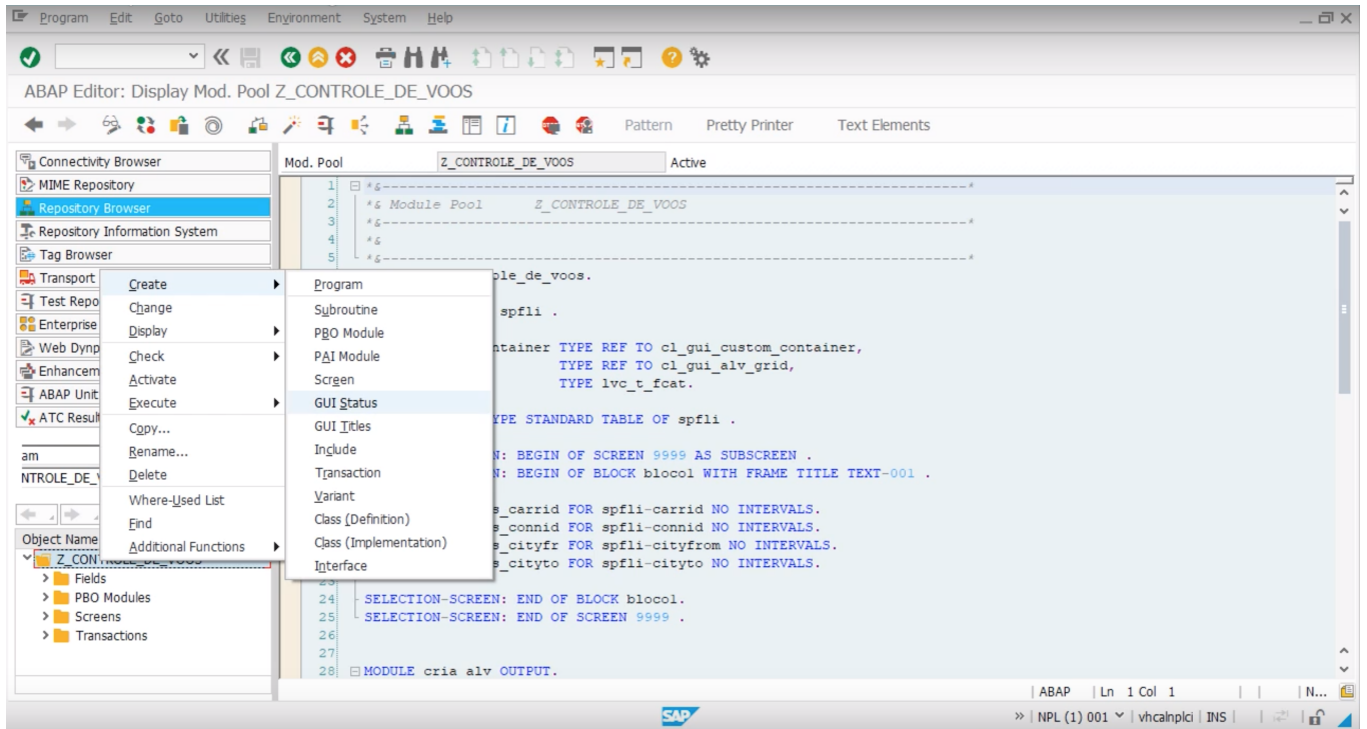


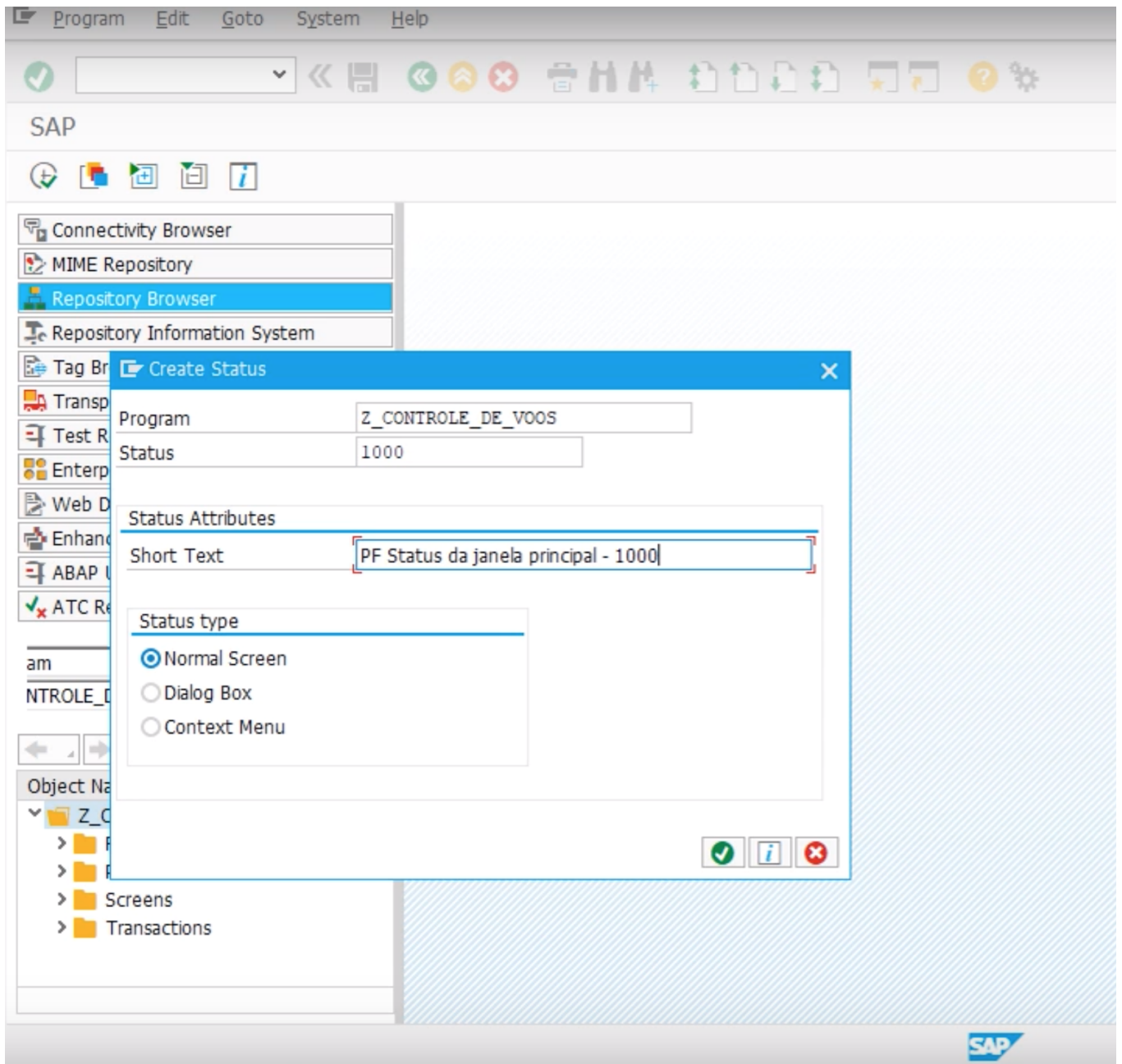
Faça o que eu fiz na aula

Até agora não implementamos as ações básicas de navegação da nossa janela. Então, é o que vamos tratar de fazer agora!

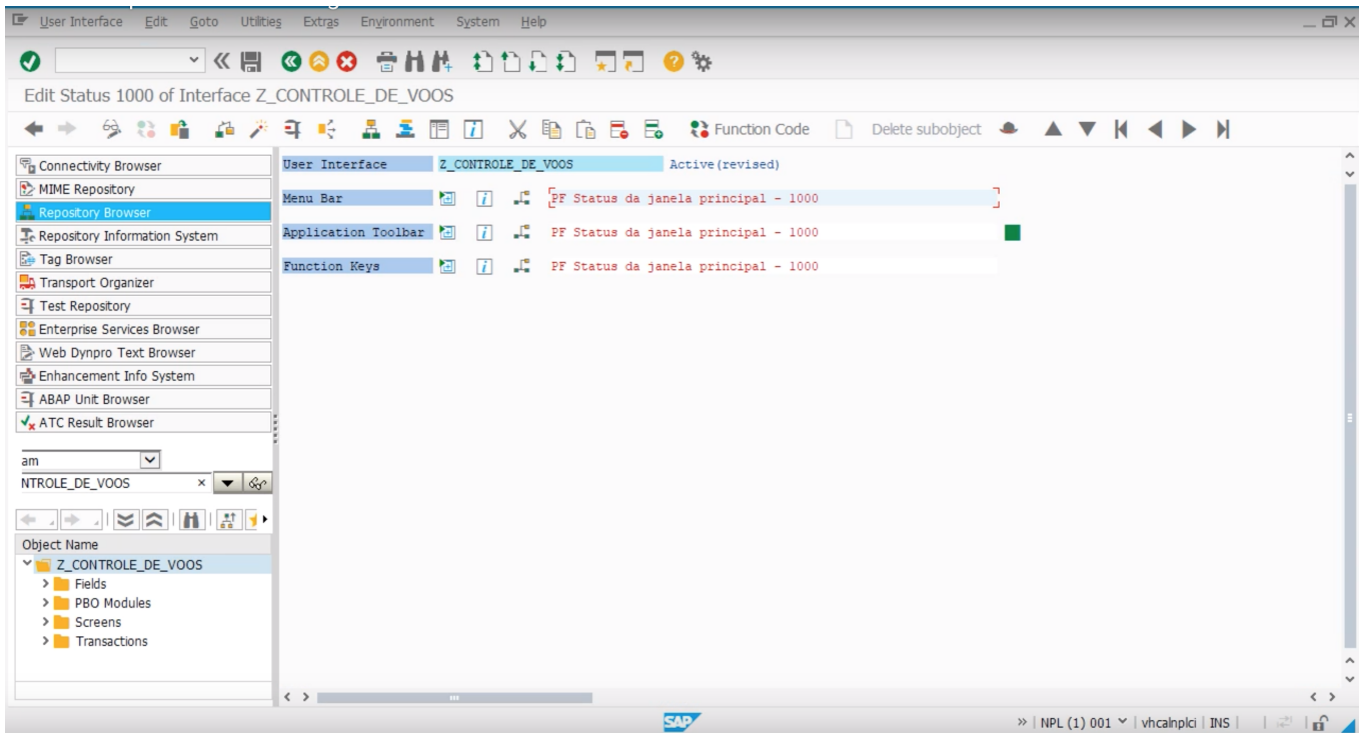
Primeiro, clicamos com o botão secundário no nome da nossa aplicação e, então, navegamos por **Create > GUI Status**.



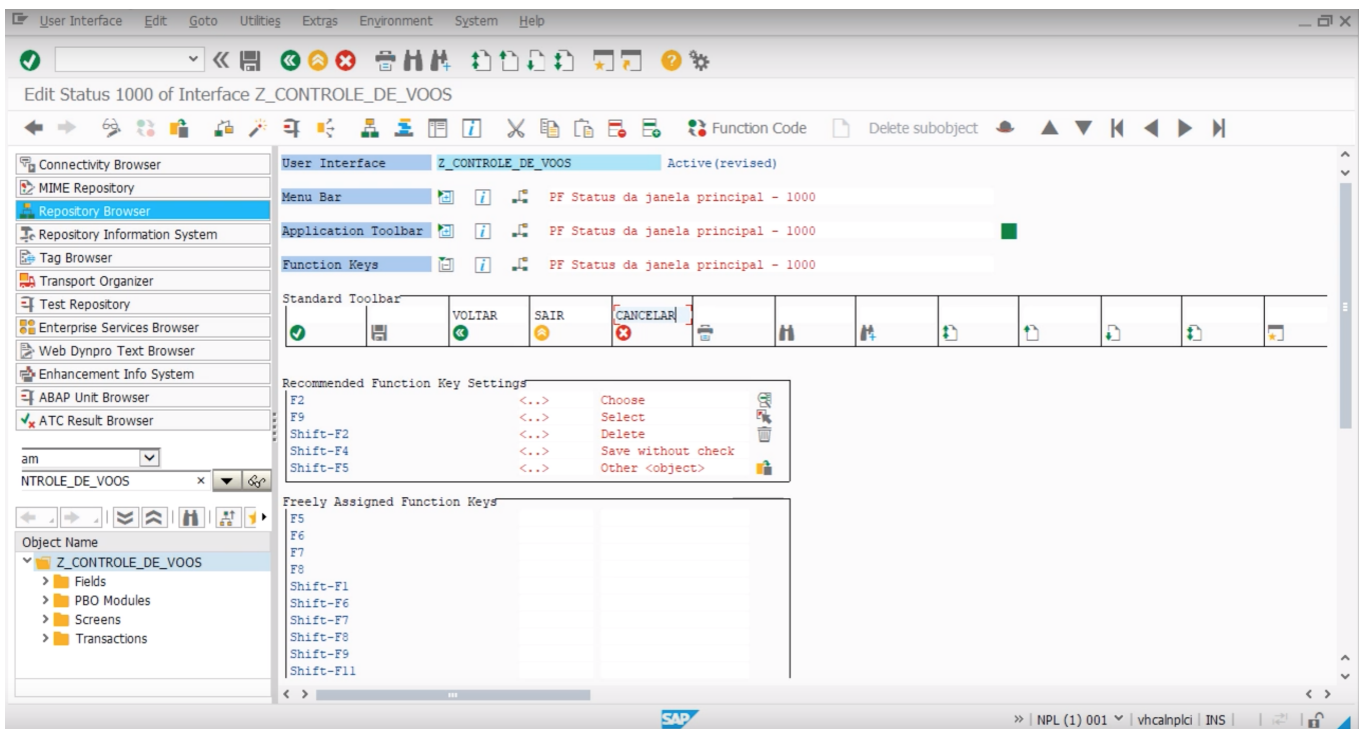
Um popup será exibido para que possamos informar o nome, a descrição e o tipo do PF-STATUS. Então, mantendo o padrão em que o nome do PF-STATUS é igual ao nome da janela, vamos inserir o valor **1000** para o nome e uma descrição e mantemos o tipo como **Normal Screen**.



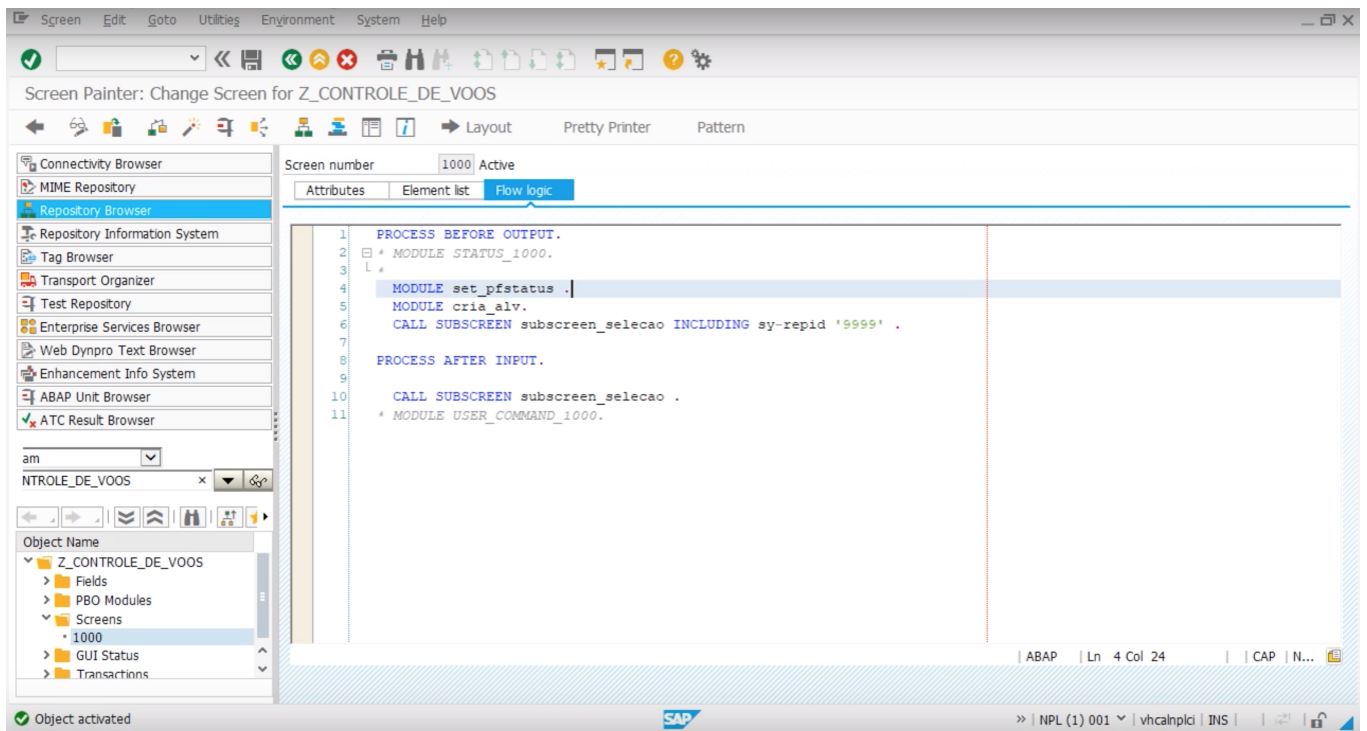
Após confirmação da criação do nosso PF-STATUS , seremos levados a uma nova janela onde iremos configurar os novos botões.



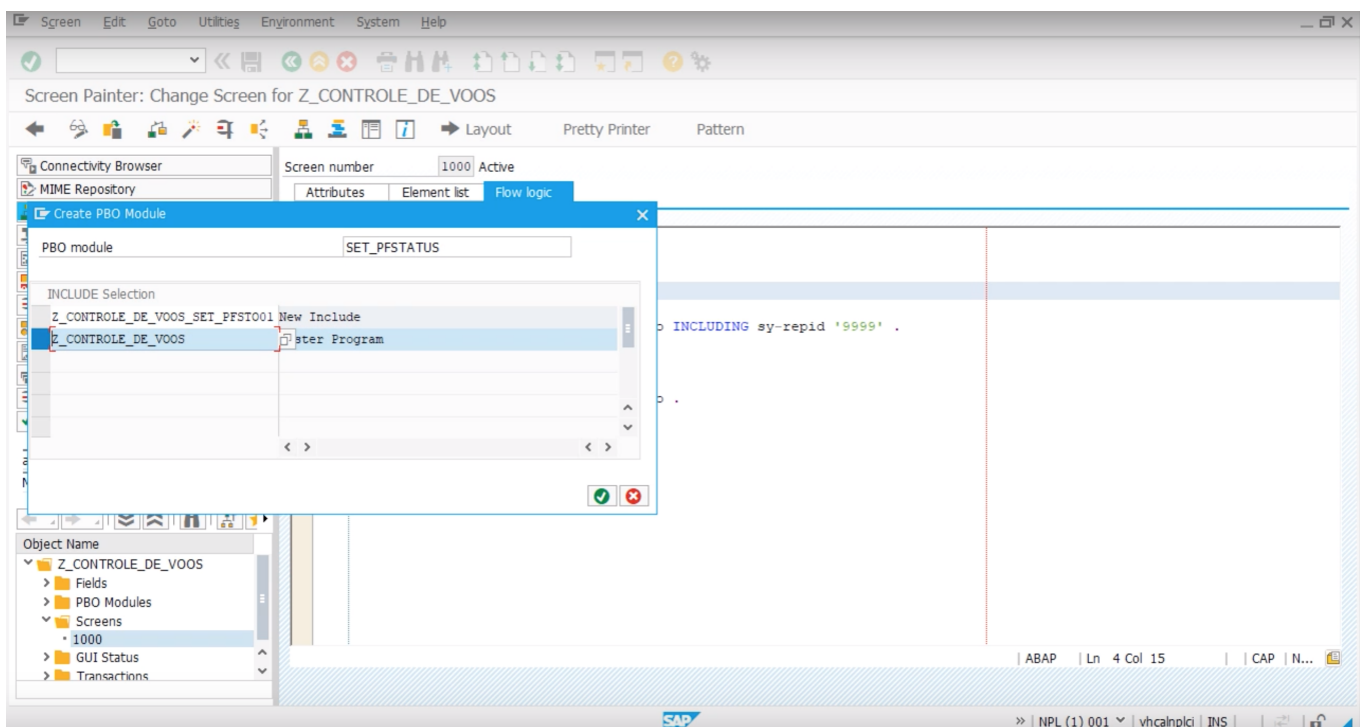
Então, vamos expandir a seção `Functions Keys` e implementar os botões: **Voltar**, **Cancelar** e **Sair**.



No fluxo lógico da aplicação, vamos implementar um novo módulo chamado `MODULE SET_PFSTATUS`.



Este módulo ainda não existe, então, ao tentarmos navegar por ele será aberto um popup para confirmação de sua criação.

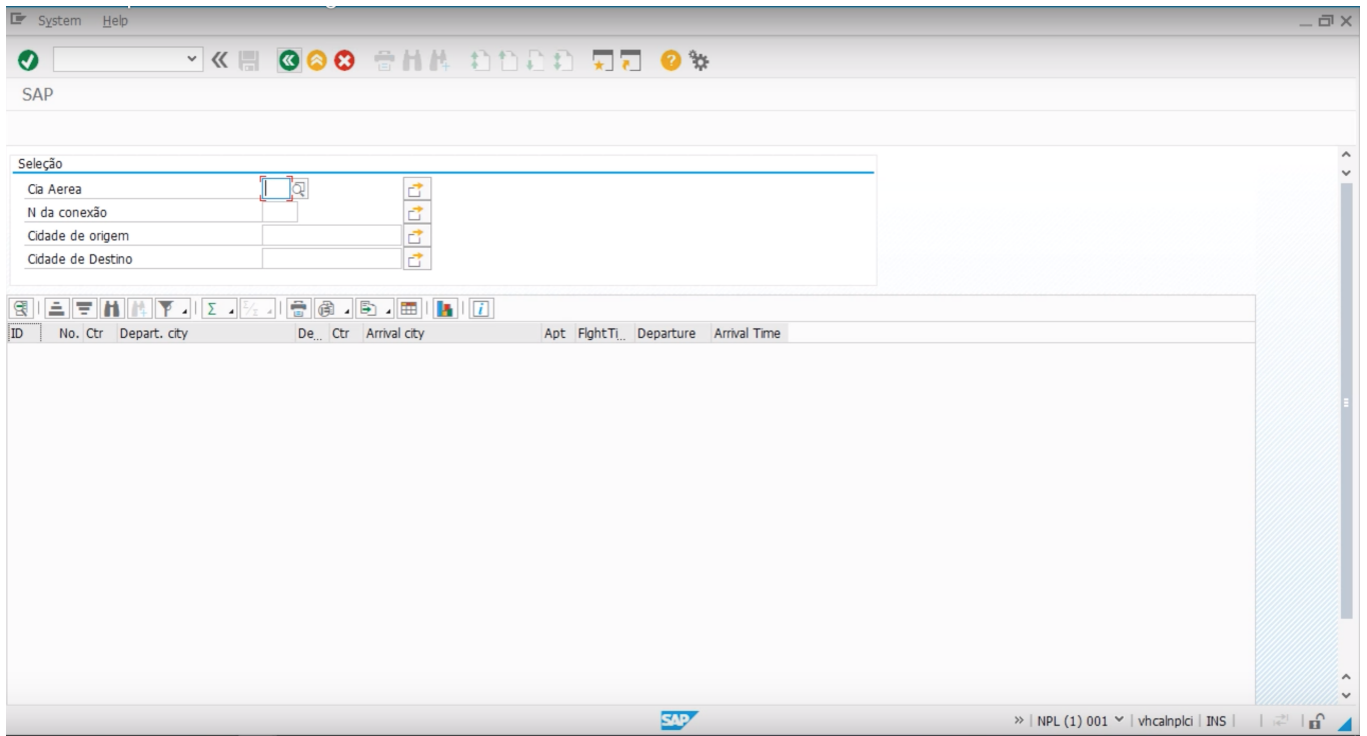


Vamos implementar o seguinte trecho de código dentro do módulo que acabamos de criar:

```

MODULE set_pfstatus OUTPUT.
  SET PF-STATUS '1000'.
ENDMODULE.
  
```

Após essa implementação já somos capazes de executar a aplicação e confirmar que os botões de **Voltar**, **Cancelar** e **Sair** estão ativos. Entretanto, eles ainda não possuem ação nenhuma.

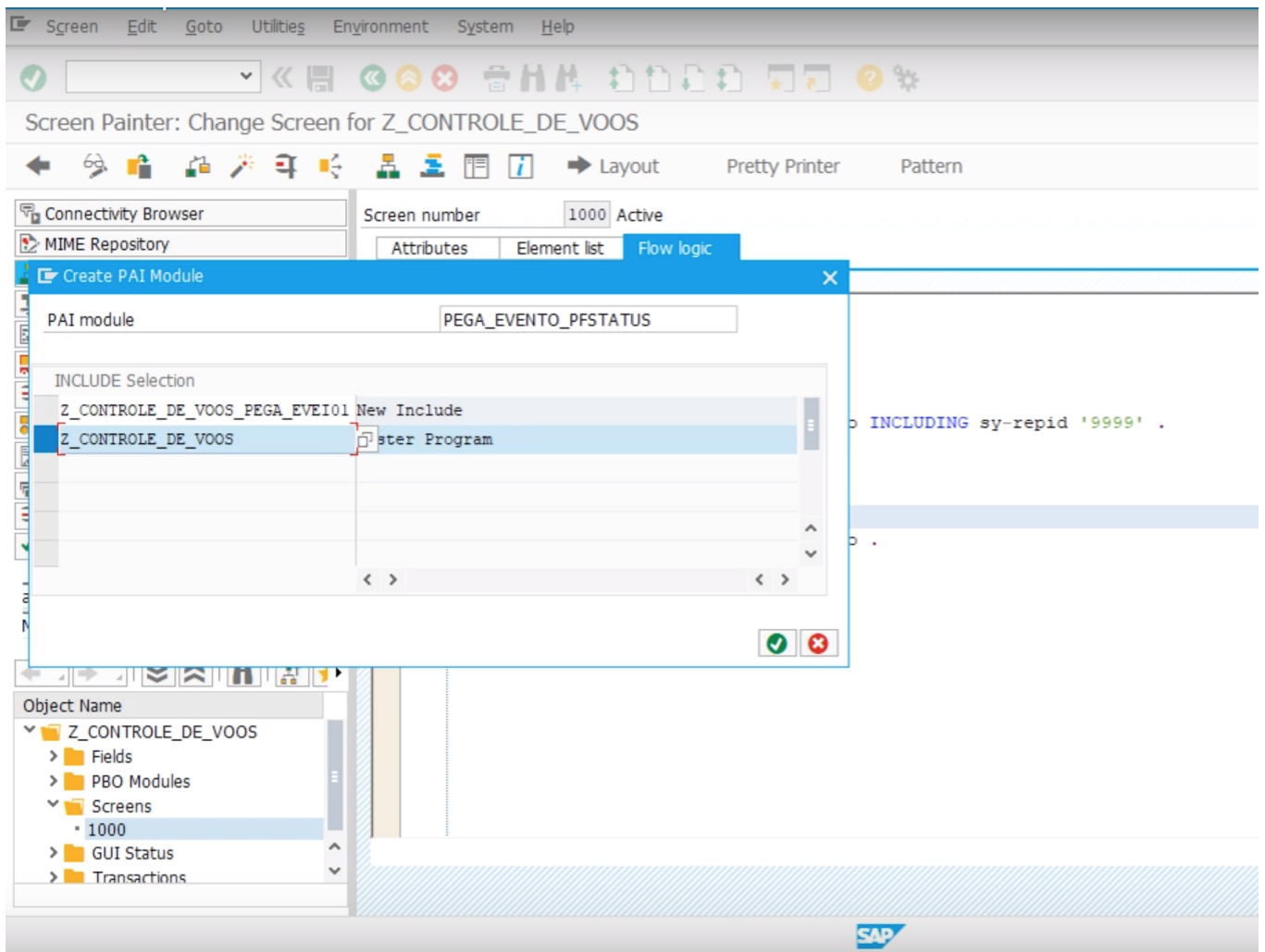


Iniciando a implementação de eventos dos nossos botões, vamos implementar um novo módulo no nosso fluxo lógico, mas dessa vez vamos implementar o módulo no evento PAI.

Vamos chamar esse módulo de `MODULE PEGA_EVENTO_PFSTATUS`.

 aula04_img08.png

Vamos confirmar a criação desse novo módulo e iniciar a sua implementação:



Após criado o módulo vamos inserir o seguinte trecho de código:

```
MODULE pega_evento_pfstatus INPUT.

  CASE sy-ucomm.
    WHEN 'VOLTAR' OR 'CANCELAR' OR 'SAIR'.
      LEAVE PROGRAM.
    ENDCASE.

ENDMODULE.
```

Então, com essa implementação já seremos capazes de sair da nossa aplicação apenas clicando nos botões Voltar, Cancelar e Sair.

Código completo

```
*&-----*
*& Module Pool      Z_CONTROLE_DE_VOOS
*&
*&-----*
*&
*&
*&-----*
PROGRAM z_controle_de_voos.
```

```
DATA spfli TYPE spfli .

DATA: custom_container TYPE REF TO cl_gui_custom_container,
      alv               TYPE REF TO cl_gui_alv_grid,
      fieldcat          TYPE lvc_t_fcat.

DATA it_spfli TYPE STANDARD TABLE OF spfli .

SELECTION-SCREEN: BEGIN OF SCREEN 9999 AS SUBSCREEN.
SELECTION-SCREEN: BEGIN OF BLOCK blk1 WITH FRAME TITLE text-001 .

SELECT-OPTIONS: s_carrid FOR spfli-carrid NO INTERVALS.
SELECT-OPTIONS: s_connid FOR spfli-connid NO INTERVALS.
SELECT-OPTIONS: s_cityfr FOR spfli-cityfrom NO INTERVALS.
SELECT-OPTIONS: s_cityto FOR spfli-cityto NO INTERVALS.

SELECTION-SCREEN: END   OF BLOCK blk1.
SELECTION-SCREEN: END OF SCREEN 9999.

MODULE cria_alv OUTPUT.

  DATA coluna LIKE LINE OF fieldcat .

  CREATE OBJECT custom_container
    EXPORTING
      container_name = 'CC_ALV'.

  CREATE OBJECT alv
    EXPORTING
      i_parent = custom_container.

  CALL FUNCTION 'LVC_FIELDCATALOG_MERGE'
    EXPORTING
      i_structure_name = 'SPFLI'
    CHANGING
      ct_fieldcat      = fieldcat.

  LOOP AT fieldcat INTO coluna .

    IF coluna-fieldname = 'DISTANCE' OR
       coluna-fieldname = 'DISTID' OR
       coluna-fieldname = 'FLTYPE' OR
       coluna-fieldname = 'PERIOD' .

      coluna-no_out = 'X' .

    MODIFY fieldcat FROM coluna .

  ENDIF .

ENDLOOP .

CALL METHOD alv->set_table_for_first_display
  CHANGING
    it_outtab      = it_spfli
```

```
it_fieldcatalog = fieldcat.
```

```
CALL METHOD alv->refresh_table_display( ) .
```

```
ENDMODULE.
```

```
MODULE set_pfstatus OUTPUT.
```

```
SET PF-STATUS '1000'.
```

```
ENDMODULE.
```

```
MODULE pega_evento_pfstatus INPUT.
```

```
CASE sy-ucomm.
```

```
WHEN 'VOLTAR' OR 'CANCELAR' OR 'SAIR'.
```

```
LEAVE PROGRAM.
```

```
ENDCASE.
```

```
ENDMODULE.
```