SCREEN PAINTER **4**
DISPOSICION DE CAMPOS EN PANTALLA (FULL SCREEN) 5
BOTONES *7*
Propiedades de los botones 8
ENTRY *8*
PROPIEDADES ENTRY 8
RADIO BUTTON *9*
Paso 1 (crear un box) 9
Paso 2 (poner los tres buttons) 10
Paso 3 (seleccionarlos y unirlos) 10
Paso 4 (cómo lo relacionamos con el programa) 11
Propiedades del radio button 12
CHECK BOX *12*
Propiedades del check box. 12
BOX *13*
Propiedades del box. 13
STEPLOOP *13*
Paso 1 (crear el objeto) 13
Paso 2 (definir el steploop) 14
Consejos 16
GRUPOS DE OBJETOS *16*
¿Son útiles? 16
¿Cómo se crean? 16
¿En que grupo estoy? 16
DICT/PROG. FIELDS *16*
MENUS DE LA EDICIÓN DE LA SCREEN PAINTER *19*
LISTA DE CAMPOS 21
LOGICA DE PROCESO 22
COMO ESTRUCTURAR UNA SCREEN PAINTER 23
MENU PAINTER **24**
BARRA DE MENÚS 25
BARRA DE SIMBOLOS 26
BARRA DE PULSADORES 27
PASO 1 (CREAR FUNCIÓN) *28*
PASO 2 (ASOCIAR FUNCIÓN A UN BOTÓN) *29*
MIRAR COMO QUEDA *30*
AVISOS *30*
FUNCIONES DEL SISTEMA 31
TITULO DE LA DYNPRO 33
DYNPROS **34**
TIPOS DE DYNPRO: 34
ATRIBUTOS DE LA DYNPRO: 34
Retener datos *34*
Posición del cursor *34*
FUNCIONES DE TRATAMIENTO DE CAMPOS 34
COMO ASOCIAR VARIAS SCREENS A UN MISMO PROGRAMA 34
COMO PASAR PARÁMETROS DE UNA DYNPRO A OTRA. 35
DYNPROS DE UN MISMO PROGRAMA *35*
ENTRE DIFERENTES PROGRAMAS *35*
EXPORT 35
IMPORT 36
COMO SE UTILIZAN 36

COMO MODIFICAR LOS OBJETOS DE UNA DYNPRO 38
PASO 1 (Poner el módulo) *38*
PASO 2 (Llamarlo desde el programa) *39*
INCOVENIENTES *40*
COMO ASOCIAR UNA TABLA A UNA DYNPRO 40
CONTROL DE PROCESO DE DIALOGO 43

EJEMPLO PRÁCTICO DE LA SCREEN PAINTER, MENU PAINTER, TRANSACCIONES Y MODUL-POOL 44

PASO 1 (CREAR MODUL-POOL) 44
PASO 2 (ASOCIAR AL MODUL POOL UNA TRANSACCIÓN) 46
PASO 3 (CREAR UNA SCREEN PAINTER) 47
PASO 4 (SOLO FALTA EJECUTARLO) *53*

EJEMPLO DE UN MODUL-POOL CON DOS DYNPROS ASOCIADAS Y EN UNA DE ELLAS UN STEPLOOP 55

PASO 1 (CREAR EL MODULPOOL) 55
PASO 2 (CREAR LA PRIMERA DYNPRO) 55
PASO 3 (CREAR LA SEGUNDA DYNPRO) 58
PASO 4 (SOLO FALTA EJECUTARLO) 62
POR ÚLTIMO 62

FORMULARIOS 65

MENSAJES EN SAP 67

TIPOS DE MENSAJES 69
VARIABLES DEL SISTEMA SOBRE MENSAJES 69

DEBUGGING 70

TIEMPOS DE VELOCIDAD 72

OPTIMIZACION DE LOS TIEMPOS DE EJECUCIÓN 74
DISTINCION DE CASOS *74*
TECNICAS SUBPROGRAMAS. VARIABLES LOCALES *74*

TRATAMIENTO DEL SPOOL 75

LLAMADA A TRAVÉS DE UN ABAP 75

JOB 76

GENERAR UN JOB MEDIANTE UN PROGRAMA ABAP/4. 80

ACTUALIZACION ASINCRONA 85

INCLUIR UNO O VARIOS REGISTROS DE ACTUALIZACION 85
F3/F11 DESPUES DE CREAR EL REGISTRO LOG 85

PANTALLAS DE SCROLL (BÁSICO) 87

VARIANTES 90

PROGRAMAS DE EJEMPLO 92

PAGINACIÓN 92
LISTADO 95

CAMPOS CURRENCY 99
VARIABLES INTERMEDIAS 99
TIPOS DE OPERACIONES 99
FUNCIONES DE CONVERSIÓN 100
RECOMENDACIONES 102
ERRORES DEL SISTEMA 103
NOMENCLATURA DE SAP 104
CO-CCA /CO-PA /FI / HR/CO-PC 104
SD / MM/PM 105
ICONOS DE SAP 107
NOTA DEL AUTOR 109
AGRADECIMIENTOS 110

SCREEN PAINTER

Para crear una SCREEN PAINTER hemos de ir a la pantalla de ABAP/4 Development Workbench.

La SCREEN PAINTER sirve para crear las pantallas de introducción de datos. Estas pantallas las denominamos dynpros.

Si en la pantalla de ABAP/4 Development Workbench no aparece el botón de SCREEN PAINTER podemos acceder a través del menú "Desarrollo" y pulsaremos F9.

Cuando seleccionemos SCREEN PAINTER nos saldrá la siguiente pantalla:

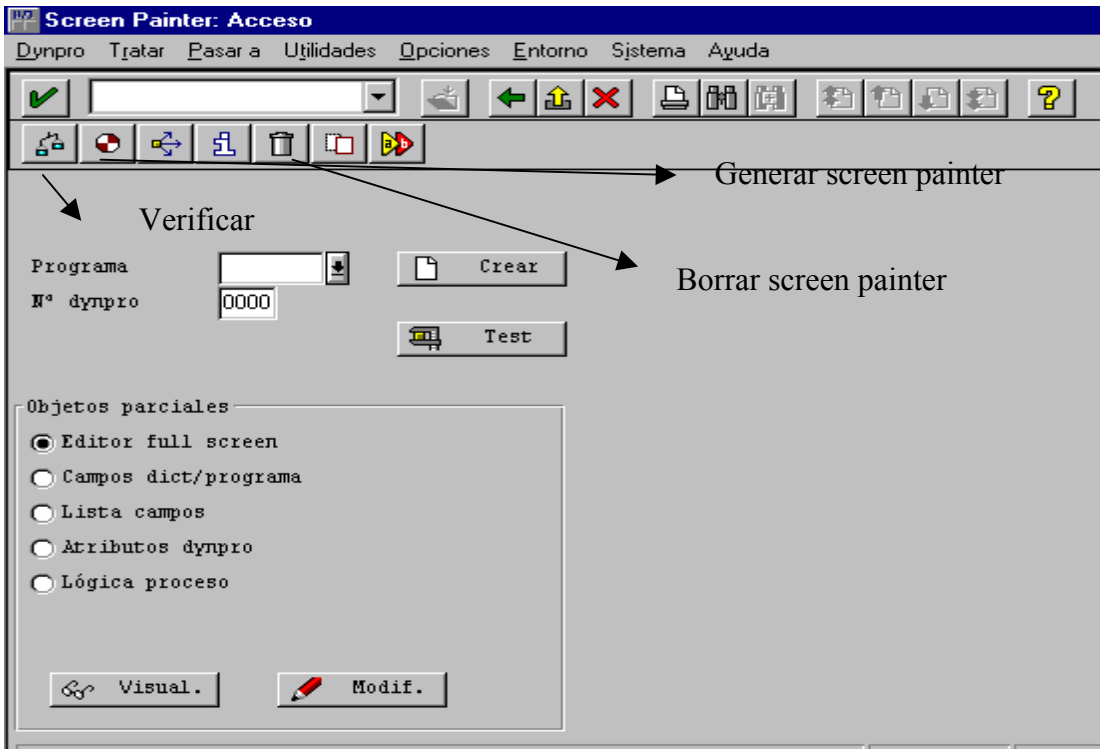


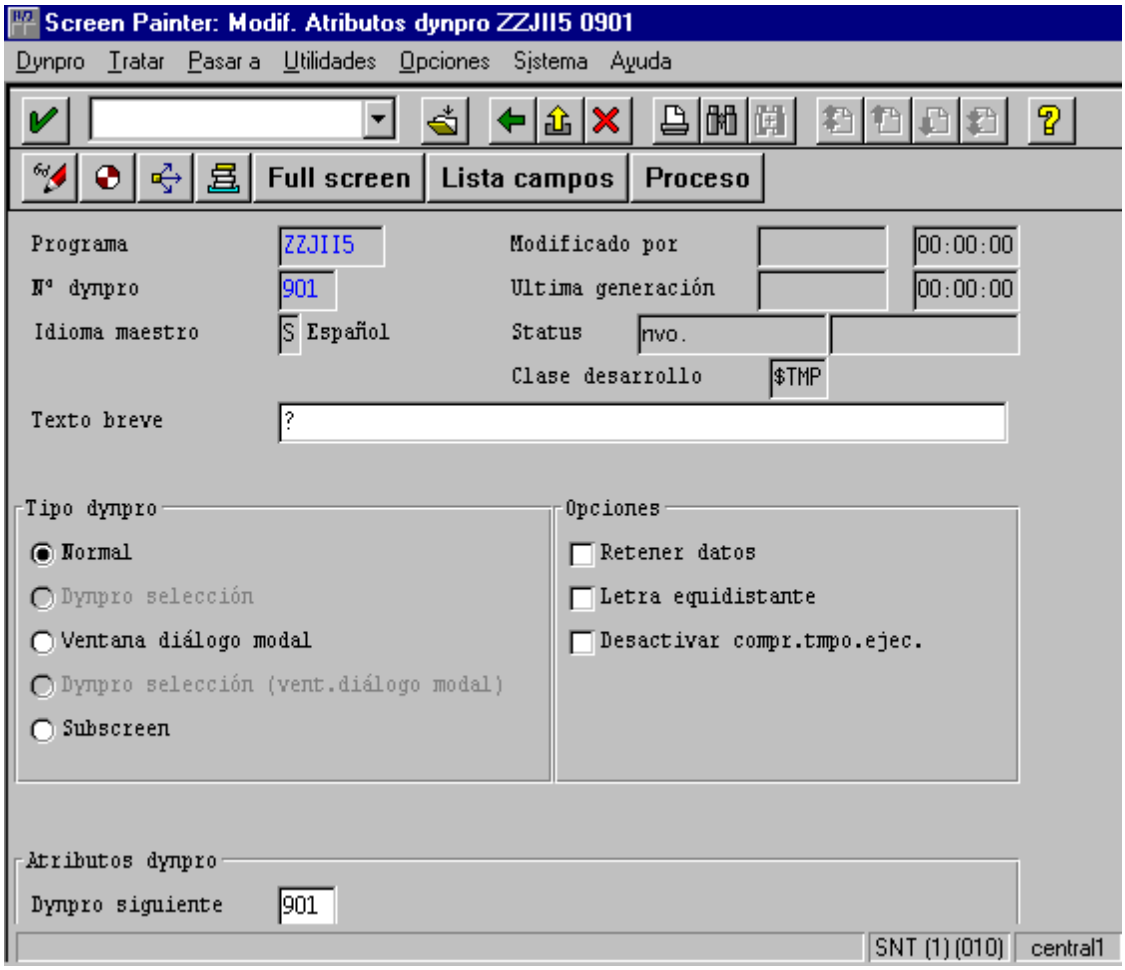
Fig. Acceso.

En "programa" introduciremos el nombre del programa al que le asociaremos la SCREEN PAINTER, que debe existir.

El Nº de Dynpro es el número de pantalla que tendrá en ese programa.

Si la creamos nos saldrá una pantalla para poner el mensaje breve de la SCREEN PAINTER.

La pantalla que sale es la siguiente:



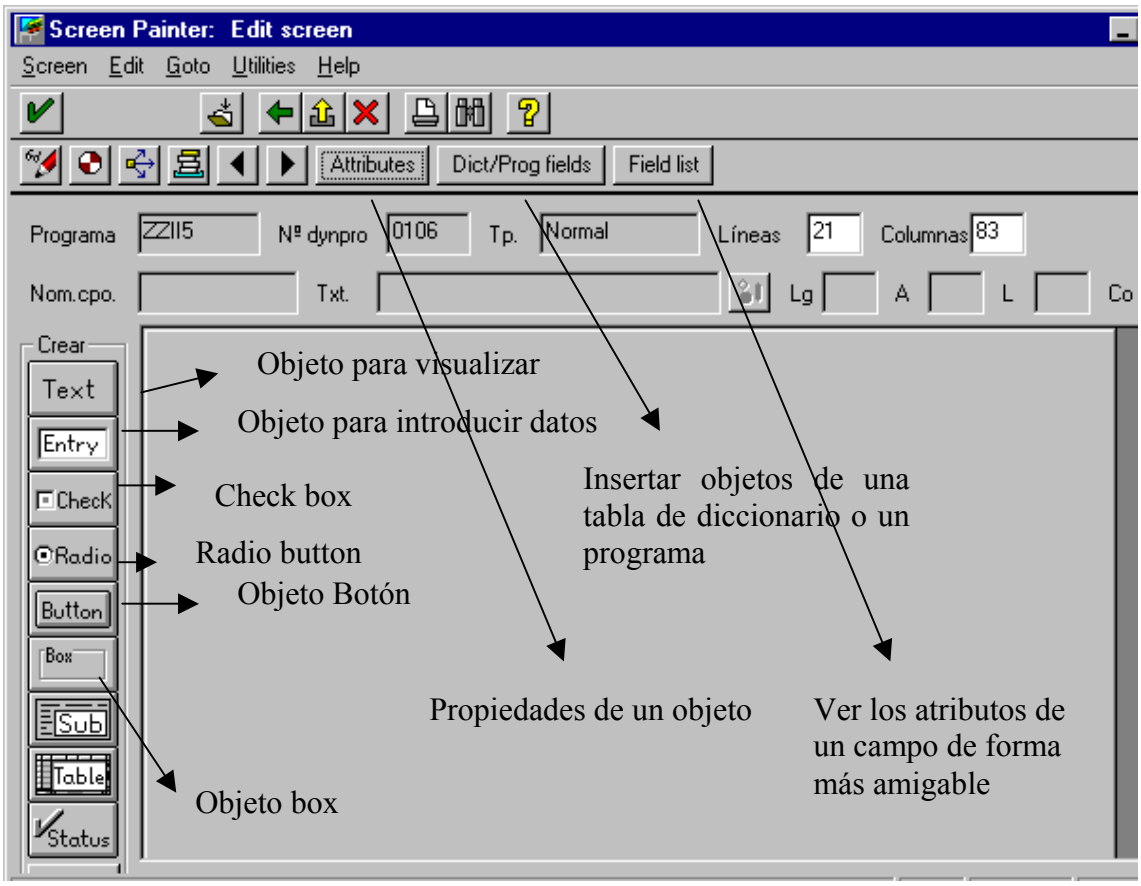
Cuando hayamos puesto los atributos de la SCREEN PAINTER, la grabaremos.

Desde aquí podemos irnos a diferentes partes de la SCREEN.

Desde el botón “FULL SCREEN” va al diseño de la pantalla.
Desde el botón “LISTA CAMPOS” va a la lista de los campos que se utilizan en la SCREEN.
Y desde el botón “PROCESO” nos vamos al código de la SCREEN PAINTER.

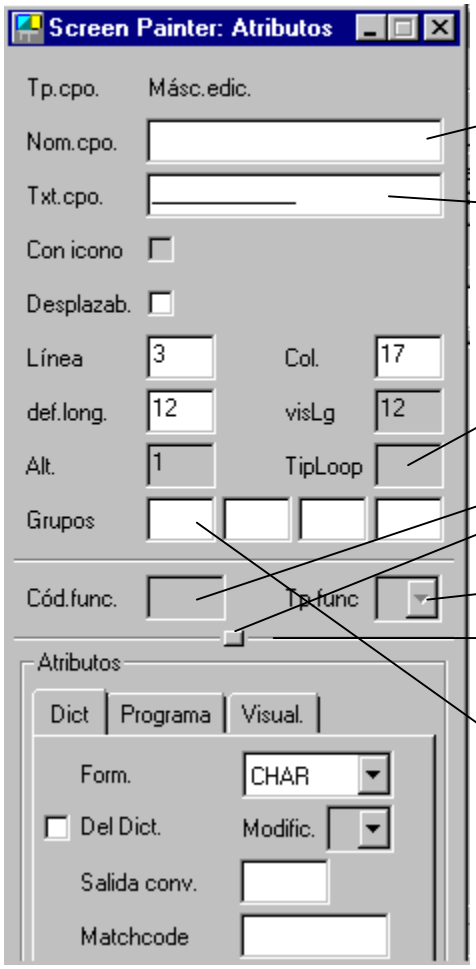
DISPOSICION DE CAMPOS EN PANTALLA (FULL SCREEN)

En la FULL SCREEN inicialmente nos saldrá la siguiente pantalla (en la página siguiente):



A partir de ahora explicaremos cómo se crean los objetos en la FULL SCREEN.

Para crearlo pulsamos sobre el objeto que queramos insertar y lo situamos donde lo queremos poner en la pantalla. Cuando esté creado hacemos doble clic sobre el objeto o pulsamos en el botón “attributes” y nos aparece la siguiente caja:



Nombre del objeto

Texto que tendrá ese campo

Tipo de steploop

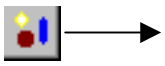
Permite mover los atributos, para poder ver lo que hay más abajo.

Función asociada a un objeto

Tipo de función

Permite mover la caja de atributos, para poder ver el contenido que hay por abajo

Grupo o grupos a los cuales pertenece el objeto



Este icono permite poner un icono a un objeto, si en objeto no aparece es que no se le puede poner.

Los campos comúnes para todos los objetos son:

Nom.cpo -> Nombre del objeto
Text.cpo -> texto del objeto, lo que saldrá en pantalla.

BOTONES

Crear botones es relativamente sencillo, como hemos visto antes, pero lo que conviene saber son los tipos de función que se pueden relacionar al código de función de un objeto (véase Fig. Ventana Propiedades).

Los tipos de funciones son las siguientes:

- <none> -> No se les asocia ningún tipo de función. Personalizada por el programador.

- E -> Exit command -> Sirve cuando asociamos un botón para salir de una pantalla. Cuando la ponemos, SAP no realiza ningún tipo de comprobación al salir por este botón, todo lo contrario ocurre cuando no lo ponemos, SAP

hace comprobaciones antes de salir (y no se sabe lo que puede pasar.....). Esta función por sí sola no se ejecuta, hay que poner una orden para que salga.

- T -> Transaction -> Es una forma de llamar a otra transacción a través de un botón, aunque también se puede hacer a través del programa. Para un programador sería más fácil y cómodo hacerlo a través del botón, pero si viene otro a modificar el programa no sabrá que se llama desde ahí (a no ser que el programa este documentado). Pero si se hace a través del programa se ve (aunque no esté documentado) lo que se hace, por ejemplo si queremos llamar a transacción "TZ10" desde un botón donde pone "Cod. func." escribiremos la transacción: "TZ10" y "Tpo. Func" le pondría el tipo T.
- H -> Help function
- S -> System function
- P -> Tabscript code

Propiedades de los botones

Nom. Icono -> Permite asociar un icono al botón. Si pulsamos el icono que está a la derecha del campo nos saldrá una lista de los iconos que SAP tiene. Si lo activamos veréis que el campo de abajo ("Quick info") se activa poniendo el nombre de la función de ese icono.

Del Dict. -> "Atributos", "pestaña Dict" -> Sirve para asociar el campo de una tabla de diccionario a un botón. El nombre del botón ha de estar puesto en el campo de la tabla que queramos, ejemplo: "SPFLI-FLDATE".

Campo salida -> Atributos, pestaña programa -> Sirve para indicarle si ese botón va a visualizar algo o no.

Invisible -> Atributos, pestaña visual -> Indicamos si se va a mostrar ese campo o no.

ENTRY

Los entry son un tipo de objeto que sirven para introducir datos por teclado

PROPIEDADES ENTRY

Nom. Icono -> Permite asociar un icono al botón. Si pulsamos el icono que está a la derecha del campo nos saldrá una lista de los iconos que SAP tiene. Si lo activamos veréis que el campo de abajo ("Quick info") se activa poniendo el nombre de la función de ese icono.

Del Dict. -> "Atributos", "pestaña Dict" -> Sirve para asociar el campo de una tabla de diccionario a un botón. El nombre del botón ha de estar puesto en el campo de la tabla que queramos, ejemplo: "SPFLI-FLDATE".

Campo salida -> Atributos, pestaña programa -> Sirve para indicarle si ese botón va a visualizar algo o no.

Invisible -> Atributos, pestaña visual -> Indicamos si se va a mostrar ese campo o no.

RADIO BUTTON

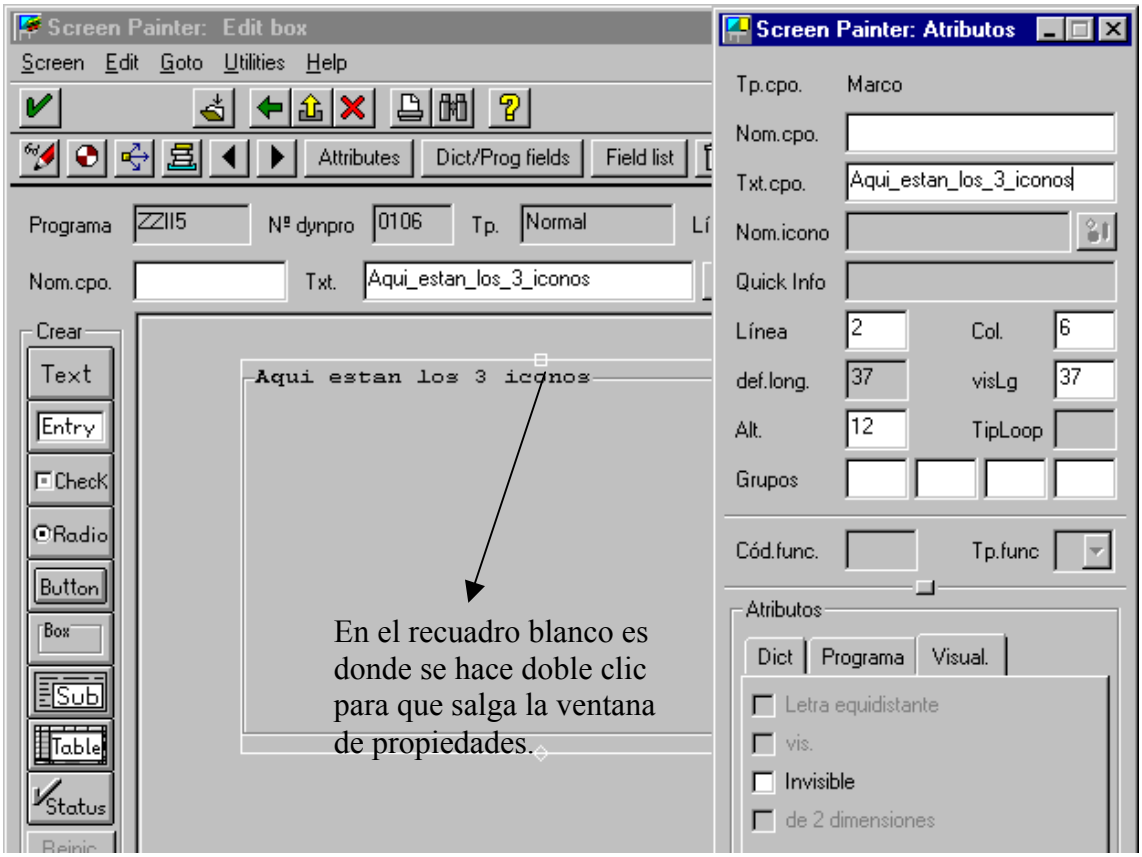
Los radio button se utilizan para que el usuario pueda escoger de una forma u otra cómo tratar los datos.

Al principio se puede pensar que es fácil de utilizar, caso correcto si sólo ponemos uno. Pero si queremos poner tres y cuando se pulse uno se desactiven los otros pulsados ya es más difícil de hacer. Vamos a poner un ejemplo ilustrado de cómo crearíamos tres radio button.

Paso 1 (crear un box)

Un box es un recuadro donde podemos introducir cualquier objeto, es muy útil cuando tenemos varios objetos y queremos separarlos unos de otros.

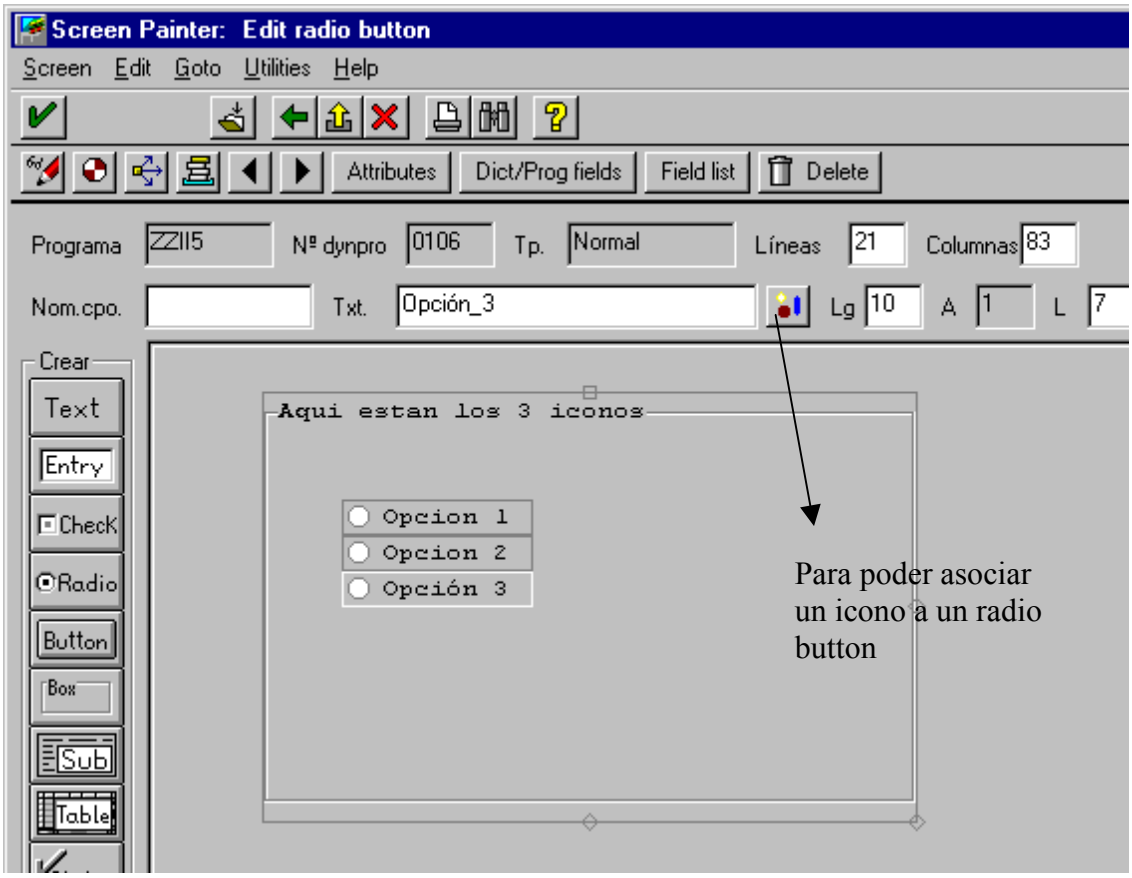
Primero haremos el box, lo haremos suficientemente grande para que entren los tres radio buttons. Hacemos doble clic en un sitio (se pondrá en la imagen) y nos saldrá la ventana de propiedades (Fig. Ventana propiedades) y pondremos un título al box. Veamos esta imagen:



En este caso, el box es bastante grande.

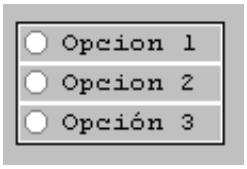
Paso 2 (poner los tres buttons)

Este paso es fácil, ponemos los tres radio buttons y a cada uno le pondremos un nombre diferente. Que no se nos olvide ponerle un nombre diferente a cada uno, ya que al generarlo no dará error pero cuando lo ejecutemos nos llevaremos una desagradable sorpresa. Así quedaría:



Paso 3 (seleccionarlos y unirlos)

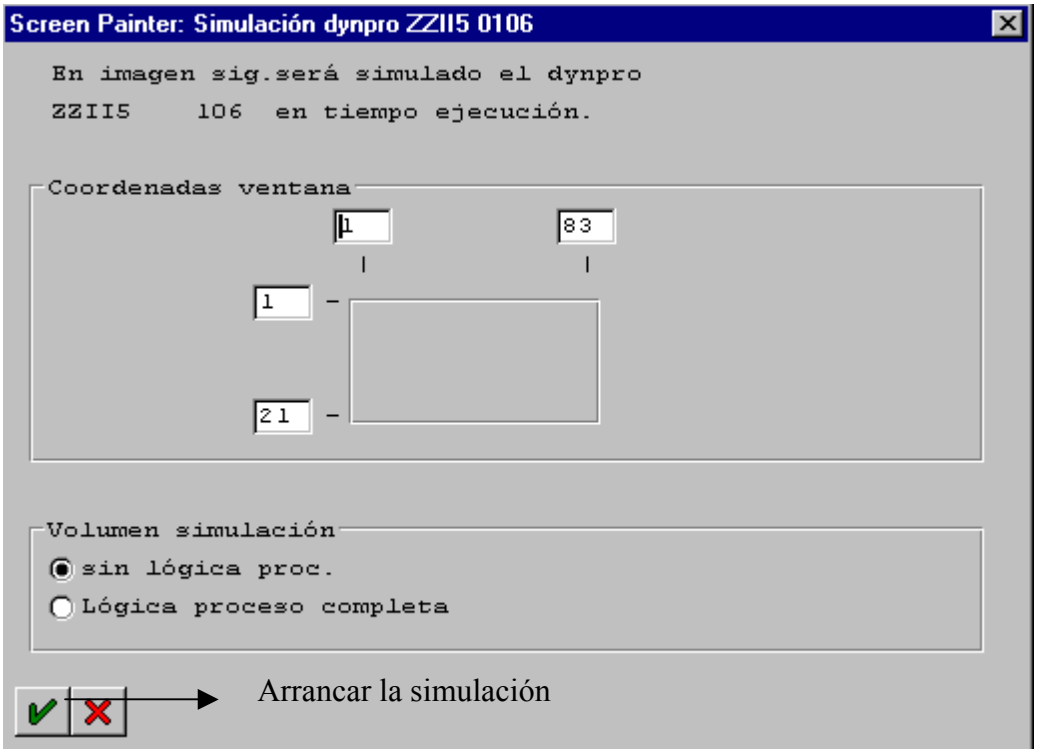
Ahora con el botón izquierdo seleccionamos los tres radios buttons. ¿Cómo se hace?. Pulsando el botón izquierdo (solo en ratones con configuración para personas diestras, si el ratón es para zurdos sería el botón derecho) en el extremo superior izquierdo y si movemos el ratón, sin soltar el botón izquierdo, veremos como aparece un recuadro, entonces lo movemos hasta seleccionar los tres. El resultado seria el siguiente:



Cuando lo seleccionamos vemos que aparece un recuadro negro que envuelve a los tres "radio buttons", aseguraros que envolvéis a los tres. Después lo grabaremos.

Ahora solo queda ir al menú: "Edit", "Radio button group", "define" y automáticamente nos unirá los tres "radio buttons". Si queremos probarlos vamos al menú: "Screen", "test". Para comprobar que se han unido aparecerá un recuadro con líneas discontinuas que envuelve a los tres botones.

Después os saldrá una pantalla para el tipo de simulación que queréis, la pantalla es la siguiente:



Si pulsáis el botón de arrancar la simulación, veréis como queda la pantalla.

Paso 4 (cómo lo relacionamos con el programa)

A los "radio buttons" no se les puede asociar un código de función.

En el programa o "modul-pool" que tenemos asociado declaramos una o más variables dependiendo de cuantos "radio buttons" tengamos. En el caso anterior tenemos tres "radio buttons" con los nombres: opcion1, opcion2, opcion3. En el programa declararíamos lo siguiente:

```
DATA: OPCION1 TYPE C.
DATA: OPCION2 TYPE C.
DATA: OPCION3 TYPE C.
```

Recordar que los nombres que declaremos han de tener el mismo nombre que los "radio buttons" que tengamos en la "SCREEN PAINTER".

Ahora para saber que "radio button" se ha pulsado SAP pone una X (mayúscula) a la variable que esta activa. Es decir, si pulsamos el radio button con el nombre opcion2 la variable opcion2 tendrá una X (mayúscula) y el resto de opciones estará en blanco.
Y a partir de ahí averiguando que radio button tiene una X haremos una cosa u otra.

Propiedades del radio button

Del Dict. -> Atributos, pestaña Dict -> Sirve para asociarle un campo de una tabla de diccionario a un "radio button". Para poder relacionarlo en nombre del radio button ha de ser el mismo que un campo de la tabla que seleccionemos.

Campo salida -> "Atributos", "pestaña programa" -> Sirve para indicar si ese "radio button" puede ser pulsado o no, o sea si esta activado o no.

Invisible -> "Atributos", "pestaña visual" -> Indicamos si se va a mostrar ese "radio button".

CHECK BOX

Los "check box" son parecidos a los "radio buttons" pero la diferencia es que de "radio buttons" solo se puede escoger uno, mientras que de "check box" podemos escoger varios de ellos.

Por ello, la opción de unirlos como los "radio buttons" no existe. Pero la forma de saber que "check box" se ha pulsado es la misma que en los "radio buttons".

Primero declaramos la variable o variables con los mismos nombres que les hemos puesto en la pantalla (acordados de poner nombre a los "check box" que hagáis). Las variables han de ser de tipo C (Char).

Para saber si se ha activado o no, hay que mirar si la variable contiene una X (mayúsculas) o esta en blanco.

Propiedades del check box.

Del Dict. -> "Atributos", "pestaña Dict" -> Sirve para asociarle un campo de una tabla de diccionario a un "radio button". Para poder relacionarlo el nombre del "radio button" ha de ser el mismo que un campo de la tabla que seleccionemos.

Campo salida -> "Atributos", "pestaña programa" -> Sirve para indicar si ese "check box" puede ser pulsado o no, o sea si esta activado o no.

Invisible -> “Atributos”, “pestaña visual” -> Indicamos si se va ha mostrar ese “check box”.

BOX

Los “box” sirven para separar unos objetos de otros. Muy útil cuando hay bastantes objetos por la pantalla.

Propiedades del box.

Del Dict. -> “Atributos”, “pestaña Dict” -> Sirve para asociar un campo de una tabla de diccionario a un “radio button”. Para poder relacionarlo el nombre del “radio button” ha de ser el mismo que un campo de la tabla que seleccionemos.

Campo salida -> “Atributos”, “pestaña programa” -> Sirve para indicar si el “box” va a visualizar algún dato o no.

Invisible -> “Atributos”, “pestaña visual” -> Indicamos si se va a mostrar el “box” o no.

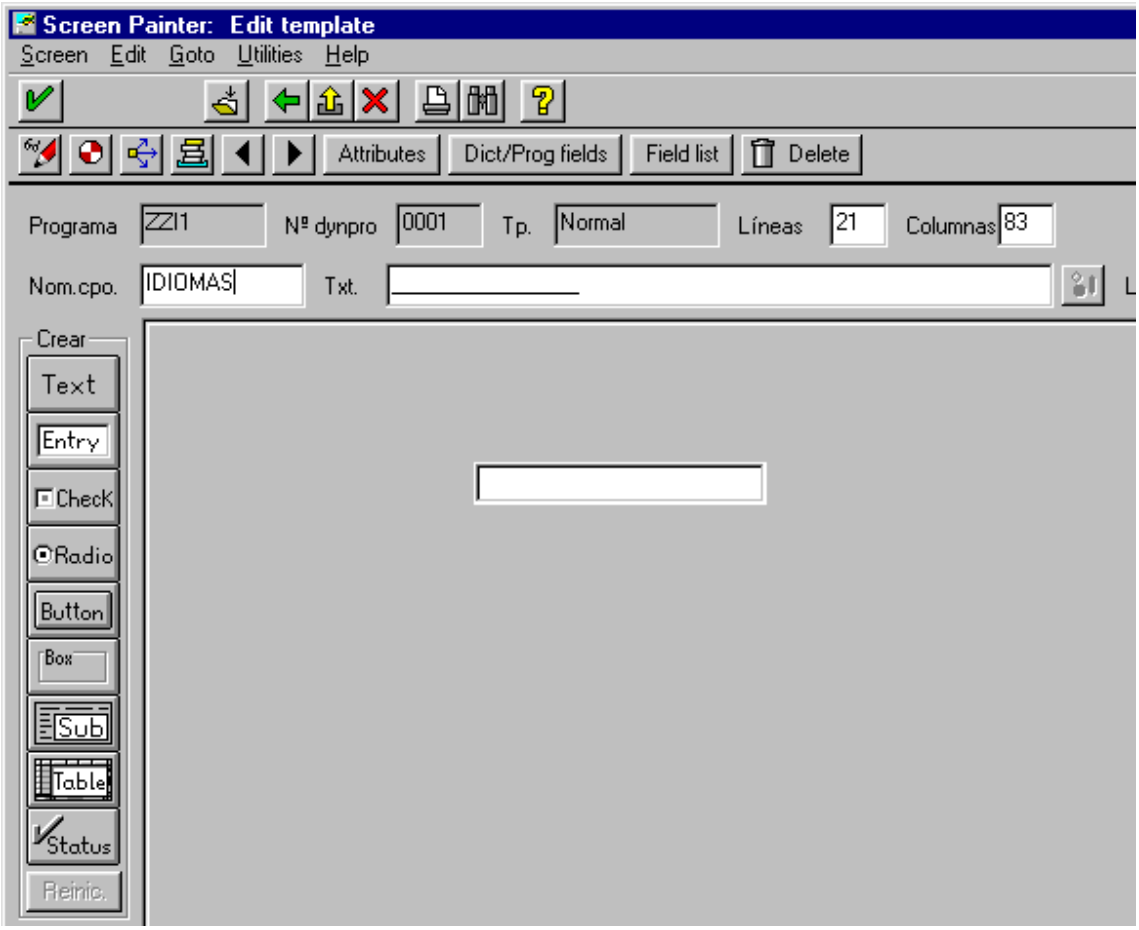
STEPLOOP

Los steploop sirven para crear un array de un campo. Solo se pueden crear como objeto de entrada, es decir de tipo “entry”.

Se crean desde la full screen de la screen painter. Para hacerlo hemos de seguir los siguientes pasos:

Paso 1 (crear el objeto)

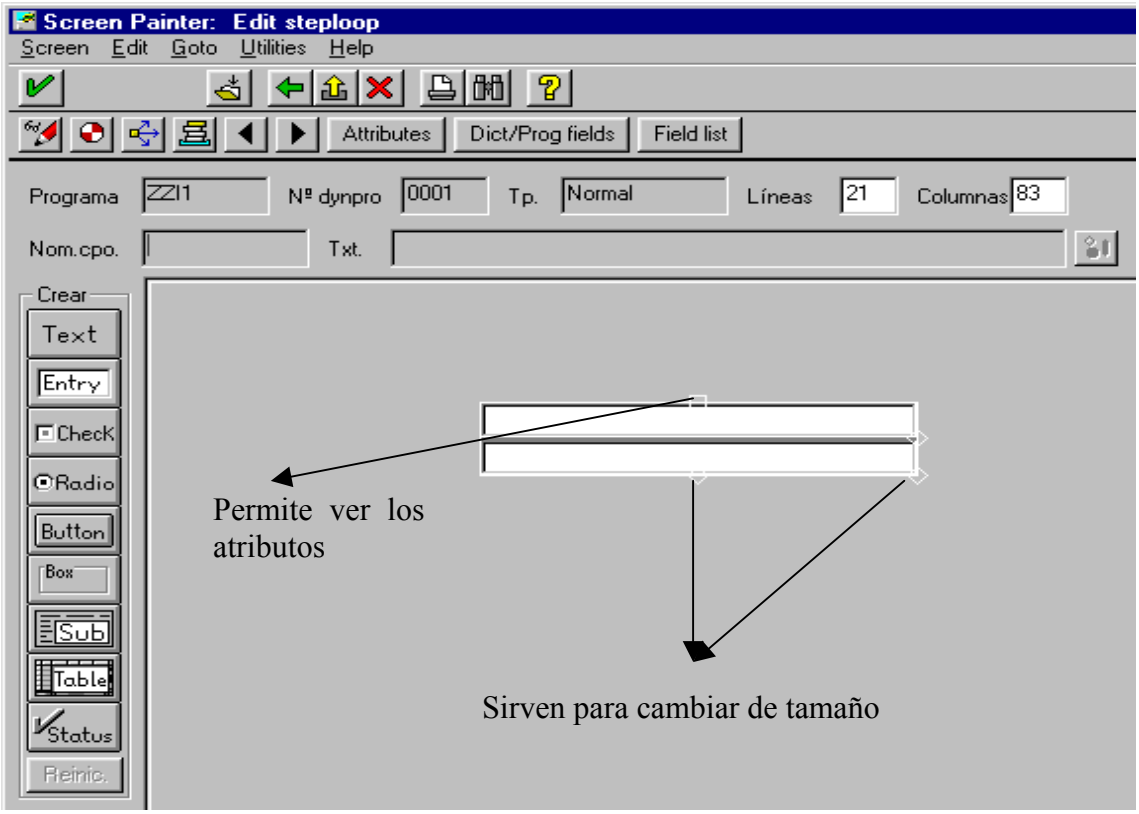
En este ejemplo haremos un sencillo “entry”, como en la siguiente pantalla:



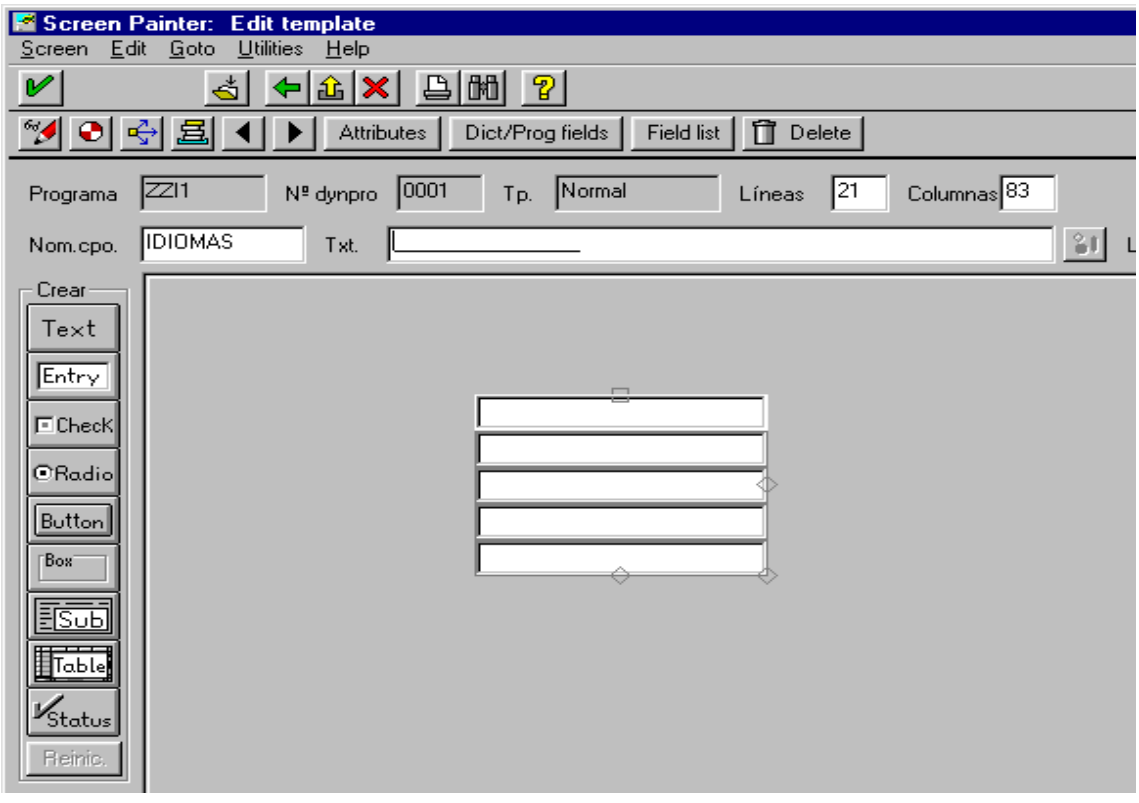
Hemos creado un objeto de tipo “entry” con el nombre “idiomas”.

Paso 2 (definir el steploop)

A continuación vamos al menú “edit”, ”steploop”, “define”. Cuando lo hagamos nos saldrá lo siguiente:



Al definirlo ha pasado a tener 2 campos (antes solo tenia 1). En está pantalla el cuadrado que sale encima del objeto sirve para ver los atributos de ese objeto (hacemos doble clic y nos saldrá) y donde aparecen los rombos sirve para cambiar de tamaño. Por ejemplo con el rombo de abajo lo pulsamos y si lo "estiramos" veremos como van apareciendo más "entrys". Como en la siguiente imagen:



Como vemos, el objeto "idiomas" ha pasado de tener 5 campos. Podemos crear tantos campos como queramos, salvo la limitación de la pantalla.

Consejos

Antes de definir el "steploop" hemos de darle un nombre al objeto, porque si no SAP no nos deja introducir el nombre más adelante.

Como he dicho solo se puede hacer como objeto de entrada, como tipo "entry".

Más adelante se explica un ejemplo de cómo se haría un modul-pool con dos dynpros y en una de ellas un steploop en el que se utiliza una tabla interna y externa.

GRUPOS DE OBJETOS

En una "SCREEN PAINTER" podemos agrupar los objetos, es más, un objeto puede estar en cuatro grupos diferentes.

¿Son útiles?

Los grupos son muy útiles cuando queremos que los objetos de un determinado grupo se escondan, que no se les puedan introducir datos o cualquier otro atributo que hemos visto en el capítulo anterior. Por ejemplo, dependiendo de los datos que introduzcamos en una pantalla, modificar los atributos de un grupo o grupos de otra pantalla.

¿Cómo se crean?

Para crearlos solo tenemos que hacer que nos salga la ventana de propiedades de un objeto (Véase. Fig. Ventana de propiedades) y donde pone Grupos, poner el nombre del grupo (de tres letras). Podemos ponerle hasta cuatro grupos diferentes a un mismo objeto.

¿En que grupo estoy?

Para saber en que grupo estamos, lo miramos a través de la tabla del sistema "SCREEN", esta tabla tiene los campos siguientes:

```
SCREEN-GROUP1.
SCREEN-GROUP2.
SCREEN-GROUP3.
SCREEN-GROUP4.
```

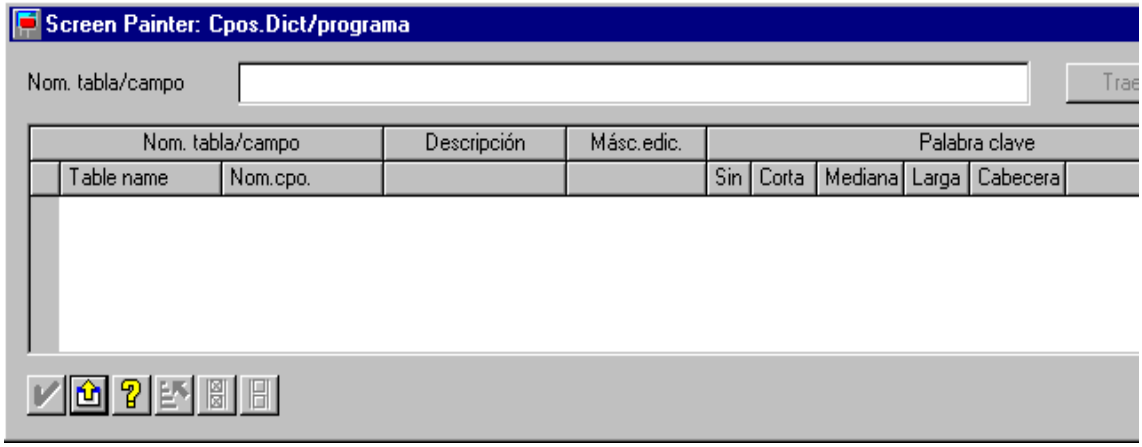
En estas variables se almacena el grupo o grupos al cual pertenece el objeto (siempre y cuando le hayamos puesto algún grupo).

DICT/PROG. FIELDS

Este botón que aparece en la parte de arriba de la full screen nos permite insertar campos de una tabla interna o de diccionario e incluso variables definidas por nosotros en el programa asociado a la dynpro

Recordar que para poder insertarse estos objetos, el programa ha de estar generado (CTRL+F3) con anterioridad a la creación de la dynpro.

Al pulsar el botón nos sale la siguiente ventana:



Donde pone "Nom. Tabla/Campo" pondremos el nombre de la tabla interna o de diccionario o una variable.
Si ponemos una tabla interna o una variable se debe pulsar el botón que pone "Traer programa" y si es una tabla de diccionario se debe pulsar el botón que pone "Traer diccionario".
Estos dos botones están a la derecha, aunque aquí no salgan (Se ve un trozo del primer botón que corresponde a "Traer diccionario").

Como ejemplo, pondremos el nombre de la tabla interna TABLA. Pulsaremos al botón de "Traer programa" y automáticamente aparecen los campos de la tabla interna.

Si hubiéramos puesto una variable o una tabla de diccionario SAP me hubiese puesto los campos relacionados con la tabla de diccionario o variable.

La pantalla que saldría sería la siguiente:

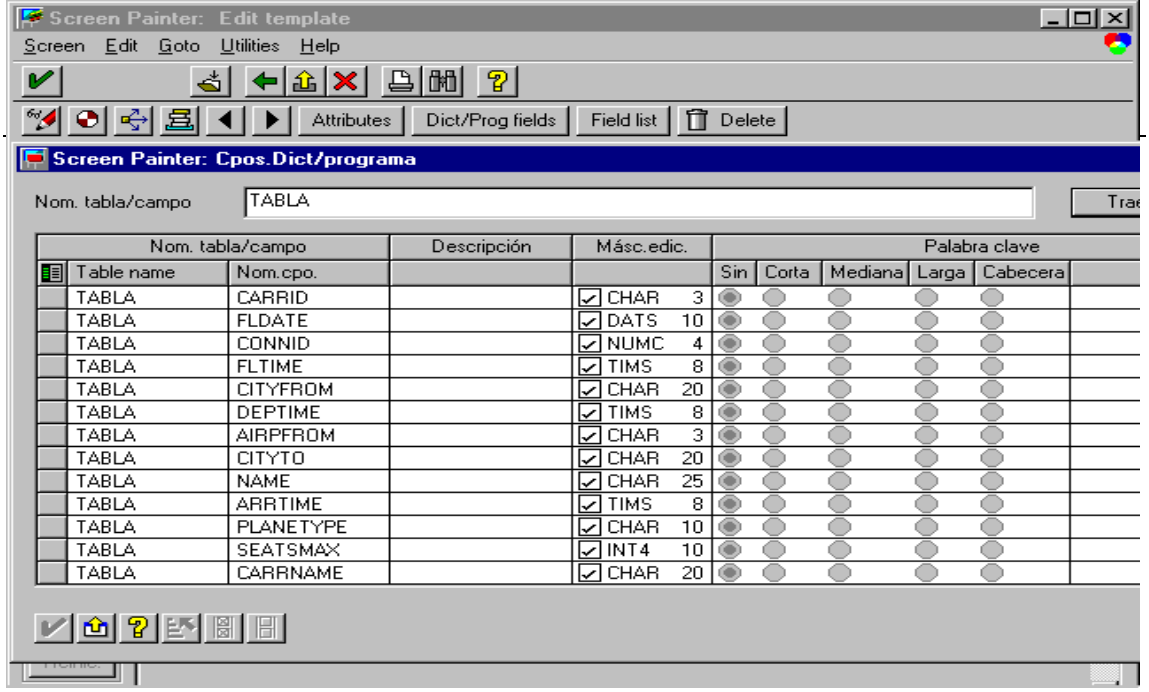
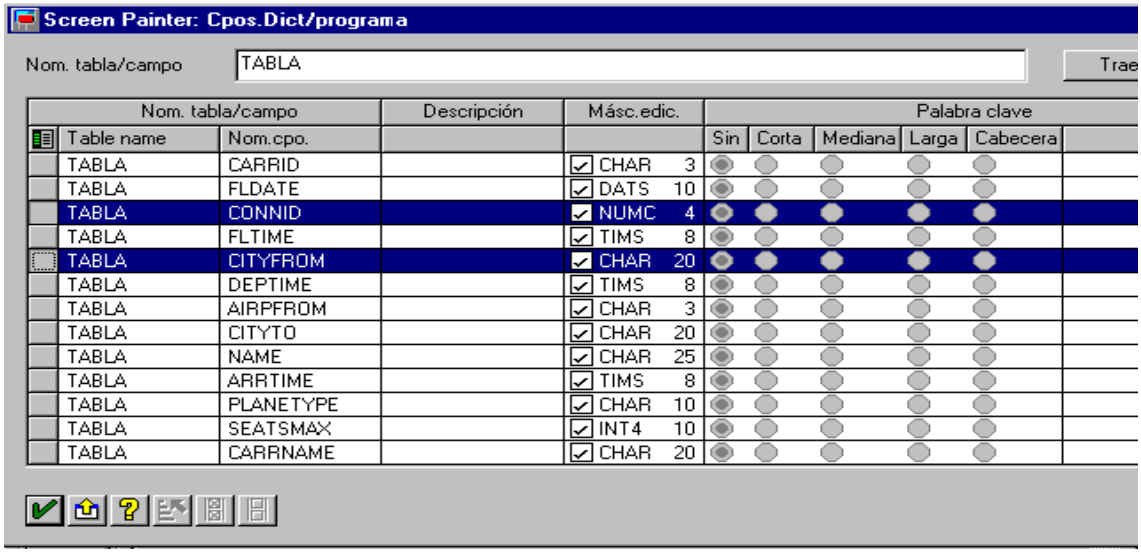
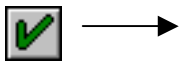


Fig. Campo Dict/programa.

Después escogeremos los campos que queramos visualizar, los escogeremos pulsando sobre el botón que está a la izquierda del nombre del campo. Véase este ejemplo:
Por ejemplo escogeremos los campos “CONNID” y “CITYFROM”.



Después los escogeremos con botón siguiente:

 Confirmación de los campos

Y sólo hace falta ponerlo en la Pantalla. Quedaría así:

Como vemos esos campos no tienen descripción (todo lo contrario ocurre cuando se utiliza una tabla de diccionario que ya posee una descripción). La descripción la pondremos nosotros con el objeto: "Text".

MENUS DE LA EDICIÓN DE LA SCREEN PAINTER

El primer menú que tenemos es "Screen" donde tenemos:

Other Screen -> Sirve para pasar de una "dynpro" a otra.
Display <-> Change -> Cambia el modo de visualización al modo de modificación y viceversa.
Check -> Realiza un chequeo de la pantalla. Dentro de este menú hay 3 submenús más: "Syntax " (Sintaxis), "Consistency " (Consistencia), "Layout".
Save without check -> Graba la pantalla pero no la chequea.
Generate -> Genera la pantalla.
Test -> Podemos ver como queda la pantalla, sin tener que ejecutar todo el programa. Muy útil para comprobar el diseño de la pantalla.

Print -> Imprime la pantalla.

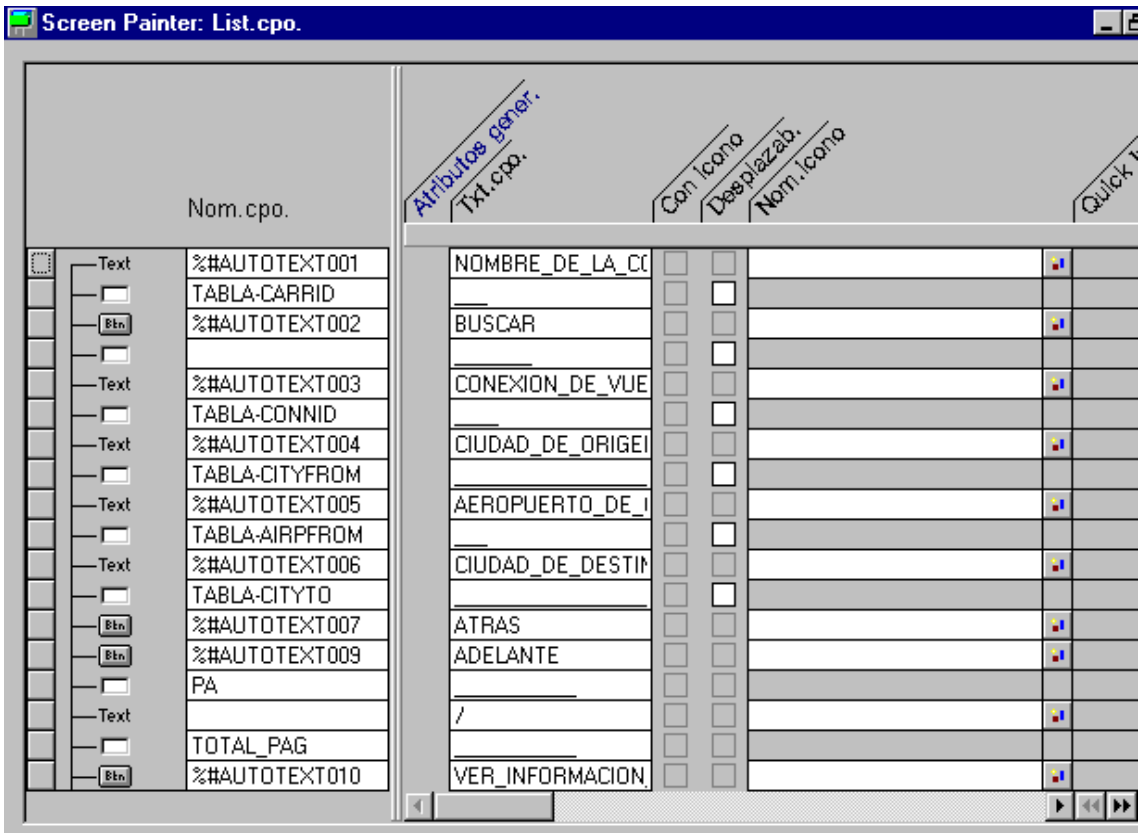
El segundo es el "Edit", que tiene las siguientes opciones:

Redraw -> Refresca la pantalla.

Radio Button group -> Cuando hemos seleccionado un grupo de “radio buttons”, permite agruparlos.
Steploop -> Los “Steploop” nos permite hacer un “array” de un campo.
Delete -> Borro un objeto que tengamos seleccionado.
Cancel -> Cancela la pantalla.

El tercer menú es del “GOTO”, que tiene las siguientes opciones:

Field list -> Saca una lista con los objetos que utilizamos, un ejemplo de lista sería este :



Aquí nos saldrían de una forma más visual los objetos que utilizamos.

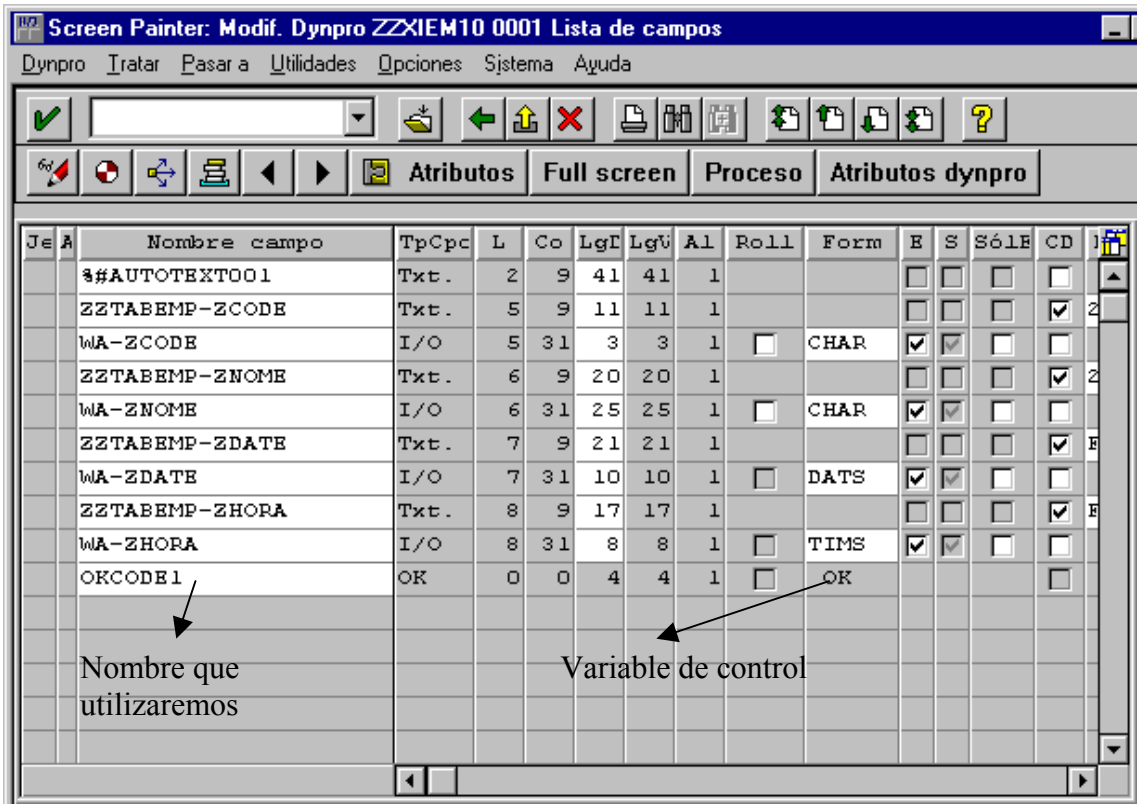
Attributes -> Nos muestra los atributos del objeto seleccionado (Véase Fig. Ventana de propiedades).
Screen attributes -> Nos saca la pantalla de propiedades de la “dynpro”.
Flow logic -> Nos va al proceso lógico del programa.
Dict/Prog Fields -> Nos saca una ventana con la que podemos insertar una variable o un campo en la pantalla (Véase Fig. Campo Dict/programa).
Next element y Previous element -> Permite pasar de un objeto a otro, es como pulsar el tabulador.
Back -> Permite volver atrás.

El cuarto menú es de “Utilities”, cuya opción más importante es la siguiente:

Global search y global replace -> Permite buscar o reemplazar un texto que introduzcamos.

LISTA DE CAMPOS

En lista de campos nos aparecerán todos los objetos que utilizamos en la "FULL SCREEN", la pantalla que sale es la siguiente:



En este caso lo he realizado de una "FULL SCREEN" que ya contiene datos.

Al final de la lista nos aparece un objeto llamado "OK" esta variable sirve para controlar que botón (nos devuelve el código de función asociado al botón) o que tecla de función (F1 a F11, ENTER, etc.) se ha pulsado. El uso de esta variable es opcional ya que podemos utilizar la variable del sistema SY-UCOMM que sirve para el mismo cometido.

Si queremos utilizarla primero le pondremos un nombre para poder utilizarla en el programa, en este caso le hemos puesto el nombre "OKCODE1". Después tenemos que declararla en el programa se haría de la siguiente forma:

```
DATA: OKCODE1(4).
```

La variable ha de ser declarada como un CHAR de 4 posiciones.

LOGICA DE PROCESO

Para ir a este punto desde la "FULL SCREEN" volveremos atrás pulsando F3 y nos iremos a la pantalla inicial ("Fig. Acceso."). Después en objetos parciales seleccionaremos "lógica de proceso" y "modificar". O si no, desde la "FULL SCREEN" a través del menú "GOTO" , "Flow Logic". La pantalla que sale es esta:

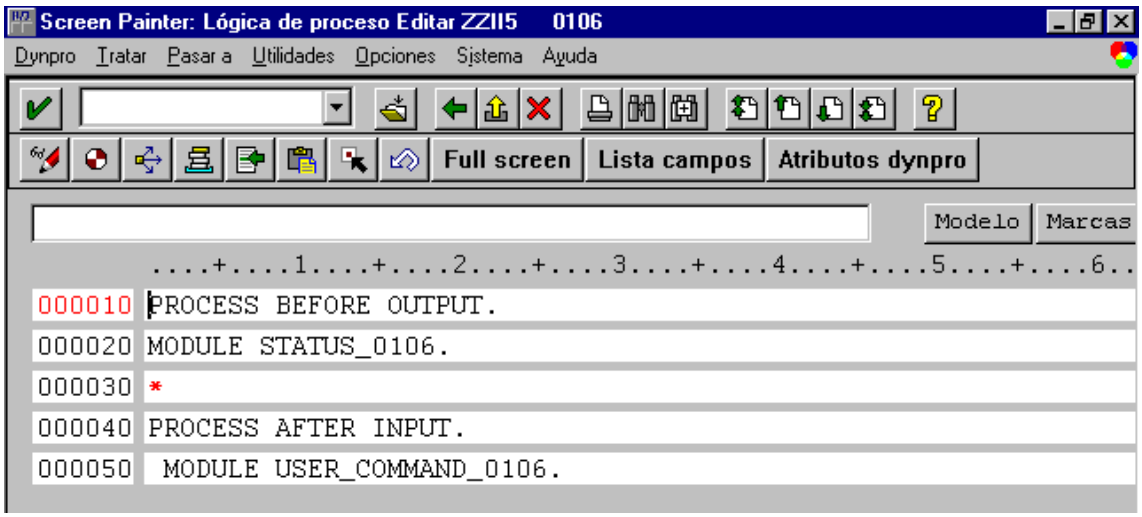
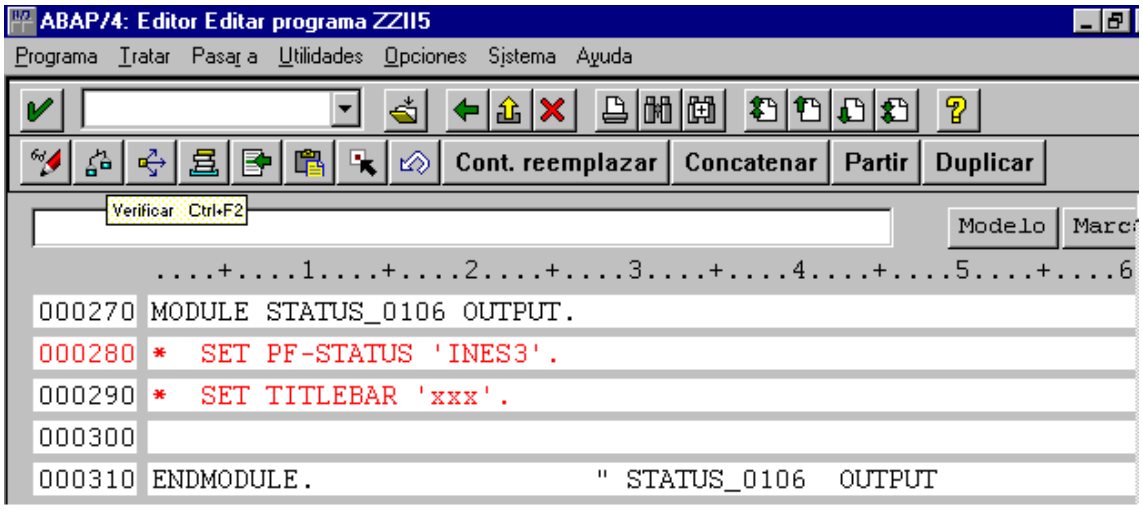


Fig. Lógica de proceso.

SAP ya pone una parte del código.

"El PROCESS BEFORE OUTPUT" es el proceso que se ejecuta antes de cargarse el programa. Si hacemos doble clic donde pone: STATUS_0106, sale la siguiente pantalla:



"El PF-STATUS" se explicará más adelante (su función es indicar que menús o botones queremos que salgan con la pantalla).

Si queremos poner los menús o los títulos tenemos que quitar él * que encontramos al principio (él * al principio es un comentario).

El “PROCESS AFTER INPUT” son las instrucciones que se ejecutan cuando se pulsa alguna tecla de función, el enter, algún botón del “MENU PAINTER” o de la “SCREEN PAINTER”.
Por defecto tiene el “MODULE USER_COMMAND_016”. Si hacemos doble clic sobre este “module” nos saldrá la siguiente pantalla:

Aquí sería donde escribiríamos el código de nuestro programa.

COMO ESTRUCTURAR UNA SCREEN PAINTER

Como ya hemos dicho una ”SCREEN PAINTER” esta ligada a un Modul-Pool, es decir, todos los módulos, procedimientos y declaración de variables están declarados en el Modul-Pool, esto puede traer confusión si un programa tiene muchas pantallas y muchos procedimientos. Para que quede bien estructurado hemos de utilizar cuatro “INCLUDES”, a saber, uno para la declaración de variables, otro para los módulos del PBO, otro para los módulos del PAI y el último para los procedimientos del programa.

MENU PAINTER

El MENU PAINTER es la herramienta que permite diseñar en la dynpro (o en una ventana) un menú con distintas opciones (barra de menú, botones, etc.).

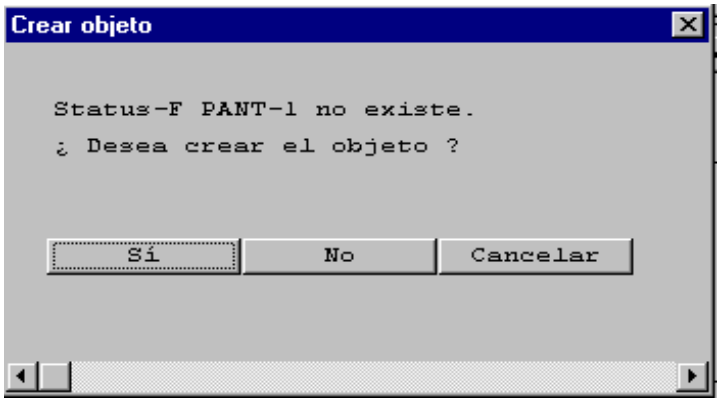
El menú painter se llama a través de la orden SET PF-STATUS, que está dentro de la lógica de proceso de la dynpro. Su sintaxis sería esta:

SET PF-STATUS ‘nombre’.

‘Nombre’ es un identificador para asociar el menú a la dynpro concreta. Por ejemplo:

SET PF-STATUS ‘PANT-1’.

Sobre PANT-1, haciendo doble clic nos saldrá la siguiente pantalla:



Si le decimos que sí saldrá esta otra:

Además de poner un texto de explicación, es aquí donde se especifica si el status a crear es para una dynpro, una ventana, etc..

Pulsando enter (o el botón del icono verde) nos saldrá la pantalla de creación de PF-STATUS. La pantalla que aparece es la siguiente:

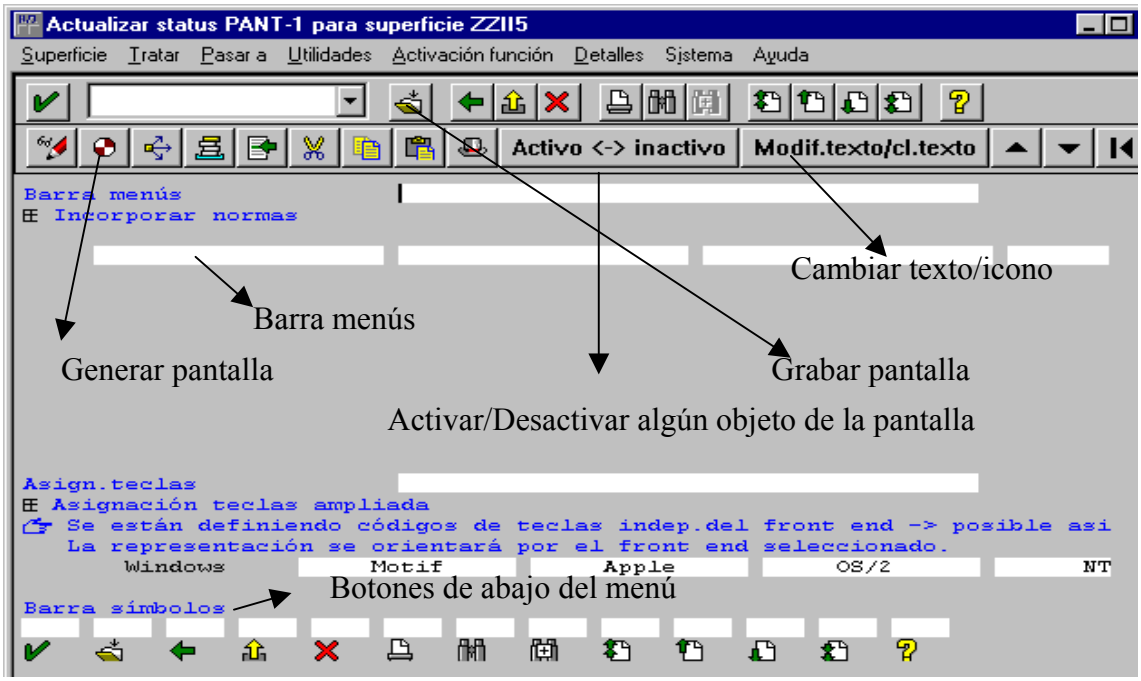


Fig. PF-STATUS.

Más abajo (no se ve) está la barra de pulsadores. Esta barra permite poner botones debajo de la barra de símbolos.

BARRA DE MENÚS

Los menús se crean en la barra de menús. A modo de ejemplo vamos a crear un sencillo menú con dos opciones y dos submenús en la primera y un submenú en el segunda. La estructura sería la siguiente:

Archivo → Buscar archivo
Salir

Edición → Pegar

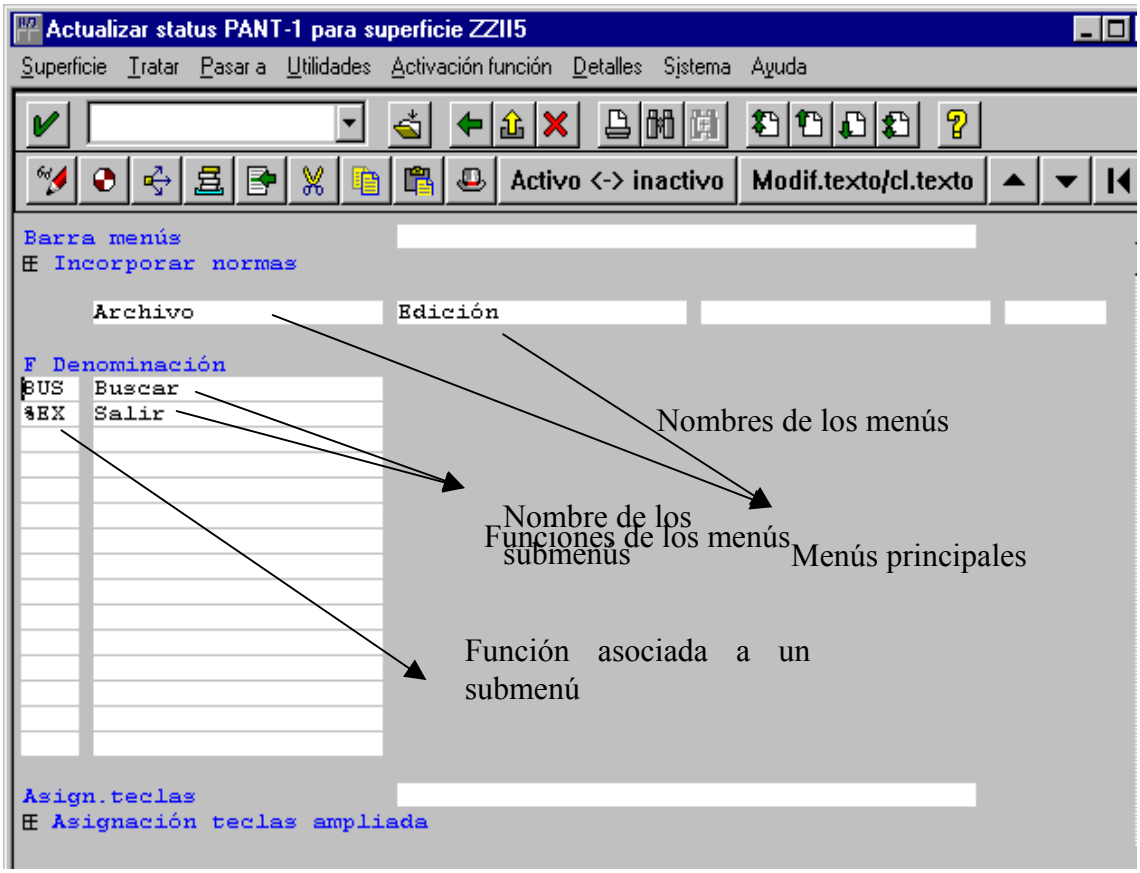


Fig. Submenú.

Donde está la 'F' en azul ponemos debajo la función que esta relacionada con el nombre del menú.
Aquí la función BUS está relacionada con el submenú BUSCAR y la función del sistema %EX (salir) está relacionada con el submenú SALIR. En el menú Edición he puesto la función PEG relacionada con el submenú PEGAR.

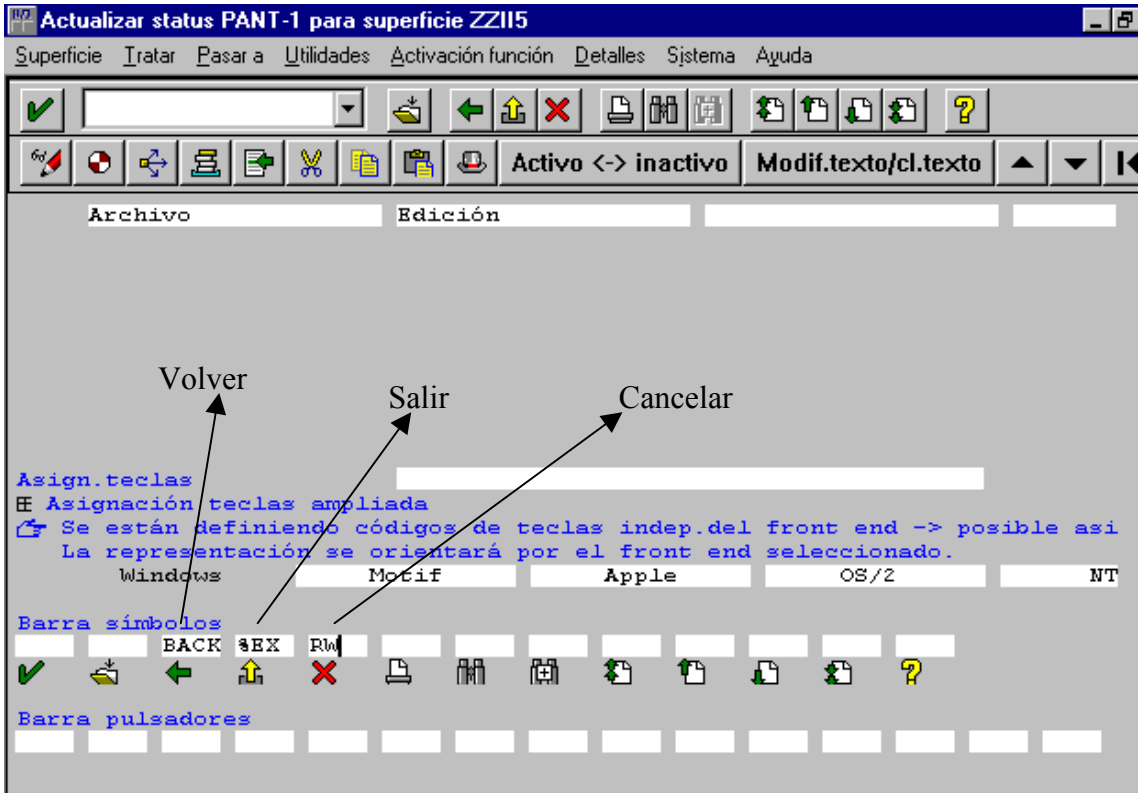
Nota: para poder introducir los submenús del menú Archivo, hacemos doble clic en Archivo y nos saldrá la pantalla de submenú (Fig. Submenú) y si se vuelve a hacer doble clic en Archivo esta pantalla se esconde.

Cabe señalar que SAP pone dos menús por defecto (Sistema y Ayuda) aunque no se vean en tiempo de diseño sí aparecen en tiempo de ejecución.

BARRA DE SIMBOLOS

Esta parte es para activar cualquiera de los iconos comunes de SAP. Solo es necesario poner el nombre de la función en los espacios que hay encima de los iconos y pulsar ENTER. Como ejemplo vamos a poner los 3 botones típicos de SAP (atrás, salir, cancelar) .

La pantalla quedaría como sigue:



Las funciones: BACK, %EX y RW son funciones ya definidas por SAP. Se suelen poner estas tres (volver, salir y cancelar). Hay más funciones internas de SAP como la de grabar, imprimir, buscar, ayuda, pero aquí no las pondremos.

BARRA DE PULSADORES

Aquí podemos crear una función propia o también podemos tomar una que ya exista

Primero crearemos una función en la barra de pulsadores. Para ello nos posicionaremos en la barra de pulsadores y escribimos un nombre de función. Vamos a crear la función "HOL", como se ve en la imagen siguiente:

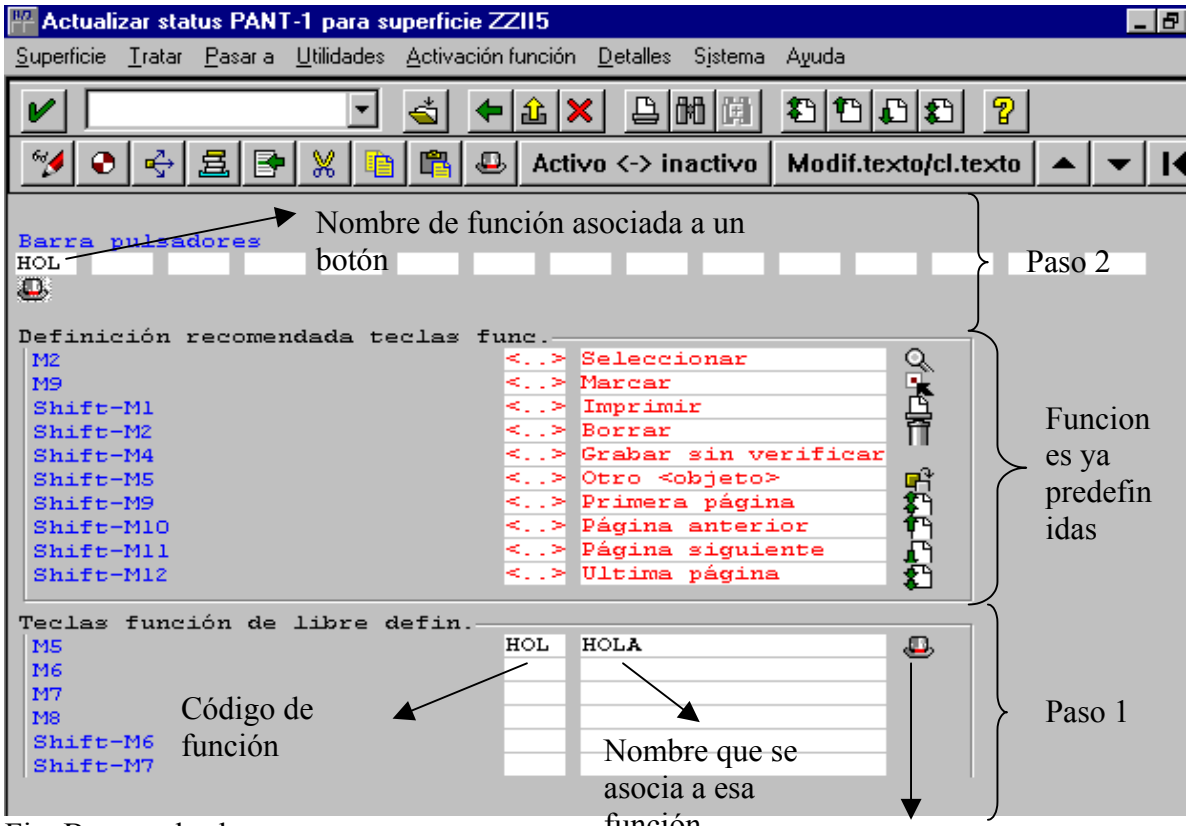
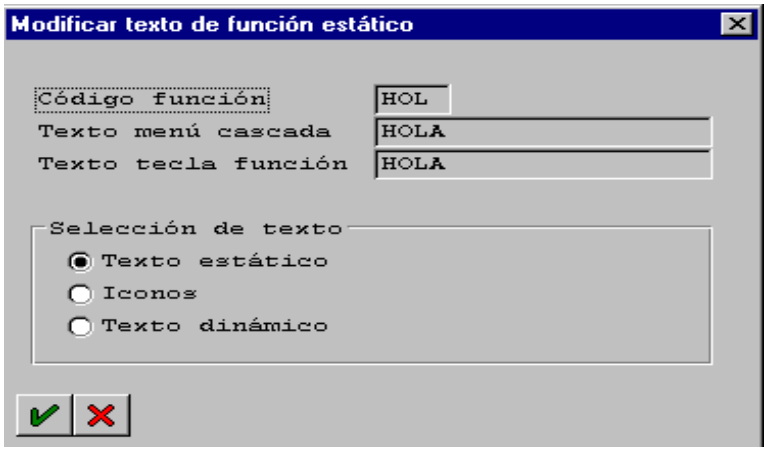


Fig. Barra pulsadores.

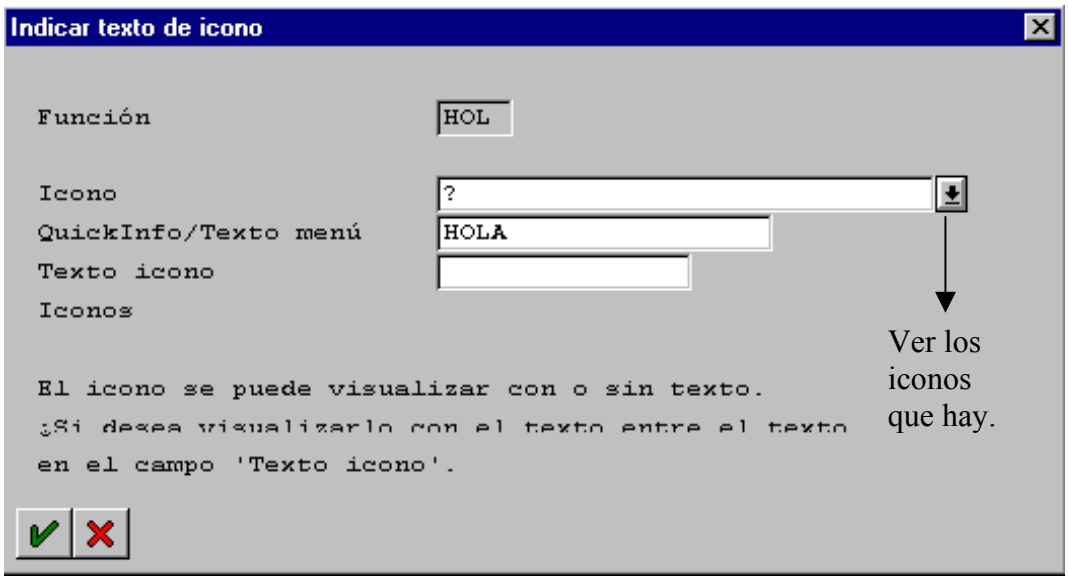
Para hacer un botón tan sencillo, se puede hacer de varias formas, una de ellas es esta:

PASO 1 (CREAR FUNCIÓN)

Donde pone "teclas función de libre defin" es donde creamos o modificamos las funciones.
En cualquiera de ellas se pone el código de función seguido del nombre que aparecerá en el botón.
Después, si queremos asociarle un icono a esa función o cambiarlo, posicionamos el cursor en el campo donde hemos escrito 'HOLA' y seguidamente pulsamos el botón del menú de arriba que pone 'Modif.texto/cl.texto'. Cuando lo pulsemos aparecerá la siguiente ventana:



Para asignar o cambiar un icono pulsamos el radiobutton que pone 'Iconos'. Después pulsando enter o el icono de confirmación sale esta otra pantalla:



Podemos ver los iconos que hay pulsando el matchcode como se ve en la figura o posicionandonos en el campo que pone 'Icono' y haciendo doble clic o pulsando F4.

Cuando hayamos seleccionado uno, pulsaremos el botón de confirmación o enter. No padezcáis ya que SAP nos "avisará" si nos hemos olvidado de poner algún dato.

Al volver veremos como sale un icono en la parte derecha del nombre de la función (véase Fig. Barra pulsadores.)

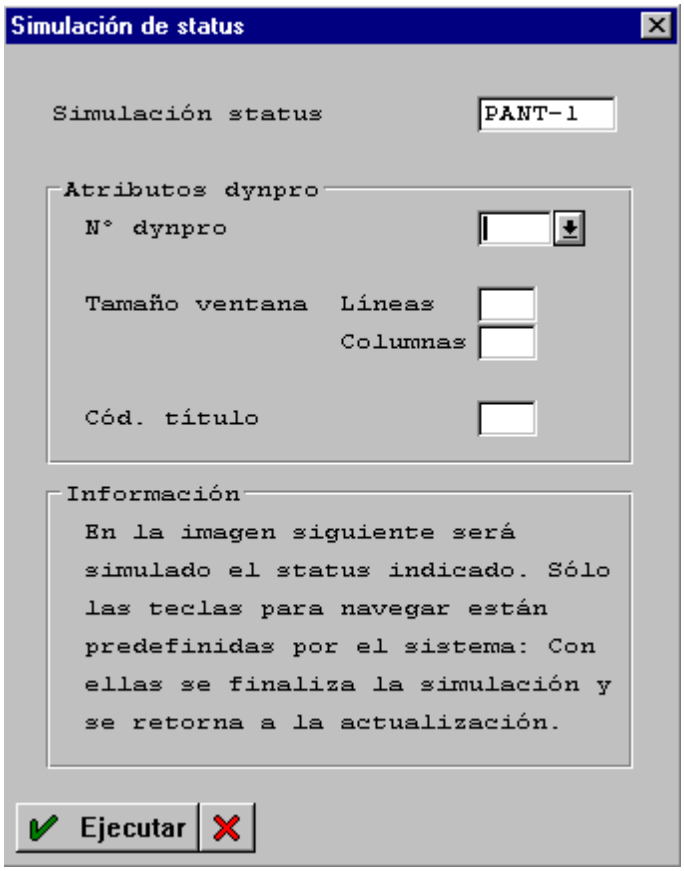
PASO 2 (ASOCIAR FUNCIÓN A UN BOTÓN)

En la barra de pulsadores donde hemos puesto el texto ('Nombre de función asociada a un botón'), justo debajo, donde pone 'Barra de pulsadores', ponemos el nombre del código de función que hemos creado (en el nuestro solo hay uno que se llama 'HOL'). Si lo escribimos y pulsamos enter veremos que debajo se pone el icono asociado a la función. Si no hay icono sale el nombre de la función.

Como ya hemos dicho antes, podemos crear funciones propias o unas que ya tiene definidas. Estas funciones se encuentran en la box "definición recomendada teclas de función", los pasos a seguir son los mismos que en una función creada por nosotros. En el recuadro que pone "<..>" escribiremos el nombre de la función y donde hemos escrito antes la función "HOL" ahora escribiremos la función escogida y automáticamente SAP nos tomará todos los valores de la función.

MIRAR COMO QUEDA

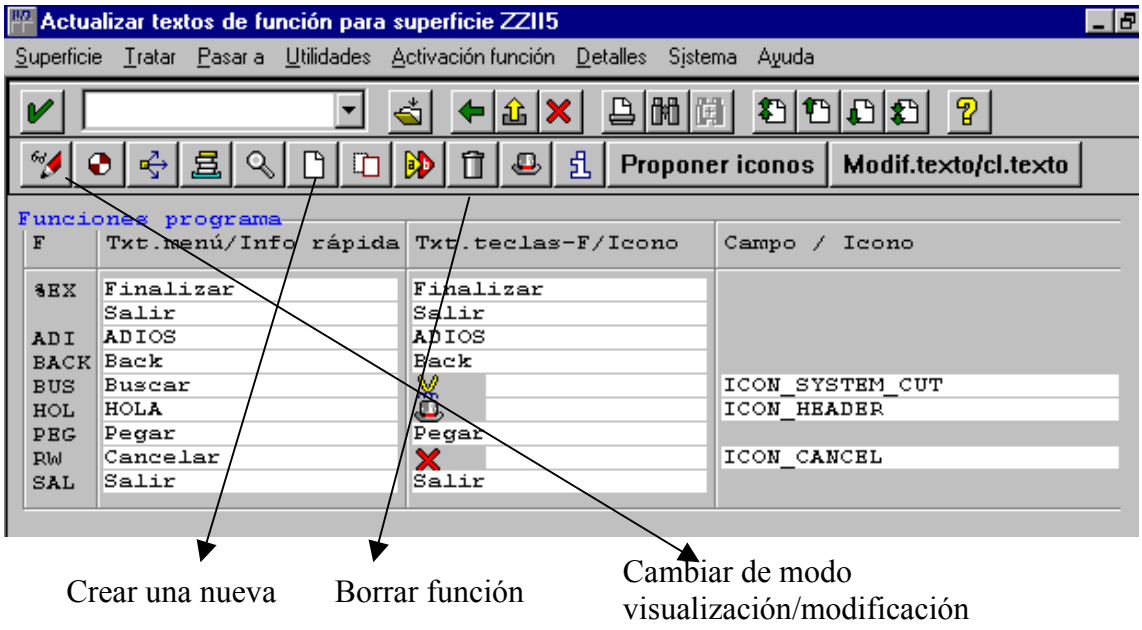
Después de haber creado todo esto grabamos el menú y lo siguiente es generarlo. Y si queremos probarlo sin tener que compilar y/o generar programas ni nada por el estilo, vamos al menú 'Superficie', 'Probar status' o F8 y nos saldrá la siguiente pantalla:



Luego pulsamos el botón: Ejecutar. Y nos sale la pantalla.

AVISOS

Ahora, cuando borramos una función con el SUPR se eliminan de la pantalla pero no de la SCREEN PAINTER. Para probarlo borramos la que hemos hecho y ponemos de nuevo sólo el código de la función y damos al enter. Veréis como aparece el nombre en el icono relacionado a ella. Para borrarlo tenemos que ir al menú 'Pasar a', 'lista funciones' o CTRL E. Y vamos a la siguiente pantalla:



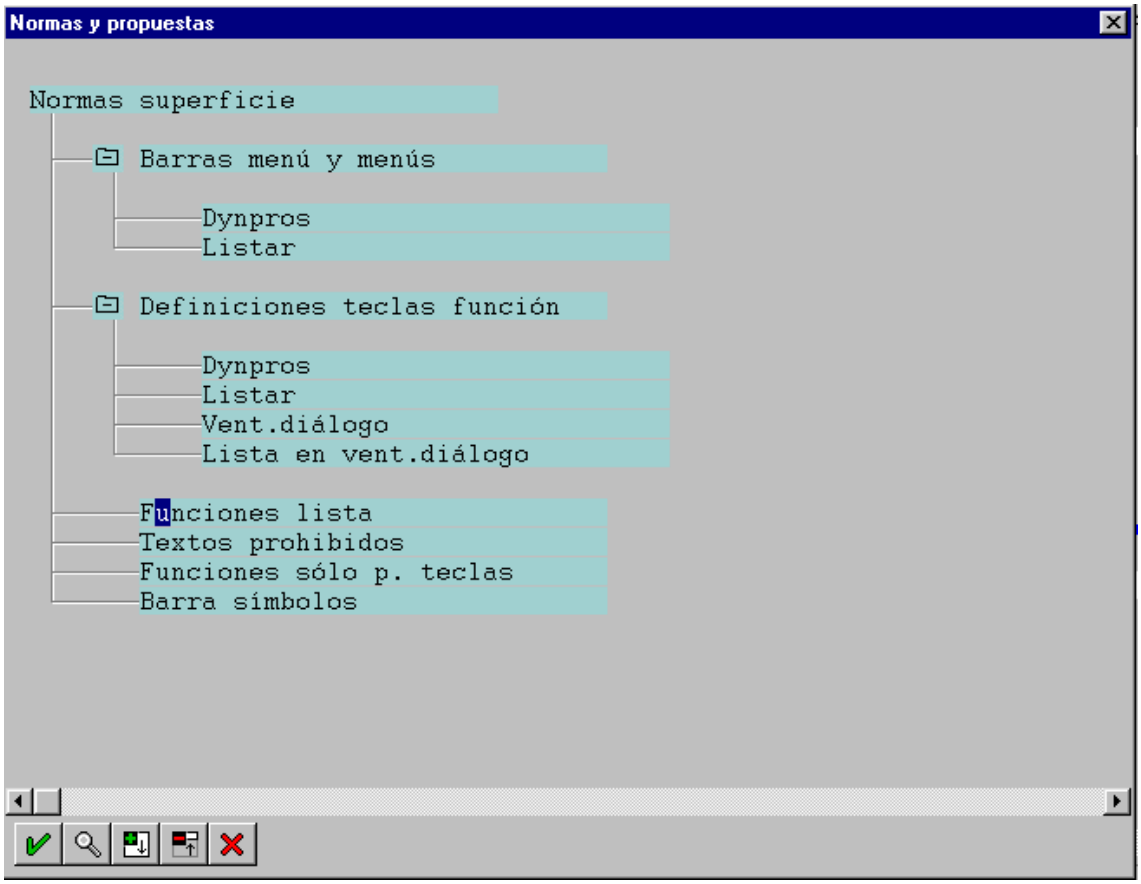
Seleccionando una función la puedo modificar y borrar

A través de esta pantalla podemos crear, modificar o borrar una función.

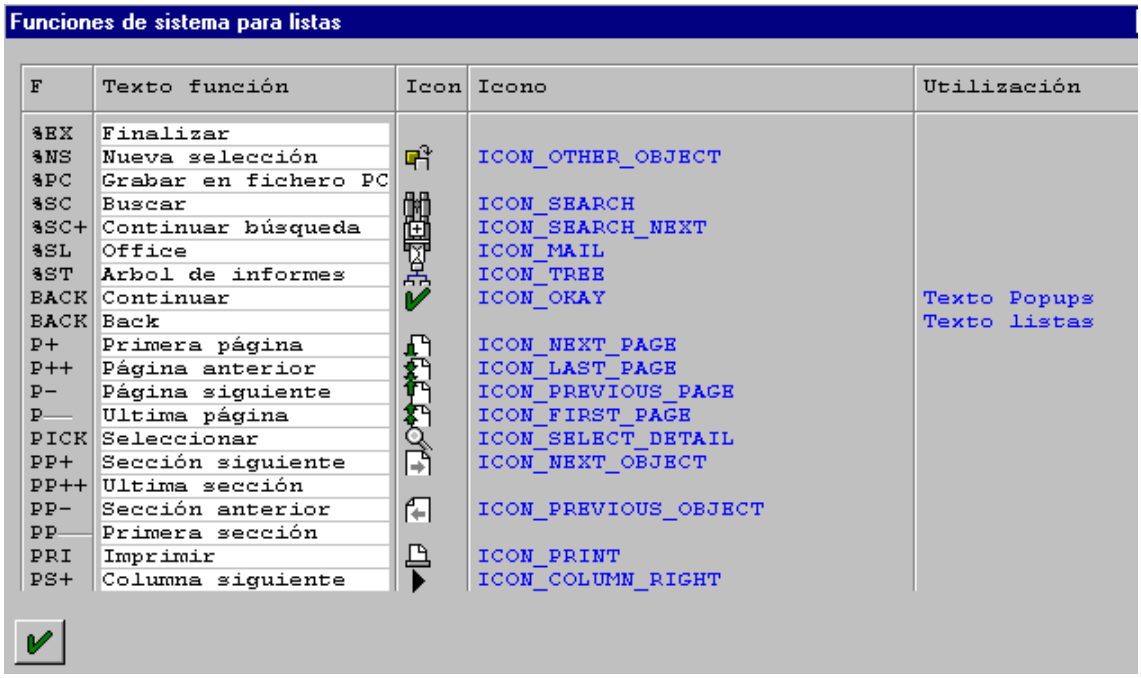
Por ejemplo para borrar una función nos posicionamos con el cursor en la función que deseemos borrar y pulsamos el icono donde sale la papelera y borraremos la función.

FUNCIONES DEL SISTEMA

Antes he utilizado funciones como RW (Cancelar), %EX (Salir), BACK (Volver). Estas tres son funciones que SAP ya las tiene hechas. Para poder ver las funciones que tienen (hay bastantes) desde la pantalla principal (Véase Fig. PF-STATUS), vamos al menú 'Utilidades', 'explicaciones', 'Normas/Propuestas'. La pantalla que sale es la siguiente:



Ahora donde pone 'Funciones lista' (la u está en azul) hacemos doble clic y nos saldrá la siguiente pantalla con las funciones del sistema :



Hay más funciones que aquí no se ven, pero con la barra de desplazamiento las podemos ver.

TITULO DE LA DYNPRO

Por defecto SAP pone un título a la ventana, para cambiar el título se utiliza la orden:

SET TITLEBAR ‘nombre-título’.

Cuando hallamos escrito esta orden, hacemos doble clic en nombre-título y nos dirá que ese título no esta creado y preguntará si lo queremos crear. Si decimos que sí nos saldrá esta ventana para introducir el título:

Cuando lo hallamos introducido lo grabaremos y volveremos al editor de programa. Ahora para que SAP “coja” ese título tenemos que grabar el programa, compilarlo y generarlo, si no lo hacemos SAP ignorará el título que hallamos creado.

La orden SET TITLEBAR..... va después de la orden SET PFSTATUS.

DYNPROS

Las Dynpros son las pantallas con las que SAP pide o visualiza datos. Como hemos visto, se construyen en la SCREEN PAINTER.

Tipos de Dynpro:

- Dynpro de include: La dynpro será definida como un subscreen y la lógica de proceso deberá contener tanto en el PBO como en la PAI la llamada **CALL SUBSCREEN** que procesará la parte PBO y PAI del subscreen. Un subscreen podrá ser llamado por distintas dynpros pero un subscreen a su vez no deberá llamar otros subscreens.
- Dynpro normal: La descrita en la SCREEN PAINTER.
- Ventana de diálogo modal: La llamada se realiza mediante:

CALL SCREEN STARTING AT <arriba a la izquierda>
ENDING AT <abajo a la derecha>.

Atributos de la Dynpro:

Retener datos

Una dynpro retiene los datos incluidos en ella. Al procesar la dynpro la siguiente vez, el sistema presentará automáticamente en los campos de entrada los datos retenidos. Si no nos interesa es suficiente refrescar la memoria.

Posición del cursor

Cuando no se desee que el cursor sea posicionado en el primer campo de entrada, sino el otro campo, es necesario indicar el nombre de este campo. Esta función puede ser ejecutada de forma dinámica en el Modul-Pool mediante la instrucción ABAP: SET CURSOR FIELD <nombre_campo>.

Funciones de tratamiento de campos

Desplazar.
Copiar.
Borrar.

Para mover o copiar los campos en la pantalla, deberán marcarse primero.

Como asociar varias screens a un mismo programa

El programa siempre será el mismo, pero cambiaremos el número de “dynpro” (cada “dynpro” tiene su propia “full screen”). Ejemplo:

```
MODULE USER_COMMAND_0001 INPUT. →  Dynpro nº 0001
      INSTRUCCIONES
ENDMODULE.            " USER_COMMAND_0001  INPUT
MODULE USER_COMMAND_0002 INPUT.→  Dynpro nº 0002
     INSTRUCCIONES
ENDMODULE.            " USER_COMMAND_0002  INPUT
```

Como pasar parámetros de una dynpro a otra.

Podemos pasar parámetros entre "dynpros" diferentes de un mismo programa o entre "dynpros" de diferentes programas.

DYNPROS DE UN MISMO PROGRAMA

Para poder pasar los valores, tenemos que declarar las variables al principio del programa, para que se puedan utilizar en todo el programa. Para que los valores de las variables se pasen sin perder su valor, lo haremos con las instrucciones explicadas anteriormente, o sea, con el "SET SCREEN" y "CALL SCREEN".
Por ejemplo si queremos llamar a la "dynpro" 100 de un mismo programa se haría de la siguiente forma:

```
SET SCREEN 100. LEAVE SCREEN.
ó
CALL SCREEN '100' AND RETURN.
```

"El AND RETURN" del "CALL SCREEN" lo pongo para que después me devuelva a la siguiente instrucción de la "dynpro".

ENTRE DIFERENTES PROGRAMAS

Cuando en una pantalla realizamos una búsqueda de un dato introducido por teclado y queremos utilizarlo en una pantalla de otro programa hay que guardar primero ese dato en memoria para después recuperarlo.

Para solucionar esto se utiliza las ordenes "IMPORT" y "EXPORT".

EXPORT

"EXPORT" se utiliza para guardar una variable, tabla interna o externa en la memoria. Su sintaxis es la siguiente:

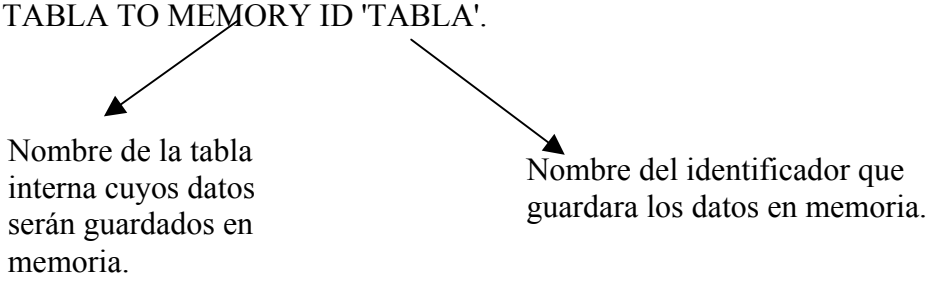
```
EXPORT variable TO MEMORY ID 'nombre'.
```

Variable -> es el nombre de una variable, una tabla de diccionario o tabla interna cuyo valor o valores se guardaran en memoria.

Nombre -> Es el nombre que le damos al identificador en la memoria. Puede ser un nombre de hasta treinta y dos caracteres.

Un ejemplo de ello sería:

EXPORT TABLA TO MEMORY ID 'TABLA'.



El nombre del identificador suele ser el mismo que el nombre de la variable o tabla a guardar.

IMPORT

Si con el EXPORT hemos guardado la variable en memoria, con el IMPORT recuperamos esa variable que está en memoria. Su sintaxis es parecida a la anterior:

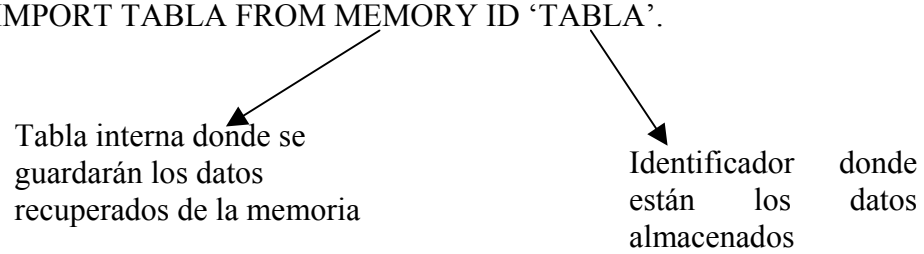
IMPORT variable FROM MEMORY ID ‘nombre’.

Variable -> es el nombre de la variable o tabla donde se guardarán los datos que están en memoria.

Nombre -> Es el nombre del identificador donde están guardados los datos en la memoria. Su longitud máxima ha de ser de 32 caracteres.

Un ejemplo del IMPORT sería este:

IMPORT TABLA FROM MEMORY ID ‘TABLA’.



COMO SE UTILIZAN

Explicaremos cómo se utilizan en un “Modulpool”. Hemos creado un programa en el que se introduce el nombre de una compañía aérea. Una vez introducido y cuando

pulsamos el botón ‘Buscar’ nos busca los vuelos de esa compañía (hay hecha una paginación que nos permite navegar por los vuelos de esa compañía) y si la encuentra nos sacará una información reducida sobre los vuelos.

Queremos hacer que cuando pulse el botón ‘Ver todos los datos’ nos lleve a una “Dynpro” en otro programa y nos saque toda la información de vuelo escogido.

Solo explicaremos cómo se guardan y recuperan los datos (el resto del problema se puede resolver utilizando las ordenes explicadas en capítulos anteriores).

La primera fase sería guardar los datos de donde estoy cuando pulse el botón ‘Ver todos los datos’. El código del programa sería este:

```
MODULE STATUS_0001 INPUT.

CASE SY-UCOMM
 WHEN 'VER'.

    EXPORT TABLA TO MEMORY ID 'TABLA'.

    EXPORT PA TO MEMORY ID 'PA'.

    CALL TRANSACTION 'ZZIV'.
MODULE STATUS_0001 INPUT.
```

Código de función que tiene el botón ‘Ver todos

```
CASE SY-UCOMM
 WHEN 'VER'.

    EXPORT TABLA TO MEMORY ID 'TABLA'.

    EXPORT PA TO MEMORY ID 'PA'.

    CALL TRANSACTION 'ZZIV'.
```

Guardo la tabla en memoria.

Guardo la variable, para indicarme qué registro estoy visualizando

Llamo a la transacción, que me mostrará toda la información de un determinado vuelo

```
ENDCASE.

ENDMODULE.
```

Como suponéis, el control de la tecla que se pulsa se realiza en el PAI (Process After Output). La variable PA me controla en qué registro estoy cuando hago la paginación.

La segunda fase sería, antes de visualizar la siguiente “Dynpro”, cargar los datos almacenados en la memoria. Se haría de la siguiente forma:

```
MODULE STATUS_0002 OUTPUT.
  IMPORT TABLA FROM MEMORY ID 'TABLA'.
  IMPORT PA FROM MEMORY ID 'PA'.
```

} Recupero de la memoria la tabla y la variable que he guardado en la anterior Dynpro

```
  PERFORM VISUALIZAR.
```

→ Visualizo el registro a través de la variable PA.

```
  SET PF-STATUS 'PANT-2'.
```

→ Visualizo el menú Painter de la Dynpro

```
  * SET TITLEBAR 'xxx'.

ENDMODULE.                    " STATUS_0002  OUTPUT
```

```
FORM VISUALIZAR.
  READ TABLE TABLA INDEX PA.
ENDFORM.
```

} Subrutina donde me posiciono y visualizo el vuelo escogido

Las variables o las tablas se recuperan de la memoria en el "PAI" (Process After Input), es decir, antes de que se visualice la pantalla.
Llamo a la subrutina externa para visualizar el registro donde están los datos completos del vuelo que hemos seleccionado en la pantalla anterior.

Como modificar los objetos de una dynpro

En SAP tenemos la posibilidad de cambiar los atributos de una "Dynpro" en tiempo de ejecución. Estos atributos se modifican en la "PBO" (Antes de que salga la pantalla).

Como ya hemos visto antes, la tabla donde se guardan los objetos de la pantalla se llama SCREEN. Para ver como funciona haremos un pequeño ejemplo:

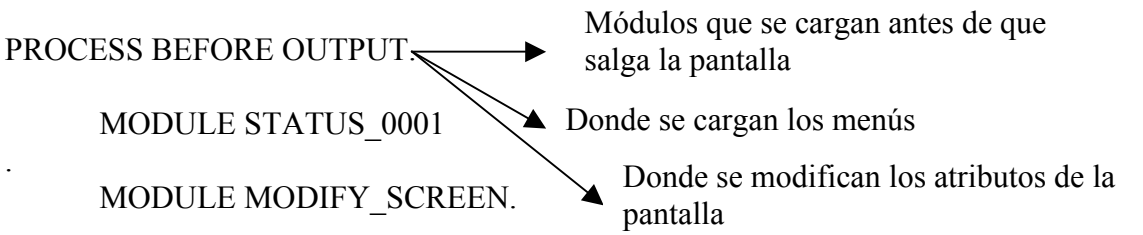
El ejemplo consistirá en que cuando demos a un botón (Su cod. func. será: DES) los campos que pertenezcan al grupo: 'MOD' se desactiven. El ejemplo se explicará en diversos pasos:

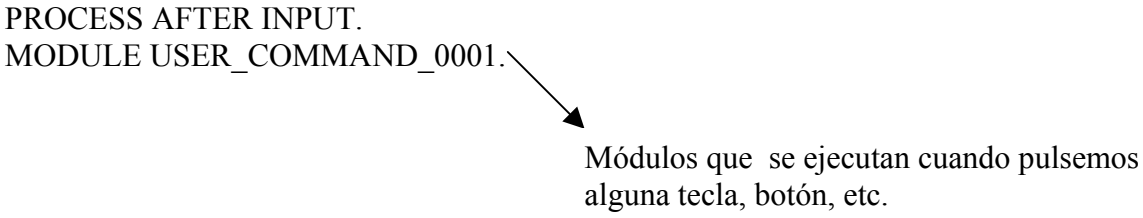
PASO 1 (Poner el módulo)

Para modificar los atributos de una pantalla hemos de crear un módulo en la "PBO". El módulo se pone en la lógica de proceso de la "SCREEN PAINTER".

Por ejemplo podemos crear uno que se llame: MODULE MODIFY_SCREEN.

Entonces la lógica de proceso de la SCREEN PAINTER quedaría de la siguiente forma:





Y dentro de este “MODULE” escribiremos lo siguiente (para acceder a ese módulo hacemos doble clic sobre “MODIFY_SCREEN” y iremos directamente al nuevo módulo):

```
CHECK MODO = 'TRATAR'.
LOOP AT SCREEN.
      CHECK SCREEN-GROUP1 = 'MOD'.
      SCREEN-INPUT = 0.
      MODIFY SCREEN.
ENDLOOP.
```

El primer “CHECK” se utiliza para comprobar el modo en que está el programa. Si el MODO contiene el valor TRATAR entonces irá al “LOOP”, en caso contrario saldrá del modulo.

Seguidamente se hace una lectura secuencial de la tabla “SCREEN” (la que contiene los datos de la pantalla).

El segundo “CHECK” comprueba si el objeto o campo pertenece al grupo 'MOD'. Si es así, no se pueden introducir datos en él.

Por último, está el “MODIFY SCREEN”. Esta orden sirve para confirmar la modificación de la pantalla. Esta orden siempre se pone cuando hagamos algún cambio en la pantalla.

PASO 2 (Llamarlo desde el programa)

Llamarlo, lo que se dice llamarlo, no lo llamamos, lo que hacemos es obligar a SAP a leer de nuevo la “SCREEN”.
Lo hacemos a través de la orden “MESSAGE”. Esta orden saca un mensaje por pantalla (suele ser de error). Cuando saca este mensaje, SAP relee de nuevo la “SCREEN”

(incluyendo el PBO). En este caso haremos que cuando se pulse un botón se visualice un mensaje. Lo codificaríamos de la siguiente forma:

```
CASE SY-UCOMM.
      WHEN 'DES'.
           MESSAGE E005 WITH 'PRUEBA DE MENSAJE'.
ENDCASE.
```

Esta parte la codificaríamos en la "PAI". Con esta sencilla instrucción hacemos que SAP lea de nuevo la pantalla.

Hacemos un ejemplo para probarlo.

INCOVENIENTES

En SAP no es oro todo lo que reluce, como diría el refranero popular. ¿Por qué digo esto?, pues porque eso de modificar los objetos está muy bien, pero (siempre hay peros) los objetos no se modifican cuando y donde nosotros queramos sino que hay que modificarlos cuando SAP nos lo deje hacer.

¿Donde nos los deja hacer? En la PBO, o sea, antes de que se visualice la pantalla.

En el ejemplo anterior solo modifica los objetos cuando cumple una condición (que ocurre cuando le damos a un botón), es decir, no siempre cuando carga esa pantalla nos modifica ese grupo de objetos.

También hay que resaltar que hay que utilizar unas instrucciones para que SAP vuelva a leer la pantalla (En nuestro caso utilizamos la orden MESSAGE).

Como asociar una tabla a una dynpro

SAP nos permite hacer una "dynpro" de una tabla.
Para hacerlo primeramente tenemos que haber hecho una vista de la tabla para poder asociarle un grupo de función. En la vista se pone un número de imagen sencilla, que es el número de dynpro de esa tabla.

En la tabla de prueba que hemos hecho en el capítulo de tablas, le hemos puesto de nombre ZZP1, en la vista hemos puesto el grupo de función ZZF1 y en el número de imagen hemos puesto el uno.

Para ver esa tabla en la "SCREEN PAINTER", tenemos que hacer lo siguiente:

1.- Primero vamos a la "SCREEN PAINTER"

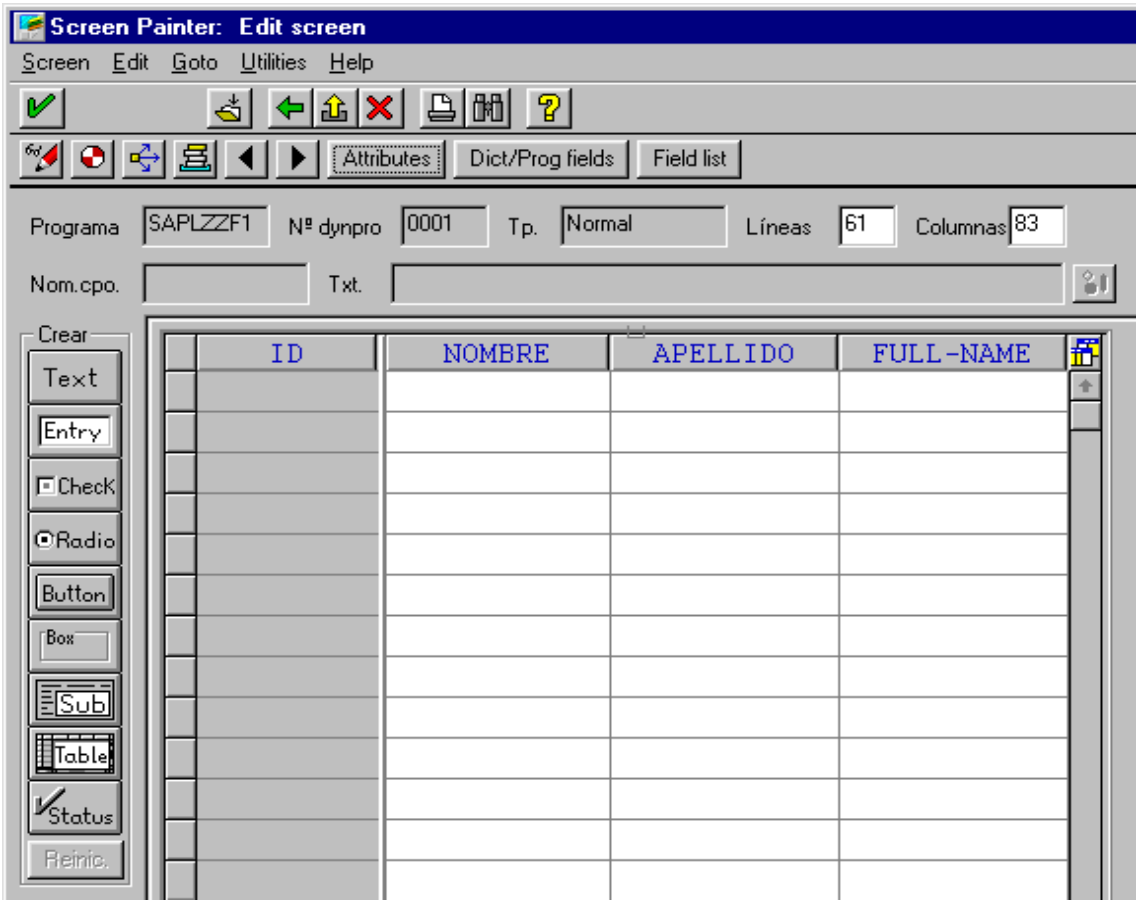
2.- En la "SCREEN PAINTER", donde pone programa, ponemos SAPLZZF1. Donde ZZF1 es el grupo de función y SAPL es la constante. La sintaxis sería la siguiente:

```
SAPL Código de función.
```

SAPL siempre ha de ponerse y junto el grupo de función.

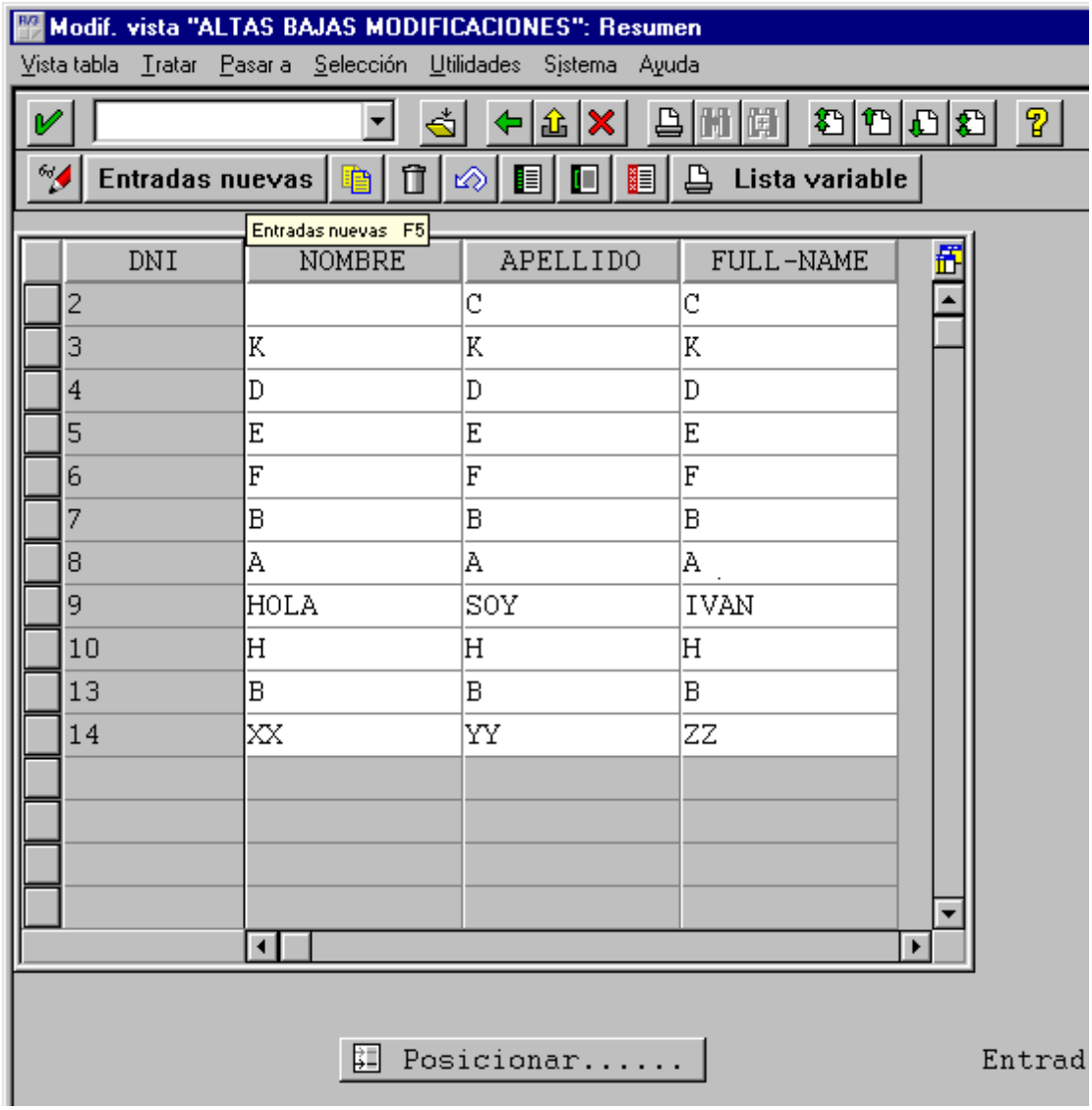
Y donde pone dynpro ponemos 0001 que es el número de la imagen sencilla.

3. - Pulsamos el botón “Modificar” y la pantalla que veremos es la siguiente:



Ahora, al objeto que pone ‘ID’ le cambiaremos el texto por ‘DNI’, y activaremos la “SCREEN PAINTER”.
Seguidamente lo que haremos es introducir los datos en la tabla. O sea, vais a la pantalla de ABAP/4 Development Workbench y al menú “Resumen”, “Data browser”. Cuando nos salga la pantalla del “Data browser” escribimos el nombre de la tabla, que en nuestro caso es ZZP1 y pulsaremos al botón que sale una hoja en blanco. Veremos que la pantalla que sale es esta (en la página siguiente):

Como veis donde antes ponía ID, ahora pone DNI.


Modif. vista "ALTAS BAJAS MODIFICACIONES": Resumen
Vista tabla Tratar Pasar a Selección Utilidades Sistema Ayuda
Entradas nuevas
Lista variable
Entradas nuevas F5
DNI NOMBRE APELLIDO FULL-NAME
2 C C
3 K K K
4 D D D
5 E E E
6 F F F
7 B B B
8 A A A
9 HOLA SOY IVAN
10 H H H
13 B B B
14 XX YY ZZ
Posicionar.....
Entrad

Control de proceso de dialogo

Screen	ABAP
	Program zzzzz.
PROCESS BEFORE OUTPUT MODULE OUTPUT1. (2) ENDMODULE	(1) MODULE OUTPUT1 OUTPUT -
PROCESS AFTER INPUT	
(1) MODULE INPUT1 INPUT. Perform xxxx.	-
	(2) ENDMODULE

(1) Control del procesador de dynpros después de ABAP/4

(2) Control del ABAP/4 después del procesador

EJEMPLO PRÁCTICO DE LA SCREEN PAINTER, MENU PAINTER, TRANSACCIONES Y MODUL-POOL.

El programa que haremos se trata de que el usuario introduzca un nombre de una compañía aérea y el programa busca datos sobre sus vuelos, sus conexiones aéreas, etc.

Para ello utilizaremos las siguientes tablas de diccionario: SPFLI, SFLIGHT.

Para mejorar el rendimiento pasaremos los datos de la tabla de diccionario a una tabla interna, la tabla interna tendrá la siguiente estructura:

```
DATA: BEGIN OF TABLA OCCURS 0,
        CARRID LIKE SFLIGHT-CARRID,
        FLDATE LIKE SFLIGHT-FLDATE,
        CONNID LIKE SFLIGHT-CONNID,
        FLTIME LIKE SPFLI-FLTIME,
        CITYFROM LIKE SPFLI-CITYFROM,
        DEPTIME LIKE SPFLI-DEPTIME,
        AIRPFROM LIKE  SPFLI-AIRPFROM,
        CITYTO LIKE SPFLI-CITYTO,
        NAME LIKE SAIRPORT-NAME,
        ARRTIME LIKE SPFLI-ARRTIME,
        PLANETYPE LIKE SAPLANE-PLANETYPE,
        SEATSMAX LIKE SAPLANE-SEATSMAX,
        CARRNAME LIKE SCARR-CARRNAME,
   END OF TABLA.
```

Esto es una declaración de un ejercicio completo pero se le pueden poner los campos que queramos.

A partir de ahora vienen los pasos a seguir:

PASO 1 (Crear Modul-Pool)

Primero hemos de crear un programa nuevo de tipo Modul-Pool.

Es igual que crear un programa normal, pero cuando nos pida el tipo de programa le pondremos de tipo M (Modul Pool).

Cuando esté en texto fuente declararemos todas las tablas, variables, etc. que utilizaremos en el programa (sólo declararemos eso).

En el programa quedaría esto:

```
PROGRAM ZZII5.                                  → Nombre del programa de tipo modul pool
TABLES: SPFLI, SFLIGHT.                         → Tablas de diccionario

DATA: BEGIN OF TABLA OCCURS 0,
        CARRID LIKE SFLIGHT-CARRID,
        FLDATE LIKE SFLIGHT-FLDATE,
        CONNID LIKE SFLIGHT-CONNID,
        FLTIME LIKE SPFLI-FLTIME,
        CITYFROM LIKE SPFLI-CITYFROM,
        DEPTIME LIKE SPFLI-DEPTIME,
        AIRPFROM LIKE  SPFLI-AIRPFROM,
        CITYTO LIKE SPFLI-CITYTO,
        NAME LIKE SAIRPORT-NAME,
        ARRTIME LIKE SPFLI-ARRTIME,
        PLANETYPE LIKE SAPLANE-PLANETYPE,
        SEATSMAX LIKE SAPLANE-SEATSMAX,
        CARRNAME LIKE SCARR-CARRNAME,
   END OF TABLA.                                → Campos de la tabla interna

DATA: NOM_CIA LIKE SPFLI-CARRID.                → Variable relacionada con un campo de la SCREEN PAINTER
```

Después de declarar todas las tablas ya sean de diccionario, internas, variables, etc. hemos de grabarlo.

 → Botón para grabar sin verificar.

Una vez creado hemos de verificar la consistencia del índice, lo haremos con el botón:

 → Verificar o Ctrl+F2

Y por último hemos de generar el programa. Se genera desde el editor de programas menú “programa”, “Generar” o CTRL+F3.

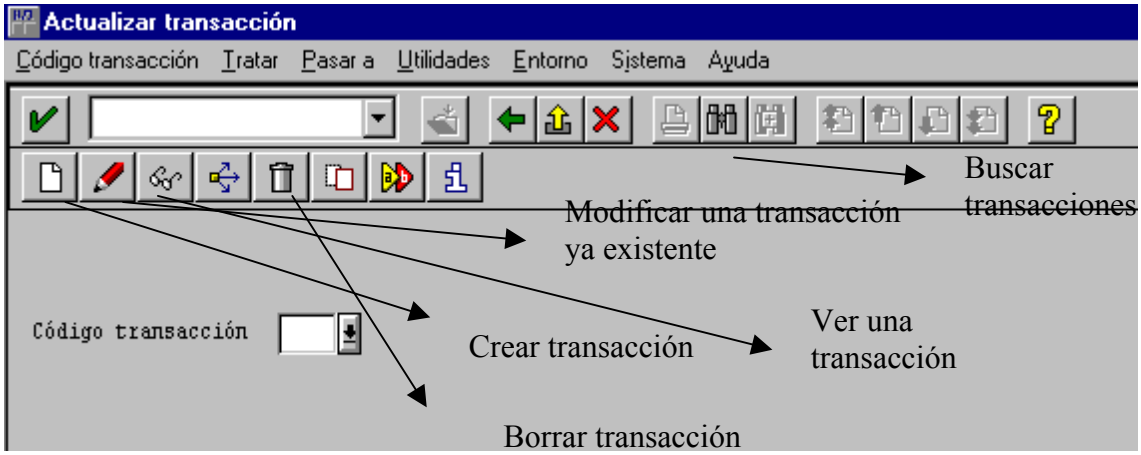
IMPORTANTE: si no lo generamos, los datos que declaremos no los podremos utilizar en el SCREEN PAINTER.

Cada vez que hagamos algún cambio en el programa lo hemos de **VERIFICAR** y **GENERAR**.

PASO 2 (Asociar al Modul Pool una transacción)

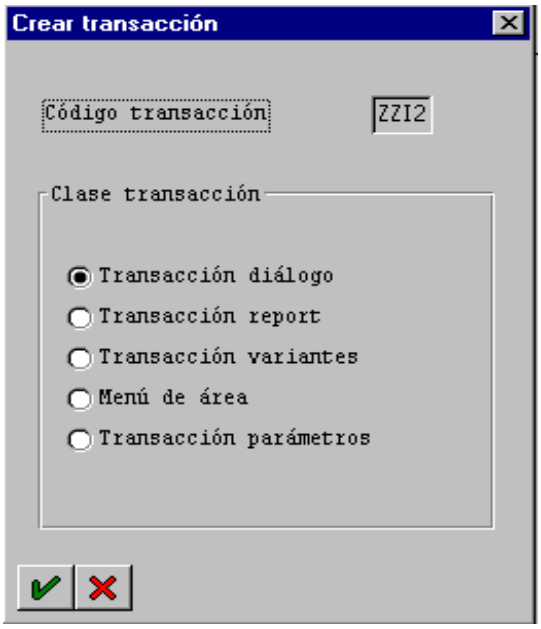
Para asociar una transacción a un Modul Pool, se hace lo siguiente:

Desde la pantalla de ABAP/4 Development Workbench en el menú “Desarrollo”, “Más herramientas”, “transacciones”, nos saldrá la siguiente pantalla:



En el “código transacción” pondremos uno nuevo o uno que ya exista.
Si no sabemos qué transacción queremos modificar podemos buscarla, pulsando F4 o los prismáticos que hay en el menú painter.

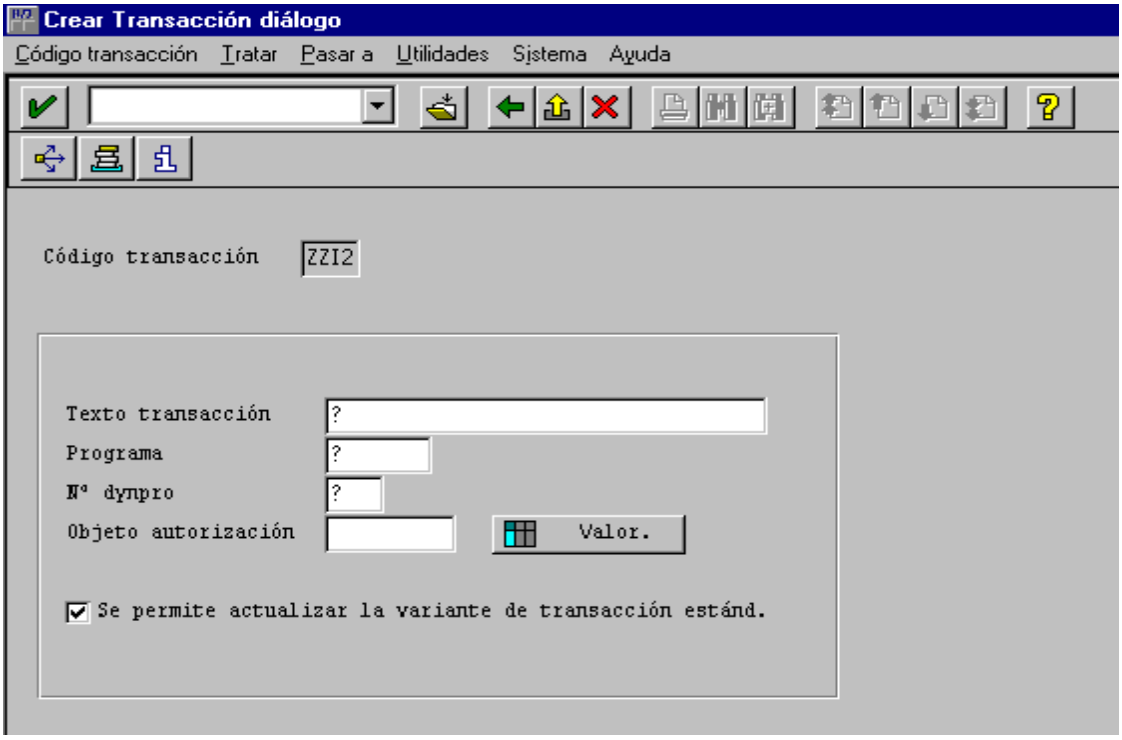
Cuando hayamos puesto una transacción y hemos pulsado crearla nos saldrá esta pantalla:



Esta pantalla solo aparecerá cuando creemos una transacción nueva.

Escogeremos la primera opción y pulsaremos enter.

Después nos aparecerá la siguiente pantalla. Esta pantalla aparecerá directamente cuando modifiquemos una transacción. La pantalla en cuestión es la siguiente:



"Texto transacción" es el mensaje explicativo de qué hace esa transacción.

"Programa" es el nombre del programa (en nuestro caso Modul Pool) al que relacionamos la transacción.

"Nº Dynpro" es el número de Dynpro que tiene ese programa. La Dynpro se la ponemos cuando creemos la SCREEN PAINTER. Por ello hay que acordarse de poner la misma Dynpro en los dos sitios, si no, no funcionará.

Después de introducir los datos, que son obligatorios, grabaremos la transacción.

PASO 3 (CREAR UNA SCREEN PAINTER)

Ahora crearemos un "entry" para introducir el nombre de la compañía al que llamaremos "NOM_CIA". Recordáis que este nombre ya está definido en el programa Modul Pool.
Se definen con el mismo nombre para poderlo relacionar tanto en la SCREEN PAINTER como el Modul Pool.

Esta forma de relacionar sirve para cualquier otro objeto.

Después escogeremos los campos que queramos visualizar, escogeremos los campos "CONNID" y "CITYFROM" de la tabla interna "TABLA".

Y sólo hace falta ponerlo en la Pantalla. Quedaría así:

Como vemos esos campos no tienen descripción (todo lo contrario ocurre cuando se utiliza una tabla de diccionario que ya posee una descripción). La descripción la pondremos nosotros con el objeto: "Text".

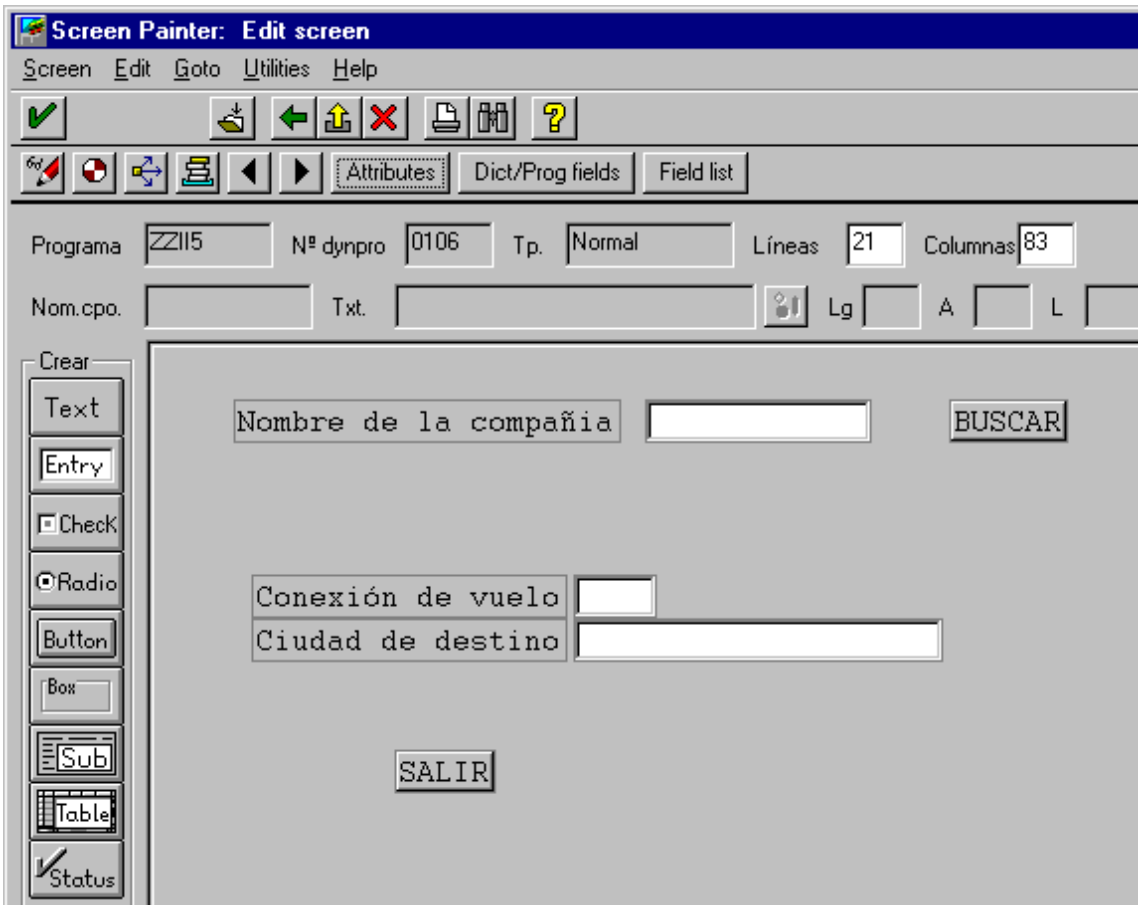
Ahora, por último, crearemos dos botones. Uno para buscar la compañía y otro para poder salir.

En el botón de buscar pondremos: "BUS" como código de función y en el de salida escribiremos: "SAL." También como código de función. En la salida pondrá el tipo de función E ("Exit") que está al lado de "Cod. Func.".

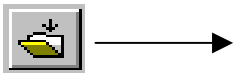
La ponemos de tipo E (Exit) ya que "SAP", cuando salimos de la "SCREEN PAINTER" (en tiempo de ejecución) se dedica a comprobar cosas internamente. Para que salga sin comprobar nada le ponemos el tipo de función E.

Un consejo: en aquellos objetos a los que no se les pueda poner "Cod. Func." y queramos relacionarlos con el programa, sólo hay que declarar en el programa una variable del mismo tipo del que hemos creado en la "FULL SCREEN", (es el caso de "NOM_CIA").

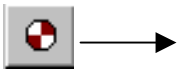
La pantalla final que nos quedaría sería la siguiente:



Por último, sólo hace falta grabarlo con el botón siguiente:

 Botón para grabar sin verificar o F11

Después para poderlo utilizar hay que generarlo, lo realizaremos con el siguiente botón:

 General CTRL+F3

Es importante que no nos olvidemos de hacer los últimos pasos cada vez que hagamos alguna modificación en la pantalla.

El siguiente paso es realizar la lógica de proceso asociada a la dynpro.

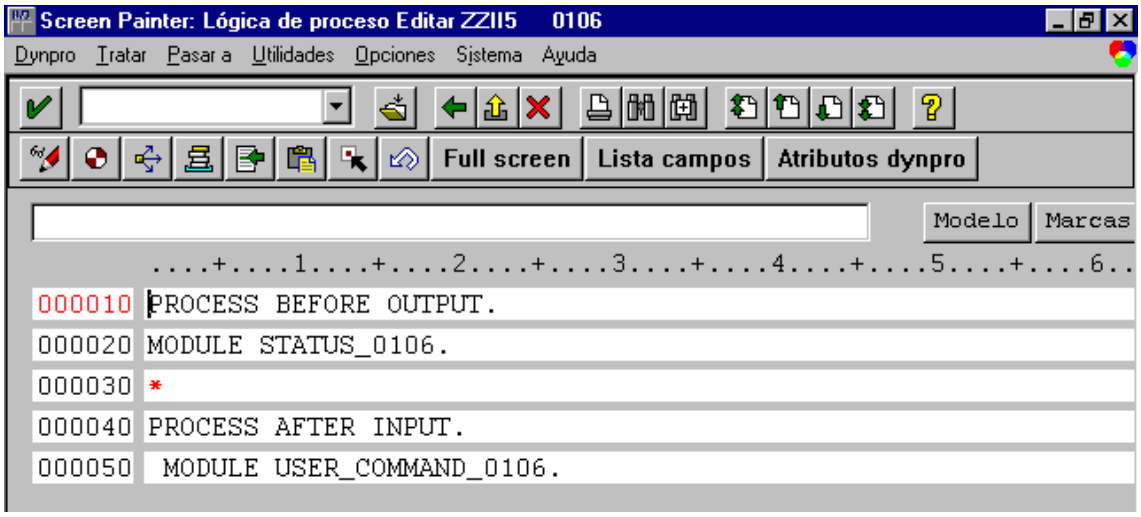
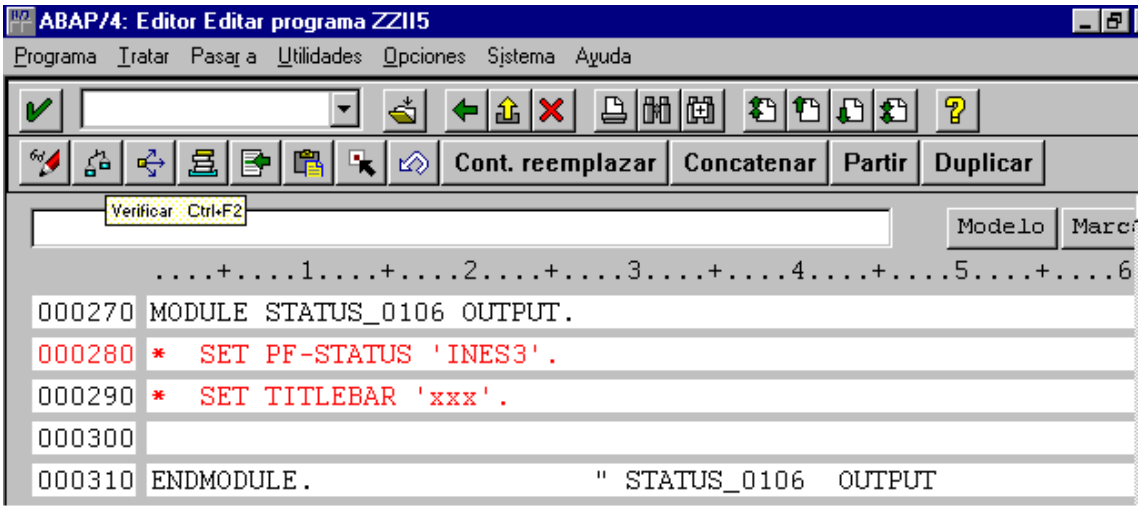


Fig. Lógica de proceso. Fig. Lógica de proceso.

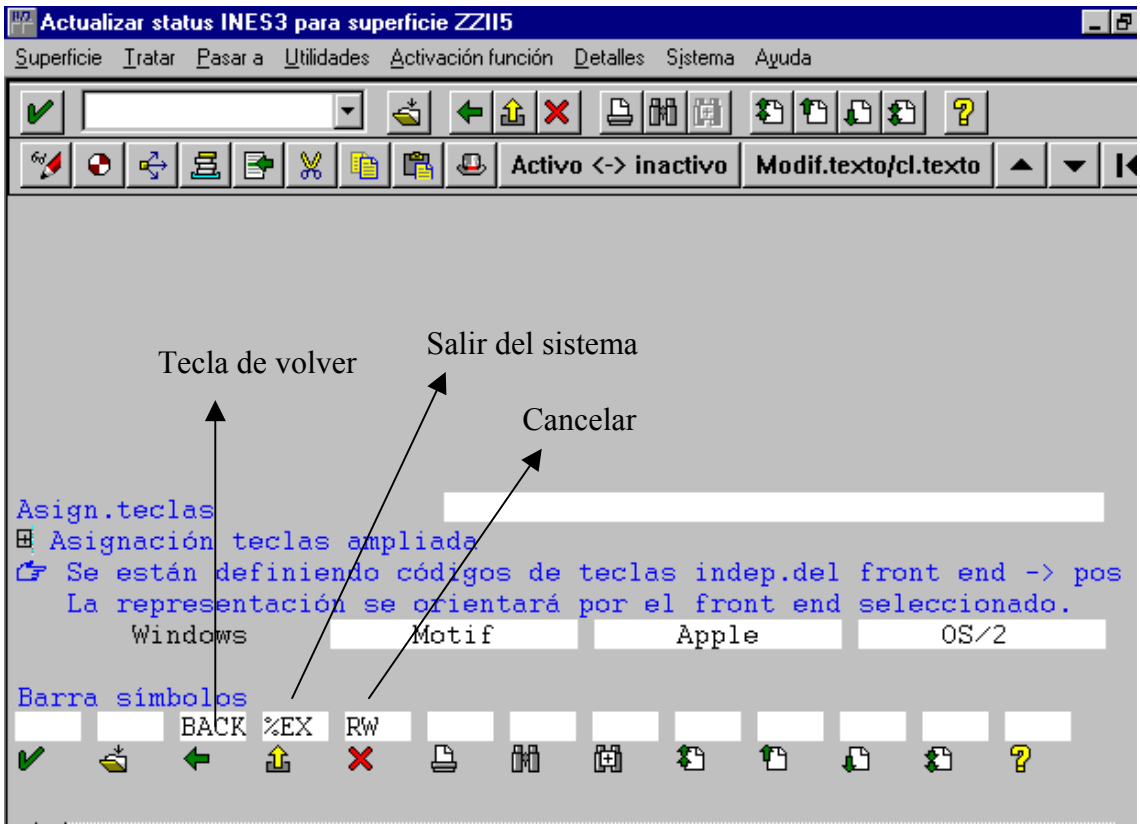
SAP ya pone una parte del código .

Realizamos primero el PBO.



Si le damos doble clic donde pone: “INES3” nos iríamos al “MENU PAINTER”.

Es aconsejable hacer un “PF-STATUS” sencillo para que pueda salir del programa (en nuestro caso no hace falta porque ya hemos puesto un botón que permite salir). Un “PF-STATUS” sería el siguiente:

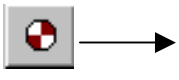


En la parte de arriba de la pantalla (no se ve) se ponen los menús y abajo (no se ve) los botones.

Cuando hayamos activado los tres iconos típicos que siempre aparecen en SAP, lo grabaremos

 Botón para grabar sin verificar o F11

Después, para poderlo utilizar hay que generarlo, que lo haremos con el siguiente botón:

 General CTRL+F3

Es importante que no nos olvidemos de hacer los pasos últimos cada vez que hagamos alguna modificación en el “MENU PAINTER”.

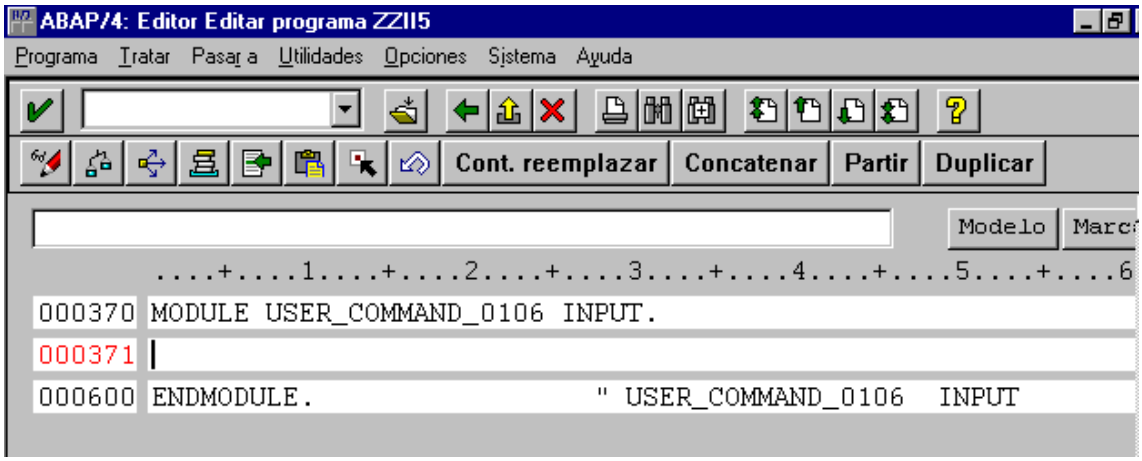
El “SET TITLEBAR xxx” es el nombre que contiene el título de la ventana. Haciendo doble clic en ‘xxx’ accedemos al título real de la ventana.

Si queremos poner los menús o los títulos tenemos que quitar el * que encontramos al principio (él * al principio es un comentario).

Después de esto lo grabaremos aunque no lo generaremos, porque aún faltan algunas cosillas.

Volviendo a la pantalla de los procesos (Fig. Lógica de proceso), realizaremos el PAI.

Por defecto tiene el "MODULE USER_COMMAND_016". Si hacemos doble clic sobre este "module" nos saldrá la siguiente pantalla:



Al principio miraremos si se ha pulsado algún botón, esto se averigua a través de la variable del sistema "SY-UCOMM". Esta variable contiene el nombre del código de función del botón pulsado (por ello la importancia de poner el "cod. Func." en los objetos que creemos en la "SCREEN PAINTER", siempre que se pueda poner). Esto se controla a través del "CASE".

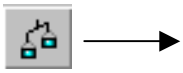
```
*&---------------------------------------------------------------------*
*&      Module  USER_COMMAND_0106  INPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE USER_COMMAND_0106 INPUT.
CASE SY-UCOMM.
    WHEN 'BUS'. ------> Corresponde al botón de buscar.
       SELECT SINGLE * FROM SPFLI
                  INTO CORRESPONDING FIELDS OF TABLA
                  WHERE CARRID = NOM_CIA.
    WHEN 'SAL'. ------> Es el botón de Salir.
       SET SCREEN 0. LEAVE SCREEN.
ENDCASE.
ENDMODULE.              " USER_COMMAND_0106  INPUT
```

Los botones que hay en "PF-STATUS", al asignarles funciones del sistema ("BACK", "%", "EX", " RW") no hace falta controlar la tecla de función pulsada.

Ahora grabamos el programa:

Botón para grabar sin verificar o F11

Después lo verificamos con este botón:



Verificar o CTRL+F2.

Seguidamente volvemos hacia atrás a través de F3 y volvemos a la pantalla de los procesos (Fig. Lógica de proceso). Por último, volvemos a grabar con el botón de siempre y lo generamos.

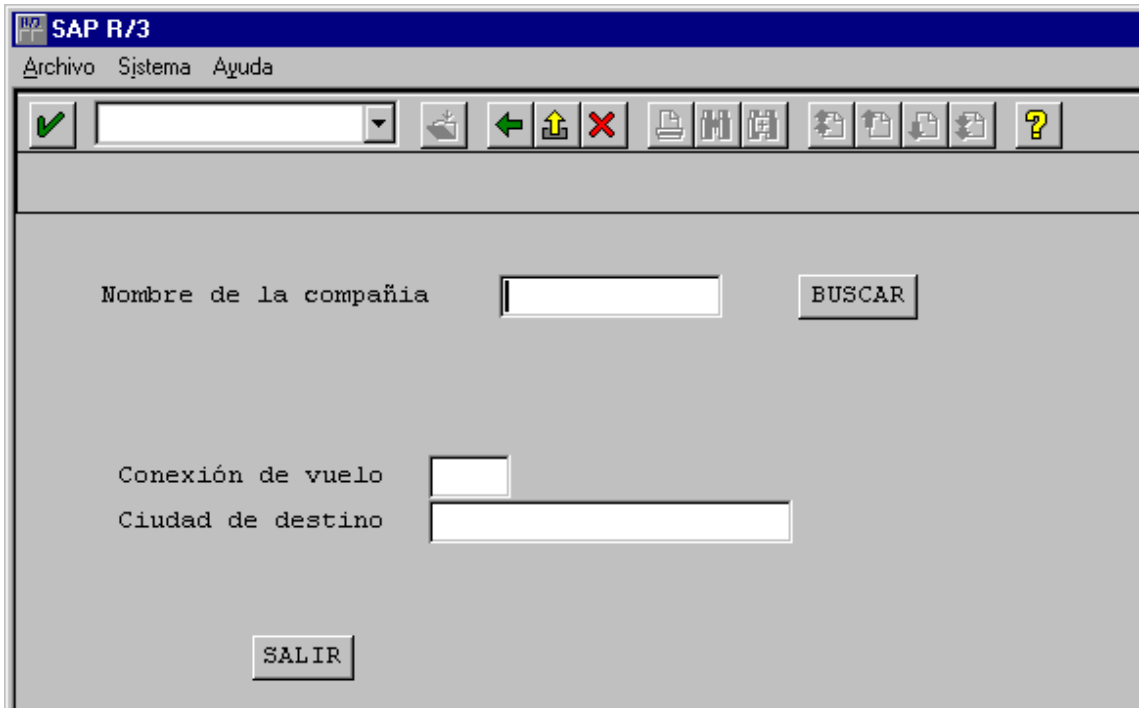
PASO 4 (SOLO FALTA EJECUTARLO)

Ahora, una vez generado, lo podemos ejecutar desde cualquier punto del SAP, en la línea de comandos.

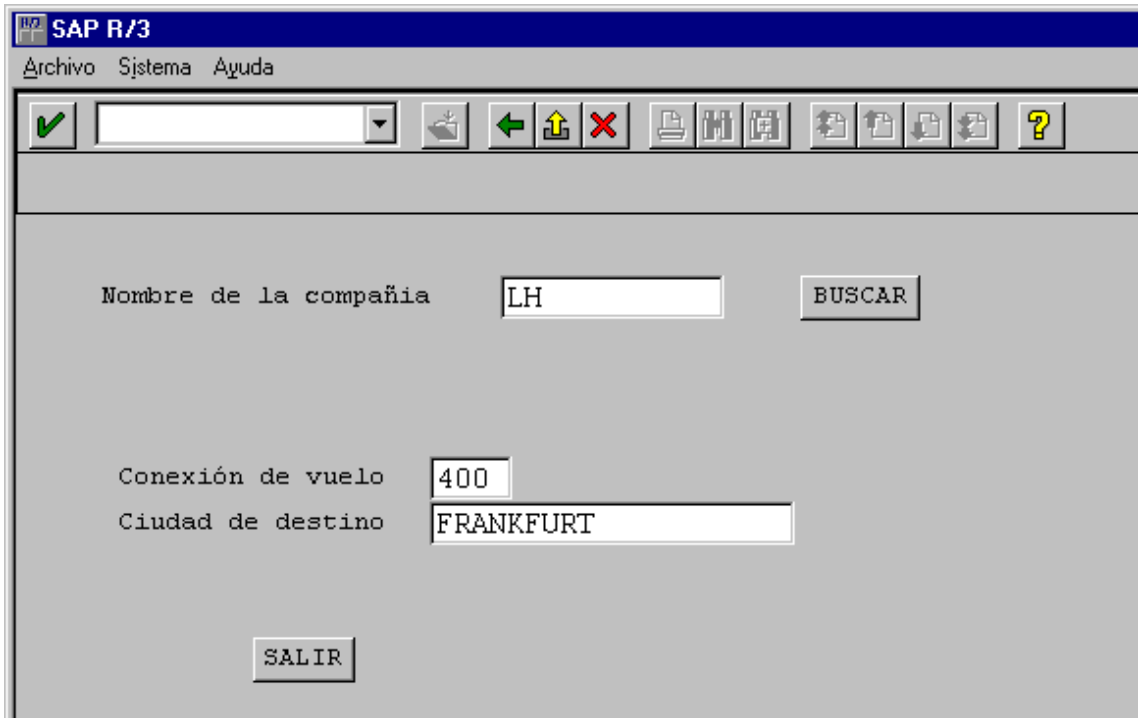
Línea de comandos

Introduciremos el código de transacción que hemos creado en el Paso 2 (¿os acordáis?)

Las transacciones se ejecutan poniendo: "/nxxxx", donde "xxxx" es la transacción. En nuestro caso hemos creado la transacción "ZZII" por lo tanto pondríamos "/NZZII" y pulsaríamos ENTER y nos saldría la siguiente pantalla:



Si en el nombre de la compañía ponemos “LH” y le damos al botón de buscar, el resultado será el siguiente:



Lo malo de este programa es que solo saca un registro (para ser más concreto el último). Para acceder a todos los registros solo hay que hacer la paginación.

EJEMPLO de UN MODUL-POOL CON DOS DYNPROS ASOCIADAS Y EN UNA DE ELLAS UN STEPLOOP

Lo que vamos a hacer es que el usuario introduzca una compañía aérea (en una dynpro) y muestre los vuelos de esa compañía (en otra dynpro). Los pasos a seguir son los siguientes:

Paso 1 (crear el modulpool)

Lo primero es crear el modul-pool al que llamaremos "ZZII" y donde escribiremos lo siguiente:

```
PROGRAM ZZII .

TABLES: SFLIGHT, SPFLI.          -> Tablas donde buscaremos los datos

DATA: BEGIN OF TABLA OCCURS 0,
        CONNID LIKE SFLIGHT-CONNID,
        CITYFROM LIKE SPFLI-CITYFROM,
        AIRPFROM LIKE SPFLI-AIRPFROM,
        CARRID LIKE SPFLI-CARRID,
      END OF TABLA.              -> Tabla interna donde se guardará los resultados de la búsqueda

DATA: DATOS LIKE TABLA.          -> Steploop con la estructura de "tabla"

DATA:  LINEA_TOPE TYPE I VALUE 1,
       NUM_LINEAS TYPE I.        -> Variables donde guardaré, el inicio y número de líneas del steploop
```

"Datos" -> Es el nombre del steploop, tendrá las mismas características que la tabla interna "tabla" ya que haremos un steploop de los 4 campos de la tabla interna.

"Linea_tope" -> Es el inicio donde escribiremos el steploop que inicialmente vale "1".

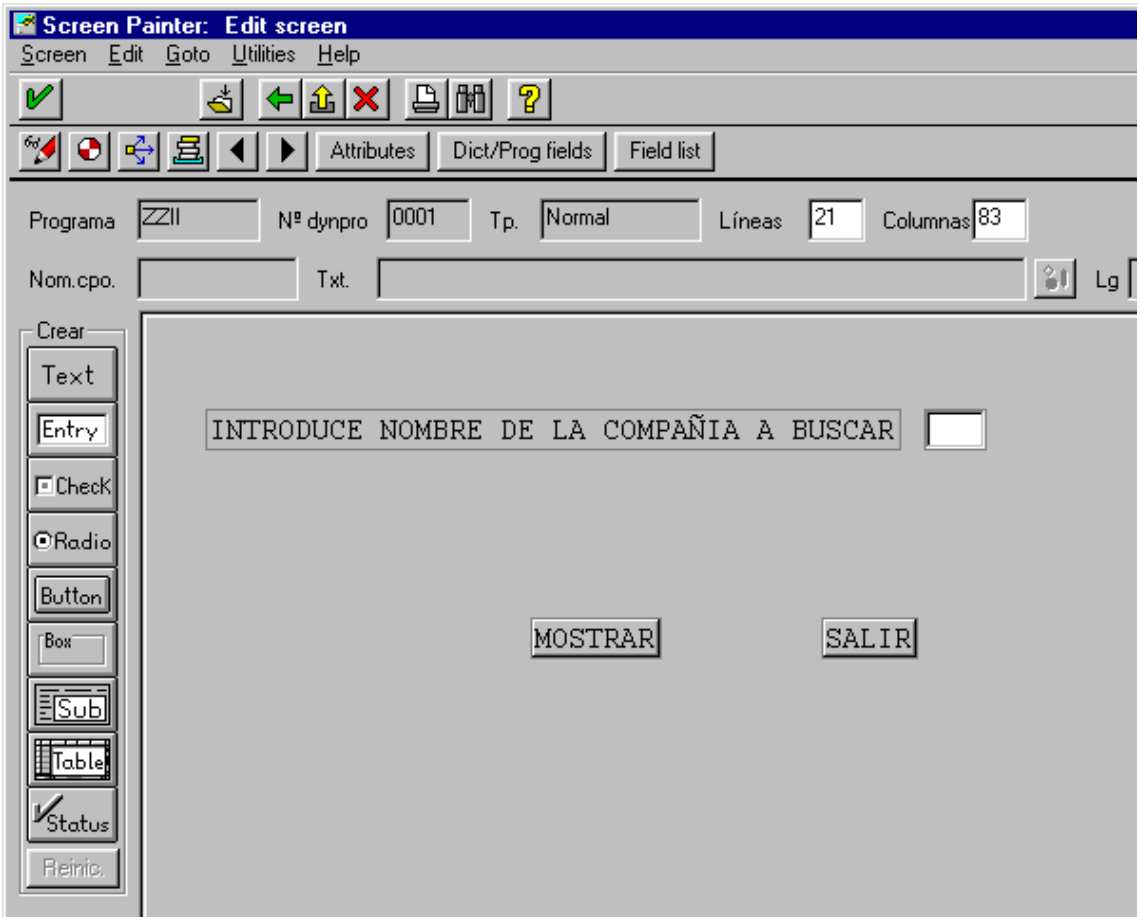
"Num_lineas" ->Número de líneas que tiene el steploop.

Por último lo grabamos, verificamos y generamos.

Paso 2 (Crear la primera dynpro)

El segundo paso será crear las pantallas con su respectivo código. En este paso haremos la pantalla donde se introducen los datos y en el siguiente paso haremos la pantalla de visualización.

Vamos a la "SCREEN PAINTER". La primera pantalla tendrá la dynpro "0001", que hecho quedaría:



El primer campo de entrada tendría el nombre "TABLA-CARRID", que lo haríamos a través del botón "DICT/PROG fields". Acordados que este botón permite insertar campos o tablas que tengamos declarados en el programa, *Recordar: Para que se puedan insertar los campos o tablas, el programa ha de estar generado.*

Después insertaremos dos botones, el botón "MOSTRAR" tiene el código de función "MOS" y el botón "SALIR" con el código de función "SAL". Después de hacer esto grabaremos el programa y lo generaremos.

La lógica de proceso de esta dynpro sería la siguiente:

```
000010 PROCESS BEFORE OUTPUT.
000020  MODULE STATUS_0001.
000030
000040 *
000050 PROCESS AFTER INPUT.
000060  FIELD TABLA-CARRID MODULE BUSCAR.
000070  MODULE USER_COMMAND_0001.
```

Como vemos los módulos son los de siempre, salvo que en la PAI pongo la orden "FIELD" (a grandes rasgos la orden "FIELD" sirve para que cuando se modifique el

campo TABLA-CARRID se ejecuten las órdenes que hay en el modulo "BUSCAR". El módulo "BUSCAR" sirve para buscar la compañía aérea introducida y poner todos sus vuelos en la tabla interna).

Si hacemos doble clic donde pone "BUSCAR" podemos crearlo. El modulo buscar tendría las siguientes instrucciones:

```
MODULE BUSCAR INPUT.
 SELECT * FROM SPFLI
    WHERE SPFLI-CARRID = TABLA-CARRID.
```

} Busco la compañía aérea en la tabla SPFLI por la compañía

```
IF SY-SUBRC = 0.
```

→ Si no hay errores, es que hay

```
   MOVE-CORRESPONDING SPFLI TO TABLA.
```

→ Muevo los datos encontrados a la tabla interna.

```
SELECT * FROM SFLIGHT
         WHERE CARRID = SPFLI-CARRID.
```

} Busco los datos que me faltan

```
     MOVE-CORRESPONDING SFLIGHT TO TABLA.
     APPEND TABLA.
   ENDSELECT.
  ENDIF.
 ENDSELECT.
ENDMODULE.                    " BUSCAR  INPUT
```

} Muevo los datos que me faltan y los añado a la tabla

Solo añadiré los datos a la tabla interna, siempre y cuando los encuentre en las dos tablas de diccionario y solo en el segundo "SELECT" cuando la compañía aérea este en la tabla SPFLI.

El módulo "USER_COMMAND_001" tendría las siguientes instrucciones:

```
MODULE USER_COMMAND_0001 INPUT.
 CASE SY-UCOMM.
```

→ Contiene el código de función del botón

```
  WHEN 'SAL'.
   SET SCREEN 0. LEAVE SCREEN.

  WHEN 'MOS'.
   SET SCREEN 2. LEAVE SCREEN.

 ENDCASE.
ENDMODULE.
```

} Si pulsamos el botón de salir, saldremos del programa

} Si pulsamos el botón de mostrar, me voy a la dynpro 0002 ó 2.

Paso 3 (Crear la segunda dynpro)

Esta dynpro servirá para visualizar los datos que tengamos en la tabla interna, que se ha rellenado en la anterior dynpro con la compañía aérea introducida.

El número de la dynpro será "0002". En este caso, primero haremos la lógica de proceso, ya que si hacemos primero la pantalla cuando la generemos nos dará error. Por ello primero haremos la lógica de proceso y después haremos la pantalla.

La lógica de proceso sería la siguiente:

```
000010 PROCESS BEFORE OUTPUT.
000020  MODULE STATUS_0002.
000030 LOOP AT TABLA CURSOR LINEA_TOPE.
000040      MODULE VISUALIZAR.
000050 ENDLOOP.
000060 *
000070 PROCESS AFTER INPUT.
000080 LOOP AT TABLA.
000090   MODULE SET_NUM_LINEAS.
000100 ENDLOOP.
000110  MODULE USER_COMMAND_0002.
```

Esta lógica de proceso ya tiene más instrucciones.
El primer "LOOP" sirve para rellenar el "steploop" con los datos de la tabla interna (rellenada en la dynpro anterior). La opción "CURSOR LINEA_TOPE" sirve para que empiece a rellenar el "steploop" a partir de la primera posición, ya que inicialmente, "LINEA_TOPE" vale 1.

Dentro del "LOOP" llamamos al módulo "VISUALIZAR" que lo que hace es poner los datos de la tabla interna al "steploop". El módulo "VISUALIZAR" tendría las siguientes instrucciones:

```
MODULE VISUALIZAR OUTPUT.
  MOVE-CORRESPONDING TABLA TO DATOS.
ENDMODULE.                    " VISUALIZAR  OUTPUT
```

Como se ve, se copian los datos que hay en la tabla interna "TABLA" en el "steploop" que se llama "DATOS".
Utilizo el MOVE-CORRESPONDING porque tanto "TABLA" como "DATOS" tienen la misma estructura. Si no la tuviesen se haría con la instrucción "=" o con el "MOVE".

Os preguntaréis por qué hacemos un módulo para una sola instrucción. Si pongo la instrucción MOVE-CORRESPONDING en la lógica de proceso, al compilar me dará el error "la instrucción MOVE-CORRESPONDING no esta definida o declarada". Este

error da porque solo puede haber una serie de instrucciones en la lógica de proceso como el “LOOP”, “CHAIN”, “FIELD” y alguna más, por ello hay que hacer el módulo.

El segundo “LOOP”, si no lo ponemos, nos da el siguiente error “El campo DATOS-CARRID no esta asignado al ‘LOOP...ENDLOOP’ debe aparecer en PBO y en el PAI”. Por este error ponemos un “LOOP” tanto en el PBO y en el PAI.

En el segundo “LOOP” llamamos al módulo “SET_NUM_LINEAS”, que guarda el número de líneas que tiene el “steploop”. Tendría el siguiente código:

```
MODULE SET_NUM_LINEAS INPUT.
  NUM_LINEAS = SY-LOOPC.
ENDMODULE.                    " SET_NUM_LINEAS  INPUT
```

“SY-LOOPC” es una variable de sistema que dice cuantas líneas tiene el “steploop” y se guardan en la variable “NUM_LINEAS”.

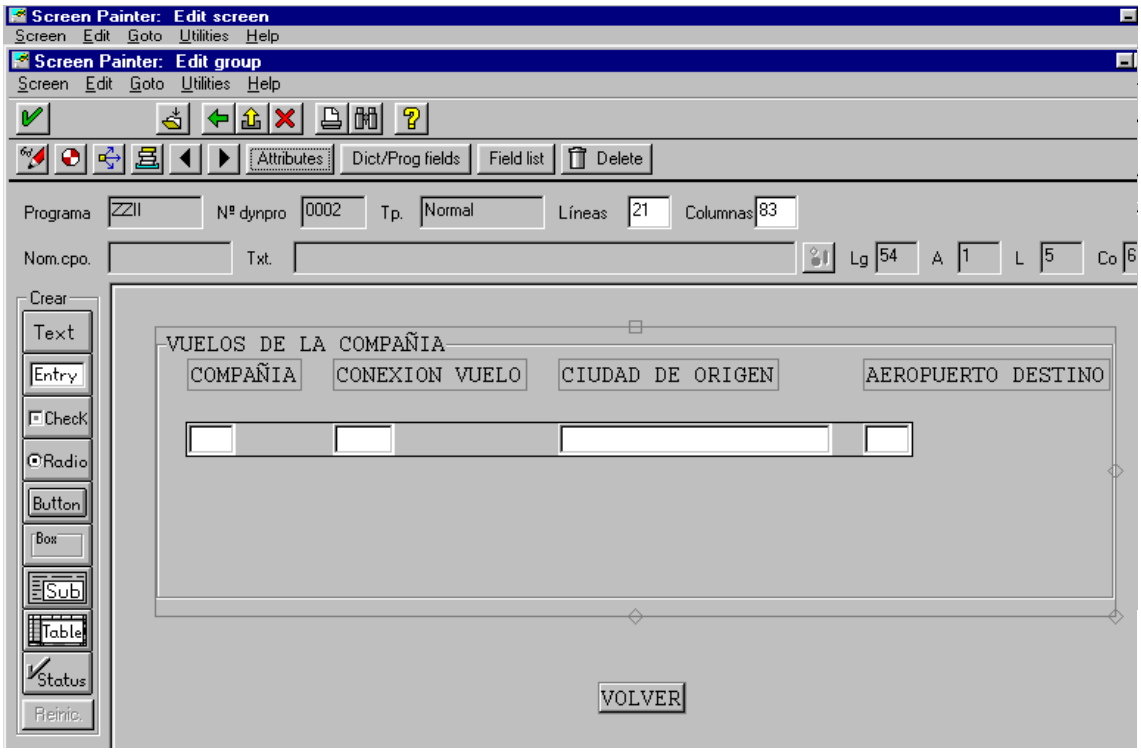
Y el módulo “USER_COMMAND_002” tendría las siguientes instrucciones:

```
MODULE USER_COMMAND_0002 INPUT.
  REFRESH TABLA. CLEAR TABLA.
  SET SCREEN 1. LEAVE SCREEN.
ENDMODULE.                    " USER_COMMAND_0002  INPUT
```

En este módulo lo que haría es limpiar el header y el contenido de la tabla interna “TABLA” y a continuación me iría a la dynpro “0001” ó “1”. Este modulo se cumpliría al pulsar cualquier tecla o botón.

Después de poner todo el código solo nos faltaría hacer la pantalla. La pantalla la haremos paso a paso.

El primer paso sería hacer esta pantalla:



Primero haremos una “box” con el texto “VUELOS DE LA COMPAÑÍA”. Dentro de la “BOX” pondremos 4 “TEXT”, cuyo texto será el nombre de los campos del “steploop” “DATOS”.

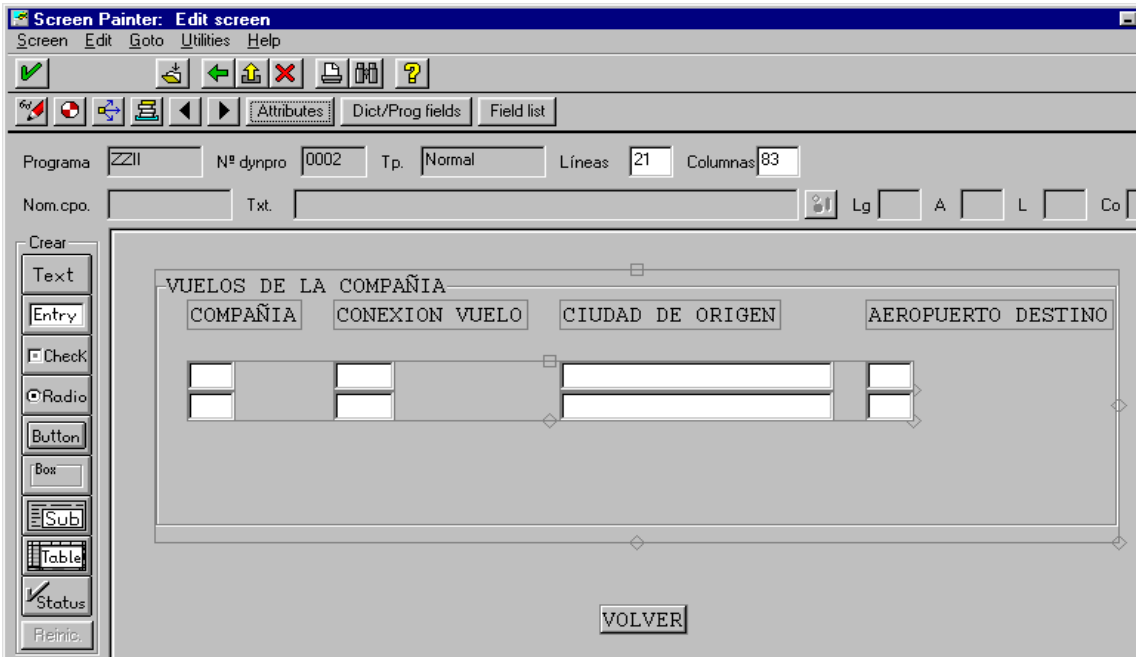
Fuera de la box pondremos un botón para volver a la dynpro anterior (como hemos visto en la lógica de proceso este botón solo esta para hacer bonito, ya que en realidad no hace absolutamente nada)

Ahora falta poner los campos del “steploop” “DATOS” en la pantalla. Como sabéis lo haríamos con el botón “Dict/Prog Fields” y nos triaríamos a la estructura “DATOS” y la insertaríamos en la pantalla. Recordar poner los campo debajo de cada etiqueta.

La pantalla que saldría sería la siguiente (en la página siguiente):

Como se ve hemos puesto 4 campos del “steploop” “DATOS”. A continuación hemos de seleccionar los 4 campos para poder hacer el “steploop”.
Después vamos al menú “edit”, “steploop”, “define” y nos creará el “steploop”.
NOTA: Tenemos que englobar los campos que queremos hacer el “steploop”, ya que en la lógica de proceso hay que hacer 2 “loops” (uno en el PAI y el otro en el PBO) por cada “steploop” que halla en la pantalla. Por eso es bueno, agrupar los campos y hacer un “steploop” de esos campos englobados.

La pantalla que saldría cuando definiríamos el “steploop” sería la siguiente:



Veréis en la imagen que el "steploop" agrupa a los 4 campos. Aquí también podemos estirar el steploop (lo hemos visto como se hacia, cuando he explicado como se hacia un sencillo "steploop"), se estira para poder insertar más campos al "steploop".

Un steploop puede ser fijo ("fix") o variable ("var"). Para saber de que tipo es miramos los atributos del "steploop" y en la ventana de propiedades (Véase Fig. Ventana de propiedades) hay un recuadro que pondrá el tipo de "steploop".
La diferencia entre uno y otro es que en el fijo si tenemos 2 campos (como en el que hemos hecho) y queremos visualizar 5 campos, solo nos mostrará los dos primeros campos mientras que en el variable el número de campos se adapta al número de campos a visualizar, es decir, si hemos creado un "steploop" de 5 campos y solo visualizamos uno, solo nos visualizará ese campo, los 4 restantes los oculta.

Para cambiar entre un tipo u otro, vamos al menú "edit", "steploop", "fix" o "variable". Este menú se encuentra en la "full screen" de la "screen painter" y automáticamente nos cambiará entre un tipo u otro.

Lo malo del tipo "variable" es que si hay muchos datos a visualizar el "steploop" nos puede ocupar toda la pantalla y lo malo del fijo es que puede ocultarnos datos por que hay más campos a visualizar que campos tiene el "steploop".

Una curiosidad es que si el "steploop" es variable y visualizamos muchos datos y estos datos sobrepasan el limite inferior de la "box", que también hemos creado, también se estira como el "steploop", pero lo malo es que no se vuelve a encoger como también lo hace el "steploop".

En resumen que elegir entre un tipo u otro depende de la cantidad de datos a visualizar.

Paso 4 (Solo falta ejecutarlo)

Para ejecutarlo tenemos que hacer una transacción (ya explicada anteriormente). Cuando lo hagamos la ejecutaremos y para probar podemos introducir la compañía "LH" o "SQ" que seguro que tiene vuelos.

Por último

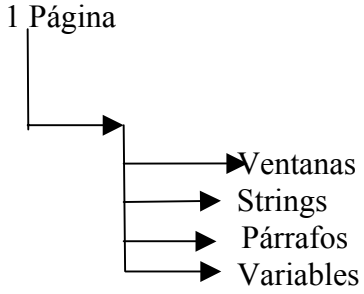
A continuación escribiré el código completo del programa "ZZII", para que podáis ver como quedaría:

```
PROGRAM ZZII .
TABLES: SFLIGHT, SPFLI.
DATA: BEGIN OF TABLA OCCURS 0,
          CONNID LIKE SFLIGHT-CONNID,
          CITYFROM LIKE SPFLI-CITYFROM,
          AIRPFROM LIKE SPFLI-AIRPFROM,
          CARRID LIKE SPFLI-CARRID,
      END OF TABLA.

DATA: DATOS LIKE TABLA.

DATA:   LINEA_TOPE TYPE I VALUE 1,
        NUM_LINEAS TYPE I.

*&---------------------------------------------------------------------*
*&      Module  USER_COMMAND_0001  INPUT
*&---------------------------------------------------------------------*
*   MIRO QUE BOTON SE HA PULSADO
*----------------------------------------------------------------------*
MODULE USER_COMMAND_0001 INPUT.
  CASE SY-UCOMM.
    WHEN 'SAL'.
      SET SCREEN 0. LEAVE SCREEN.
    WHEN 'MOS'.
      SET SCREEN 2. LEAVE SCREEN.
  ENDCASE.
ENDMODULE.
*&---------------------------------------------------------------------*
*&      Module  STATUS_0001  OUTPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE STATUS_0001 OUTPUT.
*  SET PF-STATUS 'xxxxxxxx'.
*  SET TITLEBAR 'xxx'.

ENDMODULE.
*&---------------------------------------------------------------------*
*&      Module  USER_COMMAND_0002  INPUT
```

```
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE USER_COMMAND_0002 INPUT.
  REFRESH TABLA. CLEAR TABLA.
  SET SCREEN 1. LEAVE SCREEN.
ENDMODULE.                    " USER_COMMAND_0002  INPUT
*&---------------------------------------------------------------------*
*&      Module  STATUS_0002  OUTPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE STATUS_0002 OUTPUT.
*  SET PF-STATUS 'xxxxxxxx'.
*  SET TITLEBAR 'xxx'.

ENDMODULE.                    " STATUS_0002  OUTPUT
*----------------------------------------------------------------------*
*       MODULE VISUALIZAR OUTPUT                                       *
*----------------------------------------------------------------------*
* MUEVO LOS DATOS DE LA TABLA INTERNA AL STEPLOOP                       *
*----------------------------------------------------------------------*
MODULE VISUALIZAR OUTPUT.
  MOVE-CORRESPONDING TABLA TO DATOS.
ENDMODULE.                    " VISUALIZAR  OUTPUT

*----------------------------------------------------------------------*
*       MODULE SET_NUM_LINEAS INPUT                                    *
*----------------------------------------------------------------------*
*  CUENTO EL NUMERO DE LINEAS QUE TIENE EL STEEPLOOP  *
*----------------------------------------------------------------------*
MODULE SET_NUM_LINEAS INPUT.
  NUM_LINEAS = SY-LOOPC.
ENDMODULE.                    " SET_NUM_LINEAS  INPUT
*&---------------------------------------------------------------------*
*&      Module  BUSCAR  INPUT
*&---------------------------------------------------------------------*
* BUSCO LA COMPAÑIA INTRODUCIDA Y LA GUARDO A UNA TABLA
INTERNA
*----------------------------------------------------------------------*
MODULE BUSCAR INPUT.
  SELECT * FROM SPFLI
      WHERE SPFLI-CARRID = TABLA-CARRID.
    IF SY-SUBRC = 0.
      MOVE-CORRESPONDING SPFLI TO TABLA.
      SELECT * FROM SFLIGHT
            WHERE CARRID = SPFLI-CARRID.
        MOVE-CORRESPONDING SFLIGHT TO TABLA.
        APPEND TABLA.
      ENDSELECT.
```

```
   ENDIF.
  ENDSELECT.
ENDMODULE.                    " BUSCAR  INPUT
```

FORMULARIOS



0. Cabecera:

   Donde se pone la página inicial, párrafo inicial, etc.

1. Definir páginas:

   Donde se pone, nombre de la página siguiente, modo "START" para la 1ª página, /NC, incrementar las siguientes, y el tipo de numeración que el normal es "ARABIC".

2. Definir ventanas

   Donde ponemos el nombre y la descripción, una será "MAIN", principal, y las otras pueden ser variables (si son diferentes en cada página), constantes (si siempre son iguales). En "ELEMENTOS TEXTO" se ha de codificar lo que tendrá la ventana (que serán unos determinados párrafos, identificados con el nombre en las 2 primeras posiciones.

3. Definir Párrafos:

   Nombre de las posiciones, alineación, left o right de todas las líneas y si se puede decir un margen o un interlineado, con "FONT" se puede cambiar la letra y con "TABULADORES**"** se pueden definir las posiciones de los diferentes campos.

4. Definir Strings:

   Son atributos que se asignan a los campos.

   Línea unida→ es que en el salto de línea se pueda o no partir el campo
   Oculto ⟶ que el campo sea invisible
   índice ⟶ escribiría el campo con formato índice
   subíndice⟶escribiría el campo con formato subíndice
   FONT ⟶ para cambiar tipo de letra

5. Ventana página

   A una página se le asignará todas las ventanas que tendrá, con la posición de cada una.

   Izquierda: posición a izquierda
   Arriba: nº de líneas que tendrá por encima

Ancho: nº posiciones que ocupará
Altura: nº de líneas que ocupará
Los "elementos de texto" son por cada ventana, haciendo servir los párrafos, strings que interese.

TEXTOS STANDARDS: se usan en el programa.
HERRAMIENTAS
TRATAMIENTOS TEXTOS
TEXTO STANDARD
En las includes dicen:

Nombre del texto
Nombre del objeto
identificador texto
identificador idioma
identificador párrafo
} Tablas STXH-STXL

Con el código de "ELEMENTOS DE TEXTO" se pueden tratar

Símbolos, valores que se pueden incluir en el print.

Incluir / símbolos / texto – programa – sistema día, hora standard – nuevo

Comandos de control que se pueden utilizar:

IF .. elseif ..else..endif
Case .. WHEN .. OTHERS.. endcase
Include
Define
New-Page
Protect .. endprotect
Reset
Address .. enaddress.

Por UTILIDADES / CONV. FDLOMA ORIGEN se puede cambiar el idioma del formulario, porque si no coincide el que ponemos en la 1ª pantalla con el de origen no nos dejará crear nuevos párrafos, ni ventanas, ni nada.

MENSAJES EN SAP

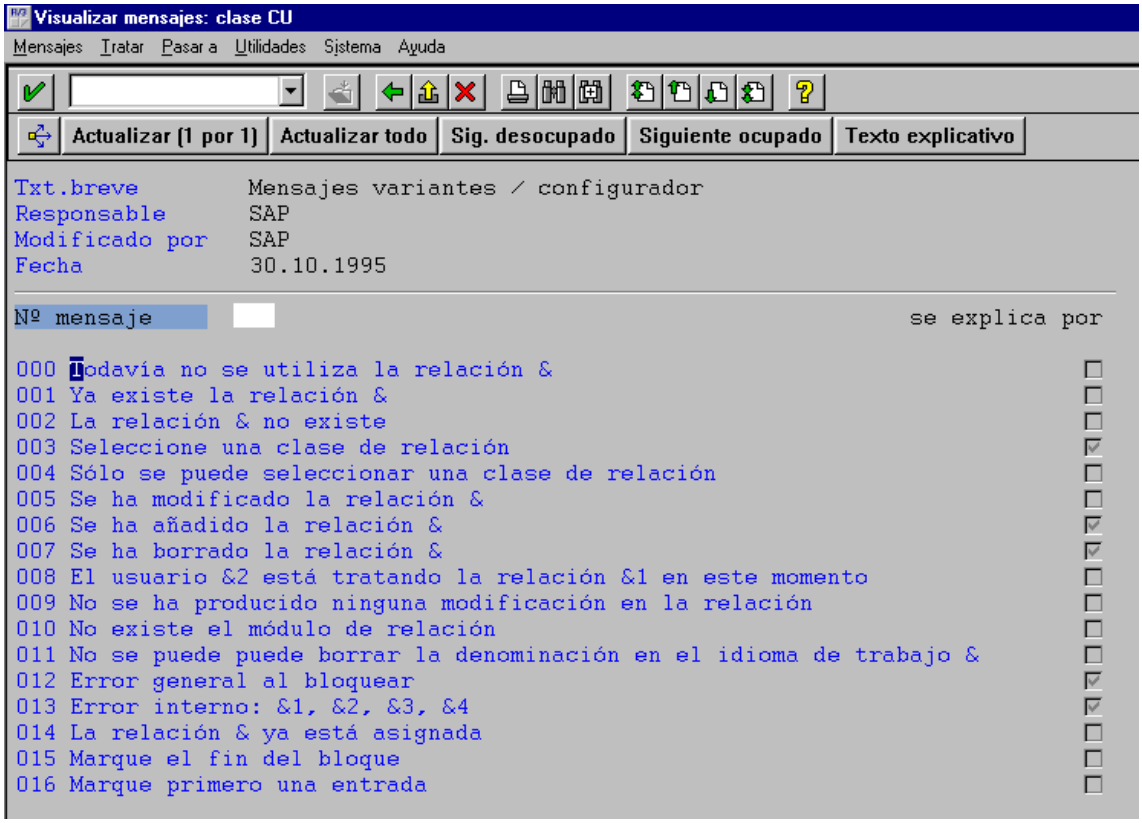
La librería donde están los mensajes se declara en el REPORT(On line) o en el PROGRAM (Modul Pool). En el siguiente ejemplo, se declaran los mensajes tanto en el REPORT como en el PROGRAM.

```
REPORT ZZIIP2 MESSAGE-ID CU.

PROGRAM ZZIIP1 MESSAGE-ID CU.
```

CU es la librería donde están los mensajes.

CU es una librería que ya existe en SAP (hay muchas más). Para ver esa librería hacemos doble clic y nos saldrá la siguiente pantalla:

Como vemos nos sale información sobre la librería. Ahora para ver los mensajes que tiene esa librería, le damos al botón que pone Mensajes y nos saldrá la siguiente pantalla con los mensajes de la librería escogida (En la página siguiente):



Como veis hay bastantes mensajes.

Para poder mostrar los mensajes se utiliza la orden MESSAGE. La sintaxis de la orden sería la siguiente:

MESSAGE Enumero WITH texto.

La E (es el tipo de error, hay muchos tipos de errores diferentes) siempre se pone y después va precedida por el número de mensaje.

En texto, son los mensajes que se sustituyen por él &, es decir, en los mensajes de arriba vemos tienen este carácter '&'. Este carácter es comodín, y sirve para ser sustituido por un mensaje nuestro, una variable o un campo de una tabla.

Ejemplo:

MESSAGE E001 WITH 'PRUEBA'.

El mensaje que saldría sería el siguiente:

Ya existe la relación PRUEBA

Existe otra forma de visualizar los mensajes sin indicar en el REPORT o en el PROGRAM a que librería hay que buscar los mensajes. Se haría de la siguiente forma:

MESSAGE Enumero(librería).

En nuestro caso el ejemplo sería el siguiente:

```
MESSAGE E005(CU).
```

El resultado es el mismo, excepto que no se puede visualizar los mensajes puestos por nosotros.

TIPOS DE MENSAJES

Los tipos son los siguientes: E, W, I, A, S.

- E -> Cuando se produce este error todos los campos se desactivan, y el usuario ha de introducir un nuevo valor.
- W ->Como en el anterior los campos se desactivan, el usuario no necesita introducir nuevos valores, se los introduce SAP y procesa de nuevo la PAI con los nuevos valores. Si pulsamos ENTER el sistema ejecuta las órdenes que hay después del MESSAGE.
- I -> El sistema se para, y automáticamente SAP nos saca un mensaje contextual (popup).
- S->El mensaje se visualiza en la PBO de la siguiente pantalla. Estos mensajes son para el funcionamiento del programa, ya que sólo son informativos.
- A-> Cuando el usuario presiona ENTER, el sistema se para y SAP vuelve al menú principal.

VARIABLES DEL SISTEMA SOBRE MENSAJES

Hay una serie de variables del sistema que se utilizan para los mensajes:

- SY-MSGID -> Identificador de mensaje.
- SY-MSGTY -> Tipo de mensaje (E, I, W, S, A).
- SY MSGNO -> Número de mensaje.
- SY-MSGVx -> Variable x del mensaje.

DEBUGGING

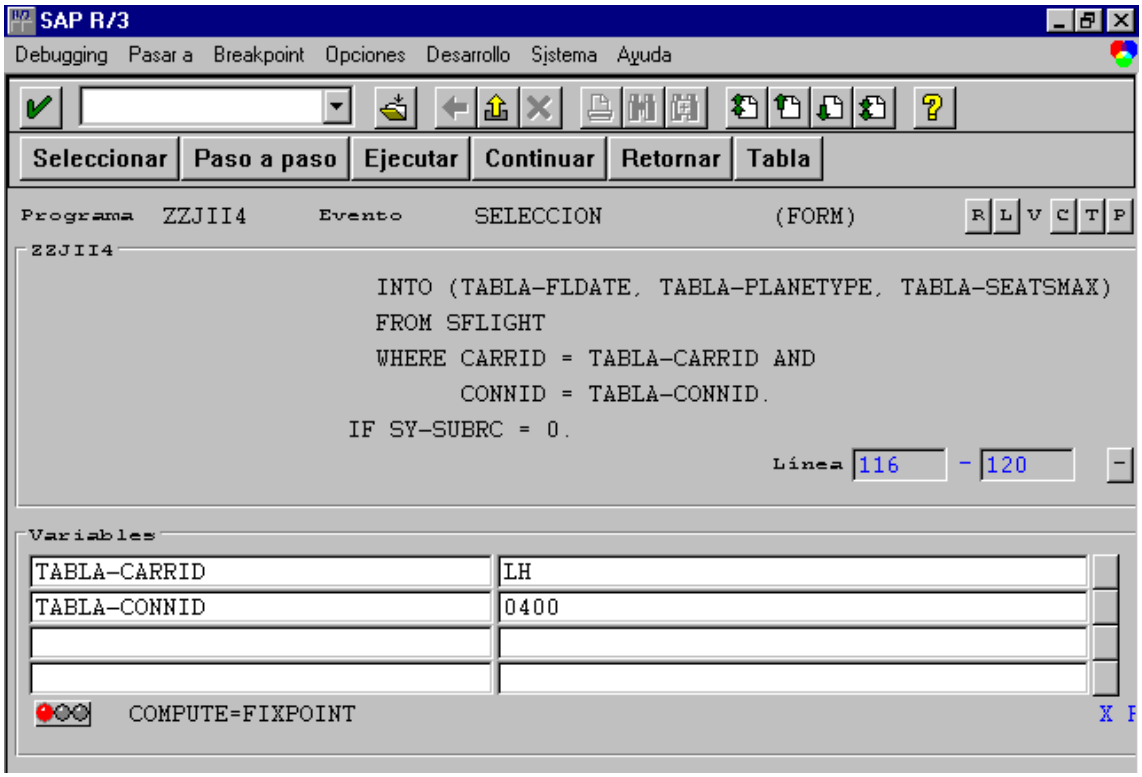
Esta función permite ejecutar el programa línea a línea.

- Se puede activar, /h en el campo de OK-CODE.
- Desde el menú de utilidades.
- Definir un BREAK-POINT en el programa

Los BREAK-POINT se definen de la siguiente forma:

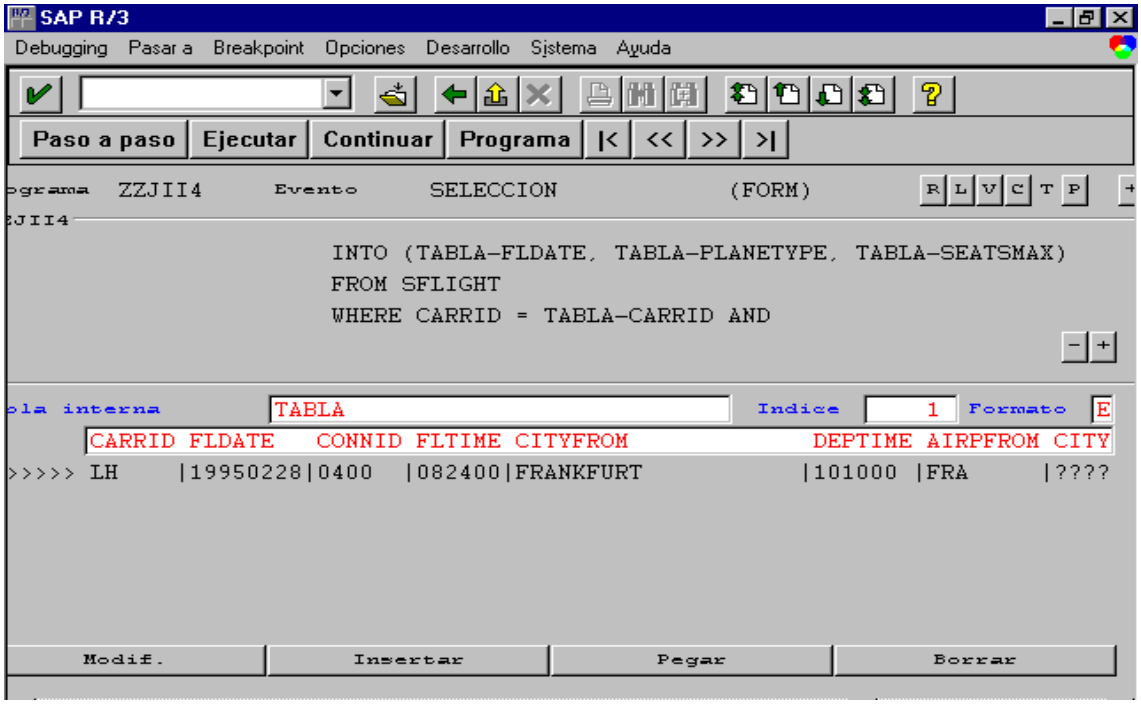
Nos posicionamos sobre la línea en la que queramos hacer la ruptura, después vamos al menú de 'utilidades', 'breakpoints', 'fijar', y entonces tendremos un punto fijado.

Yo personalmente prefiero esta opción ya que es más cómoda y no hay que activar el DEBUGGING, ya que SAP se detiene automáticamente cuando encuentra un punto de ruptura sin necesidad de activarlo. Un ejemplo de ruptura es el siguiente:

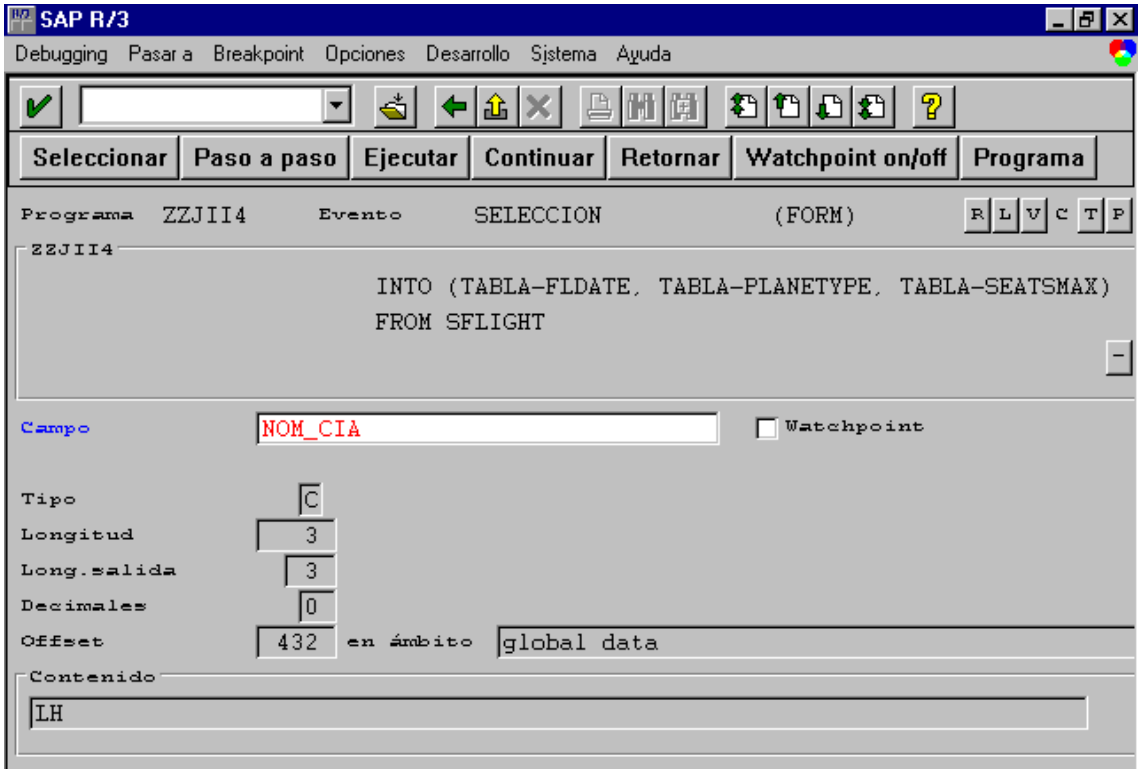


Aquí inspeccionamos las variable CARRID y CONNID de la tabla interna TABLA. Se miran haciendo doble clic sobre un campo o variable y automáticamente saldrá su contenido.

Si vamos al menú 'Pasar a', 'visualizar tabla' o SHIFT+F2, nos saldrá una pantalla, donde pondremos la tabla que queramos ver apareciendo una pantalla en el registro donde esté, la pantalla que sale es la siguiente:



Para ver el contenido de un campo, vamos al menú ‘pasar a’, ‘campos’. Nos sale una pantalla donde introduciremos el nombre de un campo y nos sale el tipo, valor, etcétera de ese campo, un ejemplo:



TIEMPOS DE VELOCIDAD

Es una forma de comprobar los tiempos de respuesta de nuestro programa, se accede a través del menú "sistema", "utilidades", "Análisis tmpo. Ejec."
Pero esto tiene un inconveniente ya que el programa encargado de realizar los cálculos va creando unos ficheros en el HOST los cuales SAP no los borra y poco a poco puede llenar el disco duro del HOST.
La pantalla que sale es la siguiente, antes de que salga la pantalla nos dice el número de procesadores que tiene el HOST:

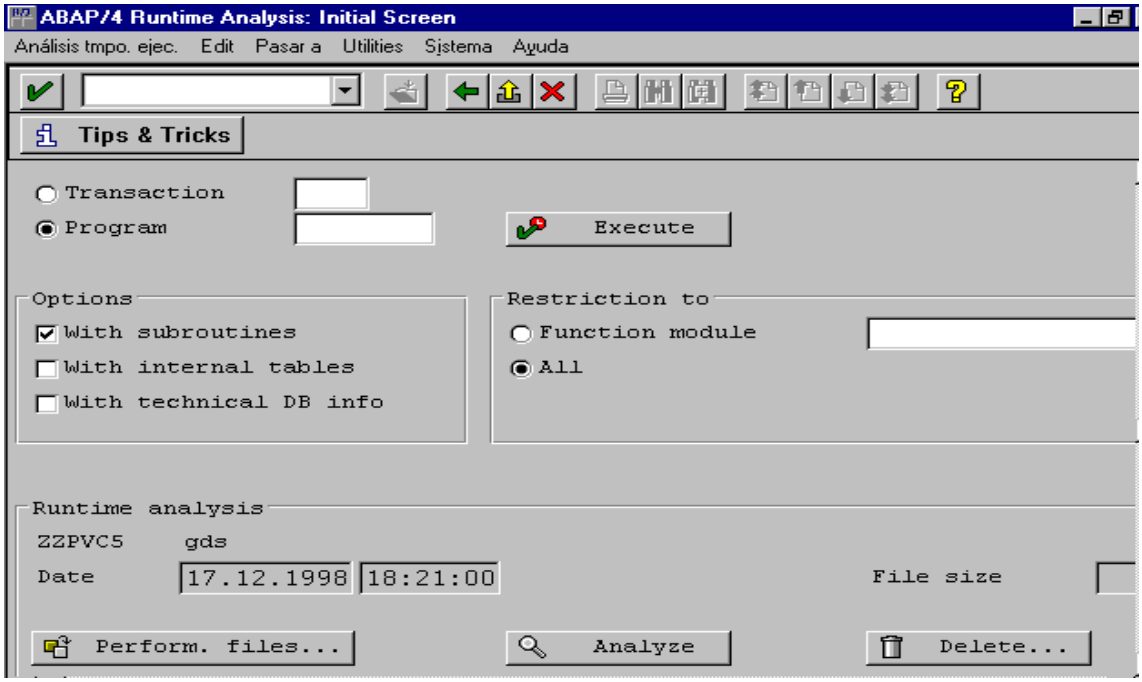
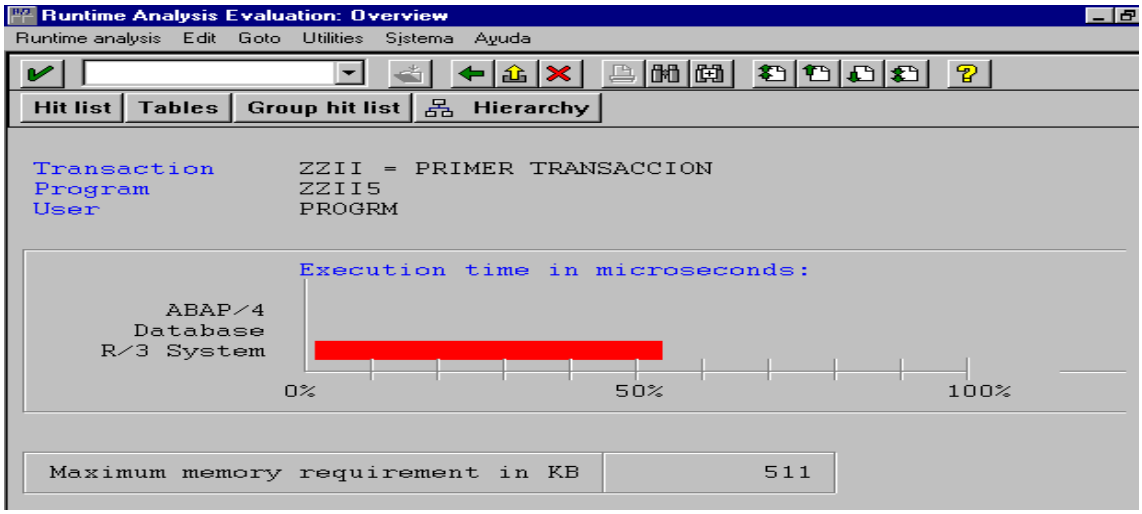


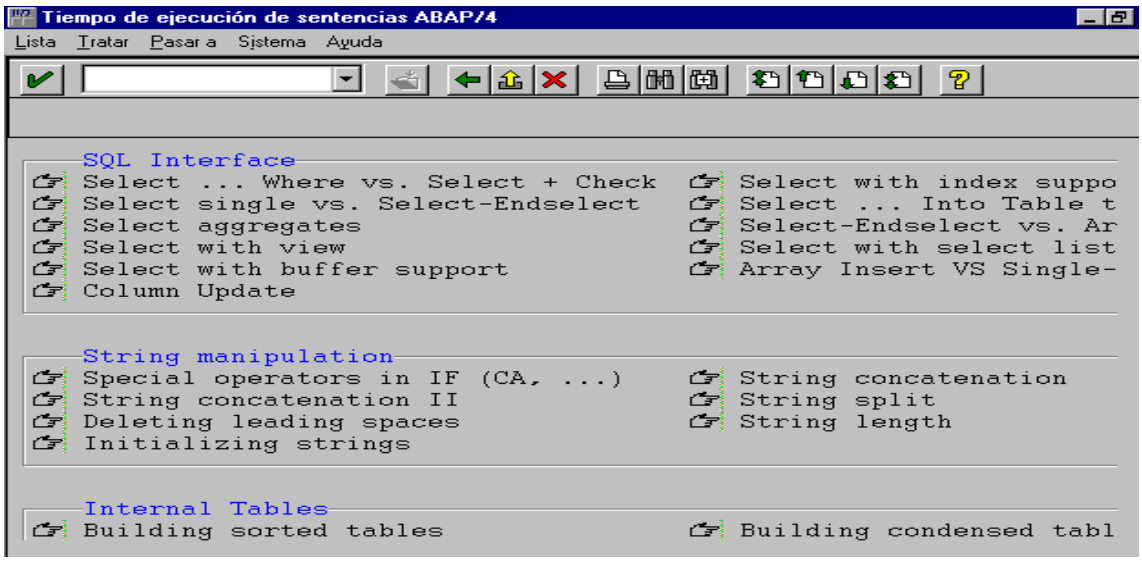
Fig. Análisis.

Podemos medir el tiempo de un programa así como el de una transaction. Ahora vamos a introducir la transacción que hemos creado antes (la ZZII). Activamos el radio button de transaction y ponemos la transaction 'ZZII'. Después ejecutamos el programa (con el botón que pone execute) después de ejecutarlo volvemos atrás (hasta volver a la pantalla de análisis, Fig. Análisis) y por último pulsamos el botón de abajo que pone 'Analyze' y saldrá la siguiente pantalla:



Nuestro programa utiliza mucho el R/3 System y sólo ocupa 511 Kb.

Volviendo a la pantalla de análisis (Véase Fig. Análisis), si le damos al botón TRIPS & TRICKS sale una pantalla donde podemos ver una comparación de velocidad de dos instrucciones que realizan lo mismo pero de diferente forma. Podemos escoger bastantes tipos de comparaciones de lectura de tablas, tratamiento de string, etcétera. La pantalla que sale es esta:



Debajo de la pantalla podemos ver más comparaciones.

También podemos calcular lo que tarda en ejecutar una serie de instrucciones a nivel de programa, esto se realiza con la orden GET RUN TIME FIELD. Con esta instrucción de ABAP/4 determina el consumo de CPU de una secuencia de instrucciones.

La sintaxis de su funcionamiento sería la siguiente:

GET RUN TIME FIELD <f>.

ORDENES.

GET RUN TIME FIELD.

En la primera llamada a esta instrucción sé inicializa el campo <f>. En cada llamada posterior se almacena en <f> el tiempo de CPU en microsegundos transcurridos desde la primera llamada.

OPTIMIZACION DE LOS TIEMPOS DE EJECUCIÓN

DISTINCION DE CASOS

Si el número de posibilidades es limitado se puede saltar de un caso a otro de forma sucesiva. Una vez alcanzada la condición deseada se procesa IF, ELSEIF y CASE la correspondiente secuencia de instrucciones. Cuando existen muchos casos (p. Ej. + de 100=, sería mejor saltar directamente a la parte de código fuente deseada (PERFORM).

TECNICAS SUBPROGRAMAS. VARIABLES LOCALES

Se deberían tener en cuenta:

- El coste de inicializar variables locales
- La utilización local de la instrucción TABLES es muy costosa.
- La definición de tablas internas locales también son costosas porque para cada llamada al FORM son construidas de nuevo y liberadas al llegar al ENDFORM.
- Consumo CPU en la cesión de parámetros, en especial en el traspaso de valores /VALUE o CHANGING)
- En las expresiones lógicas se reduce el número de instrucciones, pero se deberán utiliza de forma meditada:

  IN muy costoso.

  Las expresiones lógicas se resuelven de izquierda a derecha. En el caso de OR debería poner 1º la condición que se cumpla más frecuentemente; en el caso de AND al revés.

  La comparación entre operandos deberá ser del mismo tipo.

- Al indicar el nº de decimales disminuye el rendimiento.
- En ABAP/4 se pueden realizar cálculos con casi todos los tipos de datos, pero internamente se calculan con formatos enteros y coma flotante, siendo el cálculo con enteros el que menos tiempo de CPU consume. Los cálculos con formatos distintos requieren una conversión previa.
- En las unidades de modularización puede ser interesante unificar para los cálculos los parámetros de entrada (tipo), mediante variables locales. Se deberá evaluar los costes de conversión calculo contra los costes de las variables locales y su conversión en el transporte.

TRATAMIENTO DEL SPOOL

El SPOOL es la orden de cola que se genera en la impresión. Su tratamiento depende de las necesidades individuales de cada proceso. Se accede por la transacción "SP01" o por el menú "Sistema", "servicios", "Control impresión".

LLAMADA A TRAVÉS DE UN ABAP

Se realiza con la orden. SUBMIT .... TO SAP-SPOOL. Realiza llamadas de reports con salida de fichero de SAP.

Tiene las siguientes opciones:

1ª

- ... DESTINATION dest -> Salida a impresora
- ... COPIES cop -> Número de copias
- ... LIST NAME name -> Nombre de lista
- ... LIST DATASET dsn -> Nombre de salida de fichero de cola
- ... COVER TEXT text -> Título alternativo de la cola
- ... LIST AUTHORITY auth -> Autorización para visualizar
- ... IMMEDIATELY flag -> Imprimir inmediatamente
- ... KEEP IN SPOOL flag -> Guardar lista después de impresión
- ... NEW LIST IDENTIFICATION flag -> Nuevo fichero de cola cada vez que cambie.
- ... DATASET EXPIRATION days -> Listado de números retenidas
- ... LINE-COUNT lin -> Líneas por página
- ... LINE-SIZE col -> Número de columnas por líneas
- ... LAYOUT layout -> Imprimir formato
- ... SAP COVER PAGE mode
- ... COVER PAGE flag
- ... RECEIVER rec
- ... DEPARTMENT dep -> Nombre de departamento
- ... ARCHIVE MODE armode -> Forma de archivado
- ... ARCHIVE PARAMETERS arparams -> Estructura de archivado
- ... WITHOUT SPOOL DYNPRO -> Controlar pantalla con salto de impresión

2ª

- ... SPOOL PARAMETERS params
- ... ARCHIVE PARAMETERS arparams
- ... WITHOUT SPOOL DYNPRO

Con los parámetros IMMEDIATELY, KEEP IN SPOOL, NEW LIST IDENTIFICATION y COVER TEXT, flag el flag debe ser un literal o campo carácter con longitud 1. Si el flag está en blanco el parámetro switch esta apagado, pero ningún otro carácter de switch debe estar activado.

JOB

Un job es un proceso que lanza la ejecución de un programa ABAP/4 o un programa externo a una hora o día determinado.

Son útiles cuando queremos ejecutar un programa a una determinada hora, por ejemplo, cuando el servidor este menos ocupado(suele ser por las noches) lanzaremos un programa que haga un uso intensivo del servidor.

Para poder crear un job tenemos que ir al menú “sistema”, “jobs”, “definición jobs”. Y nos saldrá la siguiente pantalla:

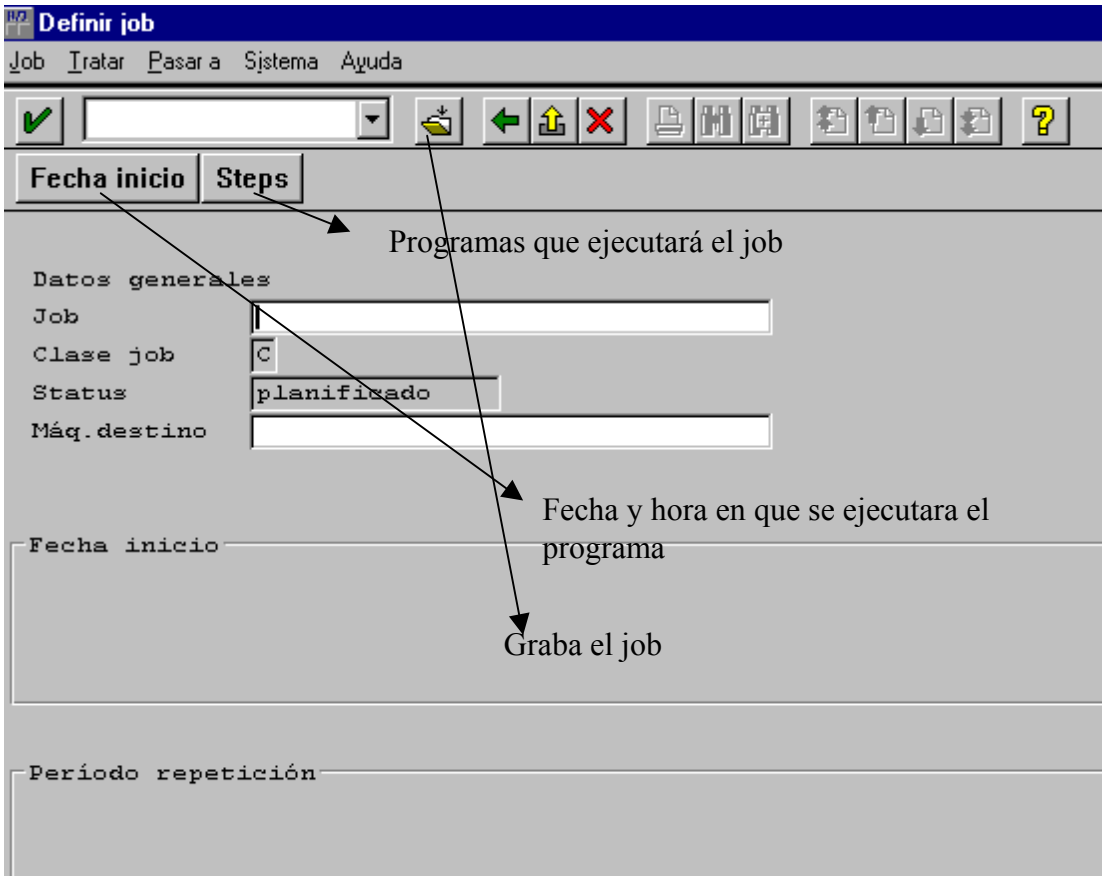
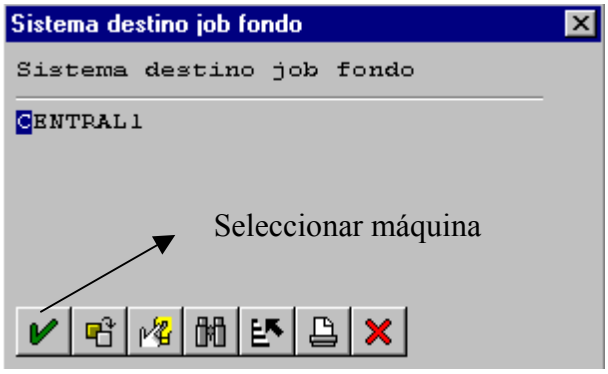


Fig. Definir job.

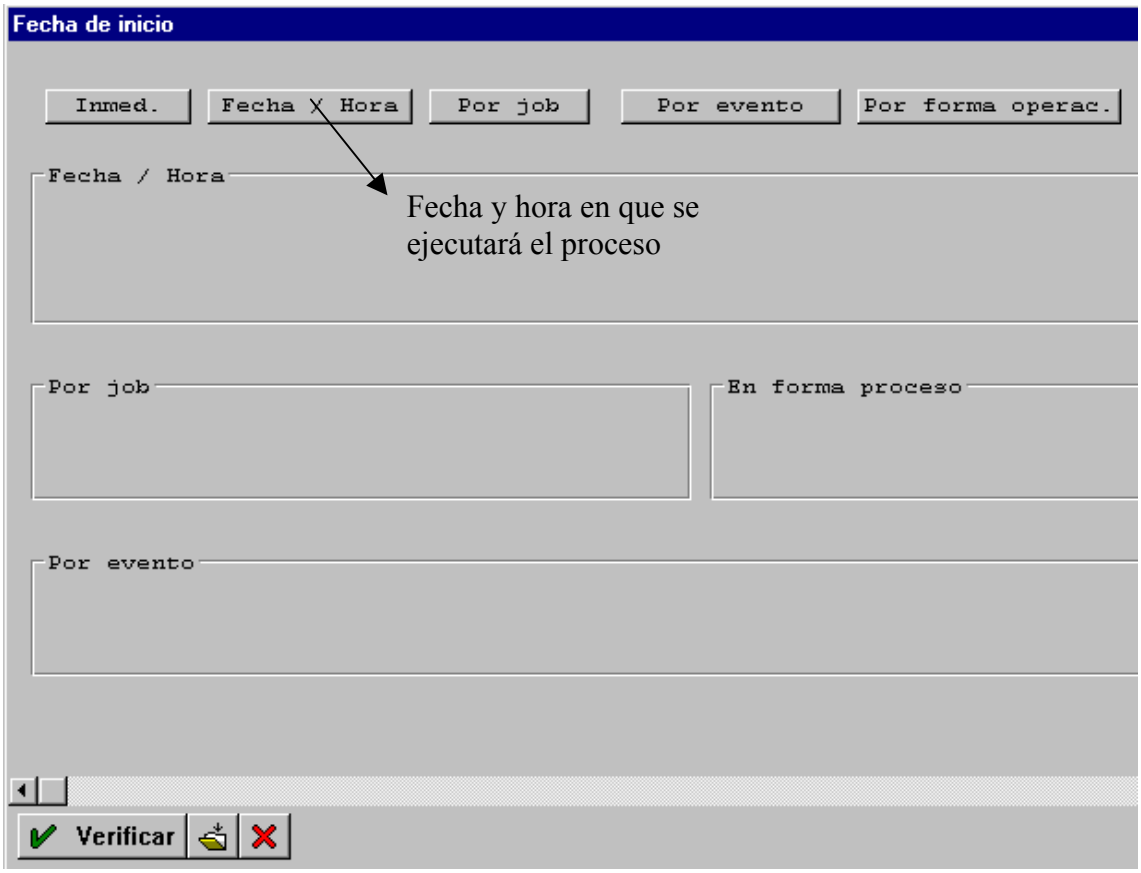
En el campo “Job” escribiremos el nombre del job que deseamos crear, como ejemplo pondremos el nombre “ZZII” y en el campo “Maq.destino” escribiremos la maquina de destino donde se ejecutará el job, para ver que máquinas hay nos posicionamos con el cursor y pulsamos F4 o le damos al mathcode, en los dos casos sale la siguiente imagen:



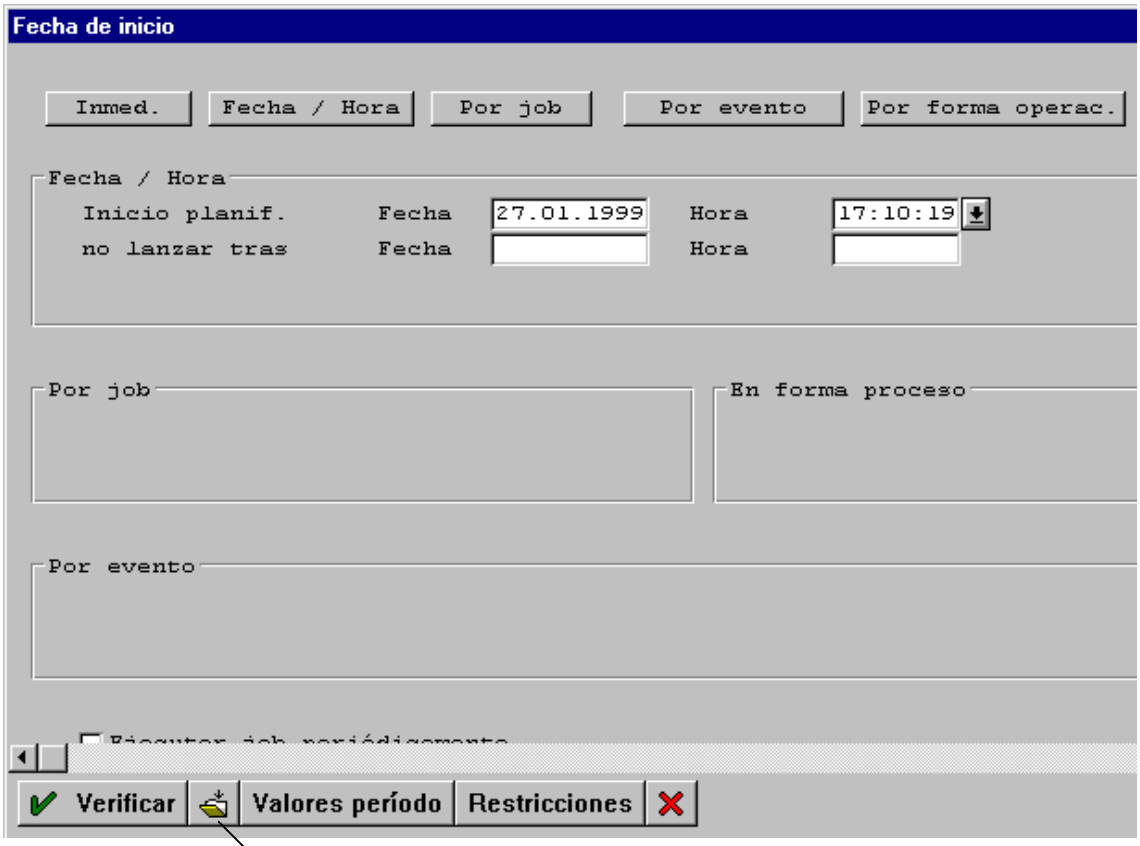
Para seleccionar una máquina, hacemos doble clic en la máquina que queramos o nos posicionamos con el cursor en la máquina en cuestión y le damos al botón de seleccionar.

Después de introducir el nombre del job y la maquina de destino donde se ejecutara el programa, solo nos falta indicarle cuando y que queremos que se ejecute.

Primero seleccionaremos la fecha y la hora en que se ejecutara el o los procesos seleccionados. Para poder introducir le daremos al botón “fecha inicio” y nos saldrá esta pantalla:



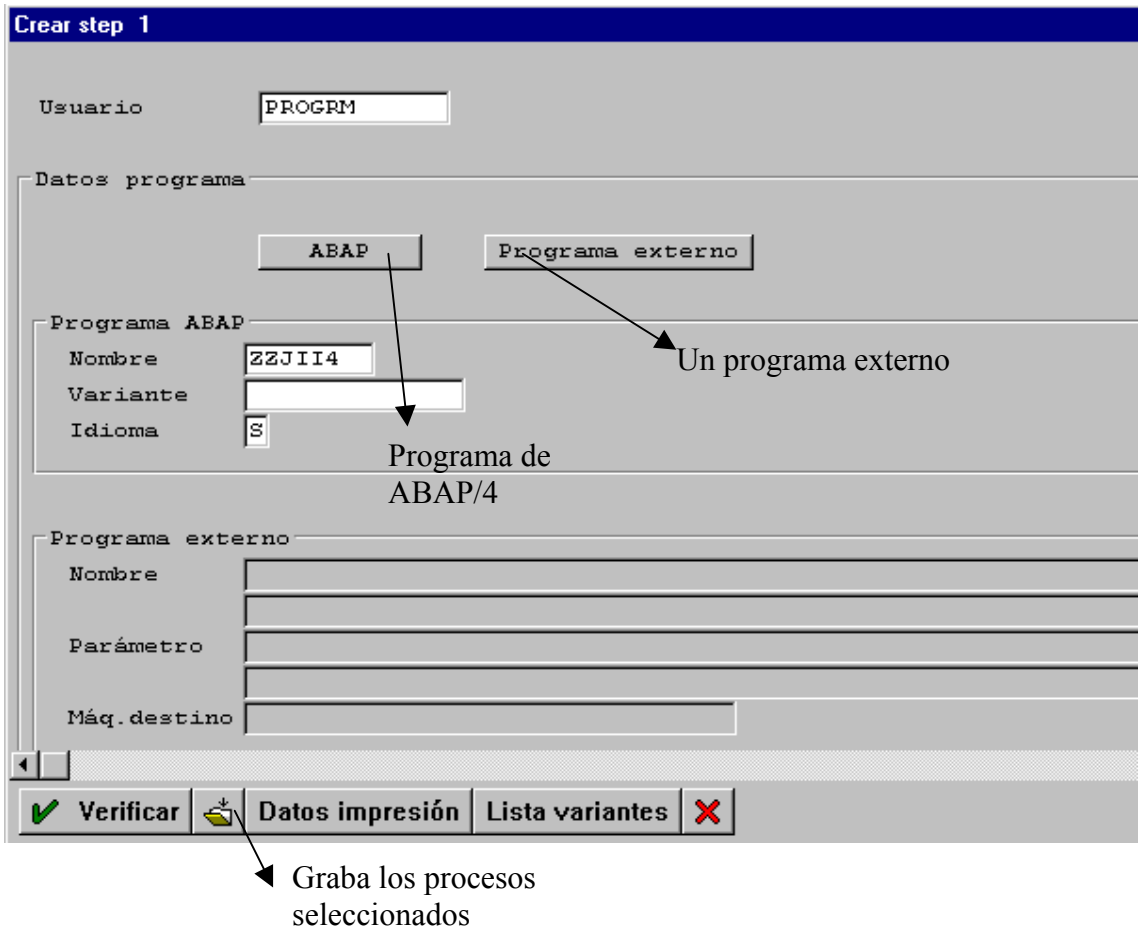
En este caso como ejemplo solo lo haremos que se ejecute ha una hora y un día determinado. Para poderlo hacer pulsaremos al botón "Fecha/Hora" y nos saldrá la pantalla anterior pero con más datos:



Tomar o grabar la hora en que se ejecutará el job

En este caso he puesto que se ejecute el día 27.01.1999 a las 17:10:19 horas. Como pongo la fecha pues, a través de los mathcodes o posicionandonos sobre el campo y pulsando F4 y entonces nos saldrá una pantalla donde podremos elegir el día o la hora dependiendo en campo hemos pulsado F4 o el matchcode.

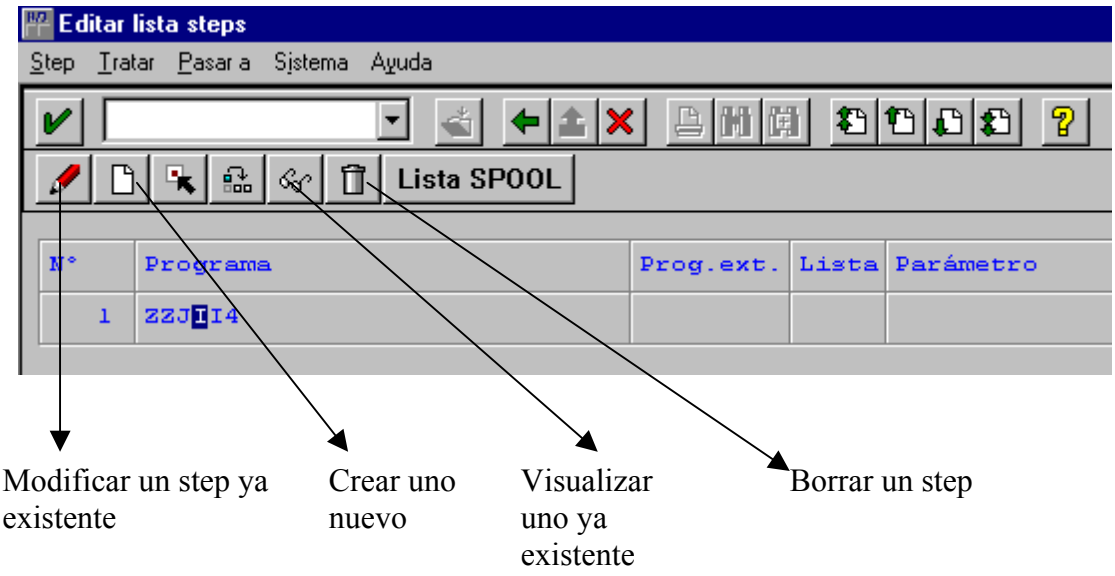
Después de seleccionar el día y la hora grabaremos la fecha dando al botón de grabar. Por último seleccionaremos que procesos se van a ejecutar, o sea, volveremos a la pantalla inicial (Véase Fig. Diseño job) y pulsaremos el botón "Steps" y nos saldrá la siguiente pantalla (en la siguiente página):



Aquí tenemos la posibilidad de ejecutar tanto programas creados por nosotros o programas externos.
Como ejemplo haremos un que se ejecute un programa creado por nosotros, para hacerlo pulsaremos el botón “ABAP”, y se activará la box “Programa ABAP/4”, después escribiremos el nombre del programa a ejecutar (en nuestro caso es un simple listado) y también nos da la posibilidad de incluir una variante ese programa (también en nuestro caso no pondremos ninguna, pero las variantes se explicarán más adelante)

Si queremos ejecutar un programa externo pulsaremos el botón “Programa externo” y se activará la box “Programa externo” y entonces escribiremos el nombre del programa, los parámetros que le pasaremos y la máquina de destino donde se ejecutarán los programas.

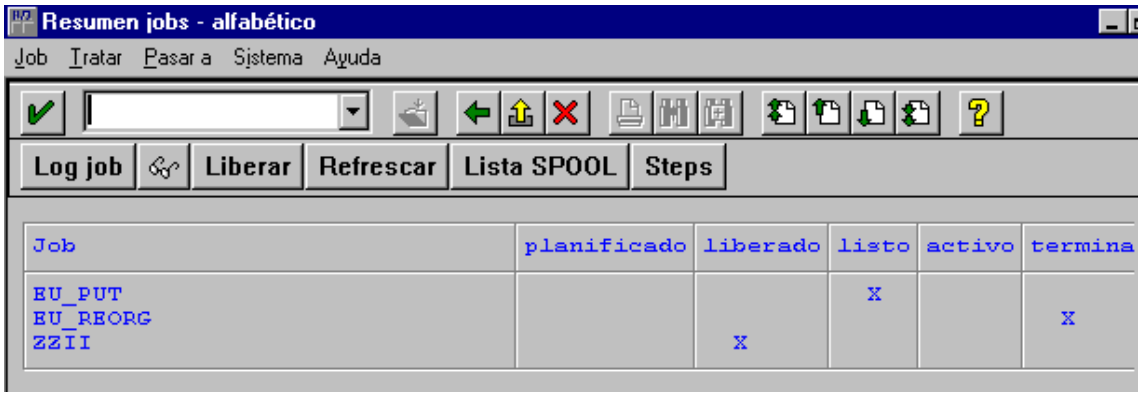
Después de todo esto lo grabaremos y nos saldrá la siguiente pantalla (en la página siguiente):



En esta pantalla podemos modificar, crear, borrar, etc. un step. Ya que un job puede ejecutar más de un proceso.

Si no queremos hacer nada más, pulsamos F3 o el botón “Volver” para volver a la pantalla donde estabamos.

Si queremos ver que jobs están pendientes de ejecución, los que sean han ejecutando ya o realizar cualquier otra operación con un job, pues entonces vamos al menú “Sistema”, “Jobs”, “Lista jobs” y nos saldrá la siguiente pantalla:



Desde aquí podemos visualizar, refrescar, etc. ha un job. El que hemos creado antes está liberado por lo que solo hace falta esperar a que llegue la hora para comprobar si se ha ejecutado.

GENERAR UN JOB MEDIANTE UN PROGRAMA ABAP/4.

También podemos generar un JOB mediante un programa ABAP, utilizando módulo de funciones.

Módulo de función 'JOB_OPEN': permite abrir un job y devuelve el número de job que se ha abierto.

```
CALL FUNCTION ´JOB_OPEN'
EXPORTING
        DELANFREP = <borrar        report        interno>
        JOBNAME = <nombre del job>
        SDLSTRTDT = <fecha arranque> (opcional)
        SDSSTRTTM = < hora de arranque> (opcional)
IMPORTING
        JOBCOUNT = <nº de job>
EXCEPTIONS
        ALREADY_LOCKED = 1
        JOBCOUNT_CANTGEN = 2
        JOBNAME_MISSING = 3
```

Módulo de función 'JOB_SUBMIT': permite insertar un paso en el job.

```
CALL FUNCTION 'JOB_SUBMIT'
EXPORTING
        AUTHCKNAM = <usuario batch>
        JOBCOUNT = <nº job>
        JOBNAME = <nombre job>
        REPORT = <nombre report>
        VARIANT = <variante del report>
EXCEPTIONS
        ALREADY_LOCKED = 1
        BTCUNAME_BADTYPE = 2
        BTCUNAME_MISSING = 3
```

Nota: la instrucción **SUBMIT <report> vía job <jobname> nombre <jobcount> WITH selección AND RETURN**
no necesita especificar una variante. Mediante el parámetro selección se especifican los
valores de selección.

Módulo de función 'JOB_CLOSE': cerrar job.

```
CALL FUNCTION 'JOB_CLOSE'
EXPORTING
        JOBCOUNT = <nº de job>
        JOBNAME = <nombre del job>
        SDLSTRTDT = <fecha de arranque> (opcional)
        SDLSTRTTM = <hora de arranque> (opcional)
        STRTIMMED = <arranque inmediato> (opcional)
EXCEPTIONS
        ALREADY_LOCKED = 1
        JOBCOUNT_CANTGEN = 2
```

```
            JOBNAME_MISSING = 3
```

Si al abrir el JOB no se ha especificado la fecha y hora de arranque, se puede especificar aquí. También se puede especificar arranque inmediato.

```
********************  EJEMPLO JOB  ********************

REPORT ZSPA006 MESSAGE-ID ZK.

TABLES:PRPS.

DATA: JOB_NAME LIKE TBTCJOB-JOBNAME,
      OK.

SELECT-OPTIONS: POSID FOR PRPS-POSID OBLIGATORY.

START-OF-SELECTION.
      CLEAR OK.
      PERFORM FONDO USING OK.
      IF OK = 'X'
* Error en la ejecución del job
        MESSAGE I151.
      ELSE.
* OK
        MESSAGE I152 WITH JOB_NAME.
      ENDIF.

********************  FORM FONDO  ********************
*   Descripción: creamos el job, lo lanzamos y lo cerramos.

FORM FONDO USING OK.
      DATA: JOBCOUNT LIKE TBTCJOB-JOBCOUNT.

* Nombre del job.

      CONCATENATE 'ZSPA_' SY-UNAME '_' SY-DATUM INTO JOB_NAME.
      CALL FUNCTION 'JOB_OPEN'
            EXPORTING
                  JOBNAME          = JOB_NAME
            IMPORTING
                  JOB_COUNT        = JOB_COUNT
            EXCEPTIONS
                  CANT_CREATE_JOB      = 1
                  INVALID_JOB_DATA     = 2
                  JOBNAME_MISSING      = 3
                  OTHERS               = 4.
            IF SY-SUBRC = 0.
            SUBMIT ZSPA0005 VIA JOB JOB_NAME NUM JOBCOUNT
                              WITH POSID IN POSID
```

```
                                AND RETURN.
        IF SY-SUBRC = 0.
                PERFORM CERRAR_JOB USING JOBCOUNT.
                IF SY-SUBRC <> 0.
                        OK = 'X'.
                ENDIF.
        ELSE.
                OK = 'X'.
        ENDIF.
        ELSE.
                OK = 'X'.
        ENDIF.

ENDFORM.



********************  FORM CERRAR_JOB  ******************
   *   Descripción: Cerramos Job


FORM CERRAR_JOB USING JOB-COUNT.
        DATA: RELEASE LIKE BTCH0000-CHAR1.
        CALL FUNCTION 'JOB_CLOSE'
                EXPORTING
                        JOB_COUNT               = JOBCOUNT
                        JOBNAME                 = JOB_NAME
                        STRTIMMED               = 'X'
                IMPORTING
                        JOB_WAS_RELEASED                = RELEASE
                EXCEPTIONS
                        CANT_START_IMMEDIATE    = 1
                        INVALID_STARDATE                = 2
        JOBNAME_MISSING                 = 3
        JOB_CLOSE_FAILED                = 4
        JOB_NOSTEPS                     = 5
        JOB_NOTEX               = 6
        LOCK_FAILED                     = 7
        OTHERS                  = 8.
ENDFORM.
```

En el programa anterior veíamos como creábamos un Job, en ese programa podemos decirle que ejecute un determinado programa, por ejemplo, el que viene a continuación:

```
REPORT ZSPA0005.
TABLES: PRPS.

DATA: BEGIN OF TABLA OCCURS 100,
        POSID   LIKE        PRPS-POSID,
```

```
        OBJNR  LIKE        PRPS-OBJNR,
        PBUKR LIKE         PRPS-PBUKR,
        PGSBR  LIKE        PRPS-PGSBR,
        PKOKR LIKE         PRPS-PKOKR,
END-OF-TABLA.

SELECT OPTIONS: POSID FOR PRPS-POSID OBLIGATORY.

START-OF-SELECTIONS.
        SELECT POSID OBJNR PBUKR PGSBR PKOKR
                INTO CORRESPONDING FIELDS OF PRPS FROM PRPS
                WHERE POSID IN POSID.
        CLEAR TABLA.
        MOVE:       PRPS-POSID  TO TABLA-POSID,
                PRPS-OBJNR TO TABLA-OBJNR,
                PRPS-PBUKR TO TABLA-PBUKR,
                PRPS-PGSBR TO TABLA-PGSBR,
                PRPS-PKOKR TO TABLA-PKOKR.
        APPEND TABLA.
        ENDSELECT.

SKIP.
ULINE: /1(80).
LOOP AT TABLA.
        FORMAT COLOR 3 ON.
        WRITE:  /1 '|' ,   2 TABLA-POSID.
        WRITE: 12 '|' , 13 TABLA-OBJNR.
        WRITE: 49 '|' , 50  TABLA-PBUKR.
        WRITE: 60 '|' , 61 TABLA-PGSBR.
        WRITE: 71 '|' , 72 TABLA-PKOKR.
        WRITE: 80 '|'.

        FORMAT COLOR 3 OFF.
ENDLOOP.
ULINE: /1(80).
```

ACTUALIZACION ASINCRONA

Para que la actualización no de tiempos de espera adicionales, se ejecute en una tarea de actualización asíncrona. La tarea de actualización tiene más prioridad que la de diálogo y puede ser ejecutada como proceso independiente en otra máquina. En el diálogo se crea un registro log, con los datos a modificar y el nombre del programa de actualización. El programa de actualización asíncrona toma de las tablas de log los datos a actualizar,

El programa de actualización deberá ser iniciado por el programa de diálogo:

```
COMMIT WORK
```

Los programas de actualización asíncrona son siempre módulos de función:

```
CALL FUNCTION "XXX" IN UPDATE TASK
```

Ejemplo:

```
CALL FUNCTION "UPDATE_ZKURS"
IN UPDATE TASK
EXPORTING KURS=ZKURS
```

La llamada a esta función escribe los datos que están definidos con parámetros EXPORTING, a la tabla log.

INCLUIR UNO O VARIOS REGISTROS DE ACTUALIZACION

Dynpro 100

```
FORM _______ ON COMMIT
    CALL FUNCTION IN UPDATE TASK
END FORM
```

Dynpro 200

```
FORM _______ ON COMMIT
    CALL FUNCTION IN UPDATE TASK
ENDFORM
COMMIT WORK
```

La actualización de todos los registros grabados mediante PERFORM _____ ON COMMIT se efectúa cuando se indica COMMIT WORK.

Si se produjera un error en la actualización, la tabla log quedaría igual y daría un status "ERR".

F3/F11 DESPUES DE CREAR EL REGISTRO LOG

Si guardo uno o más módulos de función con PERFORM _____ ON COMMIT, debo borrar la memoria intermedia a continuación del Dynpro inicial, en caso de que se haya

utilizado la función F3 = BACK. Mediante ROLLBACK se borrar la memoria de ordenes de log.

El registro de log se puede evaluar en la transacción SM13.

Los registros de log que no se puedan procesar se marcan con el status “ERR”; los registros de log actualización se borra.

PANTALLAS DE SCROLL (BÁSICO)

Cuando se genera una lista básica, la instrucción SCROLL junto con INDEX se refieren a la lista básica de la misma. Cuando los detalles de la lista son generados, se refieren a la directa subordinación de la lista que justo se visualizo.
La instrucción solamente es efectiva si se usa después de un WRITE, un SKIP o alguna instrucción similar.
Los campos de sistema que son usados para el modul-pool son documentados en *System Fields for List.*

Nota: el valor de *cód.return* se fijará de la siguiente forma:

SY-SUBRC = 0 OK
SY-SUBRC = 4 límite de la lista rechazado. Scrolling imposible
SY-SUBRC = 8 lista no existe. Scrolling imposible

Variantes

- SCROLL LIST TO FIRST PAGE

El scroll se coloca en la primera página (los comandos correspondientes son P- - o PP- -).

Apéndices: INDEX idx
LINE lin.

INDEX

El nivel de la lista de scroll corresponde a la variable del sistema SY-LSIND al generarse
la lista.

Ejemplo: SCROLL LIST INDEX 1 TO FIRST PAGE
El scroll empezará en el nivel 1 de la lista.

Nota: Si la nueva lista (la cual reemplaza la última lista visualiza) es creada en un evento y te quieres desplazar a un sitio en particular en esta nueva lista, tener en cuenta que un cambio hecho a la variable del sistema SY-LSIND es tenido en cuenta solo después del evento. Así pues, SY-LSIND debería ser manipulado después de la última instrucción del evento (por ejemplo: SY-LSIND = SY-LSIND – 1). La instrucción de scroll añadiendo INDEX idx debe ser usada para el desplazamiento en esta nueva lista. De esta forma, el desplazamiento en la lista antigua se anula.

LINE

Visualiza la lista desde la línea **lin** (corresponde al comando de entrada PLnn). El encabezado estándar TOP-OF-PAGE no se puede mover verticalmente y, por lo tanto, no puede ser ignorado cuando la línea **lin** está definida.

- SCROLL LIST TO LAST PAGE

Desplazamiento de la última página de la lista visualizada en la pantalla (corresponde al comando de entrada PP++).

Puede tener los mismos apéndices que al primera variante.

- SCROLL LIST TO PAGE pag.

Desplazamiento especificado en la página de la lista visualizada sobre la pantalla (corresponde al comando de entrada PPnn).

Puede tener los mismos apéndices que la primera variante.

Ejemplo 1: SCROLL LIST INDEX 1 TO PAGE 7
Desplazamiento sobre la lista de nivel 1 a la página 7.

Ejemplo 2: SCROLL LIST INDEX 1 TO PAGE 7 LINE 5
Desplazamiento sobre la lista de nivel 1 a la página 7 y visualizar desde la línea cinco.

Nota: la lista ahora visualizada desde la página 7 y línea 5 sale sin encabezado, a pesar de que exista encabezado para esa página (estándar y/o top-of-page). Si se quiere Visualizar una línea en particular inmediatamente después del encabezado, se puede usar el módulo de función LIST SCROLL LINE... el cual realiza los cálculos necesarios.

- SCROLL LIST TO COLUMN col.

Visualiza la lista desde la columna **col** (corresponde al comando de entrada PSnn)

Puede tener los mismos apéndices que la primera variante.

- SCROLL LIST FORWARD

Desplazamiento hacia abajo de la lista por la página de la pantalla (no más abajo de la última página) (corresponde al comando de entrada P+).

Apéndices: INDEX idx (como en la primera variante)
..n PAGES

..n PAGES

Desplazamiento de n páginas hacia delante (corresponde al comando de entrada PP+n)

- SCROLL LIST BACKWARD

  Desplazamiento hacia atrás de la lista (no antes de la primera página), corresponde al comando de entrada P-.

  Puede tener los mismos apéndices que la variante anterior (con la diferencia de que aquí vamos hacia atrás).

- SCROLL LIST LEFT

  La visualización empieza en la columna 1 de la pantalla (es decir, justificado a la izquierda), (corresponde al comando de entrada PS--.

  Apéndices: INDEX idx (como en la primera variante)
  BY n PLACES

  BY n PLACES

  Los shifts de la lista visualizan n columnas. Tú determinas la dirección usando los parámetros LEFT o RIGHT (corresponde a los comandos PS+n y PS-n)

- SCROLL LIST RIGHT

  Se visualiza justificado a la derecha. Esto solo tiene sentido en los reports en que se escoge la longitud de la pantalla mediante el comando LINE-SIZE. (Corresponde al comando PS++).

  Puede tener los mismos apéndices que la variante anterior.

VARIANTES

Cuando en un programa tenemos que introducir siempre los mismos datos para poder realizar un proceso y queremos que SAP nos introduzca estos datos de forma automática entonces hacemos una variante. Las variantes solo se pueden hacer en programas de tipo "report".

Como ejemplo haremos una variante de un listado de compañías, Supongamos que cada cierto tiempo nos saque un listado de una compañía aérea en concreto (la NJ). Para hacer la variante primero hemos de ejecutar el programa (a través del editor ABAP/4) del cual queremos hacer la variante. En nuestro caso cuando ejecutemos el programa saldrá la siguiente pantalla:

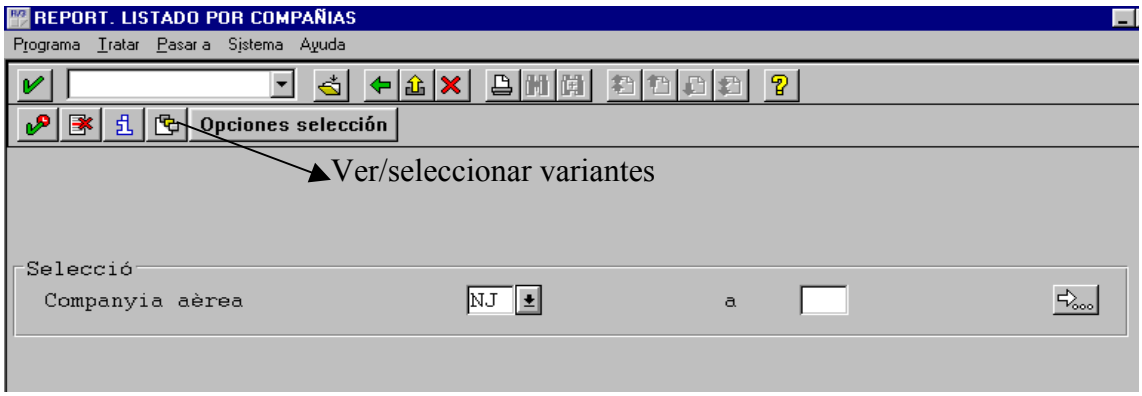
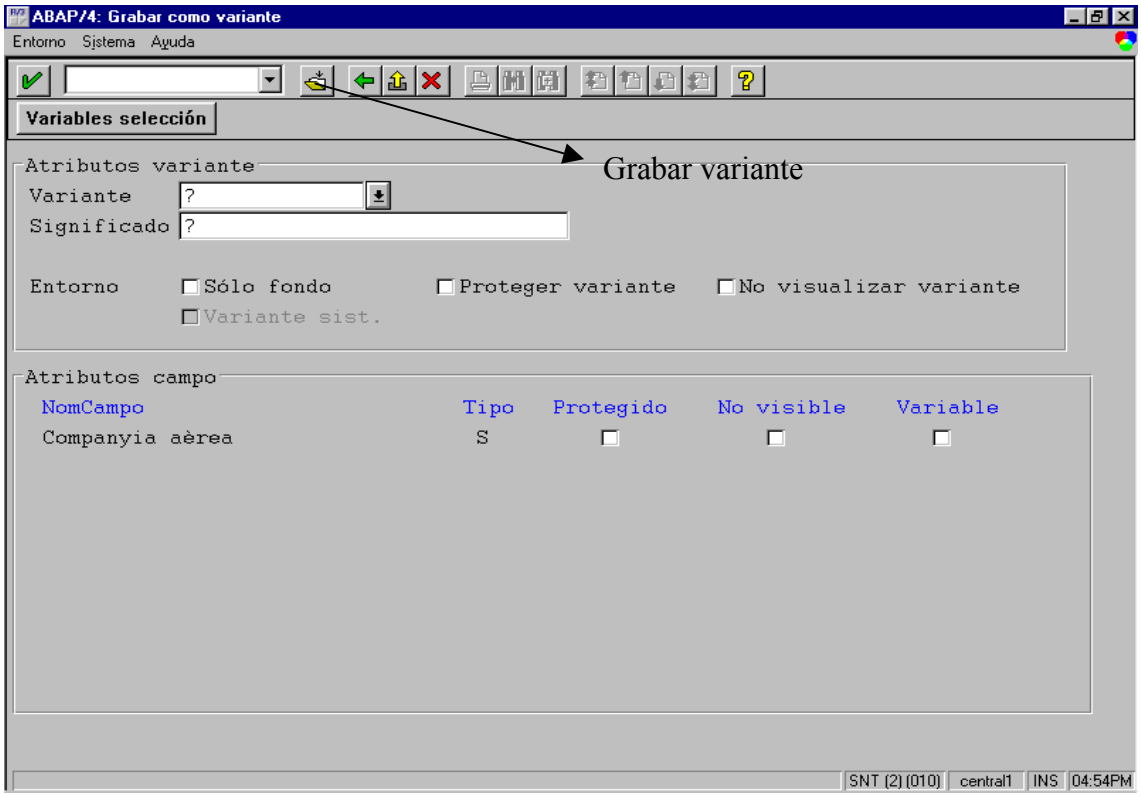


Fig. Programa.

Cuando salga la pantalla introduciremos el nombre la compañía aérea en cuestión (como ya he dicho es la NJ), a continuación, pulsaremos el botón de grabar o F4 y nos saldrá la siguiente pantalla:



Aquí tenemos que introducir dos campos obligatorios. En el campo “variante” introduciremos el nombre de la variante y en campo “significado” un explicación referente a esa variante.

Cuando los tengamos introducidos grabaremos la variante y nos volverá a la pantalla del programa que estemos ejecutando.

Si queremos ver las variantes que tiene nuestro programa o seleccionar una de ellas, pulsamos el botón de ver variantes (Véase Fig. Programa) y nos saldrá la siguiente pantalla:

Para poder seleccionarlas hacemos doble clic en la variante que queramos o nos hacemos clic en una variante y pulsamos el botón “Seleccionar”.

Después veremos como se inserta el campos o campos del que hallamos hecho la variante.

Nota: en este ejemplo solo hemos hecho una variante de un solo campo, pero se pueden hacer variantes de más de un campo.

PROGRAMAS DE EJEMPLO

Esto es una recopilación de programas que abarca varios temas. Solo expondré el código fuente y las pantallas que acompañan al programa.

PAGINACIÓN

Este es un sencillo ejemplo de paginación. El programa consta, de que el usuario introduzca un nombre de compañía y cuando presione el botón de buscar, el programa buscará los vuelos de esa compañía y solo mostrara tres campos. Y a través de dos botones de izquierda y derecha navegaremos por los vuelos de esa compañía. Y si queremos ver todos los datos de un vuelo pulsaremos otro botón donde nos mostrará la información completa de esa compañía.

Primero mostraré el código completo del programa, ya que para hacer las pantallas primero hemos de generar el programa, para poder tomar las tablas en la SCREEN PAINTER:

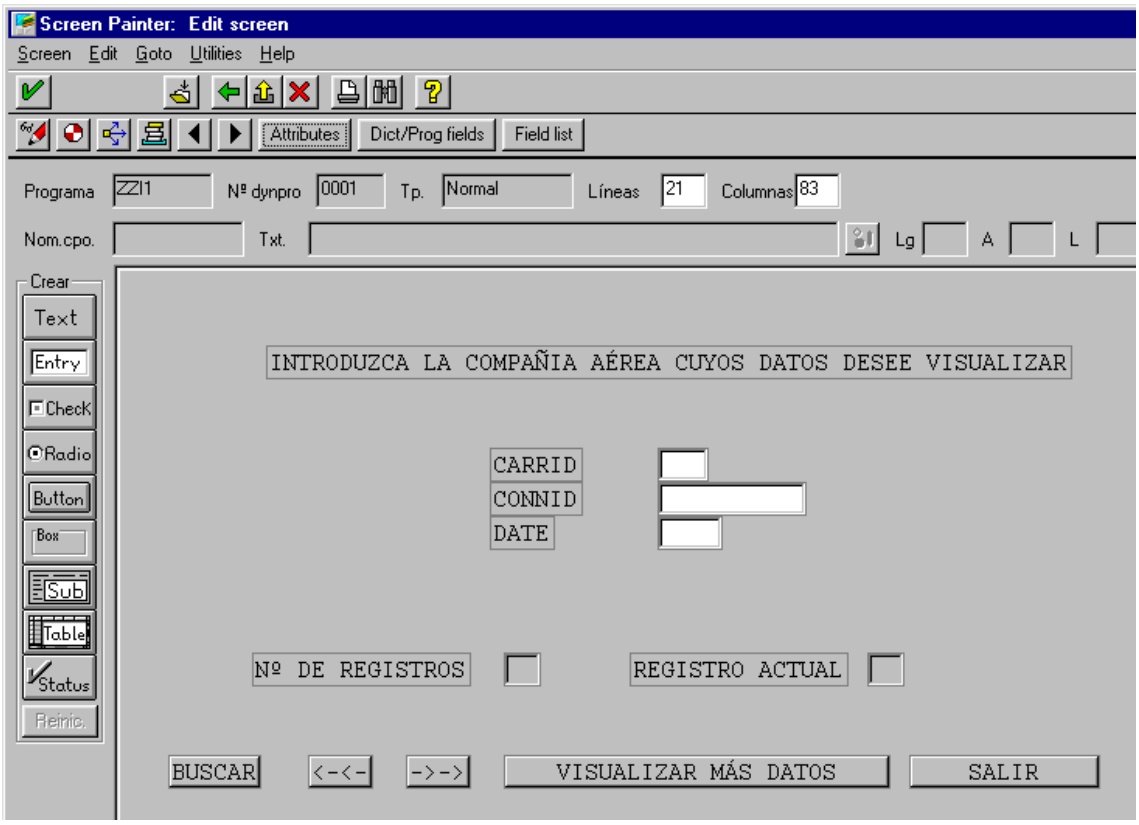
```
PROGRAM ZZI1.
*---------------------------------------------------------------------
* DECLARACIÓN DE LAS VARIABLES
*---------------------------------------------------------------------
TABLES: SPFLI, SFLIGHT.
DATA: BEGIN OF TABLA OCCURS 0,
     CARRID    LIKE SFLIGHT-CARRID,
     FLDATE    LIKE SFLIGHT-FLDATE,
     CONNID    LIKE SFLIGHT-CONNID,
     SEATSOCC  LIKE SFLIGHT-SEATSOCC,
     FLTIME    LIKE SPFLI-FLTIME,
     CITYFROM  LIKE SPFLI-CITYFROM,
     DEPTIME   LIKE SPFLI-DEPTIME,
     AIRPFROM  LIKE SPFLI-AIRPFROM,
     CITYTO    LIKE SPFLI-CITYTO,
     ARRTIME   LIKE SPFLI-ARRTIME,
     AIRPTO    LIKE SPFLI-AIRPTO,
     PLANETYPE LIKE SFLIGHT-PLANETYPE,
     SEATSMAX  LIKE SFLIGHT-SEATSMAX,
END OF TABLA.
DATA: PAG_TOT(2) TYPE N.
DATA: INDE(2) TYPE N VALUE 1.
*&---------------------------------------------------------------------*
*&      Module  STATUS_0001  OUTPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE STATUS_0001 OUTPUT.
  SET PF-STATUS 'PATAPA'.
  SET TITLEBAR 'PEPE'.

ENDMODULE.                    " STATUS_0001  OUTPUT
```

```
*&---------------------------------------------------------------------*
*&      Module  USER_COMMAND_0001  INPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE USER_COMMAND_0001 INPUT.
  CASE SY-UCOMM.
    WHEN 'BUS'.
      PERFORM BUSCAR.
    WHEN 'VER'.
      SET SCREEN 2. LEAVE SCREEN.
    WHEN 'ADE'.
      IF INDE < PAG_TOT.
        INDE = INDE + 1.
        READ TABLE TABLA INDEX INDE.
      ENDIF.
    WHEN 'ATR'.
      IF INDE > 1.
        INDE = INDE - 1.
        READ TABLE TABLA INDEX INDE.
      ENDIF.
    WHEN OTHERS.
      SET SCREEN 0. LEAVE SCREEN.
  ENDCASE.

ENDMODULE.                             " USER_COMMAND_0001  INPUT
*----------------------------------------------------------------------*
*       FORM BUSCAR                                                    *
*----------------------------------------------------------------------*
*BUSCA LA COMPAÑIA AEREA INTRODUCIDA                                   *
*----------------------------------------------------------------------*
FORM BUSCAR.
  PAG_TOT = 0.
  INDE = 1.
  SELECT * FROM SPFLI
      WHERE SPFLI-CARRID = TABLA-CARRID.
    CLEAR TABLA.
    IF SY-SUBRC = 0.
      MOVE-CORRESPONDING SPFLI TO TABLA.
      SELECT * FROM SFLIGHT
            WHERE CONNID = SPFLI-CONNID AND CARRID = SPFLI-CARRID.
        MOVE-CORRESPONDING SFLIGHT TO TABLA.
        APPEND TABLA.
        PAG_TOT = PAG_TOT + 1.
      ENDSELECT.
    ENDIF.
  ENDSELECT.
ENDFORM.

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

```
*&      Module  STATUS_0002  OUTPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE STATUS_0002 OUTPUT.
  SET PF-STATUS 'PEPE2'.
*  SET TITLEBAR 'xxx'.
  READ TABLE TABLA INDEX INDE.
ENDMODULE.                             " STATUS_0002  OUTPUT
*&---------------------------------------------------------------------*
*&      Module  USER_COMMAND_0002  INPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE USER_COMMAND_0002 INPUT.
  CASE SY-UCOMM.
    WHEN OTHERS.
      SET SCREEN 0. LEAVE SCREEN.
  ENDCASE.
ENDMODULE.                             " USER_COMMAND_0002  INPUT
```

La lógica proceso no la pongo, porque no tiene ninguna orden de importancia, y por lo tanto es la que aparece por defecto.

Ahora mostraré la primera pantalla de introducción de datos:

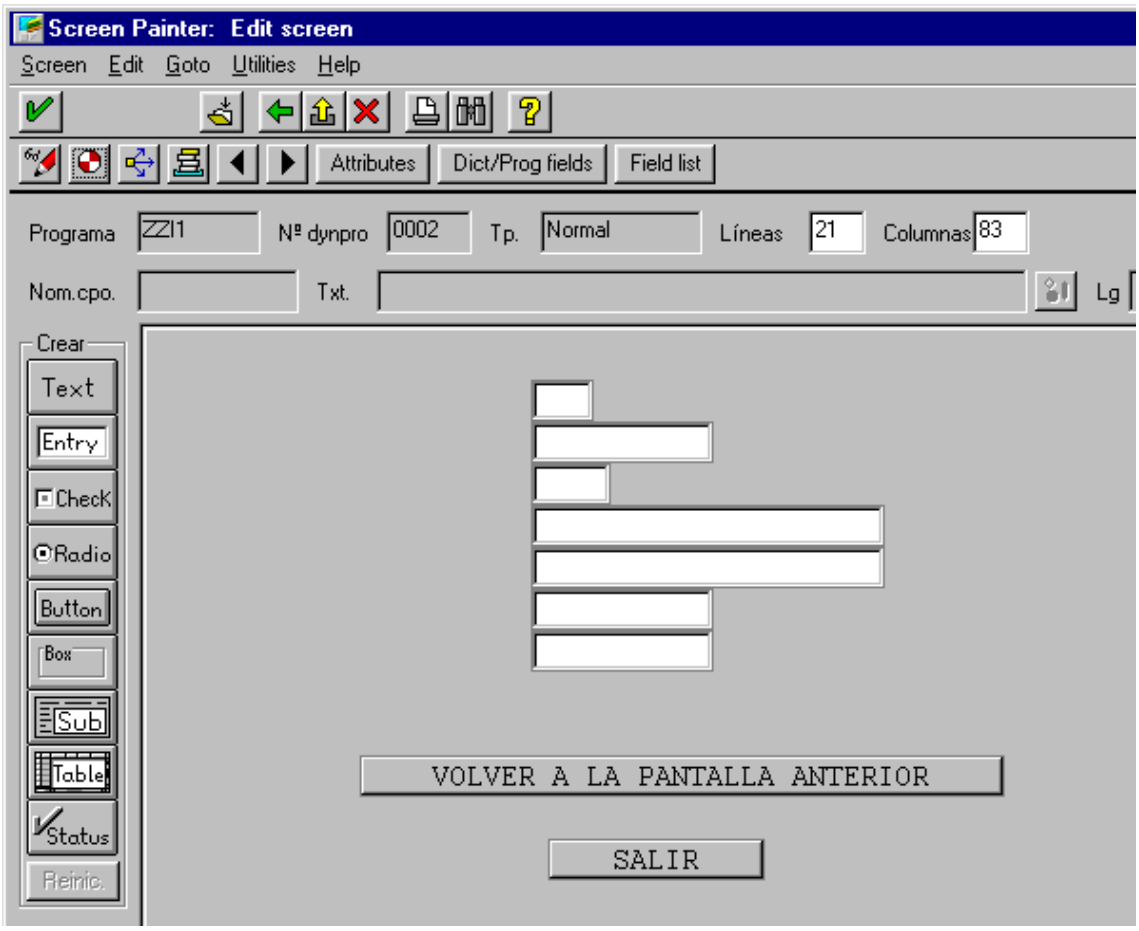


Los tres “text” tienen los siguientes nombres: TABLA-CARRID, TABLA-CONNID, TABLA-FLDATE. El primero es donde se introduce la compañía aérea.

Los dos “text” siguientes me muestran el número de vuelos totales y en que vuelo estoy.

Y los cinco botones siguientes tiene los siguientes códigos de función: BUS, ATR, ADE, VER y SAL.

La segunda pantalla sería la siguiente:



Los “text” que muestro pertenece a una tabla por lo tanto solo tenemos que insertarlos. Los dos botones tienen los siguientes código de función: ANT y SAL.

LISTADO

Este es un sencillo listado, que como el anterior realizo el listado de una compañía aérea introducida por el usuario. Y después si el usuario hace doble clic sobre una línea del listado, el programa mostrará una ventana y modificará el color de la línea seleccionada.

Al ser el programa de tipo REPORT no tiene pantallas, el listado es el siguiente:

```
************************************************************************
**
**  NOM        :  ZZJII04
**  DESCRIPCIÓ  :  LISTADO DE LOS VUELOS DE LA COMPAÑIA          **
**  AUTOR      :  INES, JOAN E IVAN                          **
**  DATA       :  04/12/1998                              **
************************************************************************
**
REPORT ZZJII4 LINE-COUNT 30
           LINE-SIZE 150
           MESSAGE-ID ZM.
************************************************************************
**
* DEFINICION DE LAS TABLAS EXTERNAS
************************************************************************
**
TABLES: SBOOK, SPFLI, SAPLANE, SFLIGHT, SCARR, SAIRPORT.
************************************************************************
**
* DEFINICION DE LAS TABLAS INTERNAS
************************************************************************
**
************************************************************************
**
DATA: BEGIN OF TABLA OCCURS 0,
       CARRID LIKE SFLIGHT-CARRID,
       FLDATE LIKE SFLIGHT-FLDATE,
       CONNID LIKE SFLIGHT-CONNID,
       FLTIME LIKE SPFLI-FLTIME,
       CITYFROM LIKE SPFLI-CITYFROM,
       DEPTIME LIKE SPFLI-DEPTIME,
       AIRPFROM LIKE  SPFLI-AIRPFROM,
       CITYTO LIKE SPFLI-CITYTO,
       NAME LIKE SAIRPORT-NAME,
       ARRTIME LIKE SPFLI-ARRTIME,
       PLANETYPE LIKE SAPLANE-PLANETYPE,
       SEATSMAX LIKE SAPLANE-SEATSMAX,
       CARRNAME LIKE SCARR-CARRNAME,
  END OF TABLA.
DATA: NOM_CIA LIKE SPFLI-CARRID VALUE 'LH'.
DATA: LINEA(2) TYPE I.
DATA: PAGINA TYPE I VALUE 0.

* LOS EVENTOS
************************************************************************
SELECT-OPTIONS: NOM_CIA FOR SPFLI-CARRID.
INITIALIZATION.
AT SELECTION-SCREEN.
START-OF-SELECTION.
    PERFORM SELECCION.
```

```
    SORT TABLA BY CARRID FLDATE CONNID.

END-OF-SELECTION.
   LOOP AT TABLA.
      AT NEW CARRID.
        PERFORM BUSCAR_COMPANYIA.
        NEW-PAGE.
      ENDAT.
        WRITE:/  TABLA-CARRID COLOR 2,
              TABLA-FLDATE COLOR 2,
              TABLA-CONNID COLOR 2,
              TABLA-FLTIME COLOR 2,
              TABLA-CITYFROM COLOR 2,
              TABLA-DEPTIME COLOR 2,
              TABLA-AIRPFROM COLOR 2,
              TABLA-CITYTO COLOR 2,
              TABLA-NAME COLOR 2,
              TABLA-ARRTIME COLOR 2,
              TABLA-PLANETYPE COLOR 2,
              TABLA-SEATSMAX COLOR 2.
* coge los datos que quiero, para cuando haga la window
       HIDE: TABLA-SEATSMAX, TABLA-AIRPFROM.
ENDLOOP.

AT LINE-SELECTION.
 IF SY-LSIND = 1. "SI ES LA PRIMERA VENTANA
  MODIFY CURRENT LINE LINE FORMAT COLOR 7. "modifico la línea escogida
  LINEA = SY-LILLI.
  SET PF-STATUS 'INES3'.
   SET TITLEBAR 'IVAN' WITH SY-LSIND. " PARA PONER EL TITULO
   WINDOW STARTING AT 2 2
        ENDING AT 85 20.   " 85 20
   WRITE: / 'PRUEBA'.
 ELSE.
 ENDIF.
  TOP-OF-PAGE DURING LINE-SELECTION.
     WRITE:/ 'TOP OF PAGE' CENTERED.
* EVENTO CUANDO EL USUARIO PULSE ALGUNA TECLA
AT USER-COMMAND.
CASE SY-UCOMM.
    WHEN 'CVEN'.
     SY-LSIND = SY-LSIND - 2.
ENDCASE.
*---------------------------------------------------------------------*
* PROGRAMA QUE SELECCIONA LOS DATOS
*---------------------------------------------------------------------*
FORM SELECCION.
    CLEAR : TABLA, SPFLI, SFLIGHT.
```

```
   SELECT * INTO CORRESPONDING FIELDS OF TABLA
         FROM SPFLI
         WHERE CARRID = NOM_CIA.
        IF SY-SUBRC = 0.
         SELECT FLDATE PLANETYPE SEATSMAX
            INTO (TABLA-FLDATE, TABLA-PLANETYPE, TABLA-SEATSMAX)
            FROM SFLIGHT
            WHERE CARRID = TABLA-CARRID AND
                CONNID = TABLA-CONNID.
           IF SY-SUBRC = 0.
            APPEND TABLA.
           ENDIF.
         ENDSELECT.
        ENDIF.
      ENDSELECT.
ENDFORM.
FORM BUSCAR_COMPANYIA.
   SELECT SINGLE CARRNAME INTO TABLA-CARRNAME
            FROM SCARR
            WHERE CARRID = TABLA-CARRID.
ENDFORM.
WRITE TEXT-001.
```

CAMPOS CURRENCY

En SAP los campos de importes se almacenan en campos de tipo **CURRENCY**. Estos campos deben hacer siempre referencia a un campo de divisa o moneda, que indica en que moneda está almacenado el importe indicado. Podemos hablar de la moneda del Documento o la de la Sociedad, incluso algunos documentos pueden contener tres monedas diferentes.
Ya es sabido que SAP almacena internamente los importes siempre con dos decimales, dividiendo o multiplicado el importe por potencias de 10 en función de los decimales que tenga la moneda. Para su representación en pantalla disponemos de la instrucción **WRITE *Importe* CURRENCY *Moneda.***
[Importe] => Variable que contiene un Valor. *[moneda]* => Variable que contiene el código de la moneda que estamos tratando, moneda del documento, de la sociedad,...

Esta instrucción hace el desplazamiento de los decimales necesarios en función de la moneda indicada, representando el valor con el numero de decimales de la moneda, por lo que no hace falta ningún tratamiento especial.

La instrucción ***WRITE*** *Importe* ***TO*** *Variable* ***CURRENCY*** *Moneda* no funciona correctamente, no se debe usar, ya que el resultado que da es un valor que no tiene ninguna relación con el importe indicado y la moneda, el resultado que da tiene la apariencia de un **owerflow**.

El principal problema aparece cuando hay que operar con los importes, acumular comparar y presentar totales.
Para la mayoría de las operaciones no es necesario realizar ninguna conversión, pero se debe tener presente que los importes con los que estamos operando, sean todos de la misma divisa.

Variables Intermedias

Si se utilizan variables intermedias para la realización de cálculos con importes, estas variables deben ser de tipo CURRENCY si no se ha realizado ninguna conversión a los valores que en ella se almacenan, o de tipo P con 4 decimales si se ha realizado la conversión a formato Real. Se utilizaran 4 decimales, por que es el numero máximo de decimales que puede tener una moneda. El tamaño del campo dependerá del valor máximo que se deba almacenar en él. Es conveniente dimensionarlo en una o dos posiciones mas como precaución.
Se recomienda operar, siempre que sea posible, con campos CURRENCY en lugar de realizar conversiones a formato real, y convertir solo en caso de necesidad.

Tipos de Operaciones

Para la mayoría de operaciones no es necesario realizar ninguna operación de transformación. Solo se debe tener presente que los campos sean de la misma moneda.

Para las ***comparaciones*** se debe tener presente la existencia de algunos campos de tablas que almacenan importes pero que no son en formato **CURRENCY**, por lo que

haría falta convertir el valor de la tabla a formato CURRENCY o el que está en formato CURRENCY convertirlo en formato real. Se recomiendo convertir el valor de la tabla a formato CURRENCY.

Para el tema de la ***impresión*** de datos por pantalla o impresora, siempre que se use un campo de tipo CURRENCY se debe utilizar la sentencia *(**WRITE** Importe **CURKENCY** Moneda),* informando en la variable moneda el tipo de moneda que se está tratando, sino se deberá convertir el valor al formato real.

Para el tema de los *Batch Input* (Siempre que sea necesario realizar un **Call Transaction** con tabla **BDC** o Prepara un Juego de Datos), si se debe enviar un importe se debe enviar el formato real en un campo de tipo CHAR, controlando los decimales que soporta cada moneda. Para realizar la conversión a formato Real en una variable de tipo **Char** con los decimales necesarios se ha desarrollado una función especial. Ver **Funciones de conversión**.

Funciones de Conversión

Para la realización de todas las conversiones de formato, **SAP=>REAL, REAL=>SAP y SAP=>CHAR**.
Se han desarrollado tres funciones especificas, que son:

1. Z_XX_IMPORTE_SAP_TO_REAL:

Esta función a partir de los parámetros de entrada: MONEDA e IMPORTE_SAP, nos

Devolverá el importe en formato REAL. La variable que nos devolverá será de tipo P con 4 decimales.

```
call function 'Z XX IMPORTE_SAI_TO_REAL'
        EXPORTING
                MONEDA              = Var_moneda
                IMPORTE_SAP         = Var_importe
                SOCIEDAD            = Var_sociedad (Opcional)
        IMPORTING
                IMPORTE_REAL   = Var_Importe_Real
         EXCEPTIONS
                 falta_moneda       = 1
                sociedad_no_existe  = 2
                others              =3.
```

El Campo de sociedad es opcional, y solo se tendrá presente en caso de que no se indique el tipo de moneda, entonces, será necesario informar la sociedad para obtener la moneda de la sociedad.

2. Z_XX_IMPORTE_REAL_TO_SAP:

Esta función a partir de los parámetros de entrada: MONEDA e IMPORTE_REAL, nos devolverá el importe en formato SAP. La variable que nos devolverá será de tipo **CURRENCY**.

```
call function 'Z XX_IMPORTE_REAL_TO_SAP
     EXPORTING
          MONEDA          =Var_Moneda
          IMPORTE_REAL    = Var_Importe
          SOCIEDAD        = Var_Sociedad
     IMPORTING
          IMPORTE_SAP     = Var_Importe_SAP
     EXCEPTIONS
          falta_moneda    = 1
          sociedad_no_existe = 2
          others          =3.
```

El Campo de sociedad es opcional, y solo se tendrá presente en caso de que no se indique el tipo de moneda, entonces, será necesario informar la sociedad para obtener la moneda de la sociedad.

3. Z_XX_IMPORTE_SAP_TO_CHAR:

Esta función a partir de los parámetros de entrada: MONEDA e IMPORTE_SAP, nos devolverá el importe en formato **CHAR**

```
Call function ‘ZZ_XX_IMPORTE_SAP_TO_CHAR’
     EXPORTING
          MONEDA              = Var_Moneda
          IMPORTE_REAL        = Var_Importe
          SOCIEDAD            = Var_Sociedad (Opcional)
     IMPORTING
          IMPORTING_CHAR      = Var_Importe_Char_real
     EXCEPTIONS
          falta_moneda        = 1
          sociedad_no_existe  = 2
          others              =3.
```

El Campo de sociedad es opcional, y solo se tendrá presente en caso de que no se indique el tipo de moneda, entonces, será necesario informar la sociedad para obtener la moneda de la sociedad.

4. Z_XX_IMPORTE_REAL_TO_CHAR:
Esta función a partir de los parámetros de entrada: MONEDA e IMPORTE_REAL, nos devolverá el importe en formato CHAR

```
call function 'Z XX IMPORTE_REAL_TO_CHAR'
     EXPORTING
          MONEDA              = Var_Moneda
```

```
        IMPORTE_REAL          = Var_Importe
        SOCIEDAD              = Var_sociedad (Opcional)
IMPORTING
        IMPORTE_CHAR          = Var_Importe_Char_Real
EXCEPTIONS
        falta_moneda          = 1
        sociedad_no_existe    =2
        others                =3.
```

El Campo de sociedad es opcional, y solo se tendrá presente en caso de que no se indique el tipo de moneda, entonces, será necesario informar la sociedad para obtener la moneda de la sociedad.

Recomendaciones

- Se debe tener muy clara la diferencia entre la moneda de la sociedad (FI, CO) y la moneda del documento, no mezclando los dos conceptos, y siempre que se dude al escoger un campo de una tabla, se debe consultar con el analista.

- Siempre que se opera con importes ha de existir una moneda asociada a dicho importe, si no se sabe que moneda es, preguntar lo al analista. Las tablas internas que contengan importes deberán contener un campo con la moneda a la que hace referencia el importe indicado, y poner comentarios clarificadores que indiques si la moneda que se usa es la de la sociedad o la del documento.

- En ningún caso se debe introducir código fuente en los programas que realicen un tratamiento especial en función del tipo de moneda que se está procesado. En caso de que fuera absolutamente necesario, se deberá consultar con el analista encargado del desarrollo y con los encargados de desarrollo del proyecto.

Ejemplos No correctos:

WRITE *Var_Importe* CURRENCY *'PTS'. =>No es correcto.*

```
IF Var_Moneda = 'PTS'.
   Var_lmporte=Var_lmporte * 1OO.
Endif.                                  => No es correcto.
```

ERRORES DEL SISTEMA

Ya sabemos que la variable SY-SUBRC controla los errores del sistema, lo valores que puede tomar depende de la instrucción que utilizamos, por ello voy a dar que error da en las siguientes instrucciones:

- Cuando hacemos operaciones con tablas ya sean de diccionario o internas:

  0 -> No hay errores.
  2 -> Se ha producido algún error ya sea a leer, añadir, borrar o modificar.

- Cuando realizamos un algún SCROLL, con la orden SCROLL:
  0 -> OK
  4 -> Límite de la lista rechazado. Scrolling imposible
  8 -> Lista no existe. Scrolling imposible

- Cuando leemos un fichero secuencial:
  0-> No ha llegado al final del fichero.
  4-> Ha llegado al final del fichero.

- Cuando hacemos la orden OVERLAY:
  0->La sustitución ha sido un éxito.
  4->Cuando STR1 es mayor que STR2

- Cuando realizamos la orden SEARCH:
  0->Ha encontrado algo
  4->Ha ocurrido cualquier otra cosa.

NOMENCLATURA DE SAP

CO-CCA /CO-PA /FI / HR/CO-PC

Z m s xy nn v

m = IDENTIFICACION MODULO SAP
X = Base, común a todos los módulos (Rutinas,,,,)
Y = Include

I = Plant Maintence

K = CO
F = FI
P = HR
Z = prueba no asociada a ningún módulo

S = IDENTIFICACION SUBMODULO

X = Base, común a todos los modulos (Rutinas,,,,)

CO	**E** = CO-PA	
	S = CO-CCA	
	K = CO-PC	
FI	**E** = FI-GL	CONTABILIDAD EXTERNA
	D = FI-AR	DEUDORES
	A = FI-AP	ACREEDORES
	T = FI-TR	TESORERIA (+CARTERA)
	B = FI	SISTEMA DE BASE FI
	I = FI-AA	ACTIVOS FIJOS
HR	**M** = HR-	DATOS MAESTROS
	N = HR-	ADMINISTRACION NOMINA
	P = HR-	PLANIFICACION DE PERSONAL
	B = HR-	SISTEMA DE BASE HR

x = NATURALEZA DEL TRATAMIENTO

A = Apoyo
C = Conversión
E = Enlace
F = Fusión
G = General
I = Inflac
K = Empleo de capitales
P = Plan
R= Real
T = Rutina (Utilizada via include)

W = EW1/Ew2/plan

Y = ACTUALIZACION DATOS (actualización tablas, grabación lotes,etc)

Si x = K

A = Apoyo
C = Conversión
E = Enlace
F = Fusión
G = General
P = Plan
R = Real
T = Rutina (Utilizada via Include)

Sino

A = Actualiza datos
X = No actualiza datos

nn = NUMERADOR CORRELATIVO (numerar de 00 a 99, A0 a ZZ)

v = VERSION (numerar de 0 a 9)

SD / MM/PM

Z m s xy nnn v

m =IDENTIFICACION MODULO SAP
X = base, común a todos los módulos (Rutinas,,,)
I = Plant maintenace
V = SD
M = MM/PP/PPI
W = WM
Z = prueba no asociada a ningún módulo

s = IDENTIFICACION SUBMODULO

SD	**V** = Ventas
	D = Distribución
	F = Facturación
	C = Clientes
	K = Condiciones de venta
	I = Programas EDI
MM	**M** = Maestro materiales
	S = Gestión de stocks e inventario

V = Valoración de stocks
C = Compras
A = Autofacturación
F = Fórmulas/Rutas/Recetas
P = Producción
E = Explosiones e Implosiones
B = Datos Maestros (Cuotas, Ordenes de serie,...)

WM **S** = Listados ubicaciones,materiales,lotes...
B = Batch Input
P = Pre-picking
E = Entrada Ord. Transporte
O = Salida Ord. Transporte

PM **M** = Datos Maestros

X = NATURALEZA DEL TRATAMIENTO

C = Conversión
E = Enlace (Interfases...)
F = Fusión
G = General
T = Rutina (Utilizada via include)
I = Impresión de documentos (factura,albaran,...)

nnn = NUMERADOR CORRELATIVO (numerar de 000 a 999)

V = VERSION (numerar de 0 a 9)

ICONOS DE SAP

SAP tiene unos iconos (grabar, compilar, generar, etcétera) que siempre son los mismos, no importa en que pantalla estemos.

Los iconos son los siguientes:

 → Grabar sin verificar o F11.

 → Verificar o Ctrl+F2

 → Activar o generar un objeto

 → Cambiar el modo de visualización/modificación

 → Borrar un objeto de SAP (tablas, programas, etcétera)

 → Seleccionar valores de un campo, es lo mismo que pulsar F4 sobre un campo. Más conocido como MATCHCODE

 → Permite volver a la pantalla anterior o F3

 → Finaliza la ejecución de cualquier proceso o SHIFT+F3

 → Cancela la ejecución de cualquier proceso o F12

 → Si pulsas este botón, es lo mismo que pulsar ENTER

 → La fantástica ayuda de SAP o F1

 → Imprimir un documento de SAP o CTRL+P

 → Buscar cualquier cosa en SAP

 → Copiar un objeto a otro o CTRL+F5

Renombra el nombre de un objeto o SHIFT+F6

Ref. utilización, que programas utilizan un objeto
CTRL+SHIFT+F3

Ejecutar algún proceso de SAP o F8, también para buscar algún elemento.

Ayuda sobre la imagen seleccionada o manual online o
CTRL+SHIFT+F6

Trae una variante a la pantalla donde estemos

Saca la opciones de la pantalla donde estemos

Permite realizar selecciones múltiples

Crear un objeto nuevo o F7

Modificar un objeto o F6

Visualizar un objeto o F5

Transporte, envía un objeto a transporte

Herramientas, para reparar algún objeto

Ver o seleccionar una variante de un programa

NOTA DEL AUTOR

Todos los ejemplos aquí explicados están realizados en la versión: SAP FRONTED 3.1H.

Además, las explicaciones primero han sido realizadas en SAP y después explicadas, por lo tanto todos los ejemplos si se realizan bien, funcionan correctamente.

Hay que decir, que los ejemplos aquí explicados, los he intentado explicar y realizar de la forma más sencilla posible. Pero en SAP como las cosas se pueden realizar de varias formas diferentes (confirmado por mi propia experiencia) seguro que encontraréis alguna forma más corta y sencilla de realizar los ejemplos aquí explicados.

Muchos y muchas advertiréis que hay comentarios y explicaciones muy personales, que según mis compañeros le dan un toque más personal al manual.

Y quiero dejar muy claro, que este manual solo contiene la explicación de una pequeña parte que ofrece SAP, por ello este manual nunca se estancara sino todo lo contrario crecerá y crecerá hasta convertirse en un buen manual.

El autor de este manual es un creyente convencido de que la programación es un "arte experimental";nada vale nada, en particular la verborrea típica en la literatura de SAP ABAP/4, si no se prueba en la máquina. Por ello, todas las preguntas planteadas deben experimientarse en máquina y, sobre todo, las respuestas han de verificarse en máquina para ver si son correctas.

AGRADECIMIENTOS

Este manual no habría sido posible sin la inestimable ayuda de mis compañeros de curso.

Este soy yo, Iván Rodrigo. El Autor del manual.

Esta chica tan maja, es Inés Campo. Mi mano derecha, sin ella la imagen del manual no hubiera sido la que es.

Esta es la matemática, Montse Martínez. La mano izquierda, gracias a ella el manual tiene tantas páginas.

Este chicarrón es Víctor Lorente. Él ha sido el corrector ortográfico y léxico de este manual.