

Academia ABAP With

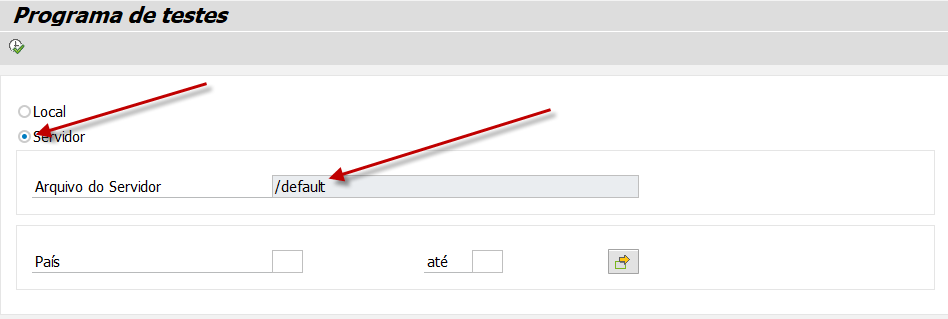
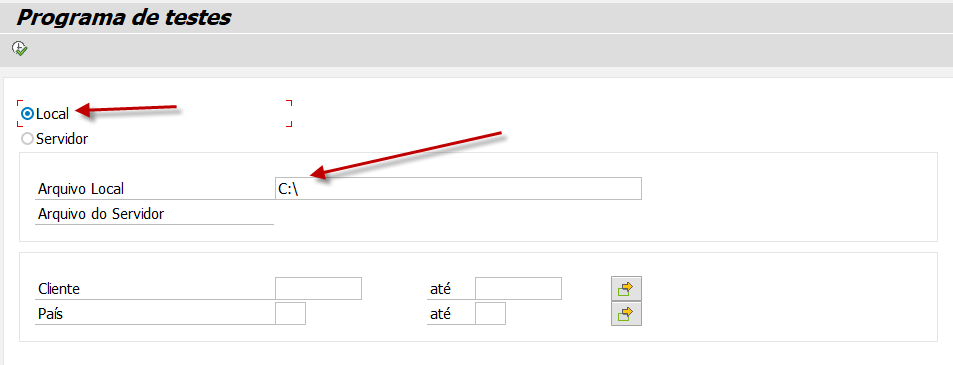
Hana & Fiori

o **Tela de seleção dinâmica**

Uma tela de seleção dinâmica pode variar em sua exibição de acordo com opções selecionadas, no exemplo abaixo, temos uma tela de entrada para o usuário selecionar um caminho do **Windows** ou do **Servidor SAP**, onde está hospedado o sistema.

Ao selecionar a opção local, serão exibidas as opções para a seleção de Arquivo e ao selecionar a opção de servidor, serão exibidas só as opções para o servidor (imagens 1 e 2)

### Exemplo de tela dinâmica: (Programa: Z\_APASSARELLI\_004\_5)



o **Eventos da tela de seleção**

Para entender a estrutura de um código, precisamos entender que existem diversos eventos na execução de um programa, isso define a estrutura lógica obrigatória que sempre vamos respeitar na construção de nossas demandas.

Por exemplo quando falamos acima da tela de seleção, ela marca o início de um programa, pois o usuário partirá dessa tela para as lógicas programadas para serem executadas após a ação do usuário.

Exemplos – Programa: **Z\_APASSARELLI\_004\_9**

* + AT SELECTION-SCREEN ON
  + START-OF-SELECTION
  + INITIALIZATION

Vamos entender o conceito de eventos durante a explicação da aula de sintaxes em vídeo, para melhor experiência com essa ferramenta no código.

### Sintaxes ABAP:

* **Conhecendo o Help de Sintaxes (F1) e as sintaxes abaixo detalhadas com exemplos práticos:**

Exemplos das sintaxes no programa: **Z\_APASSARELLI\_004\_1**

* ADD
* APPEND
* AT FIRST
* AT LAST
* AT NEW
* AT SELECTION-SCREEN ON
* AT USER-COMMAND
* AUTHORITY-CHECK OBJECT
* ASSING f1 TO <f>
* CALL TRANSACTION
* CASE .. WHEN .. ENDCASE
* CHECK
* CLEAR
* COLLECT
* COMMIT WORK
* CONCATENATE
* CONTINUE
* CONDENSE
* DELETE
* DESCRIBE TABLE
* DO .. ENDDO
* EXIT
* FORM
* FREE
* IF .. ELSE .. ENDIF.
* IF NOT .. IS INITIAL
* INITIALIZATION
* INSERT
* INSERT LINES
* LEAVE PROGRAM
* LOOP AT
* MESSAGE
* MESSAGE-ID
* MODIFY
* MOVE
* MOVE-CORRESPONDING
* PARAMETERS
* PERFOM
* READ TABLE
* READ\_TEXT
* REFRESH
* REPLACE
* SAPGUI\_PROGRESS\_INDICATOR
* SEARCH
* SELECT
* SKIP
* SORT
* SPLIT .. AT .. INTO
* START-OF-SELECTION
* STRLEN
* SUM
* SY-BATCH
* SY-DATUM
* SY-LANGU
* SY-LINNO
* SY-LISEL
* SY-MANDT
* SY-PAGNO
* SY-SUBRC
* SY-TABIX
* SY-TVAR0 .. SY-TVAR9
* SY-UCOMM
* SY-UNAME
* SY-UZEIT
* SY-VLINE
* SY-ULINE
* SY-CPROG
* SY-TCODE
* SY-DBCNT
* TABLES
* TRANSLATE
* TYPES
* ULINE
* UNPACK
* UPDATE
* WHILE .. ENDWHILE
* WRITE

### Blocos/Performs:

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

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

PERFORM: f\_form1,

f\_form2.

FORM f\_form1.

ENDFORM.

FORM f\_form2.

ENDFORM.