'Amazonas'.
WHEN '07'.
  P_ESTADO = 'Piaui'.
WHEN '08'.
  P_ESTADO = 'Maranhão'.
WHEN '09'.
  P_ESTADO = 'Para'.
ENDCASE.

ENDFORM.                " ESTADO
```

“tudo posso naquele que me fortalece” (Filipenses 4:13).