# CL\_GUI\_ALV\_TREE - Montagem completa – Raphael

Observações:

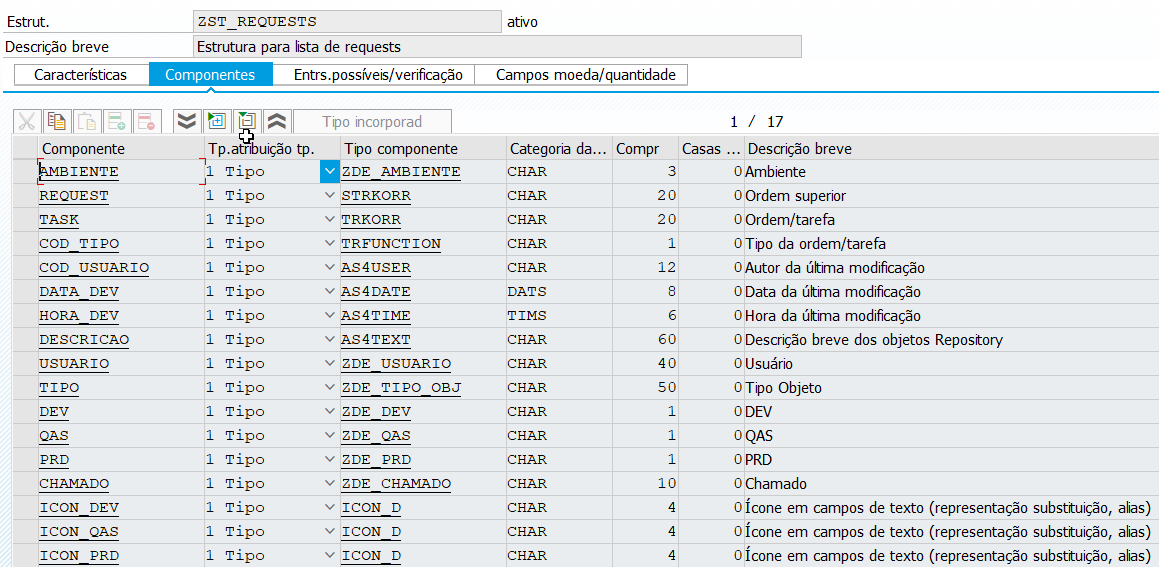
Basta copiar os FORMS aqui mencionados e colar no código, funcionará. O que precisará de ajuste, será as estruturas de dados utilizadas.

Se atentar para a tabela interna (lt\_saida\_alv LIKE lt\_requests[]), que é a chave para o funcionamento da TREE. Toda a manipulação dos dados para eventos, são em cima desta tabela, ela é organizada de acordo com a montagem dos nós, **nunca** a preenchemos.

## Declaração de variáveis

DATA: lo\_chamados         TYPE REF TO zcl\_chamados,  
      lo\_tree             TYPE REF TO cl\_gui\_alv\_tree,  
      lo\_event\_alv        TYPE REF TO zcl\_event\_alv,   
      lt\_requests         TYPE ztt\_requests,  
      lt\_requests\_full    LIKE lt\_requests[],  
      lt\_saida\_alv        LIKE lt\_requests[],  
      lt\_request\_chamado  LIKE lt\_requests[],  
      lt\_node\_key         TYPE lvc\_t\_nkey,  
      lt\_fieldcat         TYPE lvc\_t\_fcat,  
      ls\_hierarchy\_header TYPE treev\_hhdr,  
      lt\_report\_title     TYPE slis\_t\_listheader,  
      ls\_variant          TYPE disvariant,  
      ls\_layout           TYPE slis\_layout\_alv,  
      lv\_logo             TYPE sdydo\_value,  
      lv\_okcode\_100       TYPE sy-ucomm.

**Estrutura da tabela lt\_request (ZTT\_REQUESTS)**



**A montagem deve seguir os passos abaixo:**

### Criar FORM para adicionar os nós da árvore

\*&---------------------------------------------------------------------\*  
\*&      Form  f\_ADD\_NODE  
\*&---------------------------------------------------------------------\*  
\*       text  
\*----------------------------------------------------------------------\*  
FORM f\_add\_node USING p\_title TYPE c  
                VALUE(p\_relate\_key)  
                CHANGING p\_node\_key.  
  
  DATA: ld\_node\_text   TYPE lvc\_value,  
        ls\_vazio       TYPE zst\_requests,  
        lt\_item\_layout TYPE lvc\_t\_layi,  
        ls\_item\_layout TYPE lvc\_s\_layi.  
  
  "ls\_item\_layout-t\_image   = '@3P@'.  
  ls\_item\_layout-fieldname = lo\_tree->c\_hierarchy\_column\_name.  
  ls\_item\_layout-style     = cl\_gui\_column\_tree=>style\_default.  
  ld\_node\_text             = p\_title.  
  APPEND ls\_item\_layout TO lt\_item\_layout[].  
  
  "Add node  
  lo\_tree->add\_node(  
    EXPORTING  
      i\_relat\_node\_key = p\_relate\_key  
      i\_relationship   = cl\_gui\_column\_tree=>relat\_last\_child  
      i\_node\_text      = ld\_node\_text  
      is\_outtab\_line   = ls\_vazio  
      it\_item\_layout   = lt\_item\_layout  
    IMPORTING  
      e\_new\_node\_key   = p\_node\_key  
  ).  
  
ENDFORM.

### Criar FORM para adicionar as linhas na árvore (Todos os campos da tabela interna)

\*&---------------------------------------------------------------------\*  
\*&      Form  f\_ADD\_LINE  
\*&---------------------------------------------------------------------\*  
\*       text  
\*----------------------------------------------------------------------\*  
FORM f\_add\_line USING ls\_requests TYPE zst\_requests  
                VALUE(p\_relate\_key)  
                CHANGING p\_node\_key.  
  
  DATA: ld\_node\_text   TYPE lvc\_value,  
        lt\_item\_layout TYPE lvc\_t\_layi,  
        ls\_item\_layout TYPE lvc\_s\_layi.  
  
  "ls\_item\_layout-t\_image   = '@3P@'.  
  ls\_item\_layout-fieldname = lo\_tree->c\_hierarchy\_column\_name.  
  ls\_item\_layout-style     = cl\_gui\_column\_tree=>style\_default.  
  ld\_node\_text             = ls\_requests-descricao.  
  APPEND ls\_item\_layout TO lt\_item\_layout[].  
  
  "Add node  
  lo\_tree->add\_node(  
    EXPORTING  
      i\_relat\_node\_key = p\_relate\_key  
      i\_relationship   = cl\_gui\_column\_tree=>relat\_last\_child  
      i\_node\_text      = ld\_node\_text  
      is\_outtab\_line   = ls\_requests  
      it\_item\_layout   = lt\_item\_layout[]  
    IMPORTING  
      e\_new\_node\_key   = p\_node\_key  
  ).  
  
ENDFORM.

### Criar hierarquia dos nós. Para isso, vamos precisar dos FORMS acima.

*\*&---------------------------------------------------------------------\**  
*\*&      Form  F\_CRIAR\_HIERARQUIA\_ARVORE*  
*\*&---------------------------------------------------------------------\**  
*\*       text*  
*\*----------------------------------------------------------------------\**  
FORM f\_criar\_hierarquia\_arvore.  
  
  DATA: lv\_no\_pai       TYPE lvc\_nkey,  
        lv\_no\_filho     TYPE lvc\_nkey,  
        lv\_sub\_no\_filho TYPE lvc\_nkey,  
        lt\_task         LIKE lt\_requests[],  
        lt\_ambiente     LIKE lt\_requests[],  
        ls\_node\_key     LIKE LINE OF lt\_node\_key[],  
        lv\_primeiro\_no  TYPE char1 VALUE 'X',  
        lv\_line1        LIKE ls\_node\_key,  
        lv\_fieldname    TYPE lvc\_fname VALUE '&Hierarchy'.  
  
  lt\_task[]     = lt\_requests[].  
  lt\_ambiente[] = lt\_requests[].  
  
  SORT lt\_ambiente[] BY ambiente.  
  DELETE ADJACENT DUPLICATES FROM lt\_ambiente[] COMPARING ambiente.  
  DELETE lt\_requests[] WHERE task IS NOT INITIAL.  
  DELETE lt\_task[]     WHERE task IS INITIAL.  
  
  LOOP AT lt\_ambiente[] ASSIGNING FIELD-SYMBOL(<fs\_ambiente>).  
    *"Monta o nó PAI.*  
    PERFORM f\_add\_node USING <fs\_ambiente>-ambiente  
                             ''  
                    CHANGING lv\_no\_pai. *"EDC*  
  
    *"Guarda o primeiro nó para focarmos o cursor nele.*  
    IF lv\_primeiro\_no EQ 'X'.  
      CLEAR lv\_primeiro\_no.  
      lv\_line1 = lv\_no\_pai.  
    ENDIF.  
  
    *"Controla os nós que quero expandir automaticamente.*  
    ls\_node\_key = lv\_no\_pai.  
    APPEND ls\_node\_key TO lt\_node\_key[].  
  
    LOOP AT lt\_requests[] ASSIGNING FIELD-SYMBOL(<fs\_requests>) WHERE ambiente EQ <fs\_ambiente>-ambiente.  
      PERFORM f\_add\_line USING <fs\_requests>  
                               lv\_no\_pai  
                      CHANGING lv\_no\_filho. *"DESCRICAO REQUEST + LINHA DE DADOS*  
  
      LOOP AT lt\_task ASSIGNING FIELD-SYMBOL(<fs\_task>) WHERE request EQ <fs\_requests>-request  
                                                        AND task IS NOT INITIAL.  
        PERFORM f\_add\_line USING <fs\_task>  
                                 lv\_no\_filho  
                        CHANGING lv\_sub\_no\_filho. *"TASKS DA REQUEST*  
      ENDLOOP.  
    ENDLOOP.  
  ENDLOOP.  
  
  *"Expande os nós da árvore.*  
  lo\_tree->expand\_nodes(  
    EXPORTING  
      it\_node\_key = lt\_node\_key[]  
  ).  
  
  *"Foca cursor no primeiro nó*  
  lo\_tree->set\_selected\_item(  
    i\_node\_key  = lv\_line1  
    i\_fieldname = lv\_fieldname  
  ).  
  
  *"Envia os dados para tela.*  
  lo\_tree->frontend\_update( ).  
  
ENDFORM.

Obs.: Esse exemplo possui 3 níveis hierárquicos: Amniente->request->task.

Obs.: Os eventos do ALV TREE não funcionaram como do Grid normal. Tive que criar um FORM para atualizar os eventos da TREE, como segue abaixo:

FORM f\_registrar\_eventos.  
  
  DATA: lt\_events TYPE cntl\_simple\_events,  
        ls\_event  TYPE cntl\_simple\_event.  
  
  lo\_tree->get\_registered\_events(  
    IMPORTING  
      events = lt\_events[]  
  ).  
  
  "Evento criado na classe ZCL\_EVENT\_ALV  
  ls\_event-eventid    = cl\_gui\_column\_tree=>eventid\_item\_double\_click.  
  ls\_event-appl\_event = 'X'.  
  
  APPEND ls\_event TO lt\_events[].  
  
  lo\_tree->set\_registered\_events(  
    EXPORTING  
      events                    = lt\_events[]  
    EXCEPTIONS  
      cntl\_error                = 1  
      cntl\_system\_error         = 2  
      illegal\_event\_combination = 3  
  ).  
  
  IF sy-subrc NE 0.  
    MESSAGE 'Evento não suportado!' TYPE 'S' DISPLAY LIKE 'W'.  
  ENDIF.  
  
  lo\_event\_alv = NEW zcl\_event\_alv( ).  
  
  SET HANDLER lo\_event\_alv->item\_double\_click FOR lo\_tree.  
  
ENDFORM. " register\_events

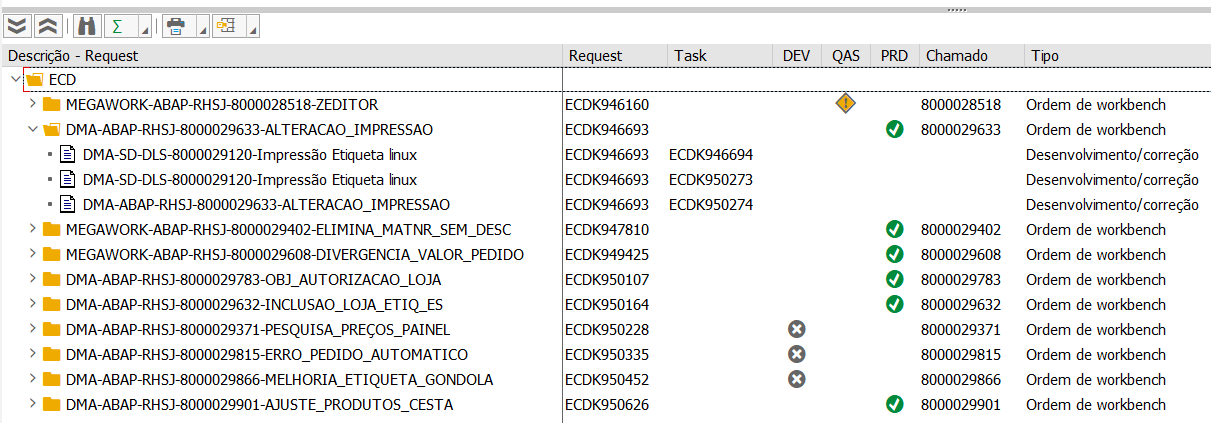
### Montar o PBO como exemplo abaixo:

\*&---------------------------------------------------------------------\*  
\*&      Module  M\_SHOWGRID\_100  OUTPUT  
\*&---------------------------------------------------------------------\*  
\*       text  
\*----------------------------------------------------------------------\*

MODULE m\_showgrid\_100 OUTPUT.  
  
  REFRESH: lt\_fieldcat[].  
  
  CLEAR: ls\_variant,  
         ls\_layout,  
         ls\_hierarchy\_header,  
         lv\_okcode\_100.  
  
  ls\_variant-report             = sy-repid.  
  ls\_layout-no\_input            = 'X'.  
  ls\_layout-colwidth\_optimize   = 'X'.  
  ls\_hierarchy\_header-heading   = 'Descrição - Request'.  
  ls\_hierarchy\_header-tooltip   = 'Request'.  
  ls\_hierarchy\_header-width     = 80.  
  ls\_hierarchy\_header-width\_pix = ''.  
  
  
    PERFORM f\_build\_fieldcat USING:  
          'REQUEST'      'REQUEST'      'ZST\_REQUESTS'  'Request'         'X'  ' '  ' '  15  CHANGING lt\_fieldcat,  
          'TASK'         'TASK'         'ZST\_REQUESTS'  'Task'            'X'  ' '  ' '  15  CHANGING lt\_fieldcat,  
          'ICON\_DEV'     '       '      '            '  'DEV'             'X'  ' '  'X'  07  CHANGING lt\_fieldcat,  
          'ICON\_QAS'     '       '      '            '  'QAS'             'X'  ' '  'X'  07  CHANGING lt\_fieldcat,  
          'ICON\_PRD'     '       '      '            '  'PRD'             'X'  ' '  'X'  07  CHANGING lt\_fieldcat,  
          'CHAMADO'      'CHAMADO'      'ZST\_REQUESTS'  'Chamado'         'X'  ' '  ' '  15  CHANGING lt\_fieldcat,  
          'TIPO'         'TIPO'         'ZST\_REQUESTS'  'Tipo'            'X'  ' '  ' '  40  CHANGING lt\_fieldcat,  
          'DATA\_DEV'     'DATA\_DEV'     'ZST\_REQUESTS'  'Data alteração'  'X'  ' '  ' '  17  CHANGING lt\_fieldcat,  
          'HORA\_DEV'     'HORA\_DEV'     'ZST\_REQUESTS'  'Hora alteração'  'X'  ' '  ' '  17  CHANGING lt\_fieldcat,  
          'USUARIO'      'USUARIO'      'ZST\_REQUESTS'  'Usuário'         'X'  ' '  ' '  40  CHANGING lt\_fieldcat.  
    
  IF lo\_tree IS INITIAL.  
    lo\_tree = NEW cl\_gui\_alv\_tree(  
      parent              = cl\_gui\_container=>default\_screen  
      node\_selection\_mode = cl\_gui\_column\_tree=>node\_sel\_mode\_single  
      item\_selection      = 'X'  
      no\_html\_header      = ''  
      no\_toolbar          = ''  
    ).  
  
    lo\_tree->set\_table\_for\_first\_display(  
      EXPORTING  
        is\_hierarchy\_header = ls\_hierarchy\_header  
        it\_list\_commentary  = lt\_report\_title[]  
        i\_logo              = lv\_logo  
        i\_save              = 'A'  
        is\_variant          = ls\_variant  
      CHANGING  
        it\_outtab           = lt\_saida\_alv[]      "Deve estar sempre SEM preencher (EM BRANCO)  
        it\_fieldcatalog     = lt\_fieldcat[]  
    ).  
  
    PERFORM f\_registrar\_eventos.  
    PERFORM f\_criar\_hierarquia\_arvore. "Coloca os dados no ALV.  
  ELSE.  
    lo\_tree->update\_calculations( space ). "Refresh

  ENDIF.  
  
ENDMODULE.

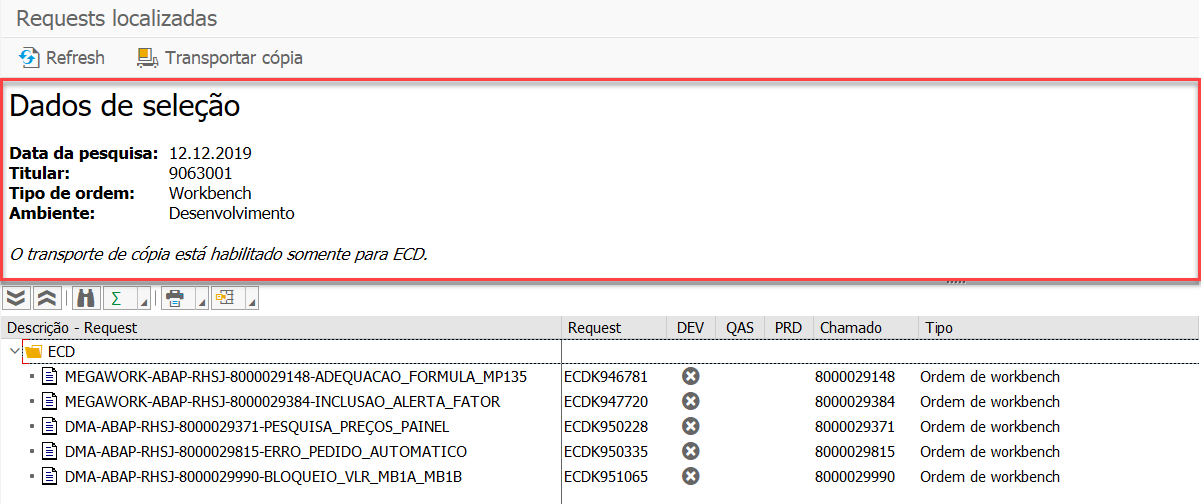
### Resultado:



### Adicionando Cabeçalho ao Grid

*\*&---------------------------------------------------------------------\**  
*\*&      Form  F\_BUILD\_REPORT\_TITLE*  
*\*&---------------------------------------------------------------------\**  
*\*       text*  
*\*----------------------------------------------------------------------\**  
FORM f\_build\_report\_title CHANGING t\_report\_title TYPE slis\_t\_listheader  
                                   p\_logo         TYPE sdydo\_value.  
  
  DATA: ls\_line     TYPE slis\_listheader,  
        lv\_uname    TYPE string,  
        lv\_tipo     TYPE string,  
        lv\_ambiente TYPE string,  
        lv\_date     TYPE char10.

p\_logo = 'DMA'. *"Adicionado em TCODE = 0FPM002 – Vefificar Arquivo da “montgem do REUSE\_ALVGRID. Ele ensina a importar o arquivo.*  
  
  *"List Heading Line(TYPE H)*  
  CLEAR ls\_line.  
  ls\_line-typ  = 'H'.  
  ls\_line-info = 'Dados de seleção'.  
  APPEND ls\_line TO t\_report\_title[].  
  
  *"Status Line(TYPE S)*  
  CONCATENATE  
    sy-datum+6(2) '.'  
    sy-datum+4(2) '.'  
    sy-datum+0(4)  
    INTO lv\_date.  
  
  ls\_line-typ  = 'S'.  
  ls\_line-key  = 'Data da pesquisa:'.  
  ls\_line-info = lv\_date.  
  APPEND ls\_line TO t\_report\_title[].  
  
  IF so\_uname[] IS NOT INITIAL.  
    IF lines( so\_uname[] ) GT 1.  
      READ TABLE so\_uname[] ASSIGNING FIELD-SYMBOL(<fs\_uname>) INDEX 1.  
  
      lv\_uname = <fs\_uname>-low.  
  
      READ TABLE so\_uname[] ASSIGNING <fs\_uname> INDEX lines( so\_uname[] ).  
  
      CONCATENATE lv\_uname '... ' <fs\_uname>-low INTO lv\_uname RESPECTING BLANKS.  
    ELSE.  
      READ TABLE so\_uname[] ASSIGNING <fs\_uname> INDEX 1.  
  
      IF <fs\_uname>-high IS INITIAL.  
        lv\_uname = <fs\_uname>-low.  
      ELSE.  
        CONCATENATE <fs\_uname>-low 'à' <fs\_uname>-high INTO lv\_uname SEPARATED BY space.  
      ENDIF.  
    ENDIF.  
  
    ls\_line-typ  = 'S'.  
    ls\_line-key  = 'Titular:'.  
    ls\_line-info = lv\_uname.  
    APPEND ls\_line TO t\_report\_title[].  
  ENDIF.  
  
  IF p\_wbench IS NOT INITIAL OR p\_ctzing IS NOT INITIAL OR p\_copia IS NOT INITIAL OR p\_task IS NOT INITIAL.  
    IF p\_wbench IS NOT INITIAL AND p\_ctzing IS INITIAL AND p\_copia IS INITIAL AND p\_task IS INITIAL.  
      lv\_tipo = 'Workbench'.  
    ELSEIF p\_wbench IS INITIAL AND p\_ctzing IS NOT INITIAL AND p\_copia IS INITIAL AND p\_task IS INITIAL.  
      lv\_tipo = 'Customizing'.  
    ELSEIF p\_wbench IS INITIAL AND p\_ctzing IS INITIAL AND p\_copia IS NOT INITIAL AND p\_task IS INITIAL.  
      lv\_tipo = 'Cópia'.  
    ELSEIF p\_wbench IS INITIAL AND p\_ctzing IS INITIAL AND p\_copia IS INITIAL AND p\_task IS NOT INITIAL.  
      lv\_tipo = 'Task'.  
    ELSEIF p\_wbench IS NOT INITIAL AND p\_ctzing IS NOT INITIAL AND p\_copia IS INITIAL AND p\_task IS INITIAL.  
      lv\_tipo = 'Workbench e Customizing'.  
    ELSEIF p\_wbench IS NOT INITIAL AND p\_ctzing IS INITIAL AND p\_copia IS NOT INITIAL AND p\_task IS INITIAL.  
      lv\_tipo = 'Workbench e Cópia'.  
    ELSEIF p\_wbench IS NOT INITIAL AND p\_ctzing IS INITIAL AND p\_copia IS INITIAL AND p\_task IS NOT INITIAL.  
      lv\_tipo = 'Workbench e Task'.  
    ELSEIF p\_wbench IS INITIAL AND p\_ctzing IS NOT INITIAL AND p\_copia IS NOT INITIAL AND p\_task IS INITIAL.  
      lv\_tipo = 'Customizing e Cópia'.  
    ELSEIF p\_wbench IS INITIAL AND p\_ctzing IS NOT INITIAL AND p\_copia IS INITIAL AND p\_task IS NOT INITIAL.  
      lv\_tipo = 'Customizing e Task'.  
    ELSEIF p\_wbench IS INITIAL AND p\_ctzing IS INITIAL AND p\_copia IS NOT INITIAL AND p\_task IS NOT INITIAL.  
      lv\_tipo = 'Cópia e Task'.  
    ELSEIF p\_wbench IS NOT INITIAL AND p\_ctzing IS NOT INITIAL AND p\_copia IS NOT INITIAL AND p\_task IS INITIAL.  
      lv\_tipo = 'Workbench, Customizing e Cópia'.  
    ELSEIF p\_wbench IS NOT INITIAL AND p\_ctzing IS NOT INITIAL AND p\_copia IS INITIAL AND p\_task IS NOT INITIAL.  
      lv\_tipo = 'Workbench, Customizing e Task'.  
    ELSEIF p\_wbench IS NOT INITIAL AND p\_ctzing IS INITIAL AND p\_copia IS NOT INITIAL AND p\_task IS NOT INITIAL.  
      lv\_tipo = 'Workbench, Cópia e Task'.  
    ELSEIF p\_wbench IS INITIAL AND p\_ctzing IS NOT INITIAL AND p\_copia IS NOT INITIAL AND p\_task IS NOT INITIAL.  
      lv\_tipo = 'Customizing, Cópia e Task'.  
    ELSE.  
      lv\_tipo = 'Workbench, Customizing, Cópia e Task'.  
    ENDIF.  
  
    ls\_line-typ  = 'S'.  
    ls\_line-key  = 'Tipo de ordem:'.  
    ls\_line-info = lv\_tipo.  
    APPEND ls\_line TO t\_report\_title[].  
  ELSE.  
    ls\_line-typ  = 'S'.  
    ls\_line-key  = 'Tipo de ordem:'.  
    ls\_line-info = 'Workbench, Customizing, Cópia e Task'.  
    APPEND ls\_line TO t\_report\_title[].  
  ENDIF.  
  
  IF p\_dev IS NOT INITIAL OR p\_qas IS NOT INITIAL OR p\_prd IS NOT INITIAL.  
    IF p\_dev IS NOT INITIAL AND p\_qas IS INITIAL AND p\_prd IS INITIAL.  
      lv\_ambiente = 'Desenvolvimento'.  
    ELSEIF p\_dev IS INITIAL AND p\_qas IS NOT INITIAL AND p\_prd IS INITIAL.  
      lv\_ambiente = 'Qualidade'.  
    ELSEIF p\_dev IS INITIAL AND p\_qas IS INITIAL AND p\_prd IS NOT INITIAL.  
      lv\_ambiente = 'Produção'.  
    ELSEIF p\_dev IS NOT INITIAL AND p\_qas IS NOT INITIAL AND p\_prd IS INITIAL.  
      lv\_ambiente = 'Desenvolvimento e Qualidade'.  
    ELSEIF p\_dev IS NOT INITIAL AND p\_qas IS INITIAL AND p\_prd IS NOT INITIAL.  
      lv\_ambiente = 'Desenvolvimento e Produção'.  
    ELSEIF p\_dev IS INITIAL AND p\_qas IS NOT INITIAL AND p\_prd IS NOT INITIAL.  
      lv\_ambiente = 'Qualidade e Produção'.  
    ELSE.  
      lv\_ambiente = 'Desenvolvimento, Qualidade e Produção'.  
    ENDIF.  
  
    ls\_line-typ  = 'S'.  
    ls\_line-key  = 'Ambiente:'.  
    ls\_line-info = lv\_ambiente.  
    APPEND ls\_line TO t\_report\_title[].  
  ELSE.  
    ls\_line-typ  = 'S'.  
    ls\_line-key  = 'Ambiente:'.  
    ls\_line-info = 'Desenvolvimento, Qualidade e Produção'.  
    APPEND ls\_line TO t\_report\_title[].  
  ENDIF.  
  
  IF p\_trkorr IS NOT INITIAL.  
    ls\_line-typ  = 'S'.  
    ls\_line-key  = 'Ordem/Request:'.  
    ls\_line-info = p\_trkorr.  
    APPEND ls\_line TO t\_report\_title[].  
  ENDIF.  
  
  *"Action Line(TYPE A)*  
  CLEAR: ls\_line.  
  ls\_line-typ  = 'A'.  
  ls\_line-info = 'O transporte de cópia está habilitado somente para ECD.'.  
  APPEND ls\_line TO t\_report\_title.  
  
  *"Insere linhas em branco para não quebrar o texto.*  
  ls\_line-typ  = 'A'.  
  ls\_line-info = space.  
  APPEND ls\_line TO t\_report\_title[].  
  
  ls\_line-typ  = 'A'.  
  ls\_line-info = space.  
  APPEND ls\_line TO t\_report\_title[].  
  
ENDFORM.



Chamar o FORM antes do SET\_FIRST\_DISPLAY da montagem.