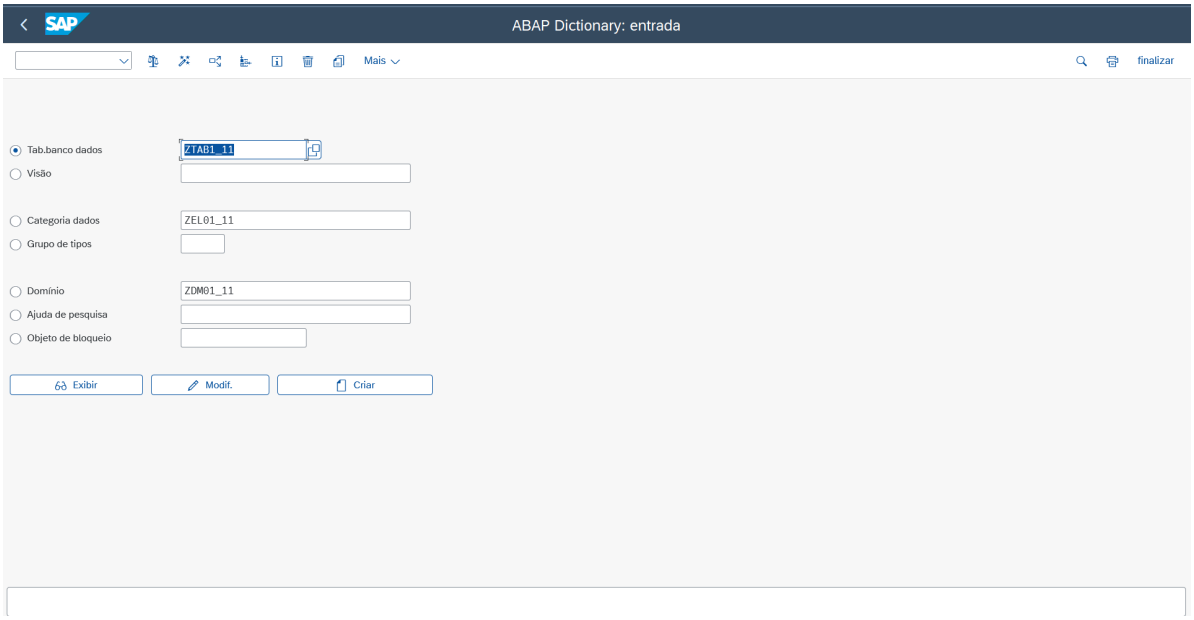
SAP ABAP

2.

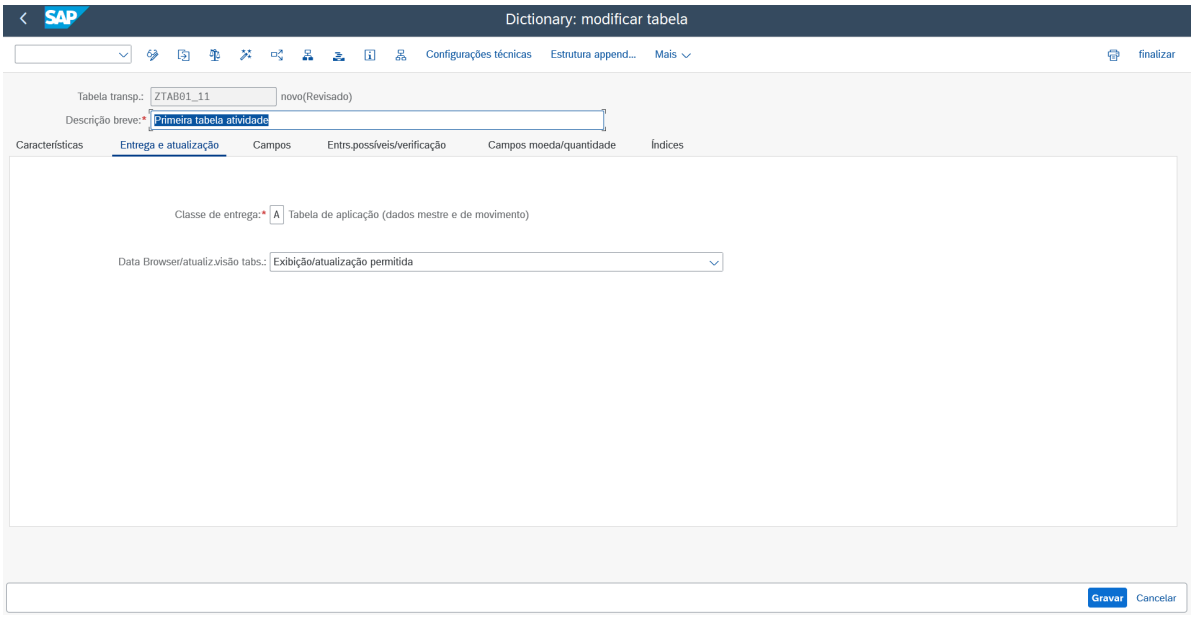
Criar primeira tabela - tabela transparente

1. Fazer o logon no ambiente correto, informado pelo cliente ou treinamento;
2. digitar na caixa de comando o código **SE11**;
3. outra informação importante é que todo objeto customizado deve ter seu nome iniciado com Z ou Y, por isso o elemento abaixo tem nome ZEL...

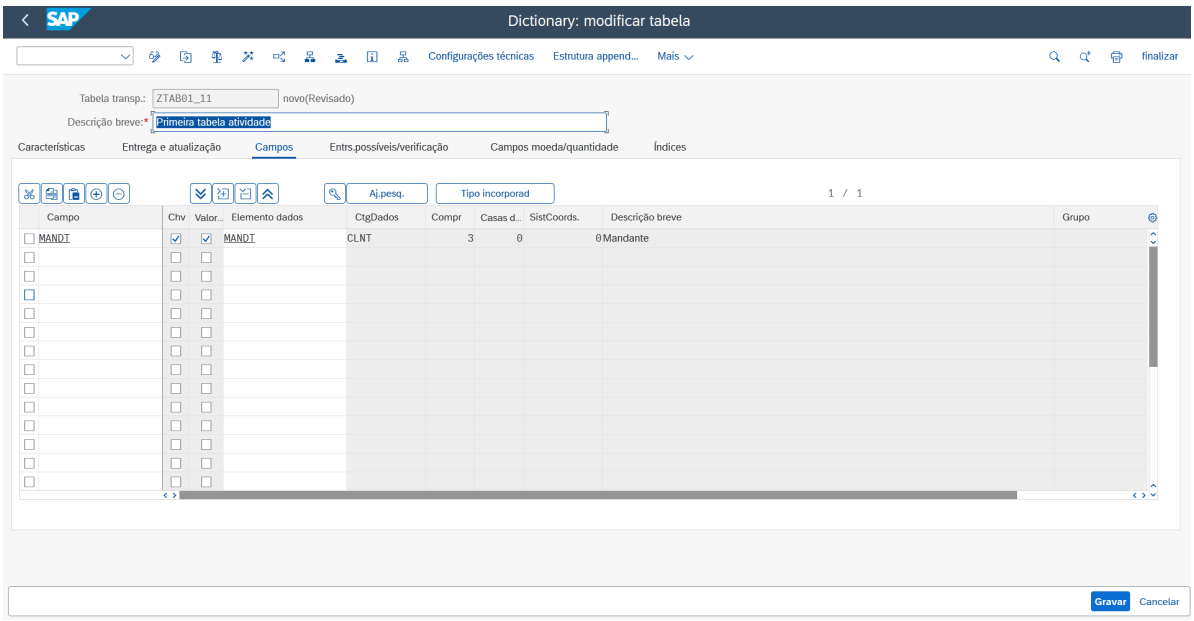
   Tela de dicionário de dados:



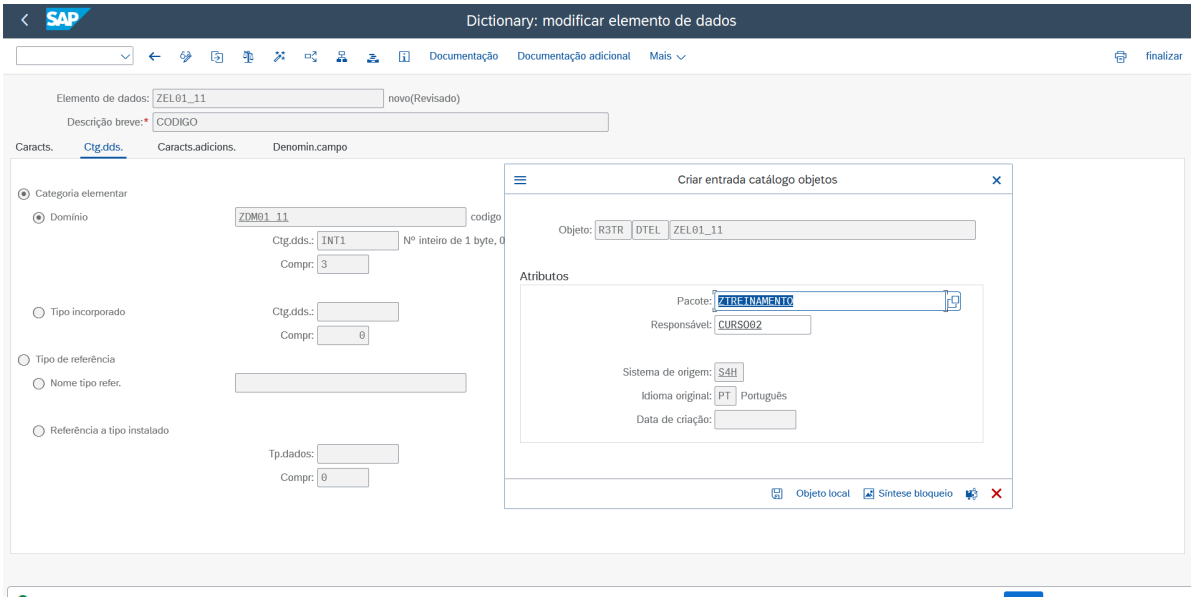
3. com a caixa do Tab. banco dados = digitar o nome da sua tabela de dados;
4. clicar no botão [criar];
5. Na tela Dicionário: modificar tabela, inserir a breve descrição da tabela;
6. Ainda na tela Dicionário: modificar tabela aba Entrega atualização preencher o campo Classe de entrega com a categoria [A], Data Browser: opção Exibição/ Atualização permitida



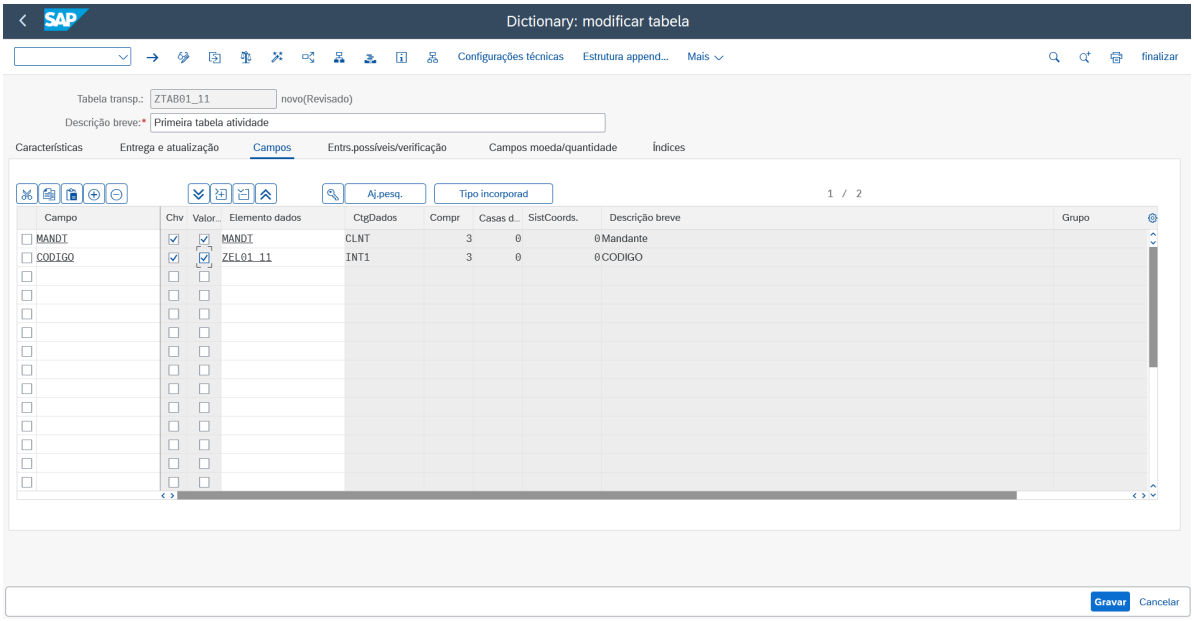
7. Agora vamos clicar na aba campos
8. primeiro campo para essa aba será MANDT(mandante) e esse terá Ch e val marcados, lembrando que o campo caso ainda não exista vc pode digitar seu nome ai, clicar duas vezes no nome que vai ficar sublinhado que o programa te direciona para uma tela onde poderá criar esse campo.



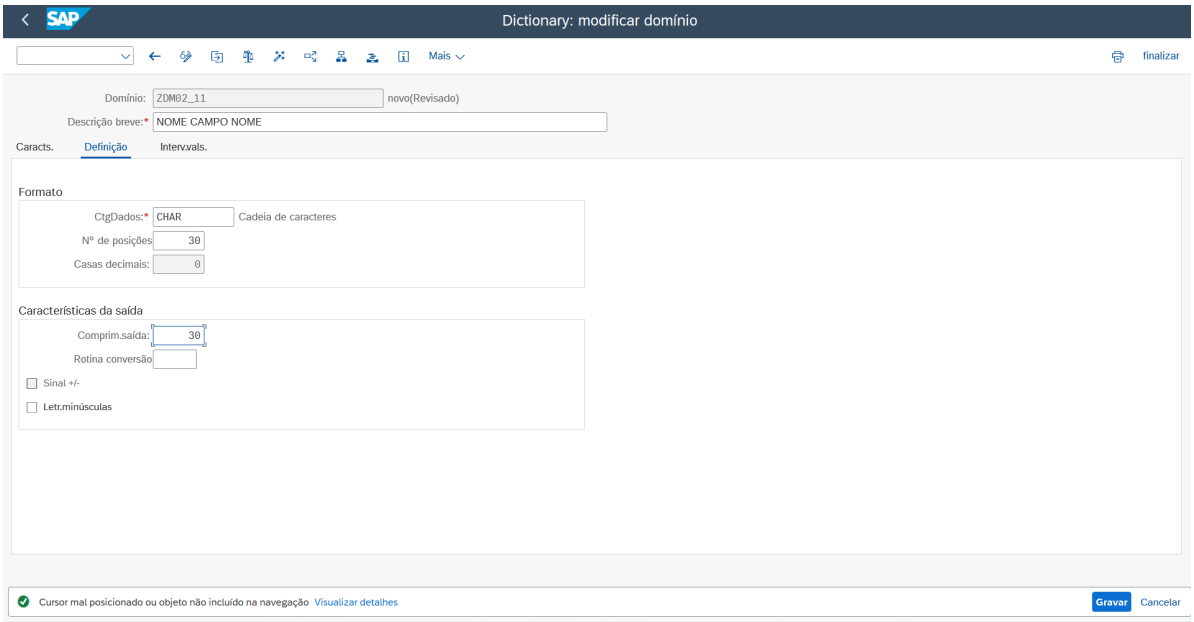
9. após o campo mandante sim vamos adicionar o campo código.



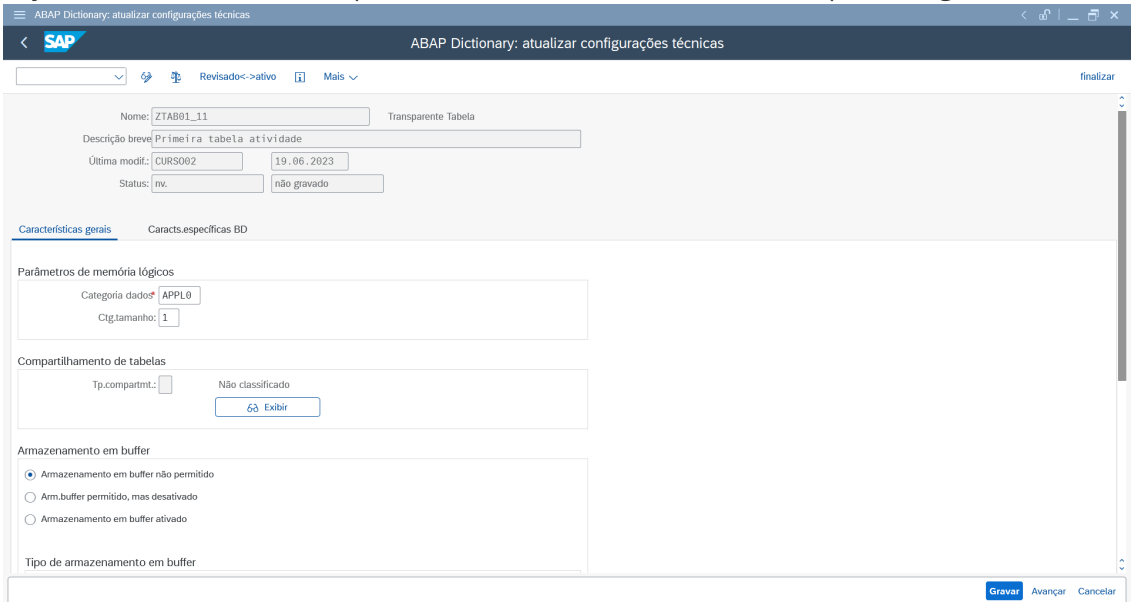
10. após clicar duas vezes no campo ele vai pedir informações do campo, como tamanho do mesmo, tipo de dados e pacote, acima temos um exemplo de preenchimento.
11. preencher tbm o Denomcampo com o nome do campo novamente.
12. Após isso vamos voltar para a tela dos campos da tabela e vamos chamar esse elemento na mesmo o resultado será:



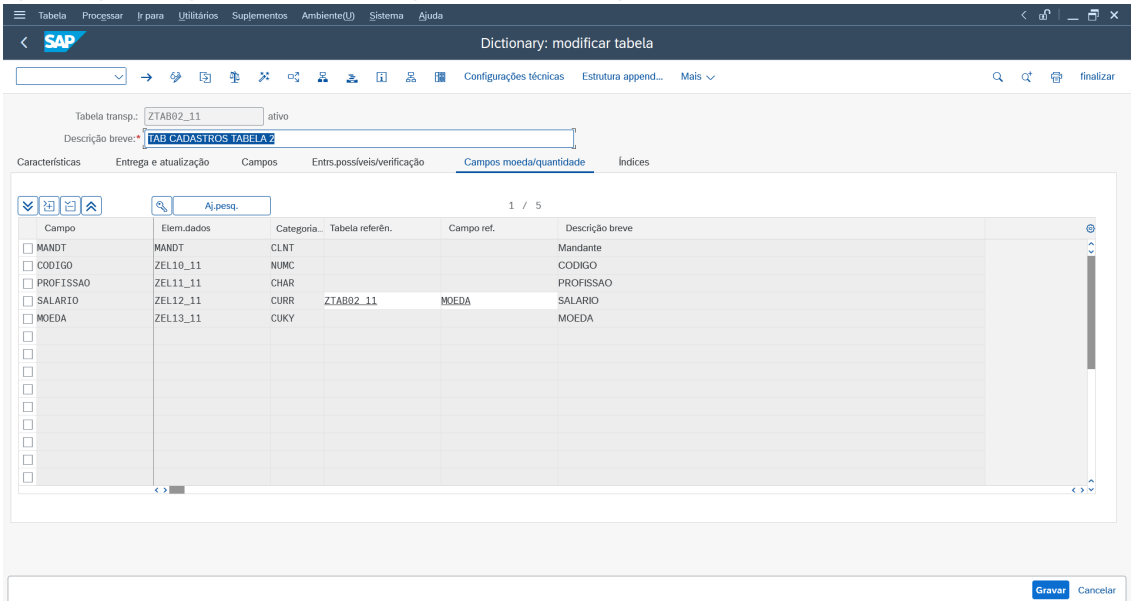
12. os passos anteriores devem se seguir para os demais campos da sua tabela sempre criando Domínio e Elemento



13. após informar todos os campos necessários, podemos ativar a tabela, para isso, caso não haja inconsistências nos campos da tabela, você vai ser direcionado para a seguinte tela:



14. no campo categoria escolher a opção APPL0 no campo ctg. tamanho podemos deixar a opção 1
15. apenas para registro de um exemplo de dado do tipo salario em tabela:

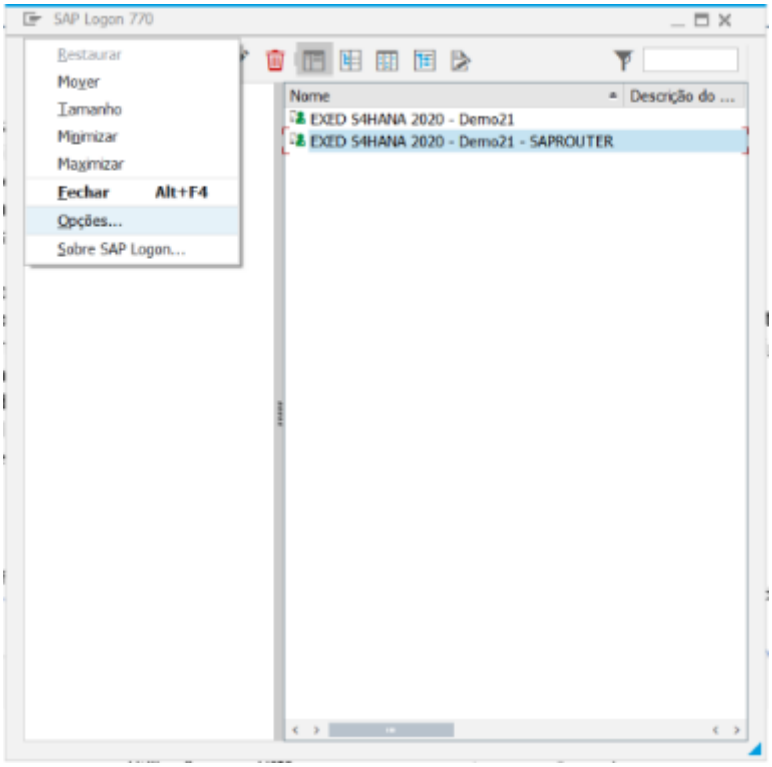


16. note na imagem acima que o campo salario esta relacionado com o campo moeda onde poderemos informar o Símbolo da moeda para o dinheiro

Trocando o tema do SAP LOGON:

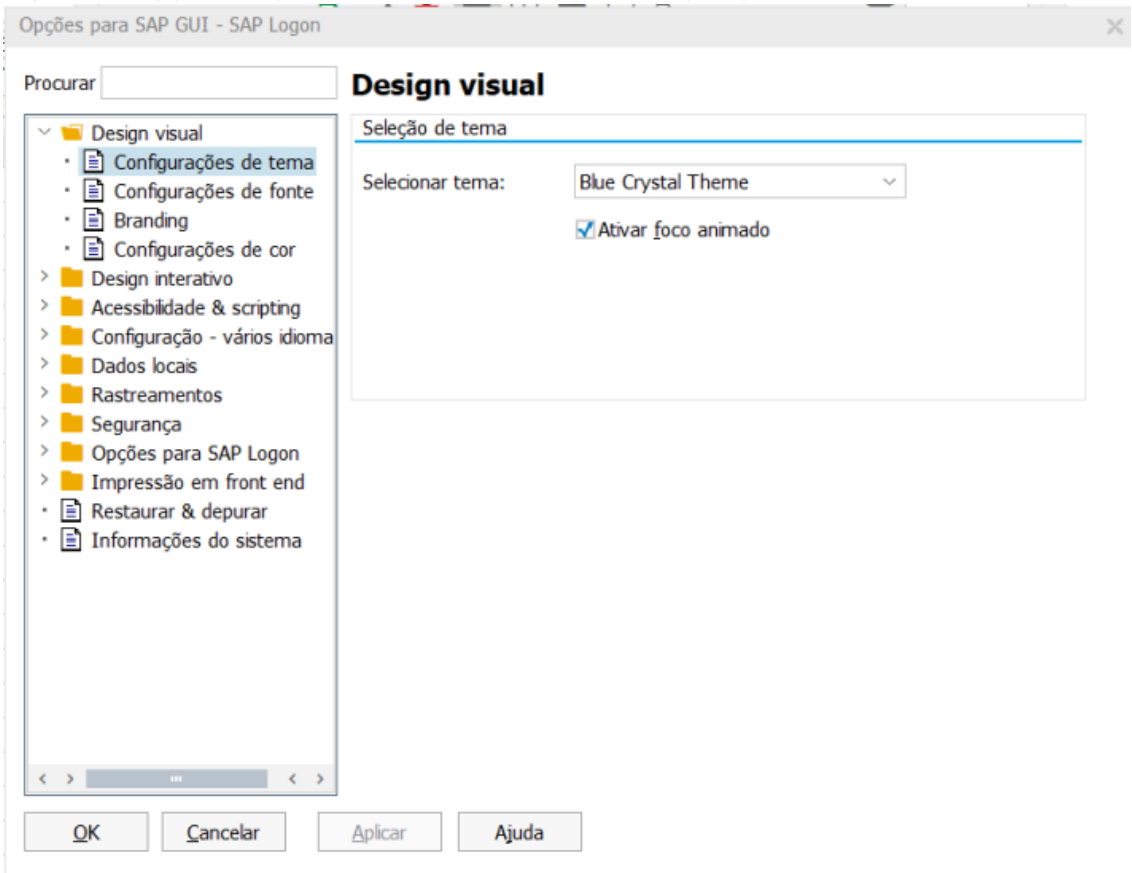
A tela do SAP LOGON dispõem de alguma opções de temas para trabalho, para alterar o tema faça o seguinte passo a passo:

1. na tela inicial do SAP Logon, antes de realizar o login você vai clicar no primeiro botão no canto superior esquerdo da tela:



2. clique na opção opções

3. expandir a opção Designer visual e clicar na opção Configurações de tema



4. selecionar no campo selecionar tema o tema desejado, estamos utilizando nesse documento o tema: Blue Crystal Theme
5. clicar no botão **Aplicar** e depois em **OK**
6. caso ainda não esteja vendo as alterações, feche o programa e abra novamente

ABAP Open SQL

O ABAP faz uso de SQL para suas consultas ao banco de dados e aqui a lógica é a mesma para quem já fez uso de SQL

- **Selecionar dados** - Comando `SELECT campo From Tabela` também é permitido utilizar critérios WHERE e relação entre tabelas com JOIN;
- **Inserir dados** no banco - comando `INSERT campo = 'valor' INTO tabela`
- **Alterar dados** no banco - comando `UPDATE tabela SET campo = "informação" WHERE codigo = 1` ou `MODIFY tabela FROM campo = "informação"`, diferença entre esses dois comandos é que o Modify cria um novo registro caso não encontrar o mesmo para atualizar no banco.
- **Apagar dados** no banco = comando `DELETE * FROM tabela WHERE codigo = 1`, uma vez apagado o registro não é possível recuperar o mesmo.

Trabalhando informações na tabela

Report - é um recurso que te permite processar os dados, como por exemplo exibir os dados. esse recurso é feito através da **transação SE38.** Esse recurso é utilizado para gerar relatórios com os dados da sua base de dados. Esse pode ser executado on-line ou em background, a diferença entre os dois modos é que o modelo background permite executar processamentos de grandes volumes de dados sem travar a tela para o usuário.

É importante manter a estrutura do programa organizada e isso pode ser feito dividindo o Reporte com linhas do cabeçalho e comentários indicando partes do código. Exemplo, podemos indicar através do comentário/ cabeçalho que uma parte do código é responsável pelos dados do relatório com dados dos funcionários da base de dados.

> OBS: veja se seu cliente já possui um modelo de cabeçalho e siga sempre o padrão já utilizado.

Outra dica sobre organização do código é separar as declarações por blocos e não criar tudo junto e misturado = tabelas, constantes...

- Tabelas internas - iniciamos o nome com t, exemplo: t_table
- variáveis globais - iniciamos o nome com v, exemplo: v_codigo
- constantes - iniciamos o nome com c, exemplo: c_dado

Tela de seleção:

Inicia-se com Selection-Screen Begin e encerra em Selection-Screen end

Condicional IF:

```
IF V_X EQ 0.
    WRITE 'TEXTO DE SAIDA'.
ELSE.
    WRITE 'TEXTO DE SAIDA'.
ENDIF.
```

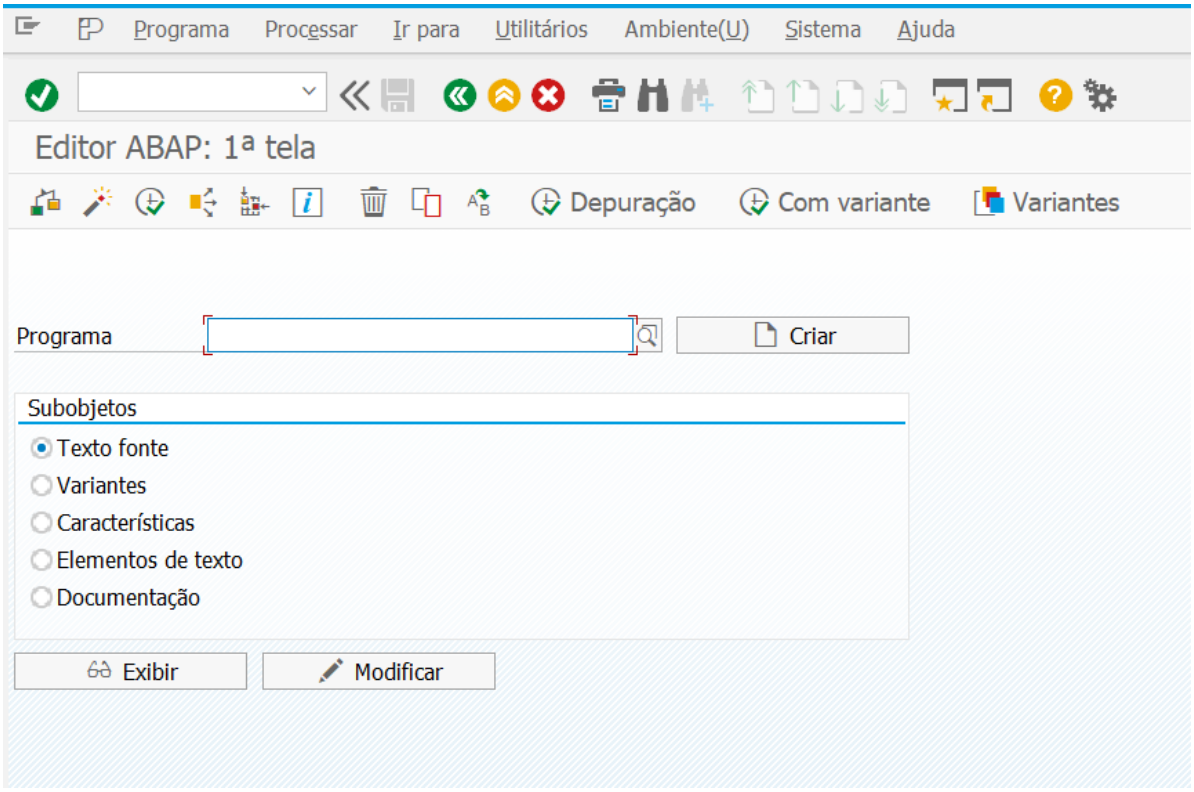
ou podemos ainda usar **CASE**:

```
CASE v_dado,
    WHEN 'valoresperado-A'.
        WRITE 'TEXTO DE SAIDA'.

    WHEN 'valoresperado-B'.
        WRITE 'TEXTO DE SAIDA'.

    WHEN otkers.
ENDCASE.
```

Tela inicial do SE38:



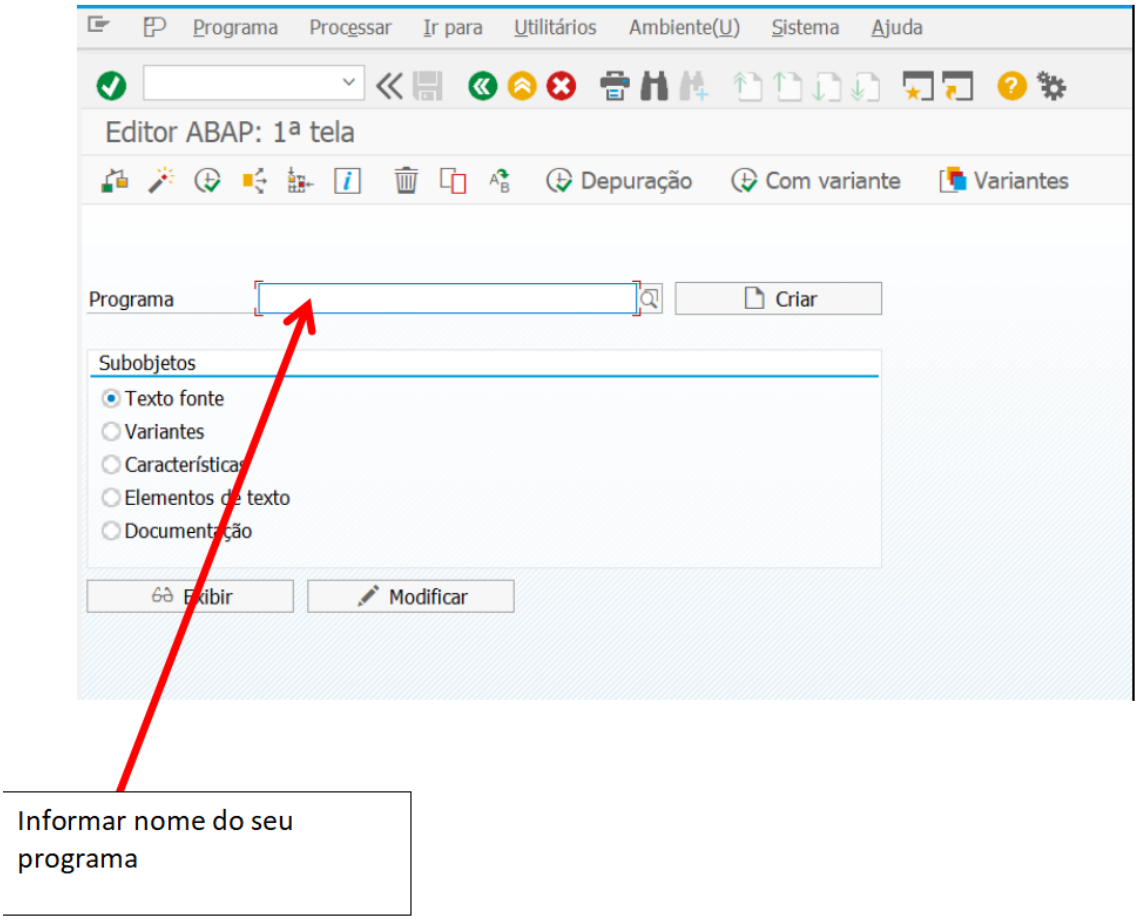
Criar Report

Para criar um Report vamos seguir a padronização a seguir: o nome deve ser ZREPX_XX onde o X é o número de report criado: primeiro, segundo... e XX é o número de aluno(a).

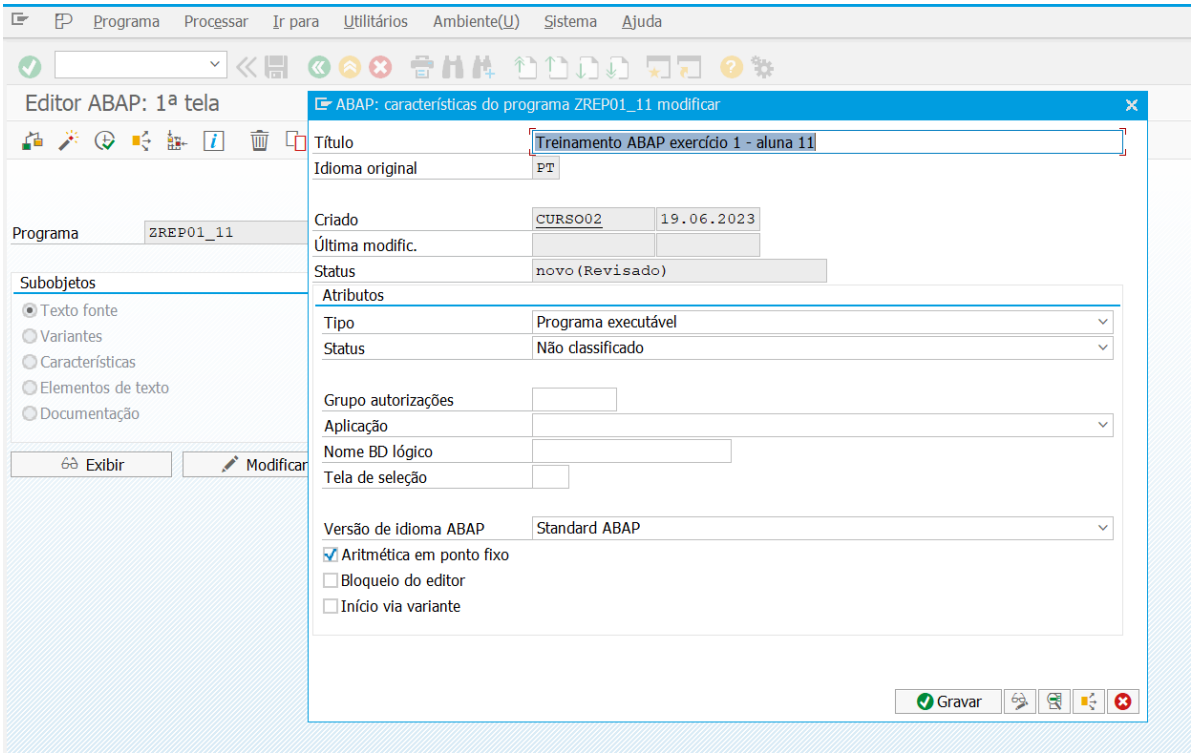
Devemos retornar o nome e pais registrado na tabela ZTAB01_xx criada na tabela 1 criada.

Passo a passo:

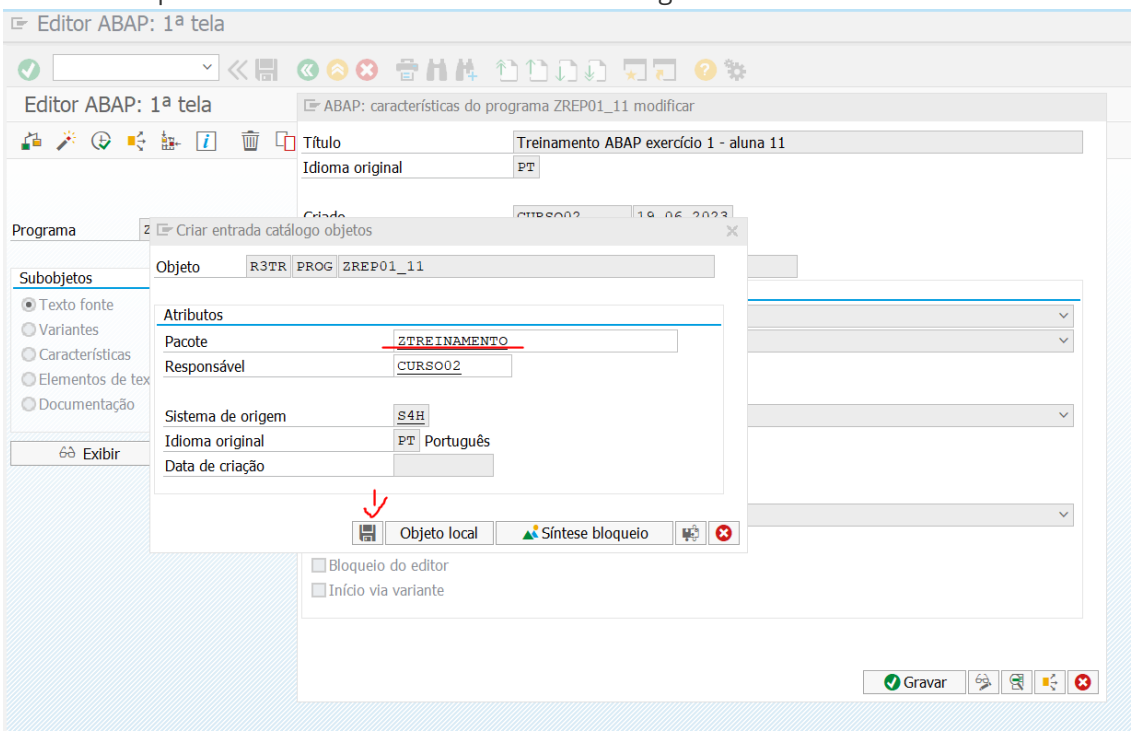
1. Preencher o nome do programa ZREP01_XX e clicar no botão Criar



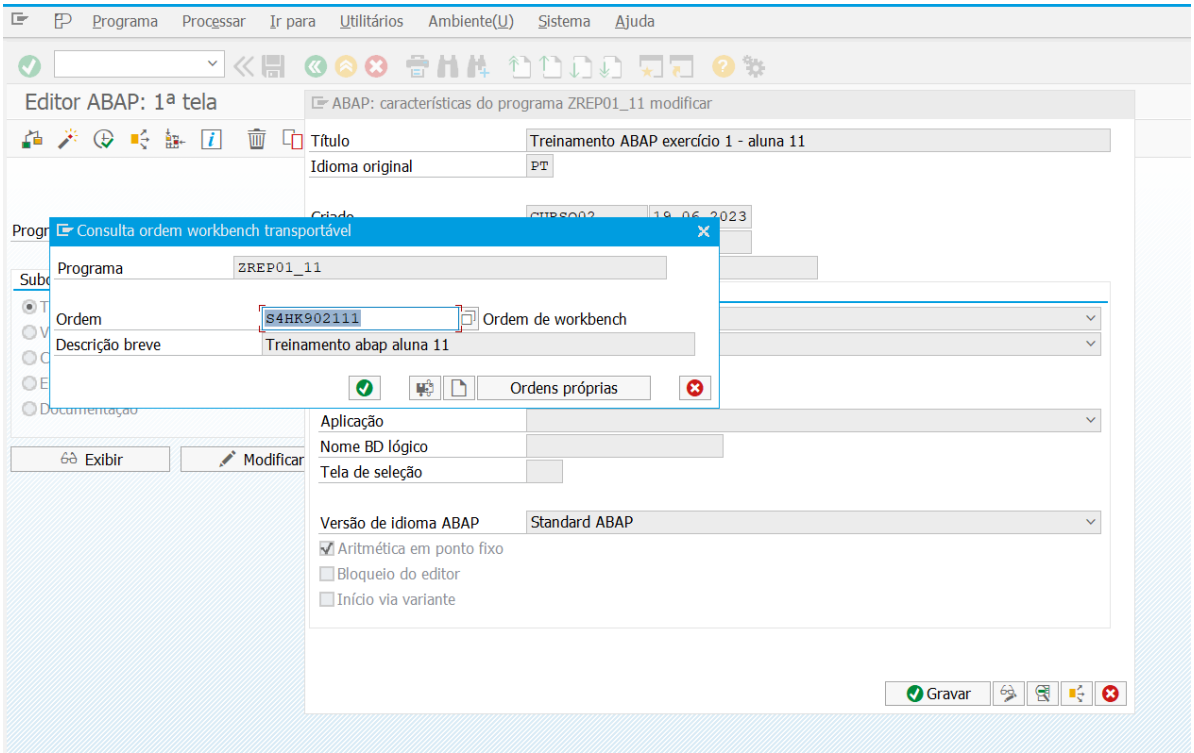
2. Em título informar um título para seu programa;
3. Em tipo selecionar a opção treinamento executavel;
4. Clicar no botão [Gravar]



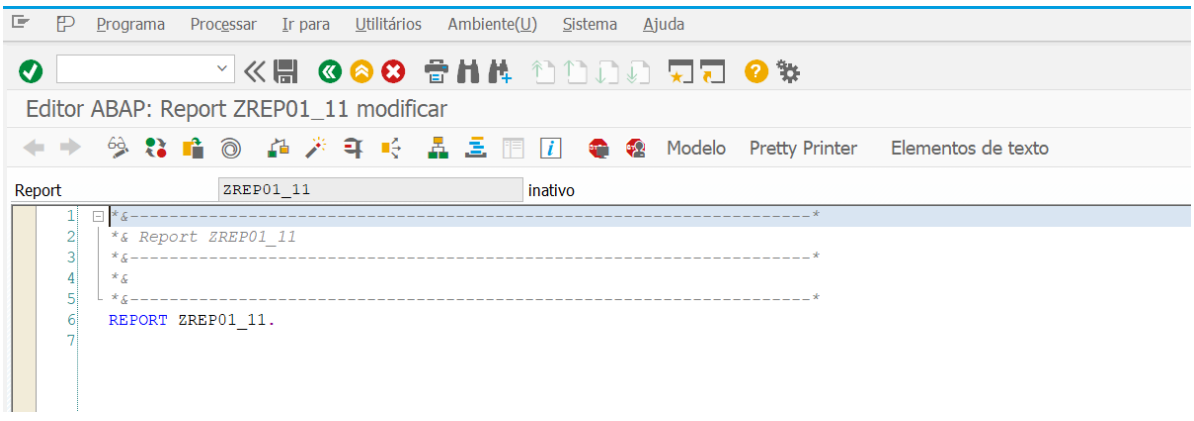
5. Preencher o pacote com ZTREINAMENTOS e clicar em gravar



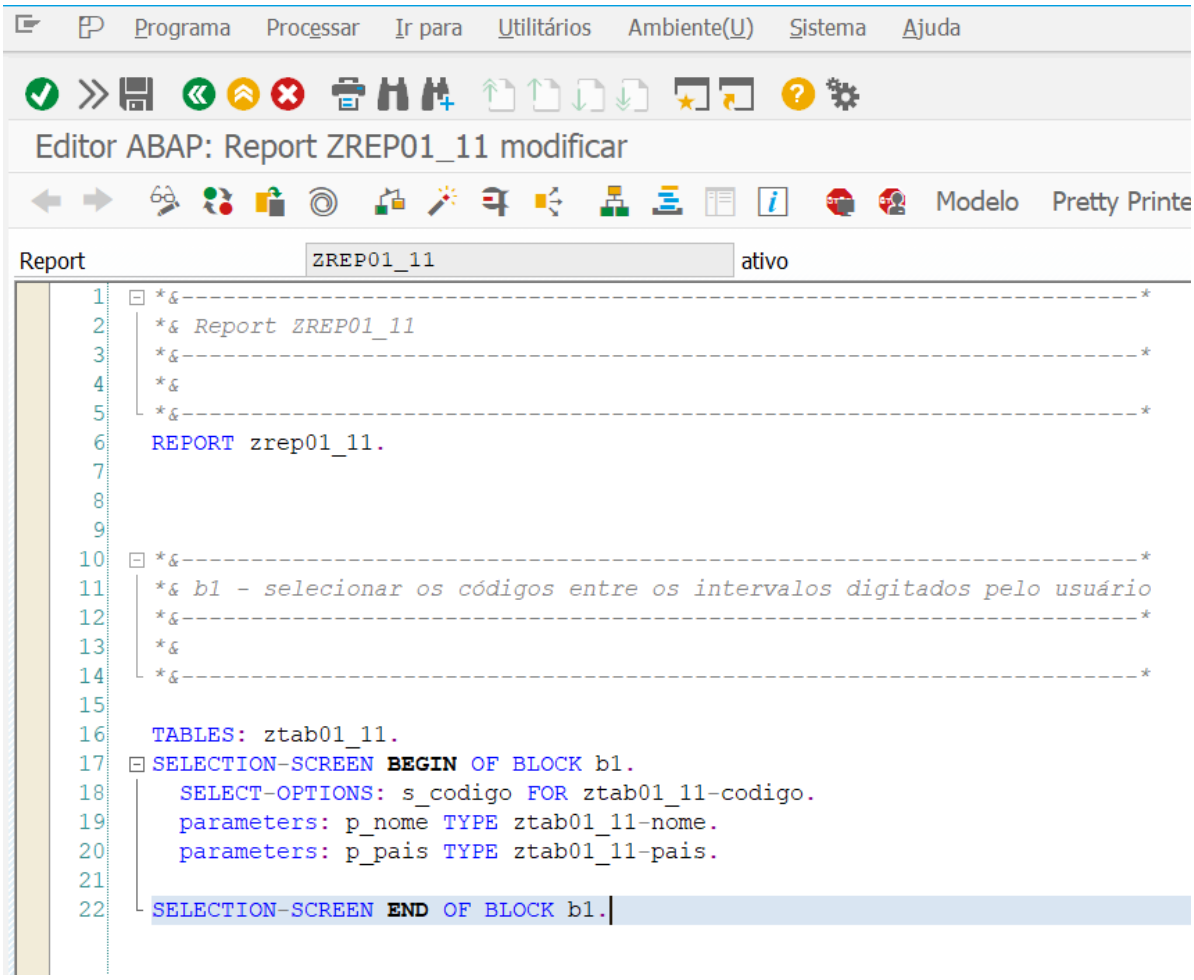
6. escolher sua request e salvar



7. o resultado esperado é:



8. Feito isso vamos preencher com o código que faz a tela desejada para o usuário



linha 16 - comando `Tables: zrep01_11.` : define a tabela que será utilizada para esse programa

linha 17 - `SELECTION-SCREEN BEGIN OF BLOCK b1.` esta iniciando a ação de seleção do report e nomeando o mesmo como **b1**

linha 18 - `SELECT-OPTIONS: s_codigo FOR ztab01_11-codigo.` essa linha indica que teremos um elemento do tipo randon - intervalo de seleção, estamos nomeando esse elemento de s_codigo e esse vai fazer sua busca na tabela ztab01_11 campo código.

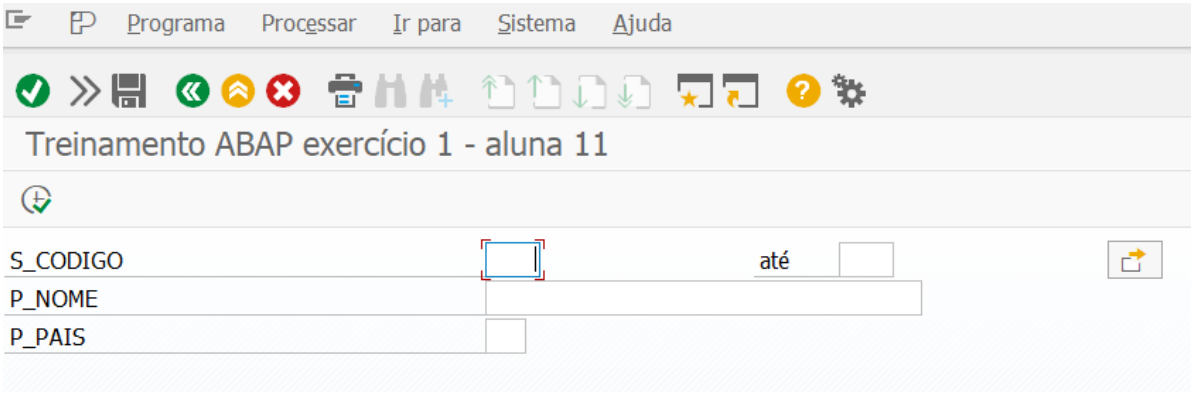
linha 19 - `parameters: p_nome TYPE ztab01_11-nome.` estamos passando um parametro de retorno de dados que retornará o campo nome da tabela

linha 20 - `parameters: p_pais TYPE ztab01_11-pais.` estamos passando um parametro de retorno de dados que retornará o campo pais da tabela

linha 22 - `SELECTION-SCREEN END OF BLOCK b1.` esta finalizando a ação do SELECTION de nome b1

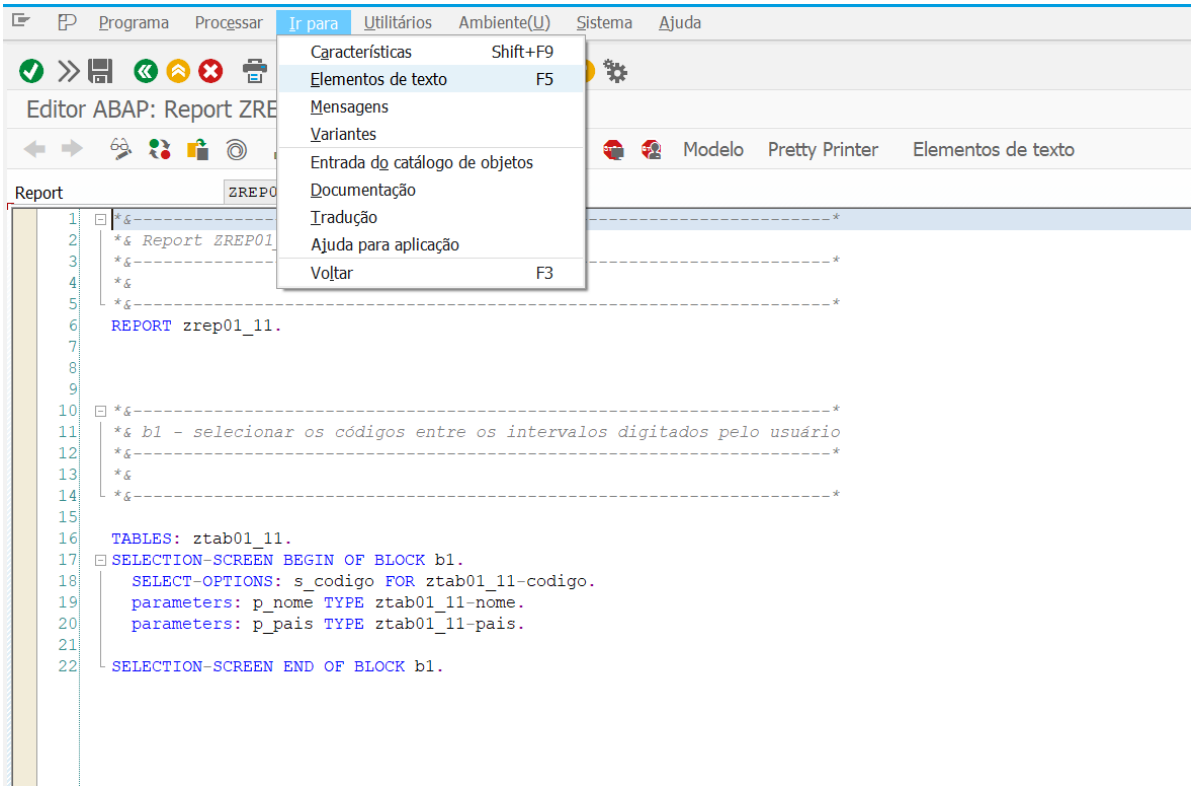
> Note que todas as linhas de comando terminam com (.) ponto, essa regra deve ser seguida para não ter erros durante o desenvolvimento. Você ainda poderá usar (,) virgula, caso for chamar um elemento do mesmo tipo na linha abaixo.

Finalizada a digitação do código podemos ativar o programa, clicando no botão ativar e caso não de nenhum erro, podemos usar o botão Direto ou Executar, assim teremos o seguinte resultado:

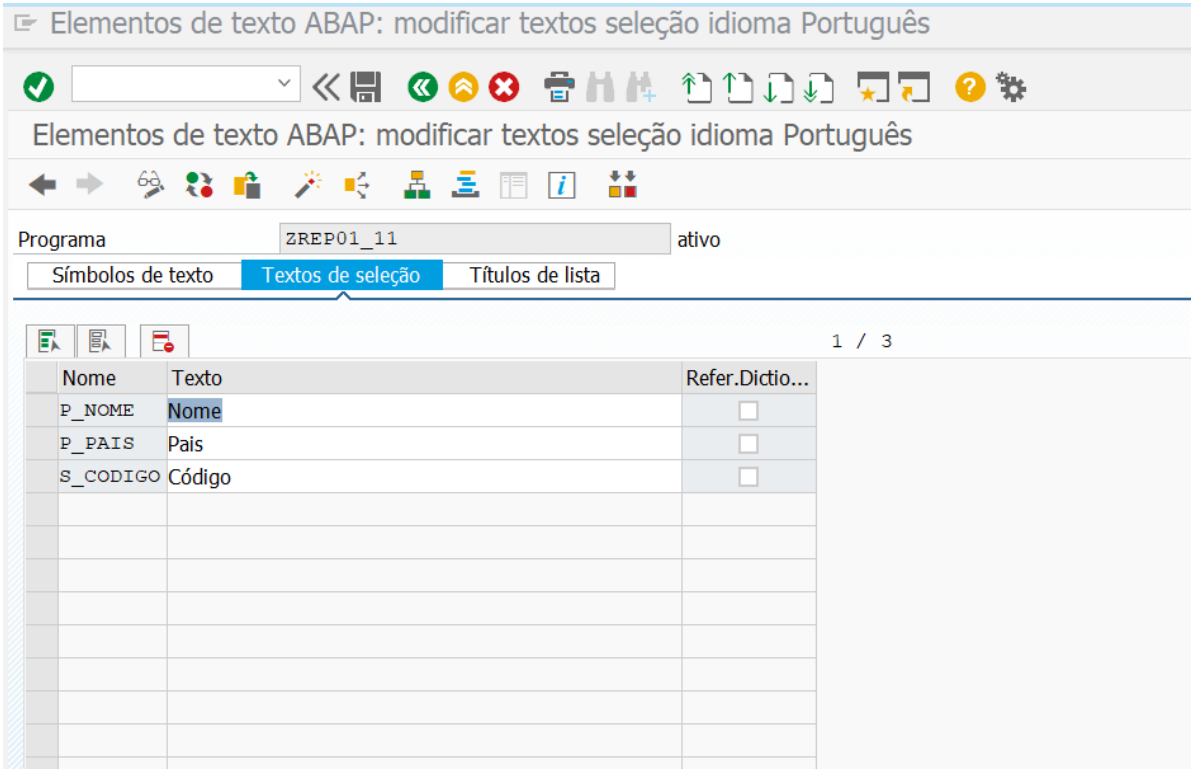


Vamos agora tratar os rótulos do formulário

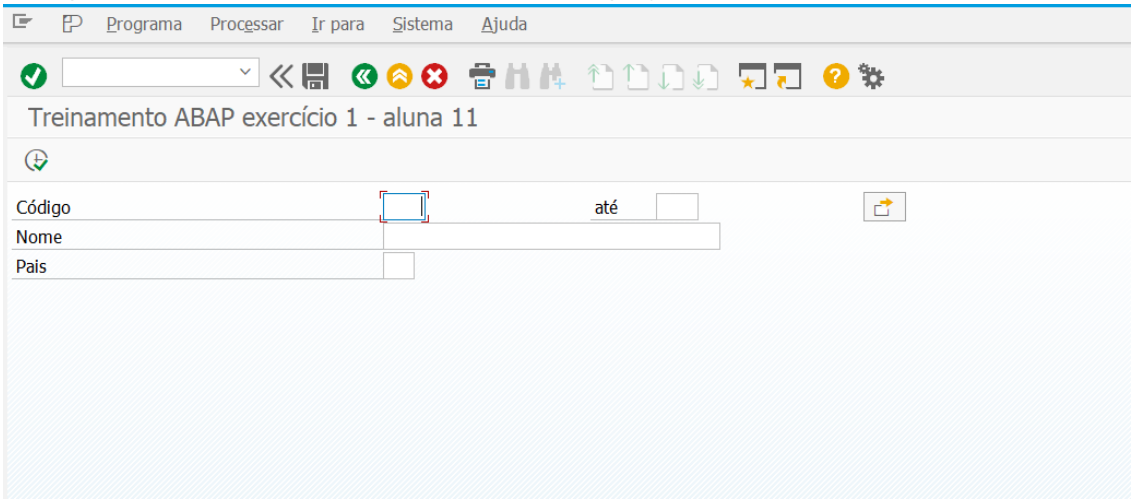
1. Voltar para tela de código e clicar no menu opção IR PARA -> opção Elementos de texto



2. Preencher o texto que deseja ter de rótulo para o nosso formulário



3. volte para a tela anterior, resultado ao executar o projeto novamente

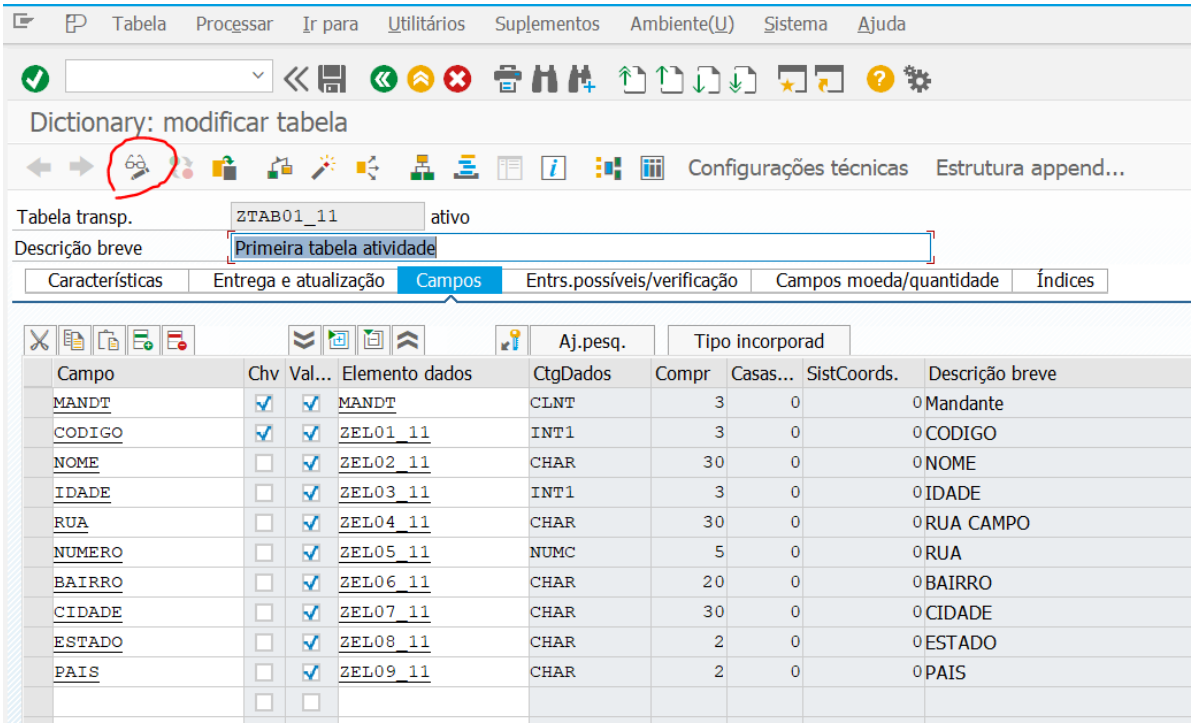


Manutenção dos dados da tabela

Para inserir, alterar ou deletar dados da tabela customizada, ou seja, tabelas que nos criamos, vamos utilizar a transação de código SM30.

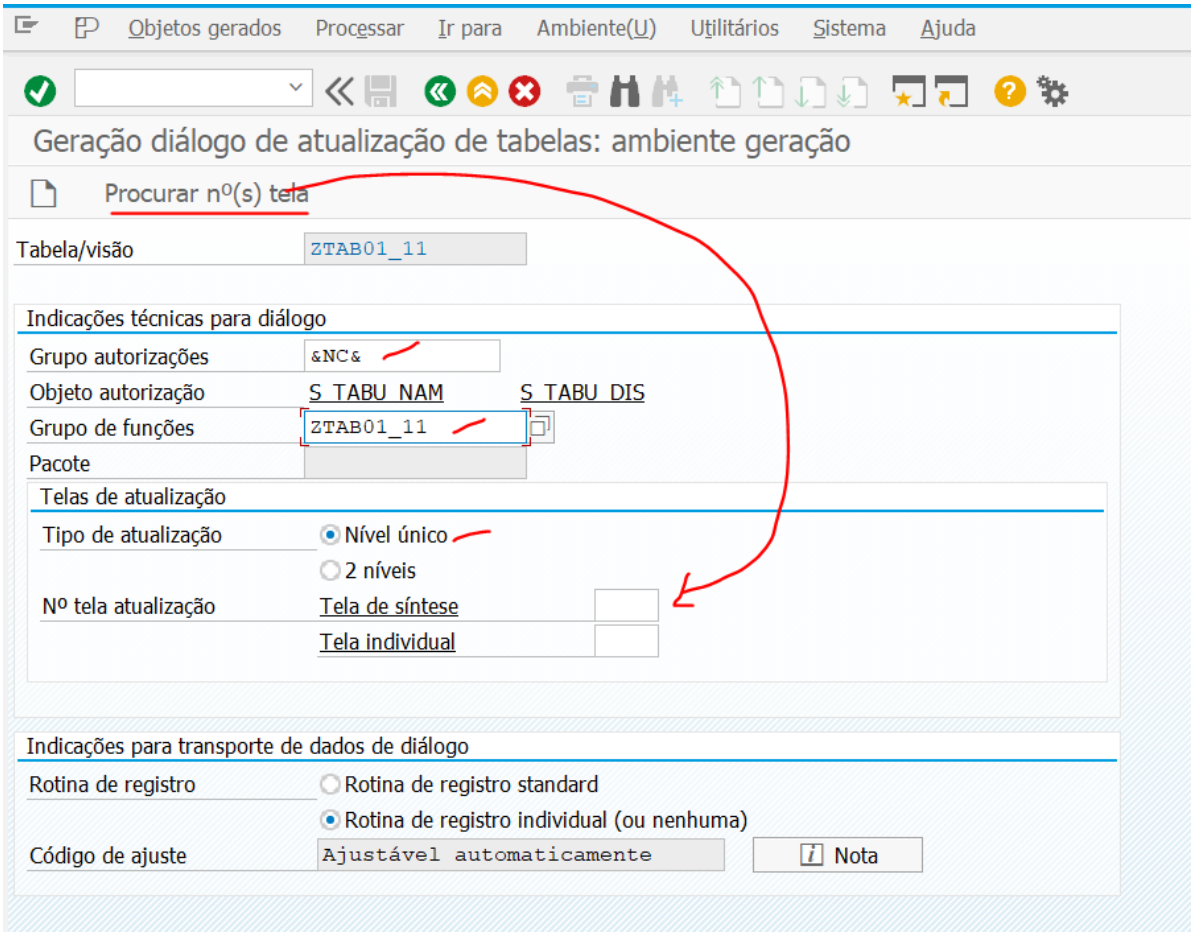
Para inserir dados na nossa tabela vamos seguir o seguinte passo a passo:

1. vamos abrir a tabela criada na operação SE11
2. vamos informar a tabela e clicar no botão modificar



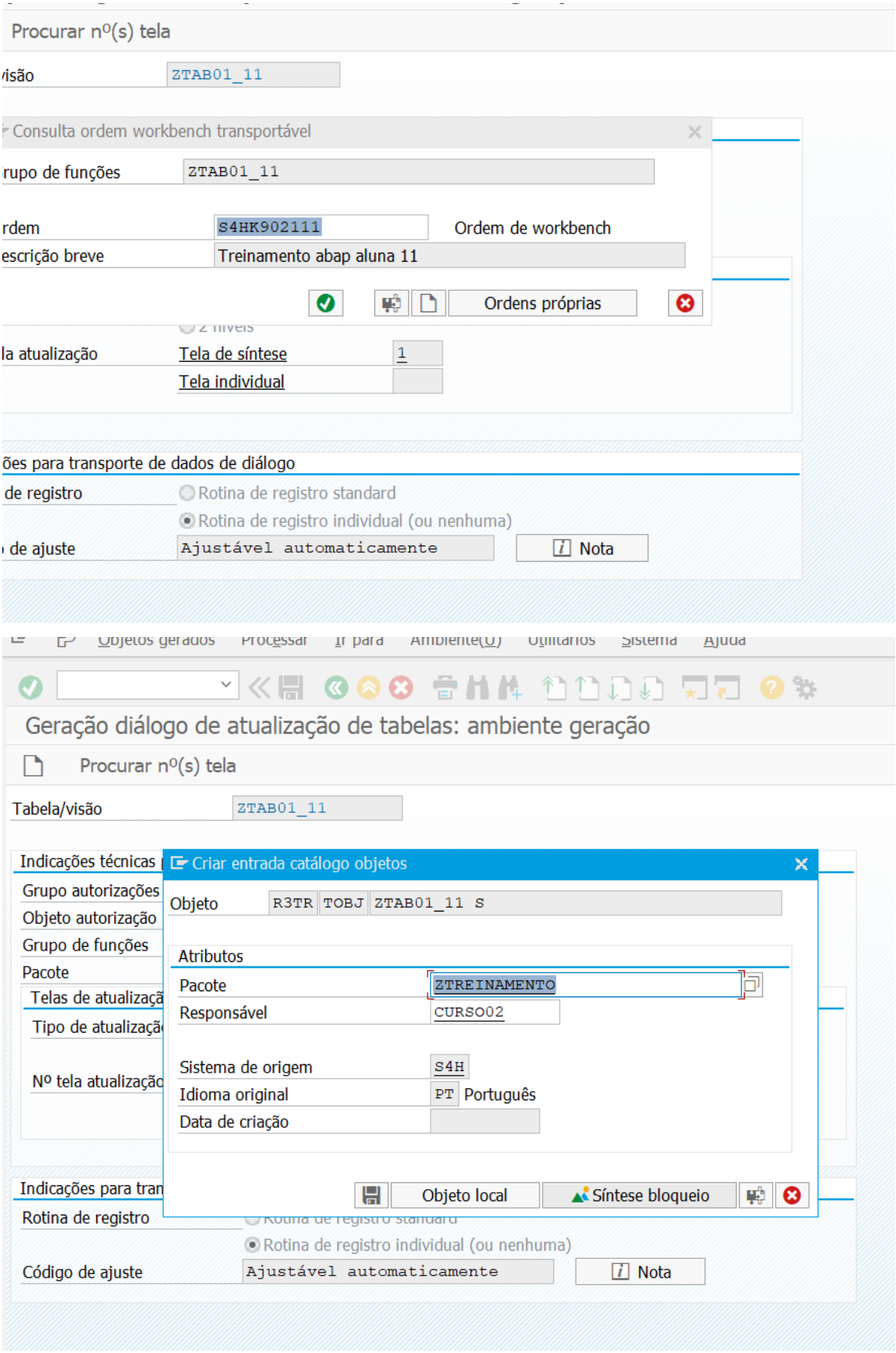
Campo	Chv	Val...	Elemento dados	CtgDados	Compr	Casas...	SistCoords.	Descrição breve
MANDT	☑	☑	MANDT	CLNT	3	0	0	Mandante
CODIGO	☑	☑	ZEL01_11	INT1	3	0	0	CODIGO
NOME	☐	☑	ZEL02_11	CHAR	30	0	0	NOME
IDADE	☐	☑	ZEL03_11	INT1	3	0	0	IDADE
RUA	☐	☑	ZEL04_11	CHAR	30	0	0	RUA CAMPO
NUMERO	☐	☑	ZEL05_11	NUMC	5	0	0	RUA
BAIRRO	☐	☑	ZEL06_11	CHAR	20	0	0	BAIRRO
CIDADE	☐	☑	ZEL07_11	CHAR	30	0	0	CIDADE
ESTADO	☐	☑	ZEL08_11	CHAR	2	0	0	ESTADO
PAIS	☐	☑	ZEL09_11	CHAR	2	0	0	PAIS
	☐	☐						

3. utilizando o menu superior vamos clicar na opção utilitários -> gerador de atualização de tabela
4. na tela seguinte vamos informar em Grupo de autorizações a opção &NC& - caso o cliente já tenha um outro grupo informe conforme documentação, caso não pode utilizar esse grupo
5. em grupo de funções informar o nome da tabela que esta sendo atualizada
6. em tipo de atualização informar nível único

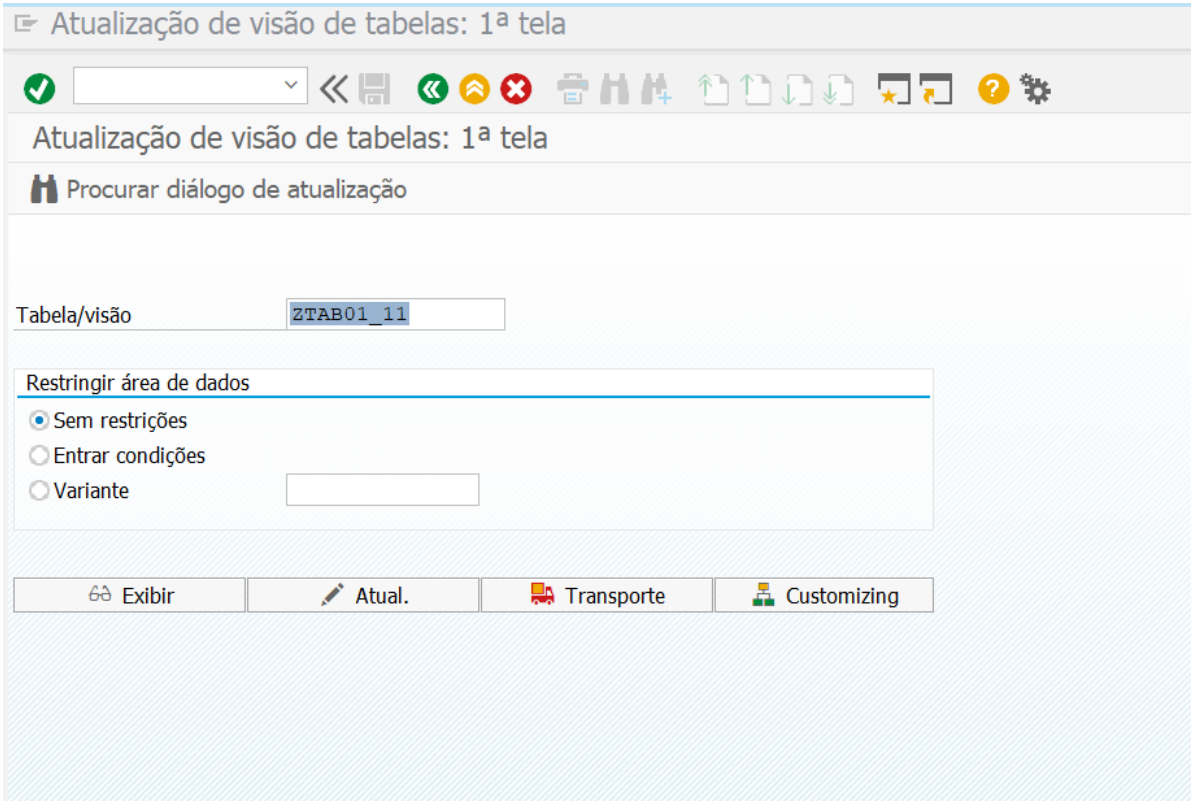


7. agora vamos clicar no botão Procurar nº de tela e ele vai preencher o campo tela de síntese
8. então vamos clicar no botão [criar] a folha em branco lado esquerdo da tela ao lado do procurar nº tela

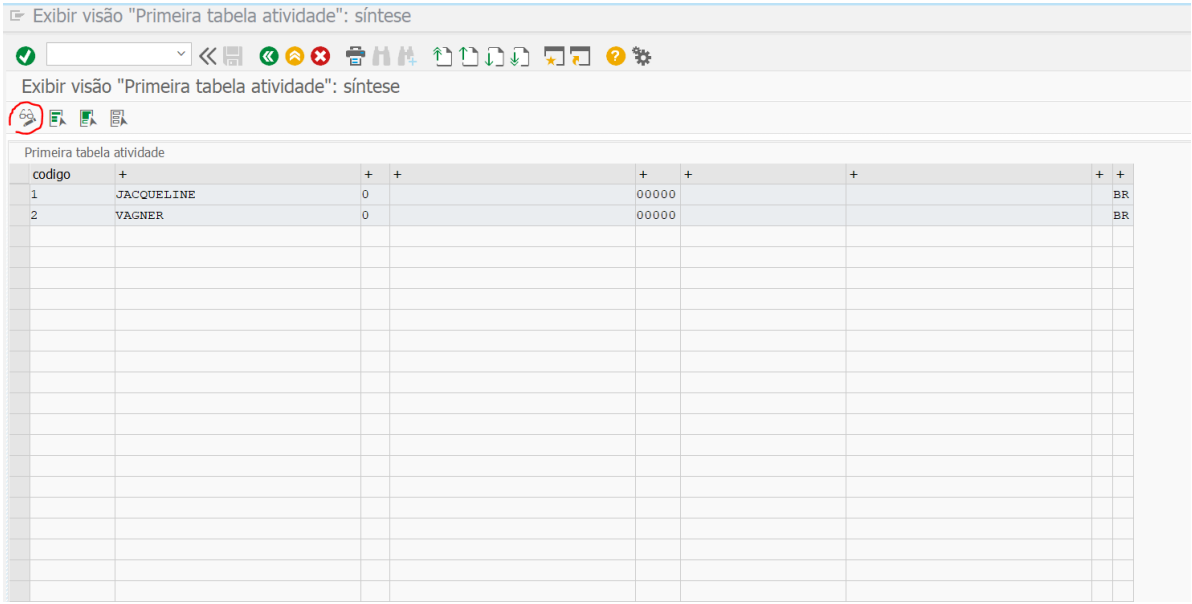
9. Aqui teremos o nome do nosso pacote você vai salvar clicando no disquete por duas vezes
10. agora vamos salvar em nossa request



11. Confirme mais uma vez em salvar
12. nesse momento, já podemos abrir uma nova janela e informar a transação sm30



13. clicar em atualizar
14. clicar no botão editar agora você já pode digitar os dados na sua tabela e também poderá clicar novamente no editar para ele gravar os seus dados



Criando Tabela Interna

TABELA INTERNA - É UMA TABELA QUE VAI ESPELHAR ALGO QUE ESTA EM TABELA - ELÁ NÃO É UMA TABELA ASSIM COMO A ZTAB01_11
A TABELA INTERNA VAI SE UTILIZADA PARA ESPERAR DADOS JUNTANDO DUAS OU MAIS TABELAS. A sua declaração vai levar nome da tabela interna e nota da tabela que ela vai espelhar.

Em outras palavras a tabela interna é uma tabela em memória apenas e vai ser utilizada para melhor desempenho e retorno de dados de relacionamento entre tabelas.

O código de operação aqui é o SE38, vamos voltar para a tela com as linhas de comando Report, e lá vamos indicar que se trata de uma tabela interna, lembre-se também de inserir os cabeçalhos e manter seu código organizado por blocos de comando.

Declarar a tabela interna passando um nome a ela e a tabela a qual se refere, aqui teremos todos os campos da tabela Ztab01_11 copiados para a tabela interna it_ztab01_11:

```
Data: it_ztab01_11 type table of ztab01_11.
```

A tabela interna pode ter todos os campos das tabelas que ela espelha, ou podemos ainda definir quais são os campos de cada tabela que vamos utilizar aqui.

Para isso vamos declarar os TYPES, dessa forma teremos a seguinte estrutura:

```
TYPES: BEGIN OF ty_table_01,
  codigo TYPE ztab01_11-codigo,
  nome  TYPE ztab01_11-nome,
  idade  TYPE ztab01_11-idade,
END OF ty_table_01.

Data: it_ztab01_11 type table of ty_table_01.
```

Work area

Também podemos criar a Work area, que é uma tabela interna que retorna um único registro, ou seja, uma única linha do banco.

Primeiro precisamos entender que um Work area é um objeto de dados estruturados de um tipo e estrutura simples, ela contem um único registro por vez. Precisamos dessa estrutura porque não é possível ler diretamente da tabela, quando executamos um SELECT os dados passam da tabela para o WORK AREA e depois para a tabela interna, podemos dizer então que é um "espaço temporário na memória do SAP e dura o tempo de execução do seu programa".

Estrutura:

```
*& --- DECLARAR WORKAREA PARA AS TABELAS *
 DATA: wa_ztab01_11 type ztab01_11,
    wa_ztab02_11 TYPE ztab02_11.
```

Entendendo o código: waztab01_11 é o nome desse espaço em memória e type indica a tabela que vai ocupar esse espaço, no caso o ztab01_11, como temos duas tabelas em uso, podemos usar virgula(,) para já declarar a baixo a próxima tabela, ou escrever outra linha DATA:. importante lembrar que os dois casos funciona e que escrever por blocos pode facilitar a manutenção futura do seu programa.

Nesse momento o arquivo ficou assim, note que passaremos as linhas de código da tela de seleção para baixo da tabela, tabela interna e work area:

```
&---------------------------------------------------------------------*
 *& Report ZREP01_11
 *&---------------------------------------------------------------------*
 *&
 *&---------------------------------------------------------------------*
 REPORT zrep01_11.

 *&---------------------------------------------------------------------*
 *& b1 - selecionar os códigos entre os intervalos digitados pelo usuário
 *&---------------------------------------------------------------------*
```

```
 *&
 *&---------------------------------------------------------------------*

 TABLES: ztab01_11,
     ztab02_11.


 *&---------------------------------------------------------------------*
 *& TABELA INTERNA - É UMA TABELA QUE VAI ESPELHAR ALGO QUE ESTA EM TABELA - ELÁ
NÃO É UMA TABELA ASSIM COMO A ZTAB01_11*
 *& A TABELA INTERNA VAI SE UTILIZADA PARA ESPERAR DADOS JUNTANDO DUAS OU MAIS
TABELAS *
 *& ---------------------------------------------------------------------*

 *& TABELA INTERNA ----------------------------------------------------*
 *& nome da tabela interna ------- nota da tabela que ela vai espelhar
 Data: it_ztab01_11 type table of ztab01_11,
    it_ztab02_11 TYPE TABLE OF ztab02_11.

 *& --- DECLARAR WORKAREA PARA AS TABELAS *
 DATA: wa_ztab01_11 type ztab01_11,
    wa_ztab02_11 TYPE ztab02_11.

 *&  TELA DE SELEÇÃO DOS DADOS *
 SELECTION-SCREEN BEGIN OF BLOCK b1.
  SELECT-OPTIONS: s_codigo FOR ztab01_11-codigo.
  parameters: p_nome TYPE ztab01_11-nome.
  parameters: p_pais TYPE ztab01_11-pais.

 SELECTION-SCREEN END OF BLOCK b1.
```

Funções

São as ações que serão realizadas propriamente, vamos primeiro declarar a Start Of Selection, faz a seleção SELECT do banco de dados.

```
*&  seleção dos dados  *
 START-OF-SELECTION.
```

SELECT

Iniciado a função de seleção vamos informar qual o select deve ser executado no banco de dados.

```
SELECT mandt
     codigo
     nome
     idade
     rua
     numero
     bairro
     cidade
     estado
     pais
  FROM ztab01_11
  INTO TABLE it_ztab01_11
```

```
  WHERE codigo IN s_codigo
      OR nome = p_nome
      OR pais = p_pais.
```

Validação

Para garantir o funcionamento da nossa busca de dados ao banco, vamos usar o comando IF perguntando se o select da tabela Ztab01_11 funcionou e tem retornos, ficará assim essa estrutura:

```
IF sy-subrc EQ 0.
\*  poderia ser if it_ztab01_11[] is not initial

* ação que deve acontecer caso tenha dado certo o select

 ELSE.

*  ação que deve acontecer caso tenha dado falha

 ENDIF.
```

Como temos uma segunda tabela relacionada a primeira, vamos incluir no caso tenha dado certo o select o select para trazer as informações relacionadas, então esse trecho do código deve ficar assim:

```
* definir o que aguardamos de resultado, o nosso SELECT propriamente falando
  IF sy-subrc EQ 0.
 \*  poderia ser if it_ztab01_11[] is not initial

   SELECT mandt
      codigo
      profissao
      salario
      moeda
    FROM ztab02_11
    INTO TABLE it_ztab02_11
    FOR ALL ENTRIES IN it_ztab01_11
    WHERE codigo = it_ztab01_11-codigo.

    MESSAGE i002(z_aula15) WITH 'Dados ok'.
  ELSE.

   WRITE: 'não foram encontrados dados para a seleção'.

  ENDIF.
```

Entendendo o código: Aqui temos o comando FOR ALL ENTRIES IN que indica que a busca deve ser feita para os dados que contenham na tabela it_ztab01_11, WERE codigo = it_ztab01_11-codigo., aqui lembre-se que o SQL vai separar nome da tabela e do campo por (-) então basicamente para a tabela 2 ele vai comparar o campo código com o campo código da tabela 1.

Até esse momento temos tudo funcionando conforme esperado, mas ainda não estamos printando para o cliente o relatório com os dados que ele buscou, então vamos agora adicionar a linha de comando responsável por ler os dados para escrevermos os mesmos na tela - sim, primeiro eu leio depois eu entrego na tela:

```
*& leitura dos dados na tela CARREGA APENAS UMA LINHA
 *& ESCRITA DOS RESULTADOS NA TELA PARA O CLIENTE - RELATÓRIO
  READ TABLE it_ztab01_11 INTO wa_ztab01_11 WITH KEY codigo = s_codigo-low.
  IF sy-subrc EQ 0.
   WRITE:
   1(10)'codigo',
   10(30)'NOME',
   20(30)'IDADE',
   35(30)'RUA',
   50(30)'NUMERO',
   65(30)'BAIRRO',
   80(30)'CIDADE',
   95(30)'ESTADO',
   110(30)'PAIS'.

   LOOP AT it_ztab01_11 INTO wa_ztab01_11 .

    WRITE:
     /1 wa_ztab01_11-CODIGO,
     10 wa_ztab01_11-NOME,
     20 wa_ztab01_11-IDADE,
     35 wa_ztab01_11-RUA,
     50 wa_ztab01_11-NUMERO,
     65 wa_ztab01_11-BAIRRO,
     80 wa_ztab01_11-CIDADE,
     95 wa_ztab01_11-ESTADO,
     110 wa_ztab01_11-PAIS.

   ENDLOOP.

  ENDIF.
```

Entendendo o código: `READ TABLE it_ztab01_11 INTO wa_ztab01_11` leitura dos dados da tabela it_ztab01_11 para o seu devido Work area. o Read lé apenas uma linha por vez.

`IF sy-subrc EQ 0` - se a leitura acima não der erros ele segue o código dentro do bloco do IF:
WRITE:
1(10)'codigo', a partir desse ponto estamos escrevendo a linha cabeçalho do relatório, posição e texto apenas.

Na linha `LOOP AT it_ztab01_11 INTO wa_ztab01_11` . estamos varrendo todos os dados do bando linha a linha para dentro do Work area.

`WRITE:/1 wa_ztab01_11-CODIGO`, a partir desse ponto estamos escrevendo os dados do work area para a tela.

O código deve estar agora dessa forma:

```
&---------------------------------------------------------------------*
 *& Report ZREP01_11
 *&---------------------------------------------------------------------*
 *&
 *&---------------------------------------------------------------------*
 REPORT zrep01_11.
```

```
 *&---------------------------------------------------------------------*
 *& b1 - selecionar os códigos entre os intervalos digitados pelo usuário
 *&---------------------------------------------------------------------*
 *&
 *&---------------------------------------------------------------------*

 TABLES: ztab01_11,
     ztab02_11.


 *&---------------------------------------------------------------------*
 *& TABELA INTERNA - É UMA TABELA QUE VAI ESPELHAR ALGO QUE ESTA EM TABELA - ELÁ
NÃO É UMA TABELA ASSIM COMO A ZTAB01_11*
 *& A TABELA INTERNA VAI SE UTILIZADA PARA ESPERAR DADOS JUNTANDO DUAS OU MAIS
TABELAS *
 *& ---------------------------------------------------------------------*

 *& TABELA INTERNA ----------------------------------------------------*
 \*  aqui vamos apenas preparar que essas tabelas vão ser utilizadas no programa
*
 *& nome da tabela interna ------- nota da tabela que ela vai espelhar
 DATA: it_ztab01_11 TYPE TABLE OF ztab01_11,
    it_ztab02_11 TYPE TABLE OF ztab02_11.

 *& --- DECLARAR WORKAREA PARA AS TABELAS *
 DATA: wa_ztab01_11 TYPE ztab01_11,
    wa_ztab02_11 TYPE ztab02_11.

 *&  TELA DE SELEÇÃO DOS DADOS *
 SELECTION-SCREEN BEGIN OF BLOCK b1.
  SELECT-OPTIONS: s_codigo FOR ztab01_11-codigo.
  PARAMETERS: p_nome TYPE ztab01_11-nome.
  PARAMETERS: p_pais TYPE ztab01_11-pais.

 SELECTION-SCREEN END OF BLOCK b1.

 *&  seleção dos dados  *
 START-OF-SELECTION.

 *&  definir o que aguardamos de resultado, o nosso SELECT propriamente falando
*
  SELECT mandt
      codigo
      nome
      idade
      rua
      numero
      bairro
      cidade
      estado
      pais
   FROM ztab01_11
   INTO TABLE it_ztab01_11
   WHERE codigo IN s_codigo
   OR nome = p_nome
   OR pais = p_pais.
```

```
 \* definir o que aguardamos de resultado, o nosso SELECT propriamente falando
 IF sy-subrc EQ 0.
 \* poderia ser if it_ztab01_11[] is not initial
  SELECT mandt
     codigo
     profissao
     salario
     moeda
   FROM ztab02_11
   INTO TABLE it_ztab02_11
   FOR ALL ENTRIES IN it_ztab01_11
   WHERE codigo = it_ztab01_11-codigo.

  MESSAGE i002(z_aula15) WITH 'Dados ok'.

 *& leitura dos dados na tela CARREGA APENAS UMA LINHA
 *& ESCRITA DOS RESULTADOS NA TELA PARA O CLIENTE - RELATÓRIO
  READ TABLE it_ztab01_11 INTO wa_ztab01_11 WITH KEY codigo = s_codigo-low.
  IF sy-subrc EQ 0.
   WRITE:
   1(10)'codigo',
   10(30)'NOME',
   20(30)'IDADE',
   35(30)'RUA',
   50(30)'NUMERO',
   65(30)'BAIRRO',
   80(30)'CIDADE',
   95(30)'ESTADO',
   110(30)'PAIS'.

   LOOP AT it_ztab01_11 INTO wa_ztab01_11 .

    WRITE:
     /1 wa_ztab01_11-CODIGO,
     10 wa_ztab01_11-NOME,
     20 wa_ztab01_11-IDADE,
     35 wa_ztab01_11-RUA,
     50 wa_ztab01_11-NUMERO,
     65 wa_ztab01_11-BAIRRO,
     80 wa_ztab01_11-CIDADE,
     95 wa_ztab01_11-ESTADO,
     110 wa_ztab01_11-PAIS.

   ENDLOOP.

  ENDIF.

  ELSE.

  MESSAGE i002(z_aula15) DISPLAY LIKE 'E' WITH 'Não foi possível encontrar o
registro solicitado'.
 ENDIF.
```

Modulando o código

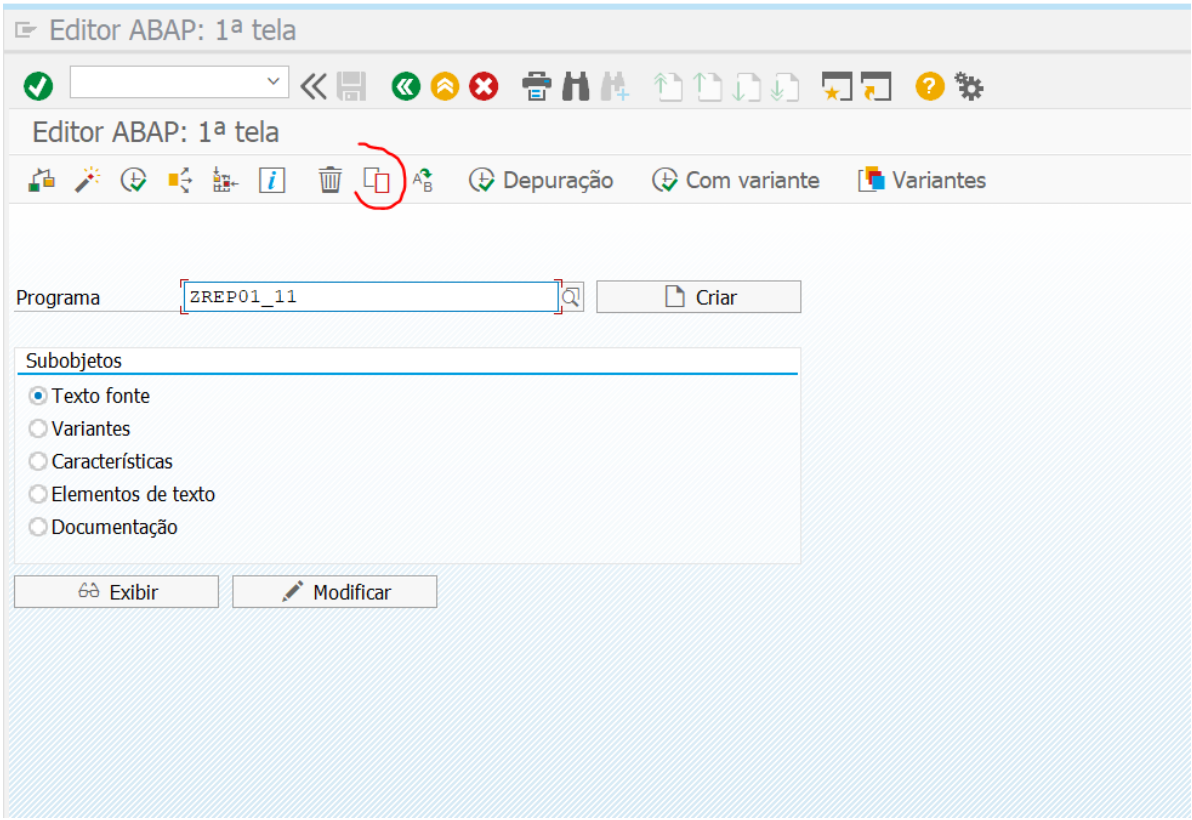
Modular seu código torna ele mais organizado e fácil de entender e dar manutenções. Modulações podem ser feitas através de Forms/ Perfomes funções e includes.

Um módulo de função é um código que pode ser chamado em vários programas, reutilizando assim o código

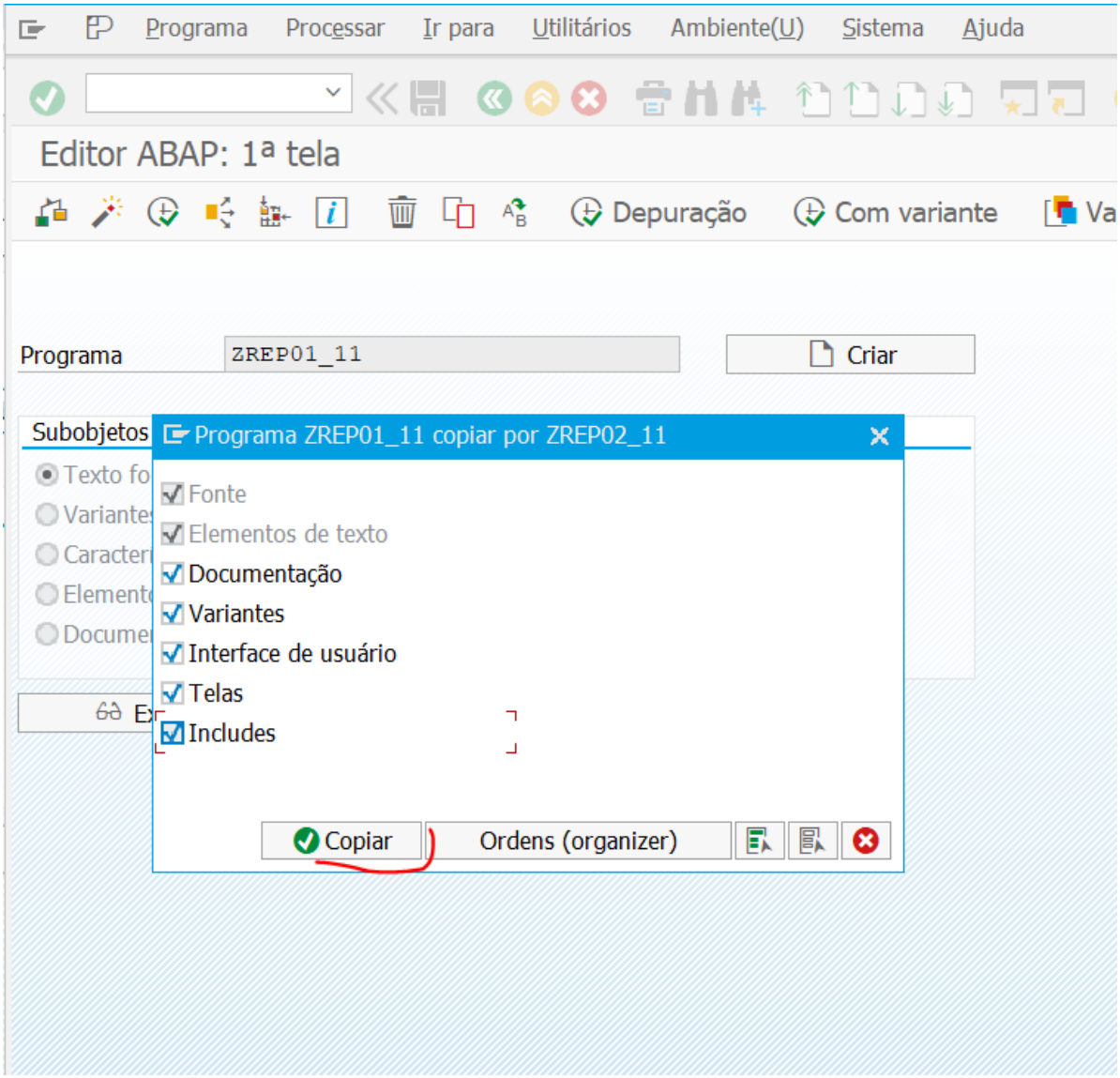
Vamos utilizar transação SE37. Lembre-se que os includes não podem ser executados sozinhos, ele não é uma programa executável, ele vai funcionar sendo chamado dentro do outro programa. São muito utilizados em programas do tipo Module Pool.

Vamos então iniciar realizando uma cópia do programa Zrep01_xx, sendo Zrep01_xx o seu report:

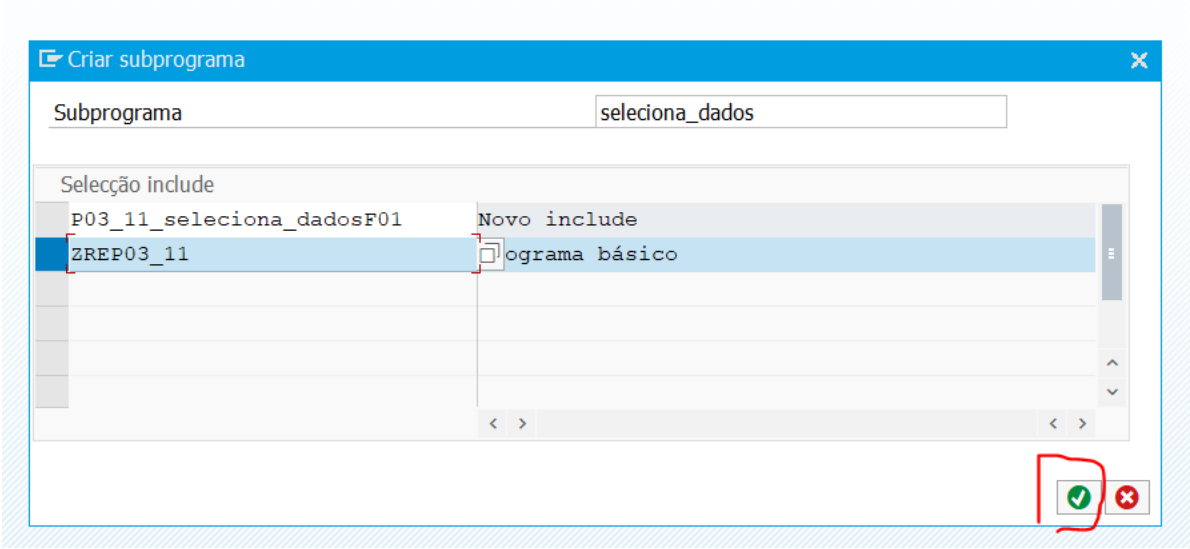
1. informar o código se38
2. na tela do modulo função vamos informar um nome válido de programa Report e clicar no botão copiar
3. na tela seguinte ajustar o nome do programa destino, SUA CÓPIA.



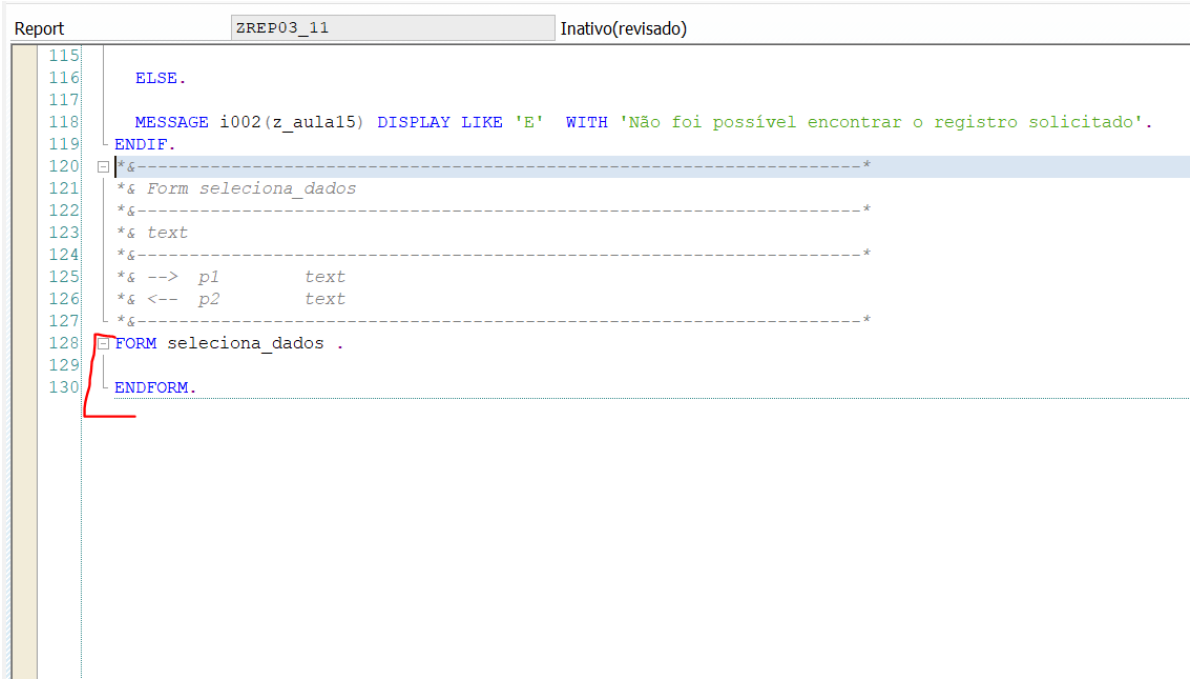
4. selecione todos os tipos de dados e clique em copiar



5. relacione o seu pacote e siga para criar a cópia
6. com a cópia do projeto feita vamos começar a editar a cópia para ter a nossa modulação:
7. após a linha START-OF-SELECTION. vamos adicionar a seguinte linha de comando `PERFORM seleciona_dados` a ideia aqui é criar uma organização e juntar as linhas de comando SELECT em um único bloco.
8. após isso clica no nome seleciona_dados duas vezes para criar ele elemento e clicar na opção **sim** na caixa de mensagens do Subprograma seleciona_dados não existe, Criar o objeto?
9. clicar na opção **sim** na caixa de dialogos pertuntando se deseja gravar o programa atual



10. na tela de confirmação onde desejo criar o nome objeto, podemos clicar no nome do nosso programa mesmo.
11. resultado esperado:



O resultado esperado após essa ação é o seguinte código para o programa copiado:

```
&---------------------------------------------------------------------*
 *& Report ZREP01_11 nome da cópia do seu programa
 *&---------------------------------------------------------------------*
 *&
 *&---------------------------------------------------------------------*
 REPORT zrep03_11.



 *&---------------------------------------------------------------------*
 *& b1 - selecionar os códigos entre os intervalos digitados pelo usuário
 *&---------------------------------------------------------------------*
 *&
 *&---------------------------------------------------------------------*

 TABLES: ztab01_11,
```

```
     ztab02_11.


 *&---------------------------------------------------------------------*
 *& TABELA INTERNA - É UMA TABELA QUE VAI ESPELHAR ALGO QUE ESTA EM TABELA - ELÁ
NÃO É UMA TABELA ASSIM COMO A ZTAB01_11*
 *& A TABELA INTERNA VAI SE UTILIZADA PARA ESPERAR DADOS JUNTANDO DUAS OU MAIS
TABELAS *
 *& ---------------------------------------------------------------------*

 *& TABELA INTERNA ----------------------------------------------------*
 \*  aqui vamos apenas preparar que essas tabelas vão ser utilizadas no programa
*
 *& nome da tabela interna ------- nota da tabela que ela vai espelhar
 DATA: it_ztab01_11 TYPE TABLE OF ztab01_11,
    it_ztab02_11 TYPE TABLE OF ztab02_11.

 *& --- DECLARAR WORKAREA PARA AS TABELAS *
 DATA: wa_ztab01_11 TYPE ztab01_11,
    wa_ztab02_11 TYPE ztab02_11.

 *&  TELA DE SELEÇÃO DOS DADOS *
 SELECTION-SCREEN BEGIN OF BLOCK b1.
  SELECT-OPTIONS: s_codigo FOR ztab01_11-codigo.
  PARAMETERS: p_nome TYPE ztab01_11-nome.
  PARAMETERS: p_pais TYPE ztab01_11-pais.

 SELECTION-SCREEN END OF BLOCK b1.

 *&  seleção dos dados  *
 START-OF-SELECTION.
 *& modulando o programa
  PERFORM seleciona_dados.


 *&---------------------------------------------------------------------*
 *& Form seleciona_dados
 *&---------------------------------------------------------------------*
 *& text
 *&---------------------------------------------------------------------*
 *& --> p1    text
 *& <-- p2    text
 *&---------------------------------------------------------------------*
 FORM seleciona_dados .
 *&  definir o que aguardamos de resultado, o nosso SELECT propriamente falando
*
  SELECT mandt
      codigo
      nome
      idade
      rua
      numero
      bairro
      cidade
      estado
      pais
   FROM ztab01_11
   INTO TABLE it_ztab01_11
```

```
  WHERE codigo IN s_codigo
  OR nome = p_nome
  OR pais = p_pais.


\* definir o que aguardamos de resultado, o nosso SELECT propriamente falando
 IF sy-subrc EQ 0.
\* poderia ser if it_ztab01_11[] is not initial
  SELECT mandt
     codigo
     profissao
     salario
     moeda
   FROM ztab02_11
   INTO TABLE it_ztab02_11
   FOR ALL ENTRIES IN it_ztab01_11
   WHERE codigo = it_ztab01_11-codigo.

  MESSAGE i002(z_aula15) WITH 'Dados ok'.




*& leitura dos dados na tela CARREGA APENAS UMA LINHA
*& ESCRITA DOS RESULTADOS NA TELA PARA O CLIENTE - RELATÓRIO
  READ TABLE it_ztab01_11 INTO wa_ztab01_11 WITH KEY codigo = s_codigo-low.
  IF sy-subrc EQ 0.
   WRITE:
   1(10)'codigo',
   10(30)'NOME',
   20(30)'IDADE',
   35(30)'RUA',
   50(30)'NUMERO',
   65(30)'BAIRRO',
   80(30)'CIDADE',
   95(30)'ESTADO',
   110(30)'PAIS'.

   LOOP AT it_ztab01_11 INTO wa_ztab01_11 .

    WRITE:
     /1 wa_ztab01_11-codigo,
     10 wa_ztab01_11-nome,
     20 wa_ztab01_11-idade,
     35 wa_ztab01_11-rua,
     50 wa_ztab01_11-numero,
     65 wa_ztab01_11-bairro,
     80 wa_ztab01_11-cidade,
     95 wa_ztab01_11-estado,
     110 wa_ztab01_11-pais.

   ENDLOOP.

  ENDIF.

 ELSE.
```

```
   MESSAGE i002(z_aula15) DISPLAY LIKE 'E' WITH 'Não foi possível encontrar o
registro solicitado'.
  ENDIF.

 ENDFORM.
```

Agora vamos passar parâmetro para a Form Seleciona_dados e validar antes de chamar o form do exibir dados:

1. criar a variável junto as outras declarações DATA do seu programa:

```
*& VARIAVEL DO CIPO CARACTER COM TAMANHO
 DATA: v_erro TYPE C.
```

2. vamos passar esse variavel para a Perform declarada do selecionar_dados:

```
PERFORM seleciona_dados CHANGING v_erro.
```

3. vamos passar essa var como parâmetro para o form do seleciona_dados:

```
FORM seleciona_dados CHANGING p_erro .
```

4. dentro do bloco else do seu seleciona dados, vamos incluir a linha dizendo um valor para o_erro caso exista erro:

```
p_erro = 'x'.
```

5. agora vamos validar o perform exibe dados da seguinte forma, caso p_erro for diferente de X, exibe os dados

```
IF v_erro NE 'x'.
```

O seu código agora deve estar da seguinte forma:

```
&---------------------------------------------------------------------*
 *& Report ZREP01_11
 *&---------------------------------------------------------------------*
 *&
 *&---------------------------------------------------------------------*
 REPORT zrep03_11.



 *&---------------------------------------------------------------------*
 *& b1 - selecionar os códigos entre os intervalos digitados pelo usuário
 *&---------------------------------------------------------------------*
 *&
 *&---------------------------------------------------------------------*

 TABLES: ztab01_11,
     ztab02_11.
```

```
 *&---------------------------------------------------------------------*
 *& TABELA INTERNA - É UMA TABELA QUE VAI ESPELHAR ALGO QUE ESTA EM TABELA - ELÁ
NÃO É UMA TABELA ASSIM COMO A ZTAB01_11*
 *& A TABELA INTERNA VAI SE UTILIZADA PARA ESPERAR DADOS JUNTANDO DUAS OU MAIS
TABELAS *
 *& ---------------------------------------------------------------------*

 *& TABELA INTERNA ----------------------------------------------------*
 \*  aqui vamos apenas preparar que essas tabelas vão ser utilizadas no programa
*
 *& nome da tabela interna ------- nota da tabela que ela vai espelhar
 DATA: it_ztab01_11 TYPE TABLE OF ztab01_11,
    it_ztab02_11 TYPE TABLE OF ztab02_11.

 *& --- DECLARAR WORKAREA PARA AS TABELAS *
 DATA: wa_ztab01_11 TYPE ztab01_11,
    wa_ztab02_11 TYPE ztab02_11.

 *& VARIAVEL DO CIPO CARACTER COM TAMANHO
 DATA: v_erro TYPE c.

 *&  TELA DE SELEÇÃO DOS DADOS *
 SELECTION-SCREEN BEGIN OF BLOCK b1.
  SELECT-OPTIONS: s_codigo FOR ztab01_11-codigo.
  PARAMETERS: p_nome TYPE ztab01_11-nome.
  PARAMETERS: p_pais TYPE ztab01_11-pais.

 SELECTION-SCREEN END OF BLOCK b1.

 *&  seleção dos dados  *
 START-OF-SELECTION.
 *& modulando o programa
  PERFORM seleciona_dados CHANGING v_erro.

  IF v_erro NE 'x'.
   PERFORM exibe_dados.
  ENDIF.

 *&---------------------------------------------------------------------*
 *& Form seleciona_dados
 *&---------------------------------------------------------------------*
 *& text
 *&---------------------------------------------------------------------*
 *& --> p1    text
 *& <-- p2    text
 *&---------------------------------------------------------------------*
 FORM seleciona_dados CHANGING p_erro .
 *&  definir o que aguardamos de resultado, o nosso SELECT propriamente falando
*
  SELECT mandt
      codigo
      nome
      idade
      rua
      numero
      bairro
      cidade
      estado
```

```
      pais
   FROM ztab01_11
   INTO TABLE it_ztab01_11
   WHERE codigo IN s_codigo
   OR nome = p_nome
   OR pais = p_pais.


 \* definir o que aguardamos de resultado, o nosso SELECT propriamente falando
  IF sy-subrc EQ 0.
 \* poderia ser if it_ztab01_11[] is not initial
   SELECT mandt
      codigo
      profissao
      salario
      moeda
    FROM ztab02_11
    INTO TABLE it_ztab02_11
    FOR ALL ENTRIES IN it_ztab01_11
    WHERE codigo = it_ztab01_11-codigo.

   MESSAGE i002(z_aula15) WITH 'Dados ok'.


  ELSE.
   p_erro = 'x'.

   MESSAGE i002(z_aula15) DISPLAY LIKE 'E' WITH 'Não foi possível encontrar o
registro solicitado'.
  ENDIF.

 ENDFORM.
 *&---------------------------------------------------------------------*
 *& Form exibe_dados
 *&---------------------------------------------------------------------*
 *& text
 *&---------------------------------------------------------------------*
 *& --> p1    text
 *& <-- p2    text
 *&---------------------------------------------------------------------*
 FORM exibe_dados .
 *& leitura dos dados na tela carrega apenas uma linha
 *& ESCRITA DOS RESULTADOS NA TELA PARA O CLIENTE - RELATÓRIO
  READ TABLE it_ztab01_11 INTO wa_ztab01_11 WITH KEY codigo = s_codigo-low.
  IF sy-subrc EQ 0.
   WRITE:
   1(10)'codigo',
   10(30)'NOME',
   20(30)'IDADE',
   35(30)'RUA',
   50(30)'NUMERO',
   65(30)'BAIRRO',
   80(30)'CIDADE',
   95(30)'ESTADO',
   110(30)'PAIS'.

   LOOP AT it_ztab01_11 INTO wa_ztab01_11 .
```

```
    WRITE:
     /1 wa_ztab01_11-codigo,
     10 wa_ztab01_11-nome,
     20 wa_ztab01_11-idade,
     35 wa_ztab01_11-rua,
     50 wa_ztab01_11-numero,
     65 wa_ztab01_11-bairro,
     80 wa_ztab01_11-cidade,
     95 wa_ztab01_11-estado,
     110 wa_ztab01_11-pais.

   ENDLOOP.

  ENDIF.
ENDFORM.
```

Debugar o código

Para testar a sua aplicação temos uma funcionalidade que pode sim ver muito válida que é o debug, para isso:

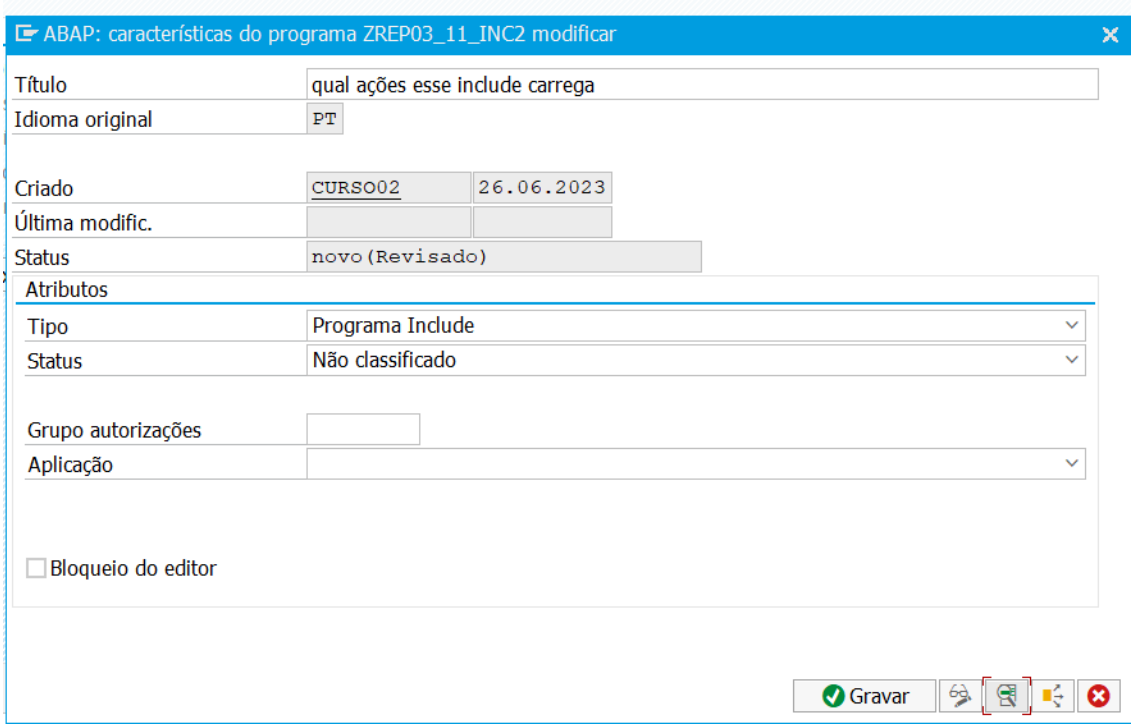
Com a tela do seu programa aberto clique na caixa de transações e digite **/h** em seguida digite **ENTER**, isso já vai te direcionar para a tela de debug onde vc pode clicar no botão **EXECUTAR** para ver a sua aplicação debugando.

Separar do programa REPORT includes

Essa ação ajuda a tornar o código mais enxuto e com menos linhas de comandos, isso facilita muito a vida da pessoa desenvolvedora a encontrar trechos de código específico para tratar de manutenções futuras.

Para isso vamos seguir o passo a passo:

1. Entre na transação **SE38** em uma nova tela
2. Colocar nome do seu include, no caso definimos o nome do programa com terminação INC - ficou zrep03_11_INC. Não esqueça de clicar no botão criar, dar a devida descrição e em Tipo: opção -> programa include.
3. botão Gravar, em seguida informe seu pacote, salvar e informar a sua request.



4. no programa que tem nome zrep03_11 vamos inserir no inicio do documento, após as linhas que inserem as tabelas no programa o código:

```
INCLUDE ZREP03_11_INC.
```

O código acima, chama o include para esse arquivo atual, assim como o include ou request do PHP, ou o import do Java.

5. Vamos ativar os dois programas (programa e include)
6. Na sequencia devemos transportar para nosso arquivo include tudo que for Perform e também os SELECT e os forms, isso para tornar o arquivo principal com o mínimo possível de código. (ATENÇÂO: vamos pensar sempre em colocar as funcionalidades dentro de includes diferentes, então se seu report tem mais de uma funcionalidade vale separar uma por inlcude)

Como ficaram os códigos:

Report:

```
&---------------------------------------------------------------------*
 *& Report ZREP01_11
 *&---------------------------------------------------------------------*
 *&
 *&---------------------------------------------------------------------*
 REPORT zrep03_11.

 *&---------------------------------------------------------------------*
 *& b1 - selecionar os códigos entre os intervalos digitados pelo usuário
 *&---------------------------------------------------------------------*
 *&
 *&---------------------------------------------------------------------*
 TABLES: ztab01_11, ztab02_11.

 INCLUDE ZREP03_11_INC.

 START-OF-SELECTION.
```

```
 *& modulando o programa
  PERFORM seleciona_dados CHANGING v_erro.

  IF v_erro NE 'x'.
   PERFORM exibe_dados.
  ENDIF.
```

Include:

```
*&---------------------------------------------------------------------*
 *& Include ZREP03_11_INC
 *&---------------------------------------------------------------------*



 *& DECLARAÇÕES

 *&---------------------------------------------------------------------*
 *& TABELA INTERNA - É UMA TABELA QUE VAI ESPELHAR ALGO QUE ESTA EM TABELA - ELÁ
NÃO É UMA TABELA ASSIM COMO A ZTAB01_11*
 *& A TABELA INTERNA VAI SE UTILIZADA PARA ESPERAR DADOS JUNTANDO DUAS OU MAIS
TABELAS *
 *& ---------------------------------------------------------------------*
 DATA: it_ztab01_11 TYPE TABLE OF ztab01_11,
    it_ztab02_11 TYPE TABLE OF ztab02_11.

 *& --- DECLARAR WORKAREA PARA AS TABELAS *
 DATA: wa_ztab01_11 TYPE ztab01_11,
    wa_ztab02_11 TYPE ztab02_11.

 *& VARIAVEL DO CIPO CARACTER COM TAMANHO
 DATA: v_erro TYPE c,
    v_parl TYPE c .


 *&  TELA DE SELEÇÃO DOS DADOS *
 SELECTION-SCREEN BEGIN OF BLOCK b1.
  SELECT-OPTIONS: s_codigo FOR ztab01_11-codigo.
  PARAMETERS: p_nome TYPE ztab01_11-nome.
  PARAMETERS: p_pais TYPE ztab01_11-pais.

 SELECTION-SCREEN END OF BLOCK b1.

 *&  seleção dos dados  *

 *&---------------------------------------------------------------------*
 *& Form seleciona_dados
 *&---------------------------------------------------------------------*
 *& text
 *&---------------------------------------------------------------------*
 *& --> p1    text
 *& <-- p2    text
 *&---------------------------------------------------------------------*

 FORM seleciona_dados CHANGING p_erro .
```

```
 *&  definir o que aguardamos de resultado, o nosso SELECT propriamente falando
*
  SELECT mandt
      codigo
      nome
      idade
      rua
      numero
      bairro
      cidade
      estado
      pais
   FROM ztab01_11
   INTO TABLE it_ztab01_11
   WHERE codigo IN s_codigo
   OR nome = p_nome
   OR pais = p_pais.


 \* definir o que aguardamos de resultado, o nosso SELECT propriamente falando
  IF sy-subrc EQ 0.
 \* poderia ser if it_ztab01_11[] is not initial
   SELECT mandt
      codigo
      profissao
      salario
      moeda
    FROM ztab02_11
    INTO TABLE it_ztab02_11
    FOR ALL ENTRIES IN it_ztab01_11
    WHERE codigo = it_ztab01_11-codigo.

   MESSAGE i002(z_aula15) WITH 'Dados ok'.


  ELSE.
   p_erro = 'x'.

   MESSAGE i002(z_aula15) DISPLAY LIKE 'E' WITH 'Não foi possível encontrar o
registro solicitado'.
  ENDIF.

 ENDFORM.
 *&---------------------------------------------------------------------*
 *& Form exibe_dados
 *&---------------------------------------------------------------------*
 *& text
 *&---------------------------------------------------------------------*
 *& --> p1    text
 *& <-- p2    text
 *&---------------------------------------------------------------------*
 FORM exibe_dados .
 *& leitura dos dados na tela carrega apenas uma linha
 *& ESCRITA DOS RESULTADOS NA TELA PARA O CLIENTE - RELATÓRIO
  READ TABLE it_ztab01_11 INTO wa_ztab01_11 WITH KEY codigo = s_codigo-low.
  IF sy-subrc EQ 0.
   WRITE:
   1(10)'codigo',
```

```
    10(30)'NOME',
    20(30)'IDADE',
    35(30)'RUA',
    50(30)'NUMERO',
    65(30)'BAIRRO',
    80(30)'CIDADE',
    95(30)'ESTADO',
    110(30)'PAIS'.

    LOOP AT it_ztab01_11 INTO wa_ztab01_11 .

     WRITE:
      /1 wa_ztab01_11-codigo,
      10 wa_ztab01_11-nome,
      20 wa_ztab01_11-idade,
      35 wa_ztab01_11-rua,
      50 wa_ztab01_11-numero,
      65 wa_ztab01_11-bairro,
      80 wa_ztab01_11-cidade,
      95 wa_ztab01_11-estado,
      110 wa_ztab01_11-pais.

    ENDLOOP.

  ENDIF.
ENDFORM.
```

Modulo de função

Para as funções temos a transação SE37, ela serve para criar ações dentro do SAP. As funções do SAP funcionam igual as funções em outras linguagens de programação. Podemos criar com esta, processamento ou uma funcionalidade específica, para assim poder ter o reuso de código para mais de um programa.

Uma função pode ter parâmetros de entrada e de saída. Sendo os parâmetros de entrada os dados que ela vai usar como critério, por exemplo. E os parâmetros de saída os dados correspondentes ao critério buscados no banco de dados por exemplo.

As funções ficam dentro do bloco de códigos `FUNCTIONe` e `ENDFUNCTION`, vale lembrar que não podemos alterar um parâmetro de entrada diretamente, podendo ser utilizado uma variável para tratar o dado do parametro.

Existem funções standard, podemos também chama-las em um programa.

A chamada de funções se dá de forma: `CALL FUNCTION 'NOME_DA_FUNÇÃO'`

Uma forma ainda mais simples de se chamar uma função é, dentro de um programa ou Report você vai no botão MODELO e no campo CALL FUNCTION informe o nome da função desejada. Clique no botão salvar e ative seu programa antes de testar.

```
CALL FUNCTION 'CONVERSION_EXIT_BEKNZ_OUTPUT'
  EXPORTING
    input  = v_input
  IMPORTING
    output = v_output.
```

Exporting - saídas

Importing - entradas

> **Atenção:** Ao escrever uma função ela vai **exportar** o dado e no programa vc vai **Importar** os mesmos. Se sua função estiver importando algum dado no seu programa vc deve exportar esses dados para a função, **um envia o outro recebe**.

Grupo de função

Ao criar uma função precisamos criar um grupo de função ou inserir essa em um grupo de funções existente, o grupo é um local para guardar funções de assuntos parecidos ou mesmo assunto: exemplo um grupo para conversão de medidas de Quilos para Libras, Metros para Milhas etc, todas as funções no grupo devem ter objetivos parecidos.

Transporte de objetos

Para transporte de objetos precisamos que o mesmo esteja dentro de uma request, que podemos entender como sendo algo parecido com um container ou um commit do git, por exemplo. Ela terá informações do seu dono, pessoas que a modificaram, histórico de status do código (versionamento do código) permitindo inclusive voltar a uma versão anterior caso necessário.

Request

As requests também podem ser migradas de um ambiente de desenvolvimento para um ambiente de produção. Levando todos os objetos que estão dentro dela.

Transações para criar, visualizar, alterar ou excluir uma request: **SE01**, **SE09**, **SE10**.

Cada pessoa desenvolvedora deve mexer e alterar apenas a sua request, evitando criar task na request de outra pessoa desenvolvedora.

Ao transportar as requests, caso tenha algum erro ela vai dar alerta de erro, é necessário ter cuidado e saber bem quais as suas requests pois um domínio não ativo pode parar uma aplicação se for feito o transporte indevido.

Criando uma função no SAP

2:25

ALV - ABAP List Viewer

O ABAP List Viewer (ALV) é uma estrutura fornecida pelo SAP para facilitar a exibição e manipulação de dados em formato de lista. Ele oferece recursos avançados de formatação, classificação, filtragem e interação com o usuário, tornando a exibição de dados mais eficiente e amigável.

O ALV é amplamente utilizado para criar relatórios e exibir resultados de consulta de maneira tabular. Ele permite que você personalize a aparência da lista, organize colunas, aplique cores e formate os dados de acordo com as necessidades específicas. Além disso, é possível adicionar recursos interativos, como ordenação dinâmica, cálculos automáticos, filtros personalizados, totalizações e agrupamentos.

Existem diferentes tipos de ALV no ABAP, incluindo:

- **ALV Grid**: É o tipo mais comum de ALV e permite exibir os dados em uma grade tabular. É altamente personalizável, com opções de formatação, filtragem e interação.
- **ALV Hierarchical**: Permite exibir dados hierárquicos em uma exibição semelhante a uma estrutura de árvore, mas com recursos adicionais, como expansão e contração de nós e funcionalidades de drill-down.
- **ALV List**: É um tipo mais simples de ALV que exibe os dados em formato de lista simples. É útil para listas simples de dados, sem a necessidade de recursos avançados de interação.

O ALV pode ser criado usando a biblioteca de classes do ALV, que fornece métodos e eventos para configurar e manipular a exibição dos dados. Também é possível criar ALVs usando a ferramenta de layout do SAP, o que facilita a configuração visual e a personalização da aparência do ALV.
Em resumo, o ABAP List Viewer (ALV) é uma estrutura poderosa no ABAP que facilita a exibição e a manipulação de dados em formato de lista. Ele fornece recursos avançados de formatação, filtragem e interação, permitindo a criação de relatórios e exibições de dados mais eficientes e amigáveis ao usuário.

É uma ferramenta para criar relatórios e telas para monitor.