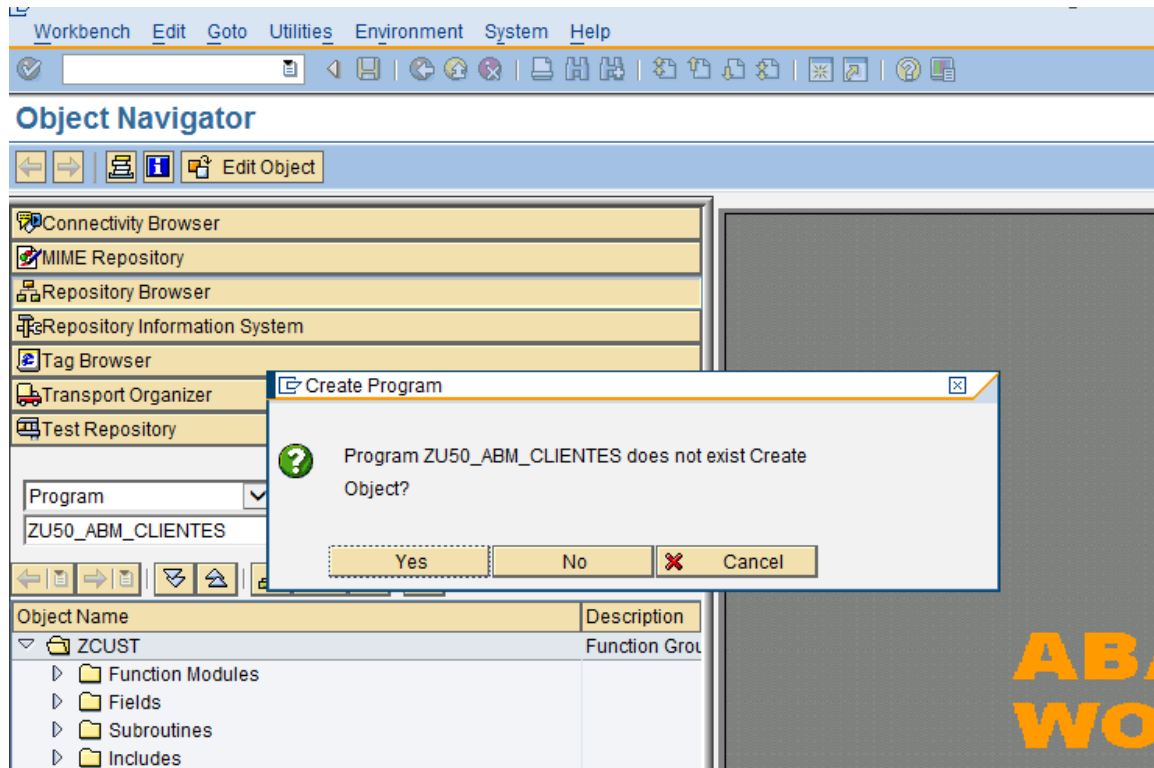



# ABAP Transaction

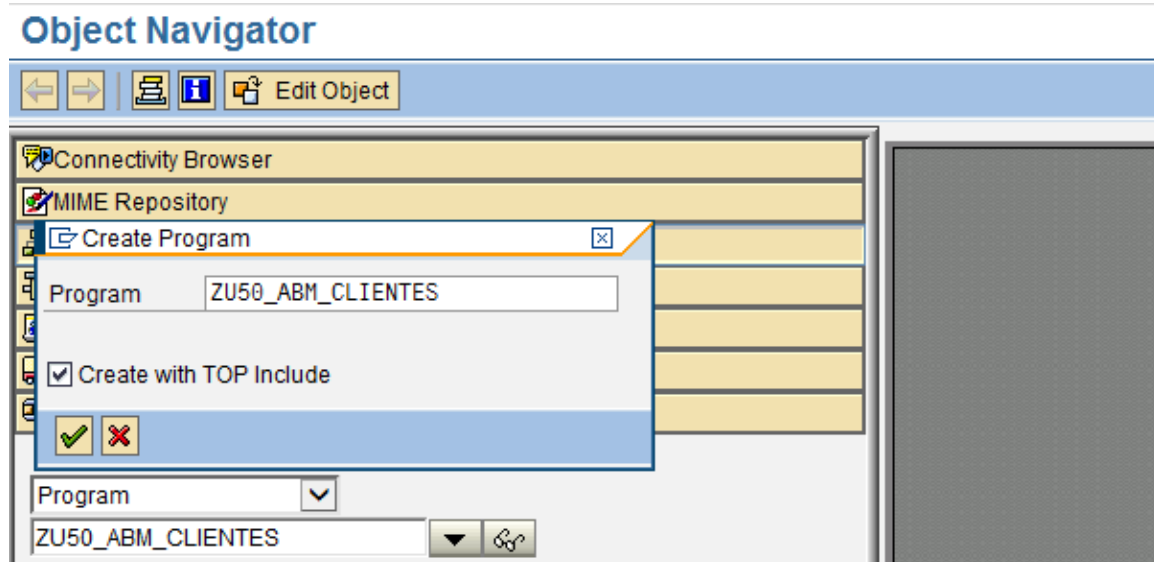
Creando un programa de Dialogo


Vamos a crear un programa de Dialogo (Transaction), usando la Transacción SE80.  
Por ejemplo el programa **ZU50\_ABM\_CLIENTES**.

Ingresamos el nombre y presionamos 

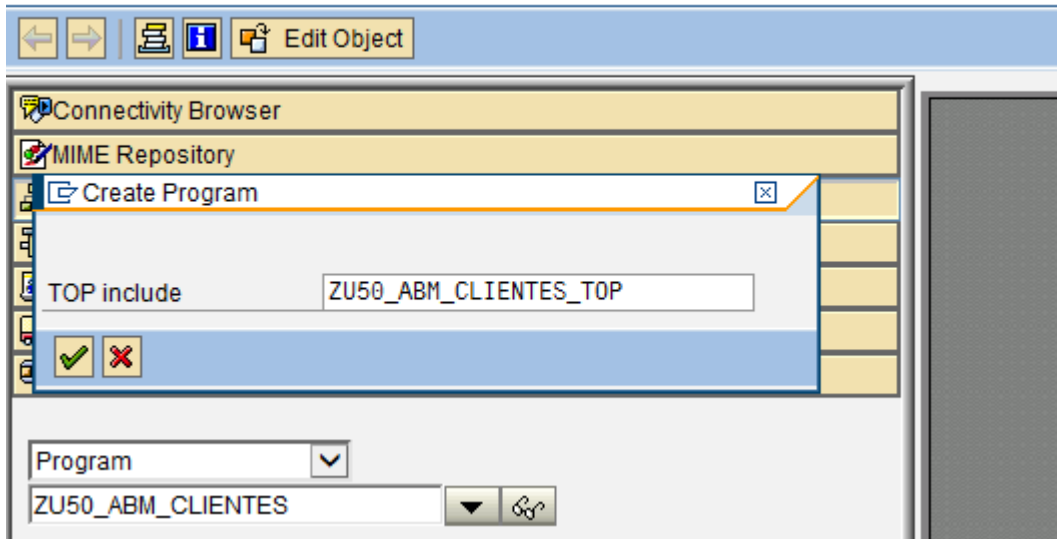


Luego nos aparece lo siguiente, la se80 detecta que es de Dialogo y propone crear un programa TOP dentro del programa principal, que usaremos para declarar variables globales, presionamos 

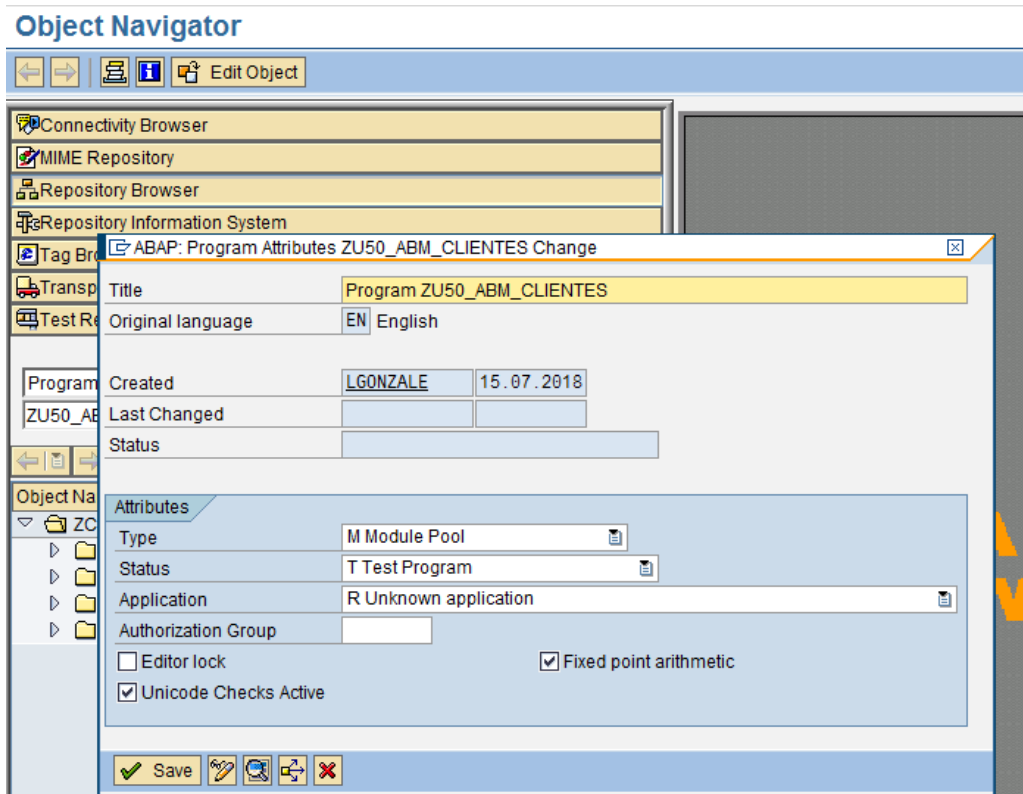


Presionamos  nuevamente (Aquí propone realmente crear el TOP)

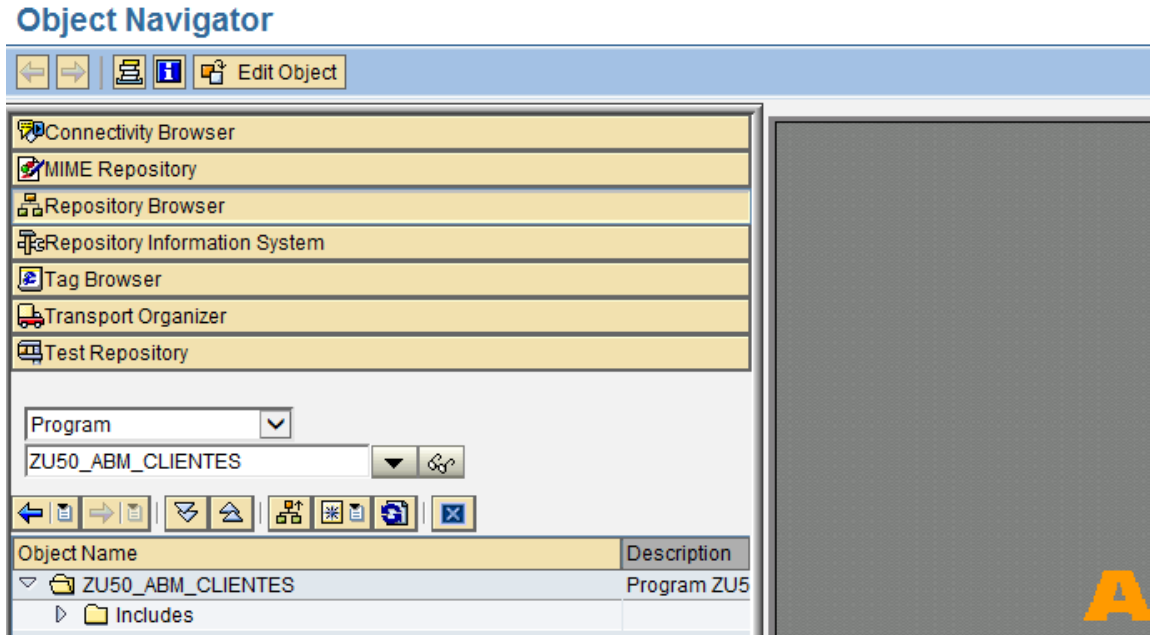
## Object Navigator



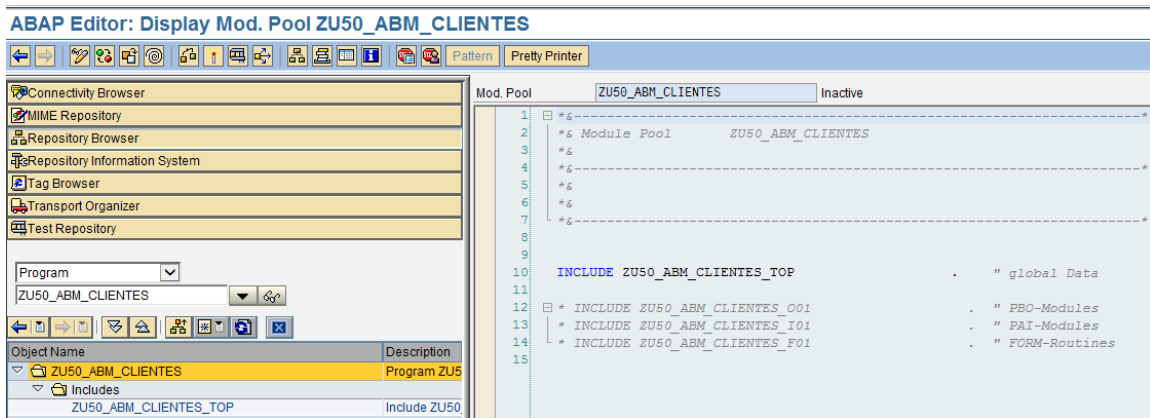
Le ponemos un Titulo al programa, debemos elegir el **Type** del programa que al ser un dialogo se le pone MODULE POOL (hay muchas formas de llamar la programación en Dialogo -> *Module Pool*, *Transaction* son las mas comunes), en **Status** le ponemos que es de TEST, y en **application** “Unknown Application” debido a que estamos haciendo un ejemplo, en el caso que se les pida hacer un dialogo para compras o ventas o finanzas, le pondrán en Application el que corresponda, esto es parte de los atributos del dialogo y sirve simplemente para ser mas informativo.






Presionamos SAVE y nos queda la siguiente pantalla

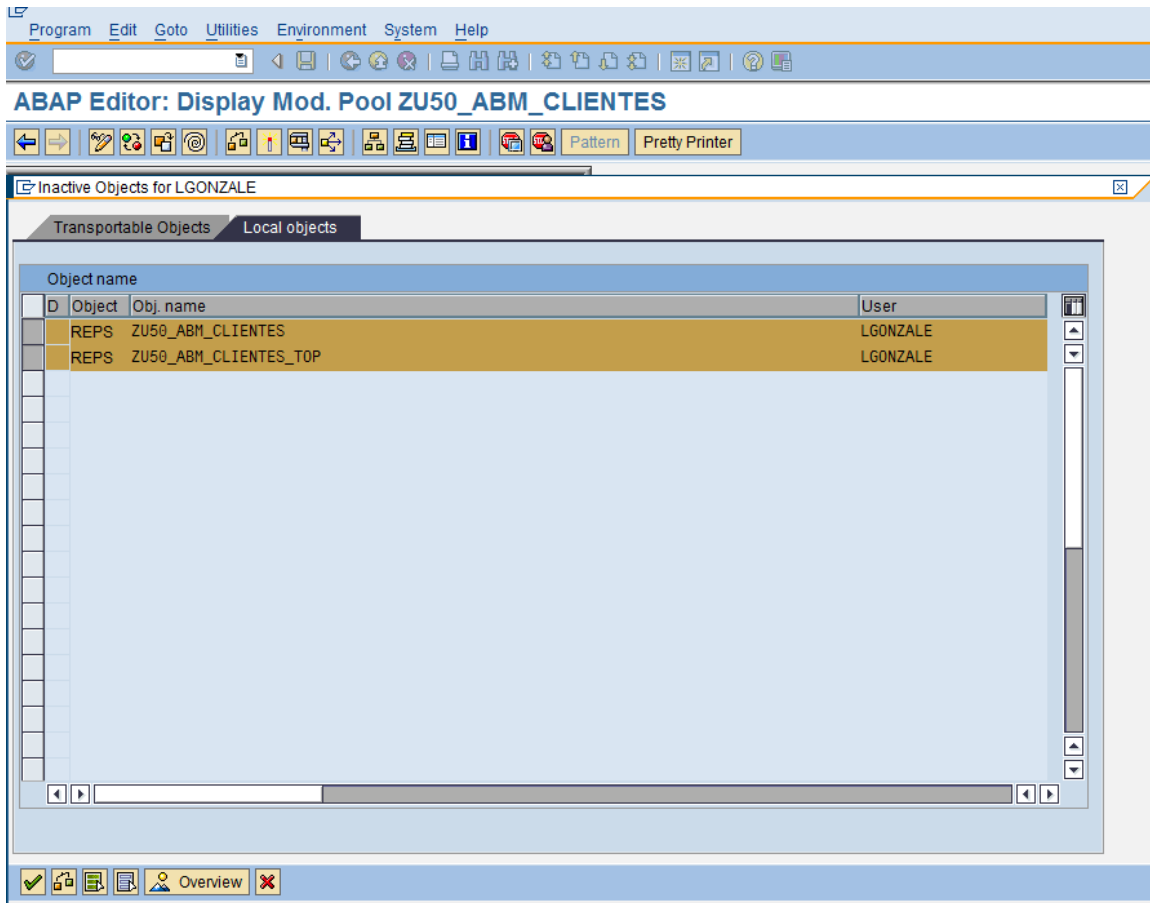


Se ve que ZU50\_ABM\_CLIENTES esta “Inactive”, eso es porque el programa todavía no esta ACTIVO, el dialogo tiene que estar siempre activo para luego si poder ejecutarlo.



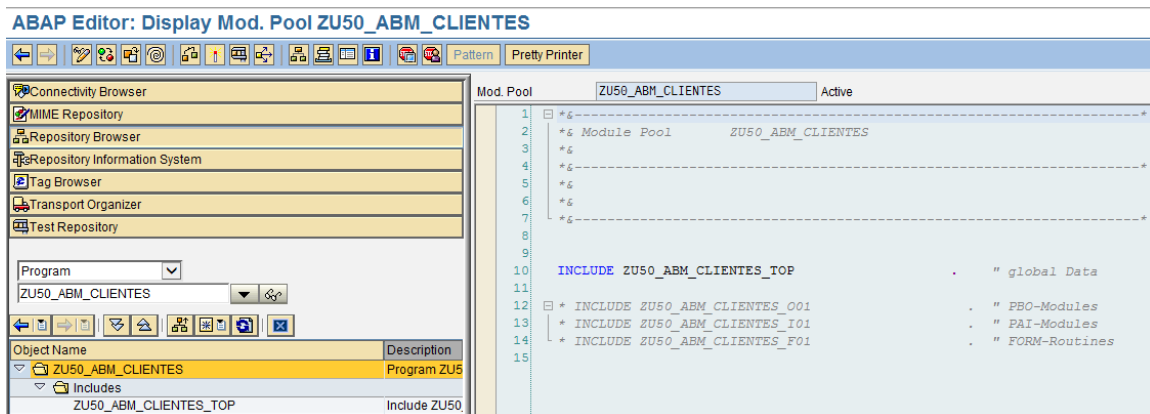
Se ve que esta Inactivo y dentro de ZU50\_ABM\_CLIENTES tenemos un INCLUDE ZU50\_ABM\_CLIENTES\_TOP, este include nos lo creo automáticamente al principio, y dentro de este include nosotros definiremos las tablas y variables globales que usaremos en todo el programa de dialogo, pero sin hacer nada dejemos todo activo, se hace haciendo doble click sobre el Include y Activándolo.

Estamos dentro del include y vemos que también esta inactivo, activémoslo  y volvamos hacia atrás con  pantalla de activación, le damos 



y vemos que queda activo. Ahora volvemos hacia atrás con

Sigue estando inactivo, bueno ahora lo activamos y no esta mas “AZULADO”, veremos mas adelante que cualquier objeto que vamos a crear en dialogo si esta azulado es porque esta inactivo y debemos siempre dejarlo activo antes de probar nuestro Dialogo.

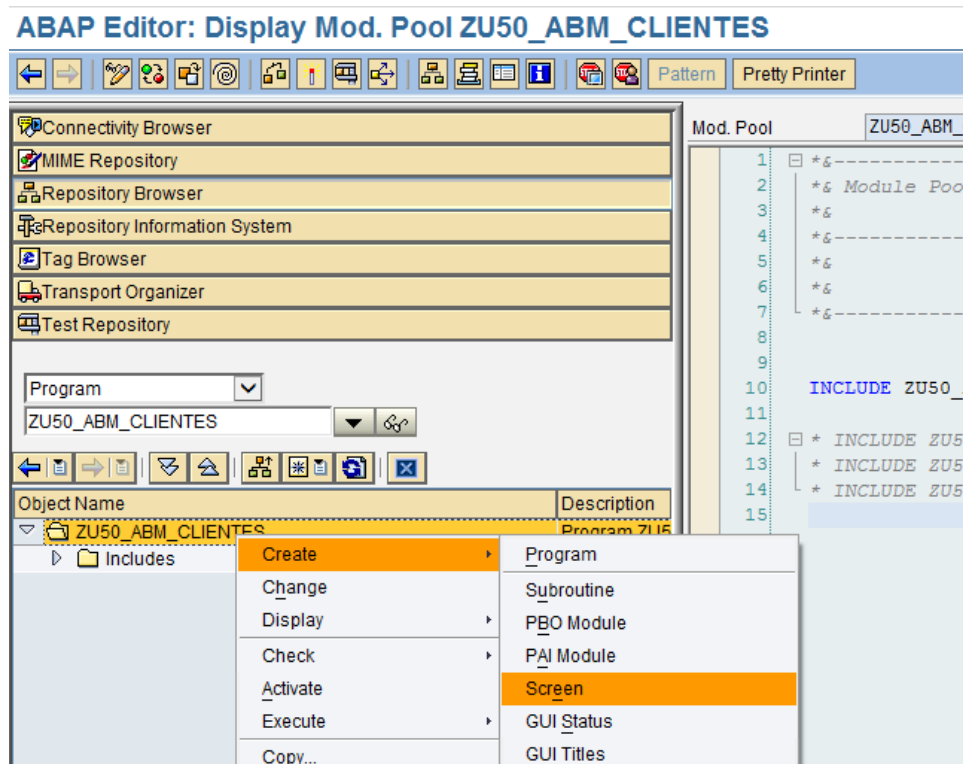


Bien ahora esta todo activo, pero en realidad no programamos nada, así que el programa no va hacer nada, lo primero que vamos a hacer es crear 3 **Screens (Dynpros)**, las Screen son las pantallas, nos permiten meter botones, campos, visualizar la información y navegar de una a otra.

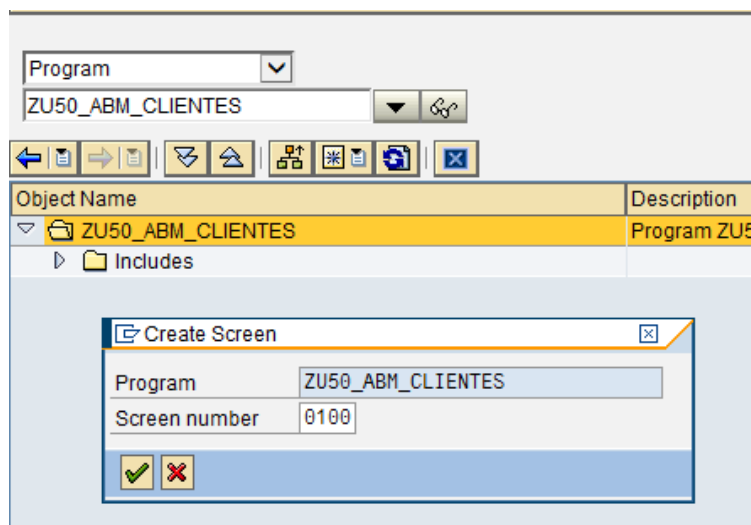
La primer Screen (0100 se les pone números a cada screen) tendrá 2 botones, cada botón tiene un código asociado, que cuando se presione el programa hará un IF y vera cual código se ha presionado y hará lo que tenga que hacer, a estos códigos se los llama OKCODE o SY-UCOMM.


Si presiona un botón ira a la screen 200 y si presiona el otro ira a la 300.  
Pero primero debemos crear la screen 0100 y los botones.

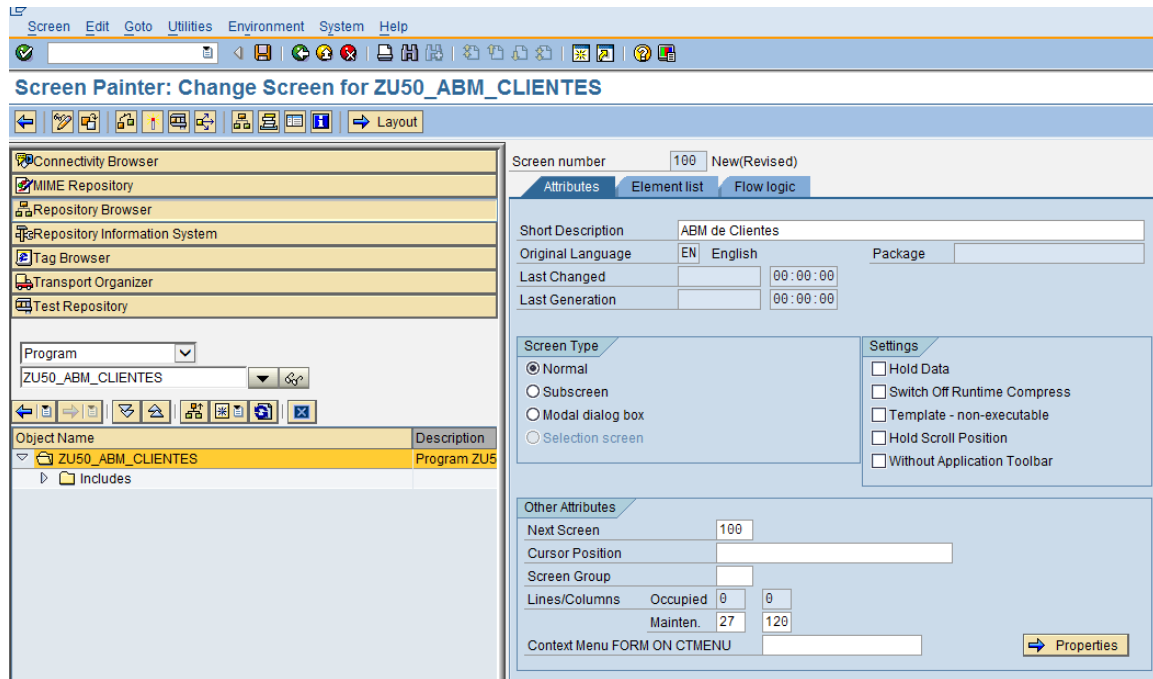
Haciendo click derecho sobre ZU50\_ABM\_CLIENTES (Ahora en negrita) -> Create -> Screen




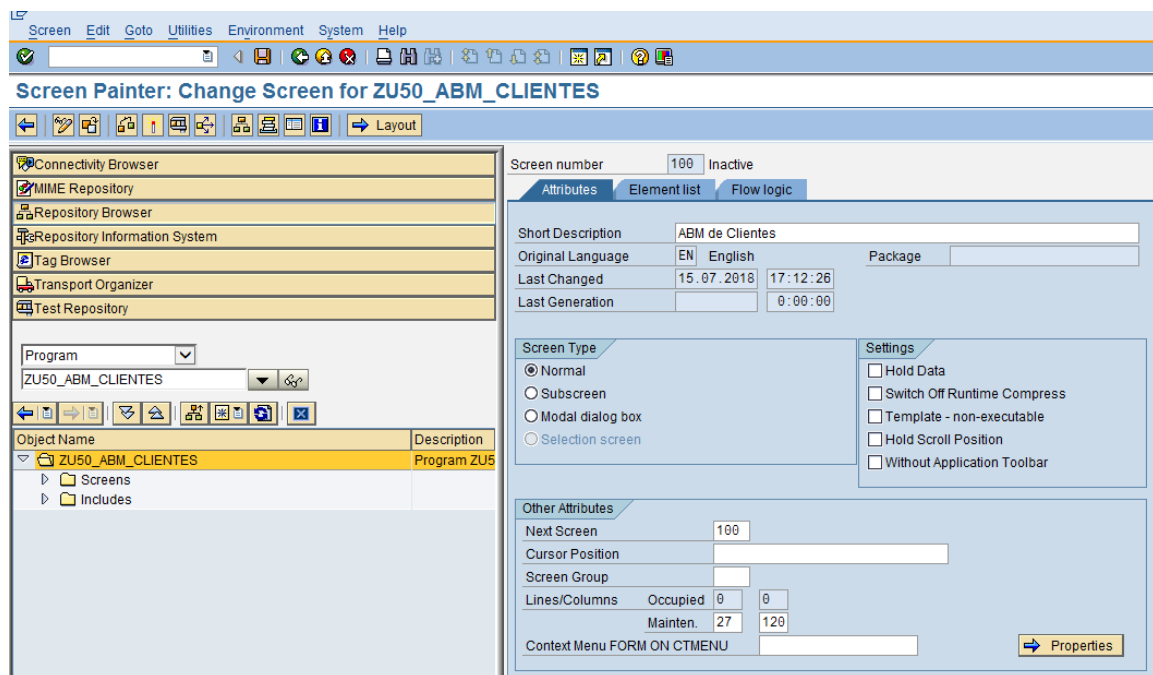
Le asignamos la screen 0100, le damos 



le ponemos un titulo a la screen 100, el tipo de screen es NORMAL, en otros ejemplos veremos como una screen puede ser definida como una Subscreen (y en este caso ser parte de una Screen Normal), luego grabamos 



Vemos que se creo un submenu de Screens, y la screen esta Inactive. Activémosla 



Vemos estas 3 solapas

Screen number  Active

Attributes Element list Flow logic

Short Description

Original Language  English

Last Changed

Last Generation

La de Atributos ya la conocemos, recién estuvimos trabajando, pasemos a la *de Element List*, en esta podemos encontrar todos los elementos creados para esta Screen, no tenemos ninguno, pero podemos ver que hay una línea creada con TYPE **OK** esperando ser ingresada.

Screen Painter: Change Screen for ZU50\_ABM\_CLIENTES

Connectivity Browser

MIME Repository

Repository Browser

Repository Information System

Tag Browser

Transport Organizer

Test Repository

Program

Object Name Description

ZU50\_ABM\_CLIENTES Program ZU5

Screens

Includes

Screen number  Active

Attributes Element list Flow logic

General attr. Texts/I/O templates Special attr. Display attr. Mod. groups / functions References

H	M	Name	Type	Li	C	D	Vi	H	Sc	Format	In	O	Out	Dj	Dict	Property list
			OK	0	0	20	20	1		OK						

Aquí es donde DIALOGO guarda el código de salida de la Screen 0100, es decir cuando presionemos un boton, ese boton va a tener asociado un código de salida y ese código de salida será almacenado en esta variable que usualmente se le pone OKCODE. Y luego lo activamos. (OKCODE pasa a ser ahora una variable, que en realidad también debemos declararla como de texto, lo haremos mas adelante)

Screen Painter: Change Screen for ZU50\_ABM\_CLIENTES

Connectivity Browser

MIME Repository

Repository Browser

Repository Information System

Tag Browser

Transport Organizer

Test Repository

Program

Object Name Description

ZU50\_ABM\_CLIENTES Program ZU5

Screens

Includes

Screen number  Active

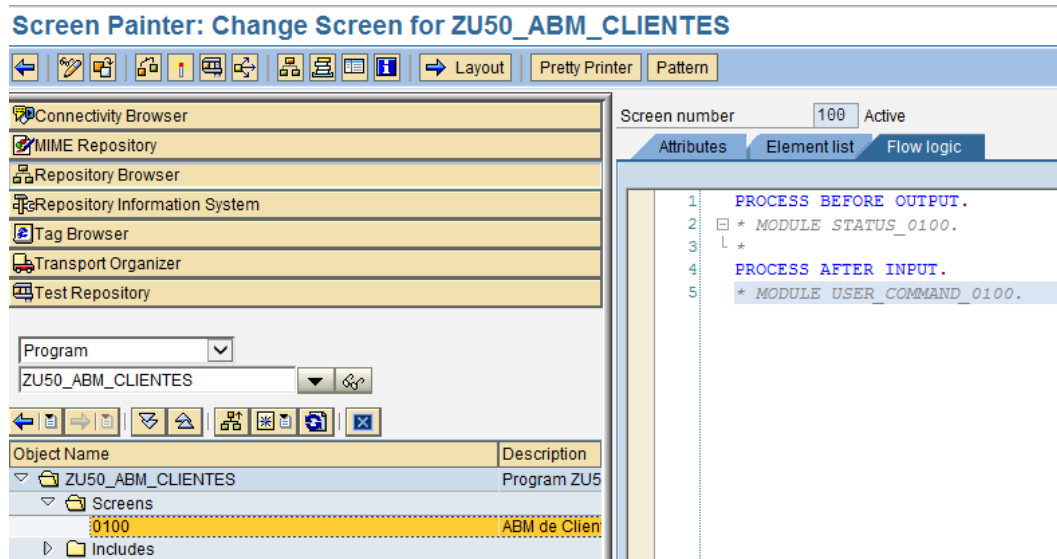
Attributes Element list Flow logic

General attr. Texts/I/O templates Special attr. Display attr. Mod. groups / functions References

H	M	Name	Type	Li	C	D	Vi	H	Sc	Format	In	O	Out	Dj	Dict	Property list
		OKCODE	OK	0	0	20	20	1		OK						⇒ Properties

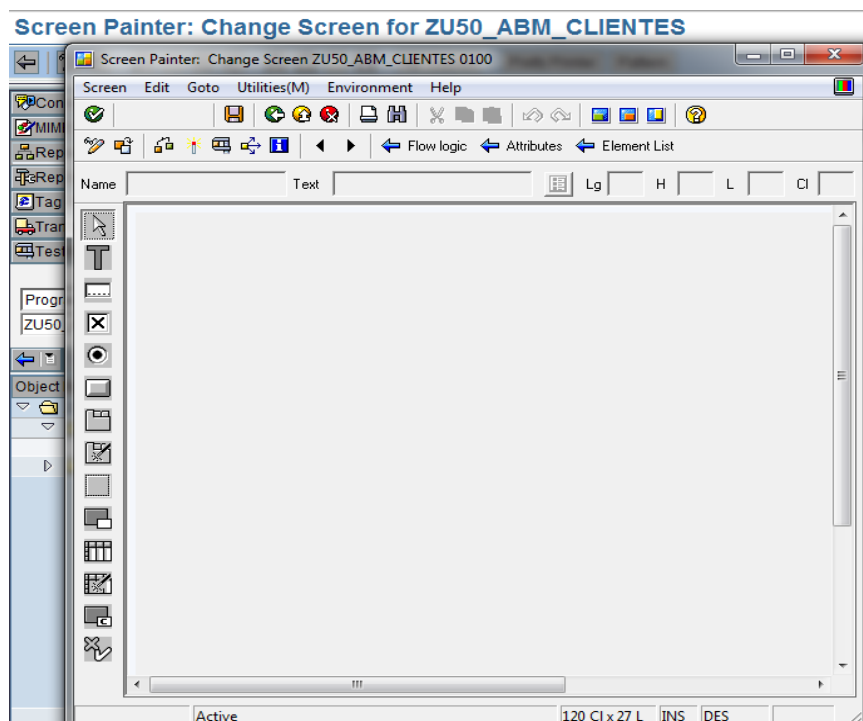



La otra solapa es la de Flow Logic, donde tenemos los procesos a ejecutar, son 2 instancias, cuando llamemos a la screen 100, antes de mostrarla se ejecutaran los PROCESS BEFORE OUTPUT, aqui podremos preparar la información que queramos mostrar, como selects a tablas y demás cosas, y al ejecutar una acción ya estando en la Screen 100, esa acción saldrá con un código de salida guardado en el OKCODE, ese OKCODE será analizado o procesado en el PROCESS AFTER INPUT, no hay ningún proceso activado, se puede ver que esta con “\*” los dos MODULES.




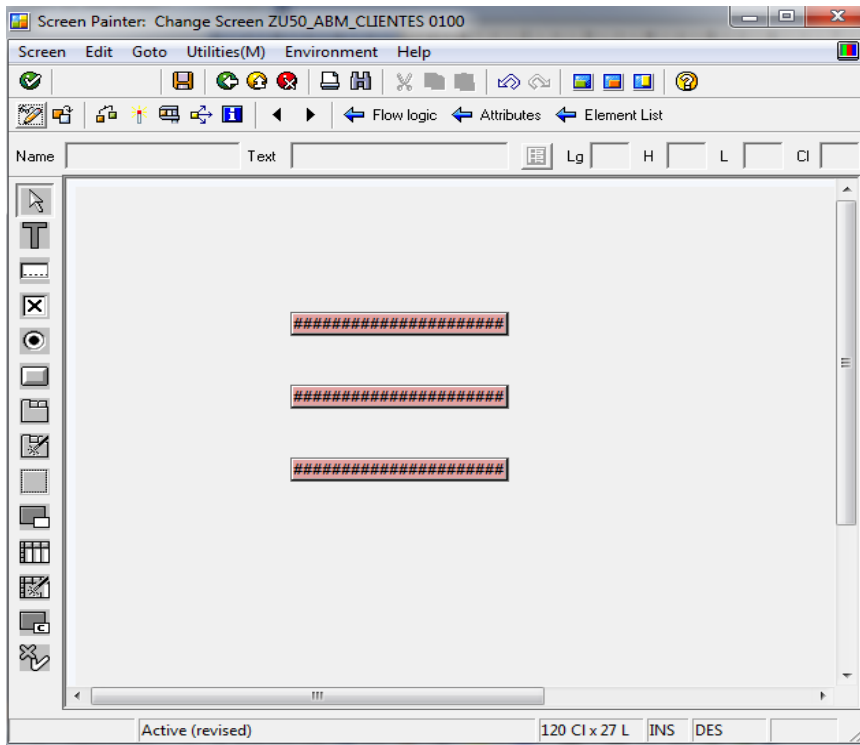
Comencemos a diseñar la screen 100, vamos a ponerle 3 botones, uno llamado “Alta de Cliente”, ”Modificación de Cliente” y otro llamado “Baja de Cliente”

Debemos presionar  Layout.

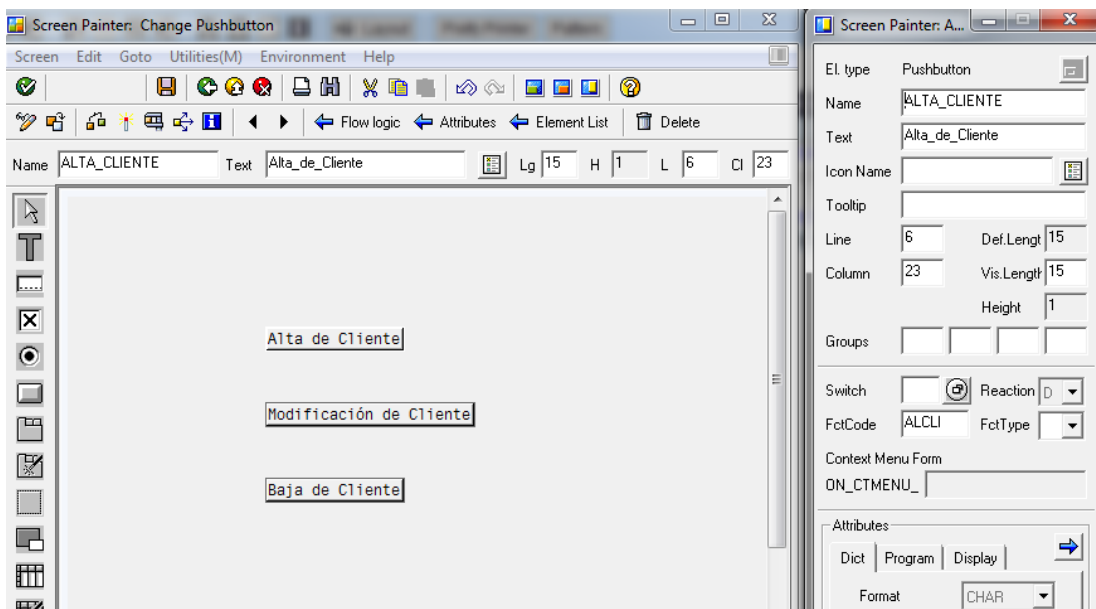


Luego con el boton  estamos en condiciones de agregar objetos a la screen 100.

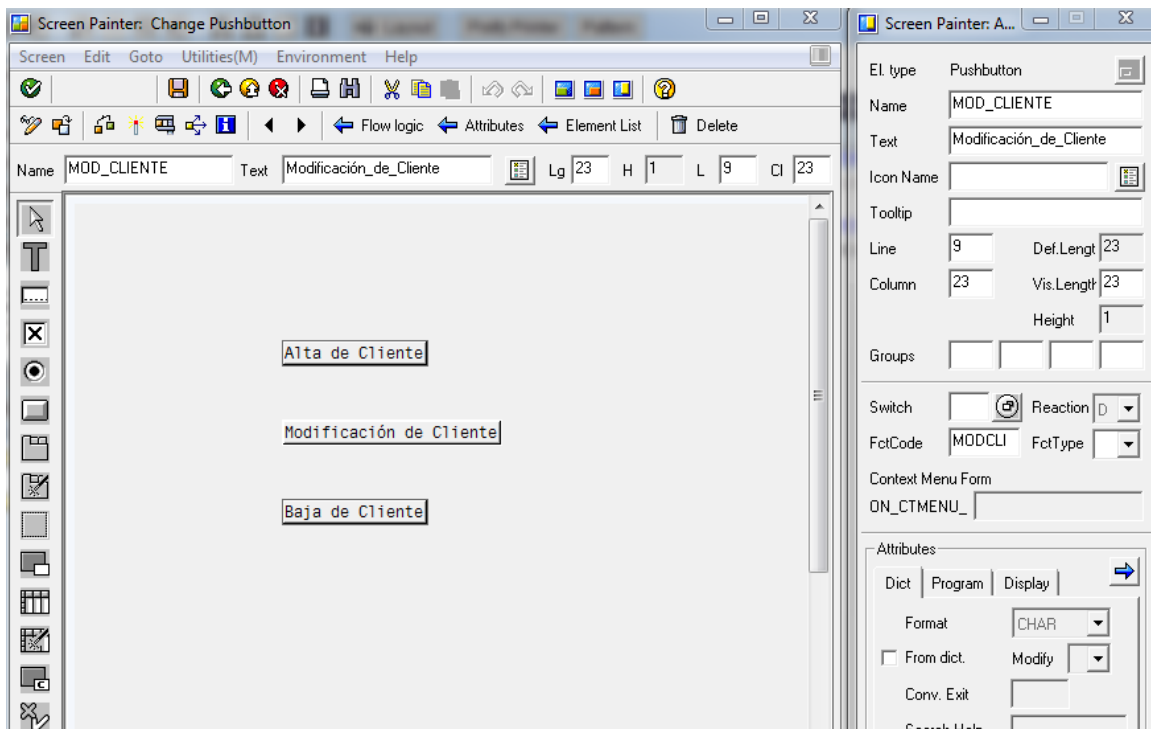
Luego haciendo un click sobre el objeto “Boton”  y soltándolo, inmediatamente movemos el mouse para darle ubicación, luego clickeando nuevamente se ubica el botón, luego hacemos lo mismo para crear el 2do botón.



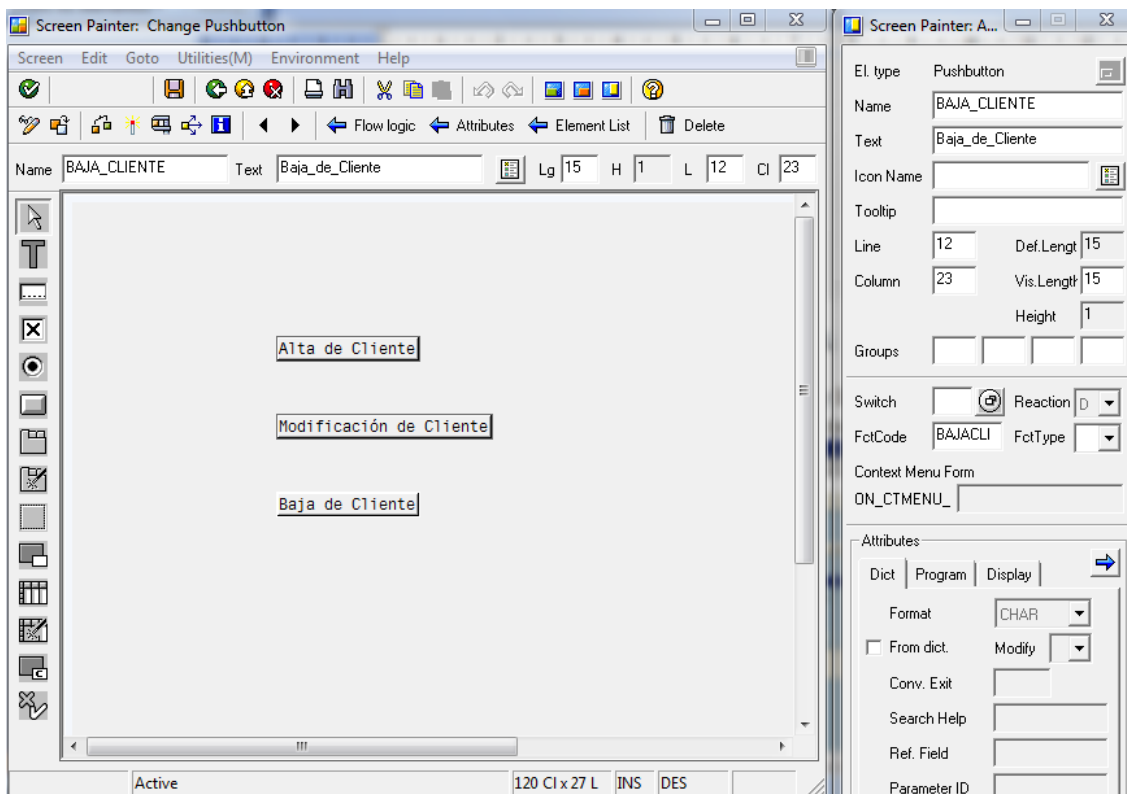
Ahora le daremos un nombre a cada botón, haciendo doble click en cada uno de ellos y llenando los atributos.



En FctCode ingresamos **ALCLI**, este va a ser el código de salida del boton Alta de Cliente.



El código de salida del botón va a ser **MODCLI**.



El código de salida del botón va a ser **BAJACLI**.

Si vamos de nuevo al **Element List** de la screen, veremos que se agregaron los dos botones como elementos de la screen 0100.

**Screen Painter: Change Screen for ZU50\_ABM\_CLIENTES**

Screen number 100 Active

Attributes Element list Flow logic

General attr. Texts / IO templates Special attr. Display attr. Mod. groups / functions References

H	M	Name	Type	Li	C	D	Vi	H	Sc	Format	In	O	Out	Di	Dict	Property list
		ALTA_CLIENTE	Push	6	23	15	15	1				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Properties
		MOD_CLIENTE	Push	9	23	23	23	1				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Properties
		BAJA_CLIENTE	Push	12	23	15	15	1				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Properties
		OKCODE	OK	0	0	20	20	1		OK					<input type="checkbox"/>	Properties

Object Name Description

▼ ZU50\_ABM\_CLIENTES Program ZU5

▼ Screens

0100 ABM de Clien

▼ Includes

La activamos



Luego se debe activar el include ZU50\_ABM\_CLIENTES\_I01

**ABAP Editor: Change Mod. Pool ZU50\_ABM\_CLIENTES**

Mod. Pool ZU50\_ABM\_CLIENTES Active

```

1  *-----*
2  * Module Pool      ZU50_ABM_CLIENTES
3  *
4  *-----*
5  *
6  *
7  *-----*
8  INCLUDE zu50_abm_clientes_top          .    " global Data
9
10 * INCLUDE ZU50_ABM_CLIENTES_001        .    " PBO-Modules
11 INCLUDE zu50_abm_clientes_i01          .    " PAI-Modules
12 * INCLUDE ZU50_ABM_CLIENTES_F01        .    " FORM-Routines

```

Object Name Description

▼ ZU50\_ABM\_CLIENTES Program ZU5

▼ Screens

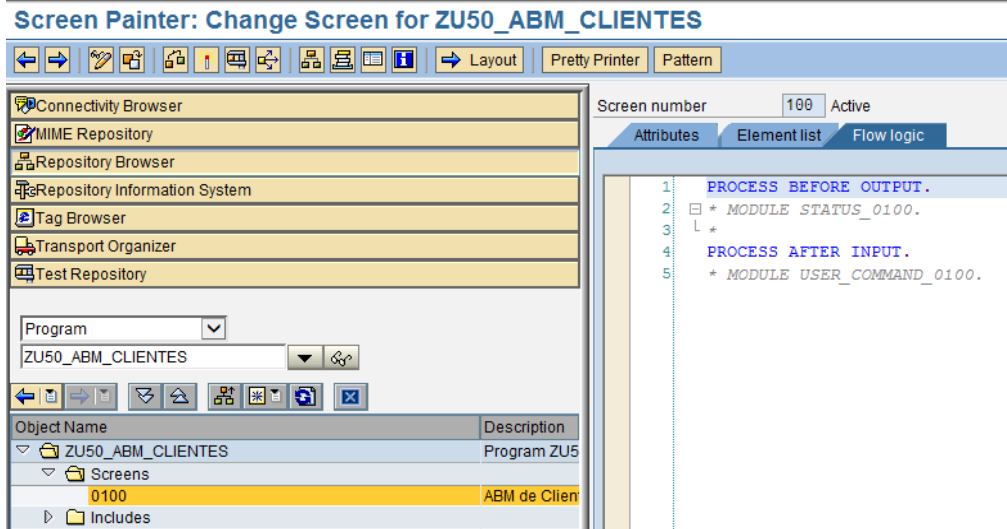
0100 ABM de Clien

▼ Includes

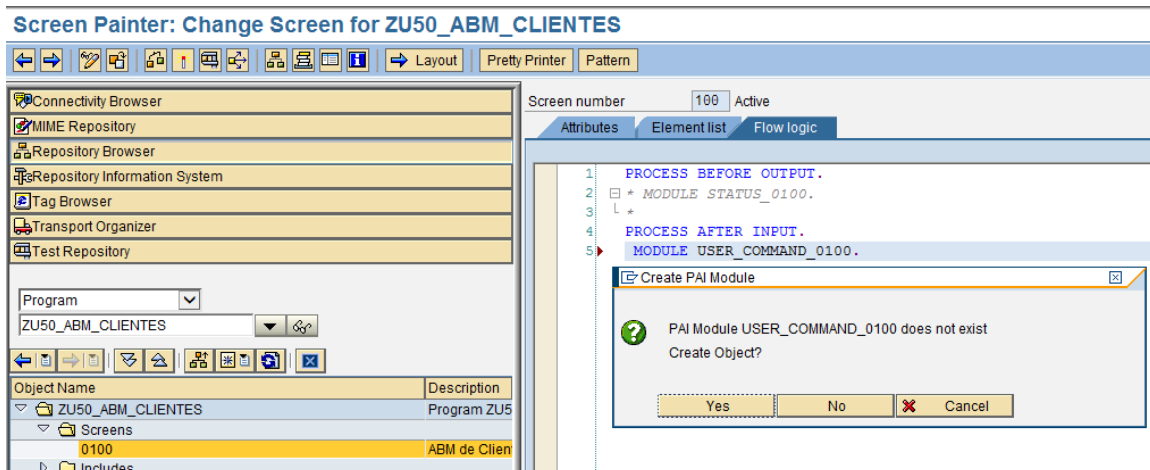
ZU50\_ABM\_CLIENTES\_I01 Include ZU50


ZU50\_ABM\_CLIENTES\_TOP Include ZU50

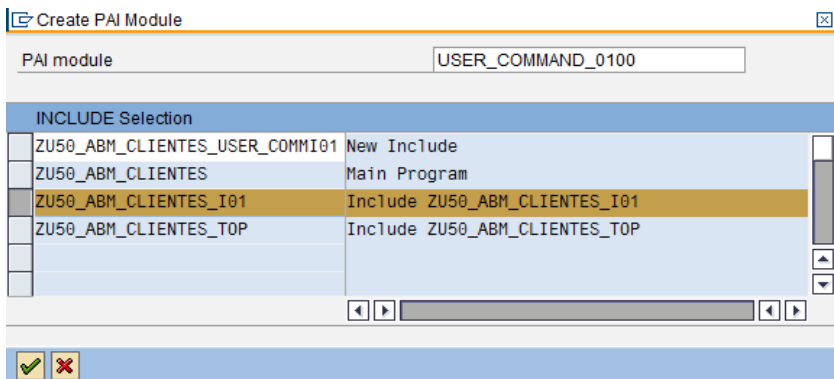
Podemos observar el Process BEFORE OUTPUT y el Process AFTER INPUT, en ambos podremos meter código, uno se ejecutara antes de mostrar la screen 0100 y otro al presionar algún botón que tenga asociado un código de salida (como bien recién definimos), entonces vamos a programar dependiendo el código de salida.




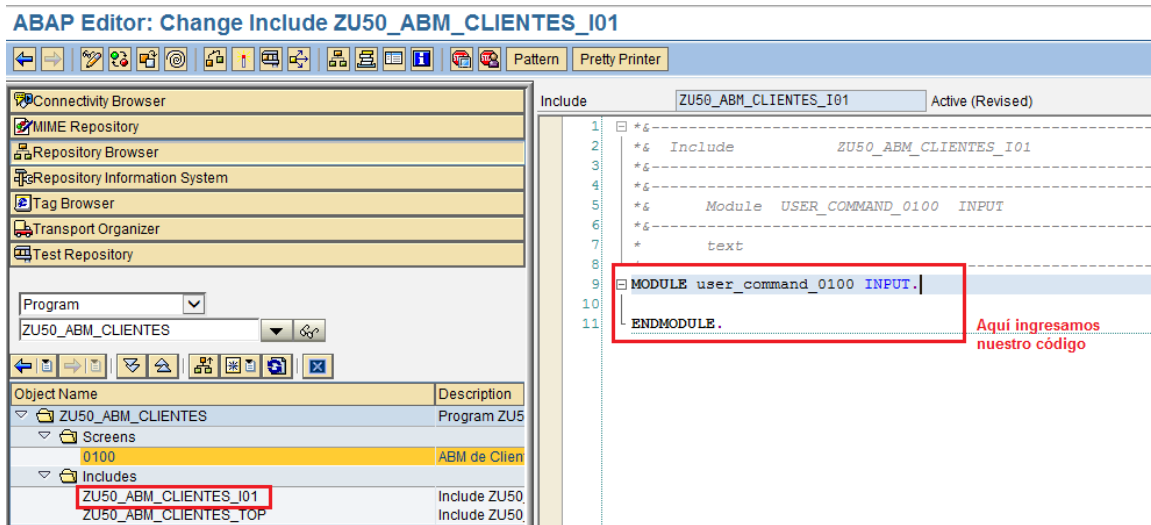
Sacamos el \* del **MODULE USER\_COMMAND\_0100** y le damos doble click al **USER\_COMMAND\_0100**



Le damos YES, y luego confirmamos , entonces creamos un nuevo INCLUDE donde almacenara el código que nosotros vamos a crear, este include estará dentro del programa MAIN ZU50\_ABM\_CLIENTES, se hace de esta forma para que quede mas claro el código.

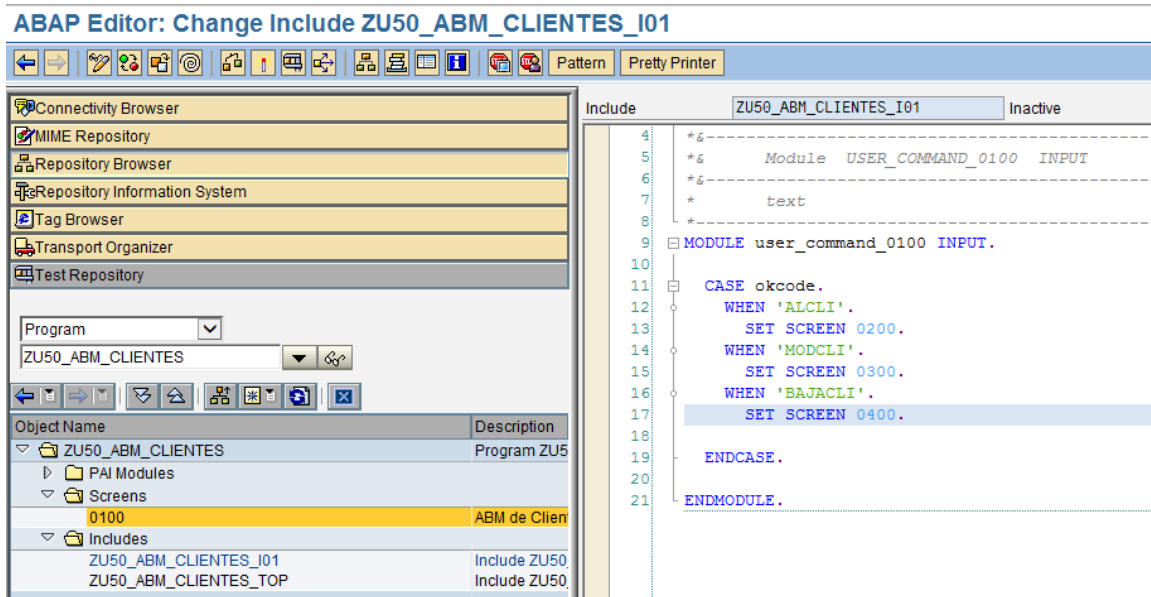



Te avisa que va a hacer y le confirmamos  :



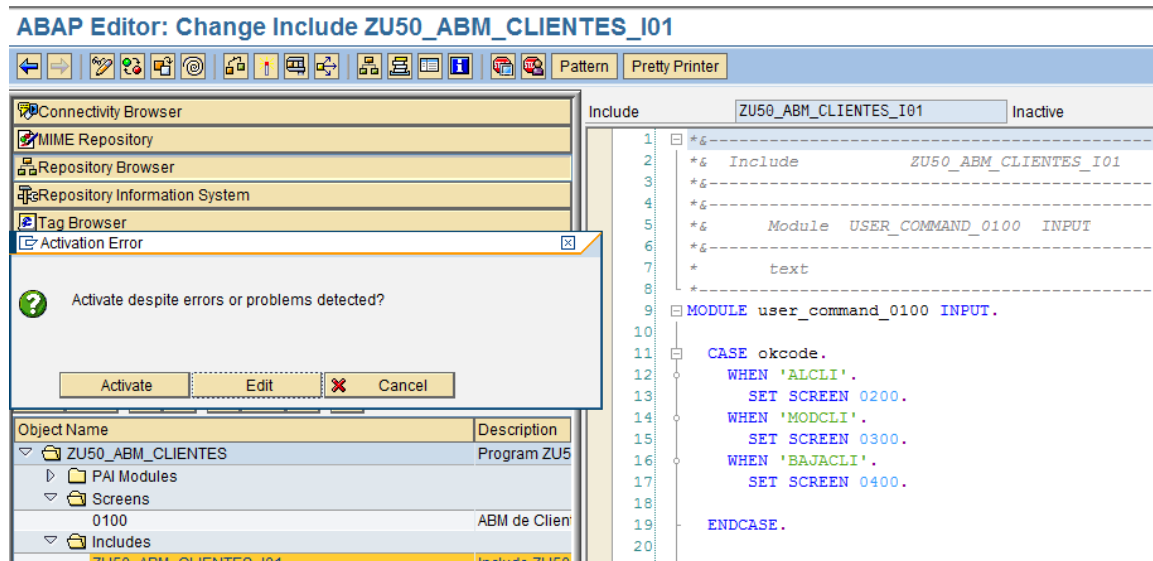
Nótese que al agregar un objeto nuevo (en este caso un include) a la screen 0100, se desactivo la screen, el programa principal y por supuesto el propio include que creamos, que luego habrá que activar los 3 para probar el programa.

Ingresamos el código, la variable a analizar donde se almacena el código de salida es OKCODE (que la definimos cuando creamos la Screen 0100), en el caso de salir con el código ALCLI, se ira a la Screen 0200, si sale con MODCLI irá a la Screen 0300 y si sale con BAJACLI irá. A la Screen 0400(Dichas Screens no han sido creadas aún).

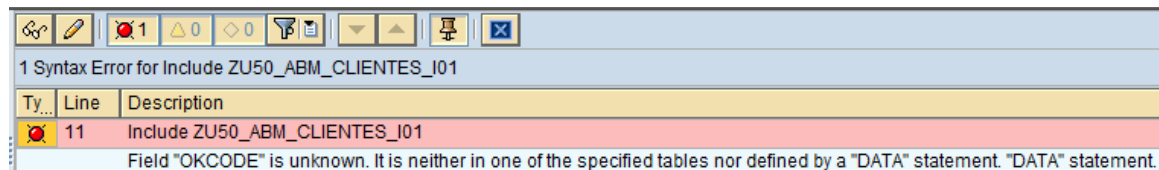


Nótese en la figura de arriba que tanto el include como la screen y el main program están desactivos, activemos primero el Include  y luego los otros.

Cuando queremos activar el Main program ZU50\_ABM\_CLIENTES, surge el siguiente Error:

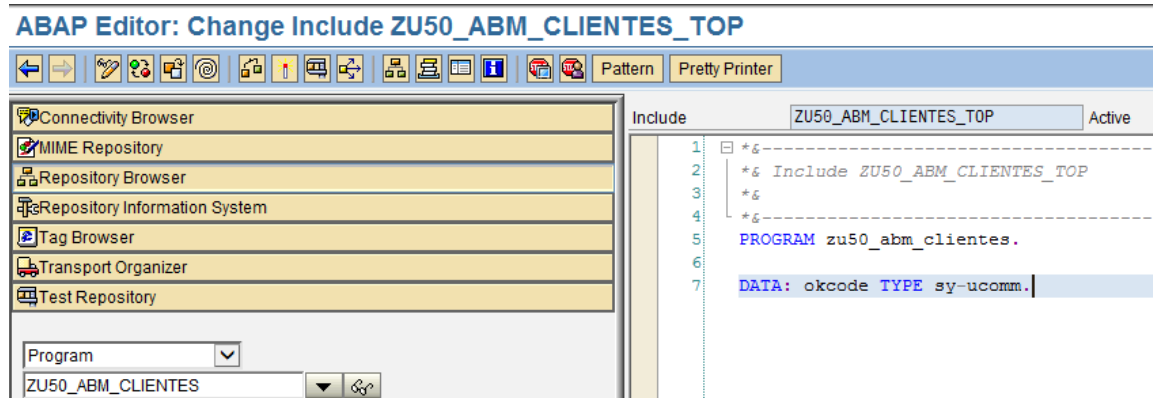


le a damos EDIT



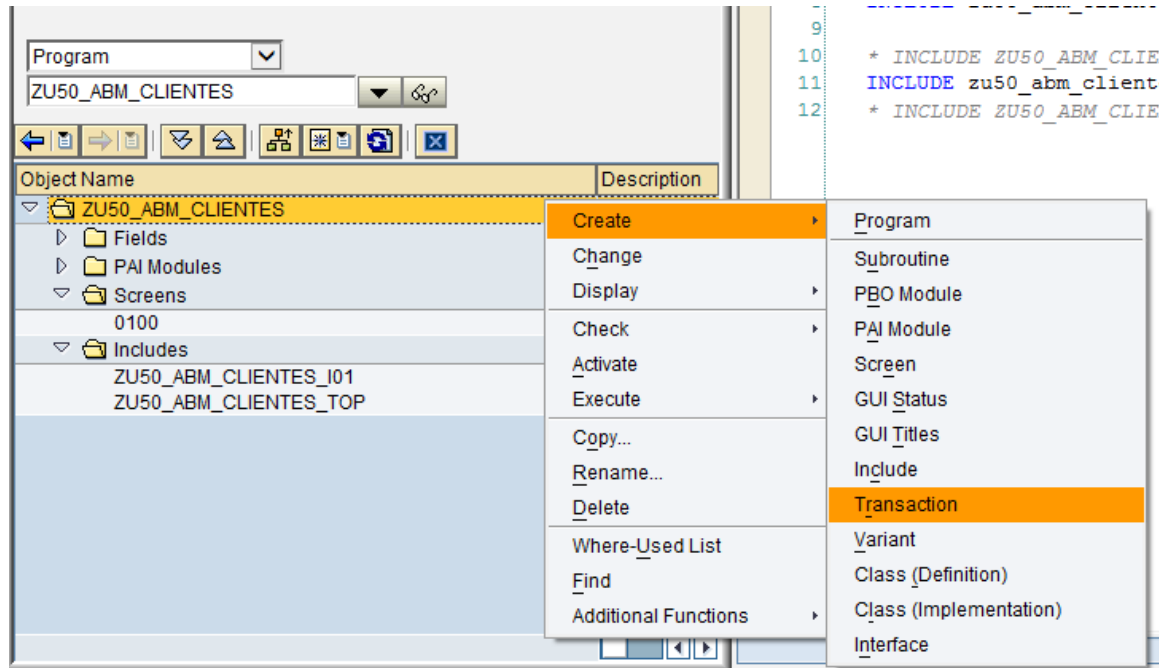
nos dice que el campo OKCODE no esta definido como una variable, esto quiere decir que en la screen 0100 le dijimos que asigne los códigos de salida a la variable OKCODE, pero no definimos OKCODE de que tipo de variable va a ser, por lo cual nos conviene definirla como variable global del programa, esto nos permitirá usar OKCODE en todas las screens proximas a crear.

Le damos doble click en ZU50\_ABM\_CLIENTES\_TOP, en este include por tener TOP, se definen las variables globales de todo el programa. (recuerden que este TOP lo creamos al principio de este programa).

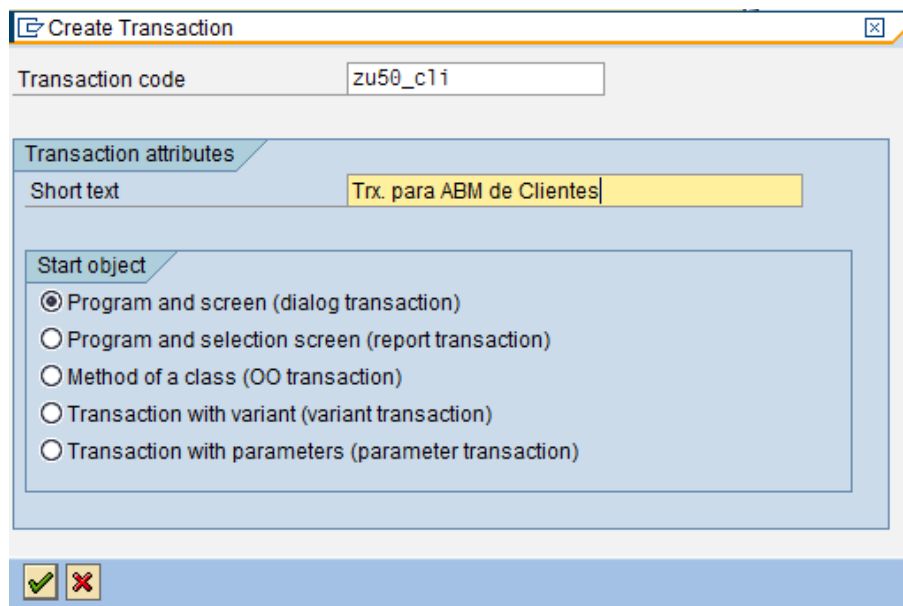


Luego activamos el TOP y el programa ZU50\_ABM\_CLIENTES después.


Ahora probemos si lo que hicimos hasta ahora está bien, para ejecutar el programa de dialogo que creamos, primero se debe crear una TRANSACCION que esté asociada a este programa, y se hace así:



Se le ingresa el nombre de la Transaccion, siempre con Z\* o Y\*, una descripción y por ultimo se le indica que es un programa de dialogo, vemos otras opciones que podrían ser asignar un reporte a una Transacción, etc.





Y en este último paso se le dice a que programa va a llamar la Transacción cuando se ejecute y que screen es la primera a ser llamada, en nuestro ejemplo el programa se llama ZU50\_ABM\_CLIENTES y la screen inicial es la 0100. Le damos .

### Change Dialog Transaction

Connectivity Browser

MIME Repository

Repository Browser

Repository Information System

Tag Browser

Transport Organizer

Test Repository


Transaction code: ZU50\_CLI

Package: \$TMP

Transaction text: Trx. para ABM de Clientes

Program: ZU50\_ABM\_CLIENTES

Screen number: 100

Authorization Object: 

☒ Maintenance of standard transaction variant allowed

Program: ZU50\_ABM\_CLIENTES

Object Name	Description
ZU50_ABM_CLIENTES	Program ZU50
Fields	
PAI Modules	
Screens	
0100	ABM de Clientes
Includes	
ZU50_ABM_CLIENTES_I01	Include ZU50
ZU50_ABM_CLIENTES_TOP	Include ZU50
Transactions	
ZU50_CLI	Trx. para ABM

Classification

Transaction classification

☒ Professional User Transaction

☐ Easy Web Transaction Service:


☐ Pervasive enabled

GUI support

☐ SAPGUI for HTML

☐ SAPGUI for Java






☐ SAPGUI for Windows


Vamos a Testear el programa, tenemos dos formas, o le damos al botón  o desde el acceso a tippear transacciones:

Workbench Edit Goto Utilities Environment System Help

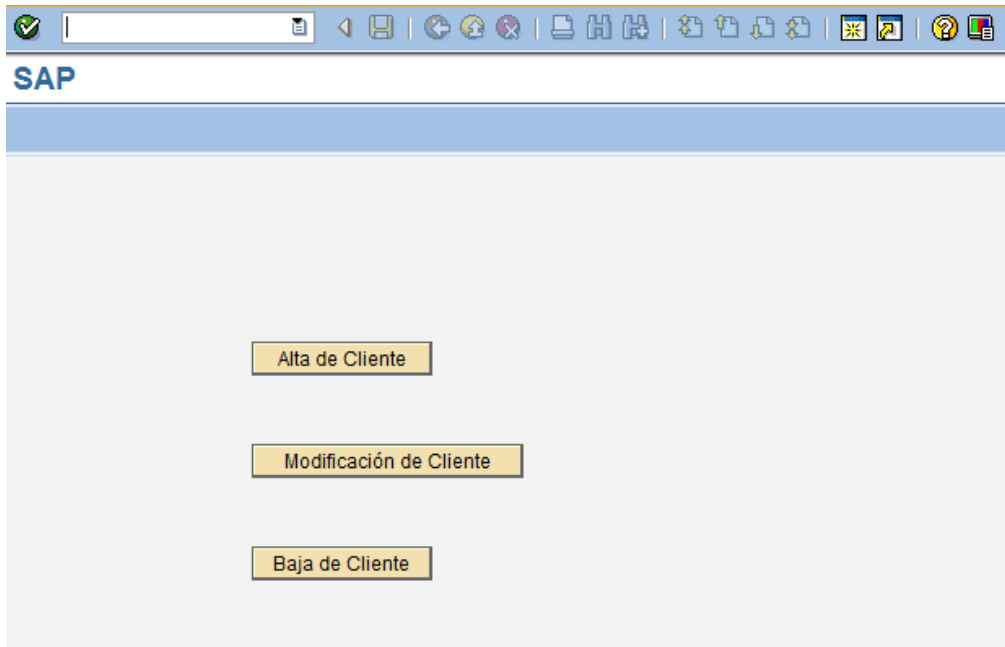
/ozu50\_cli

### Object Navigator

     Edit Object

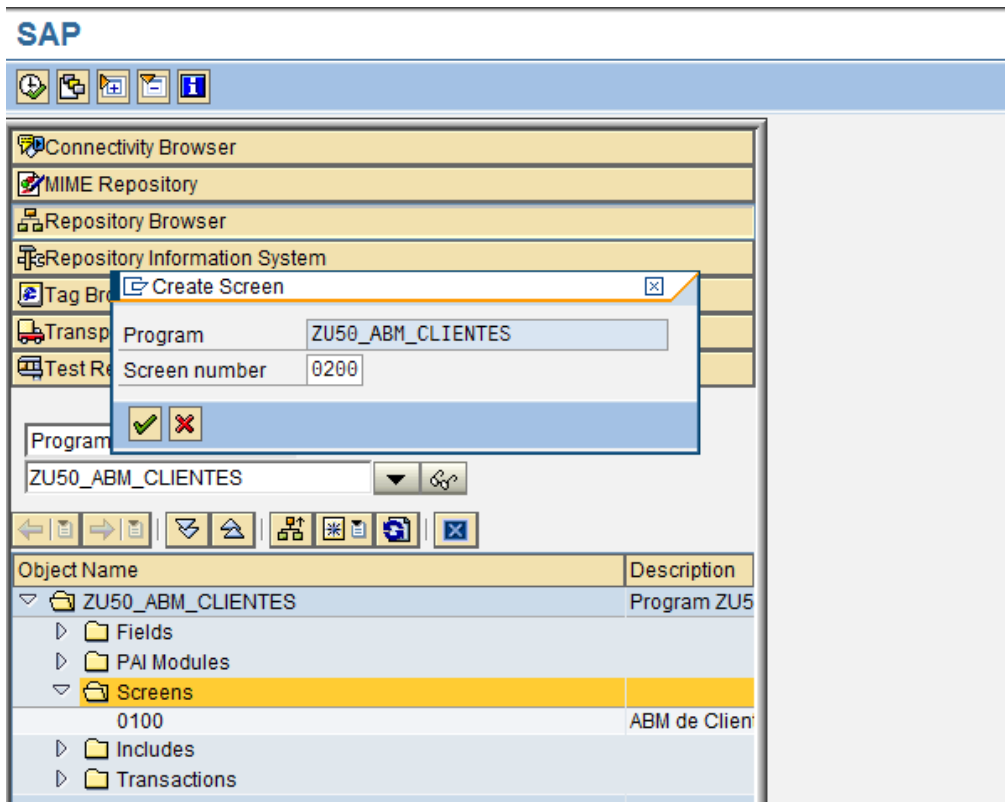
La barra indica que se va a llamar a una Transacción desde la transacción actual (no nos olvidemos que estamos en la se80) y la **O** le dice que ejecute la transacción que se le ingresa a continuación ZEJEMPLO pero que abra otra sesión de FrontEnd, esto se hace para que nosotros sigamos programando en esta sesión. **Le damos ENTER o .**

Y aca tenemos nuestro programa de dialogo.

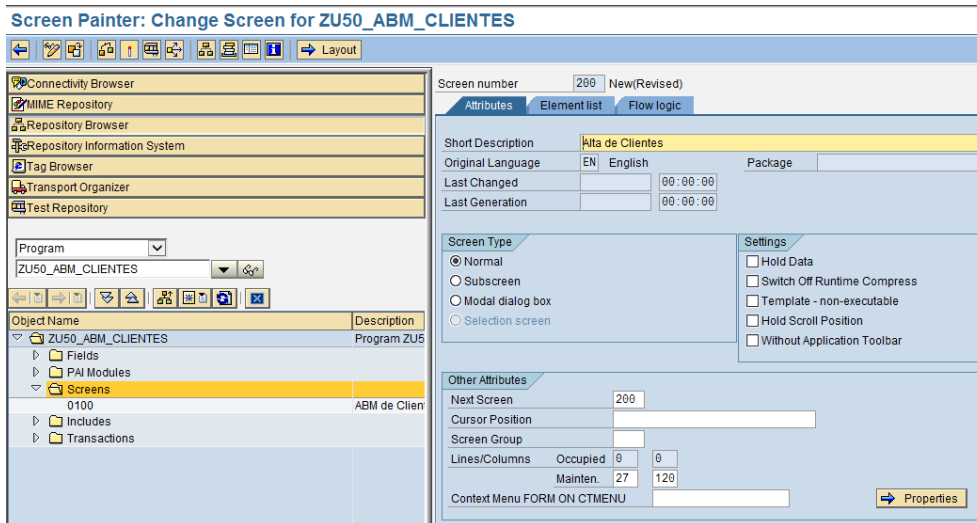


Por supuesto si presionamos en cualquiera de los tres botones, nos dará un DUMP (Error) debido a que va a querer ir a la screen 0200, 0300 ó 0400 y nosotros aún no la creamos.

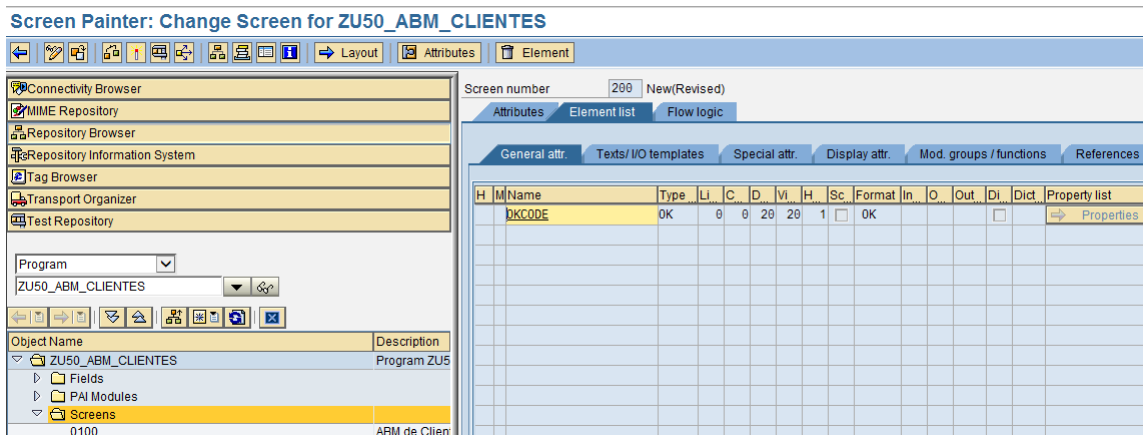
Vamos a crear la Screen 0200, donde podremos dar de alta a los clientes.

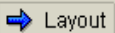


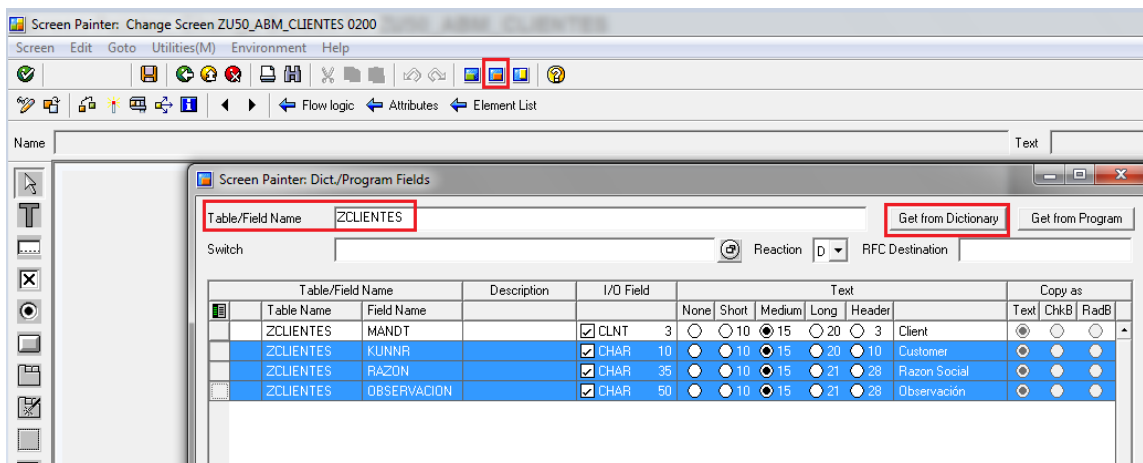
Ingresamos el título, y va a ser una Normal Screen.



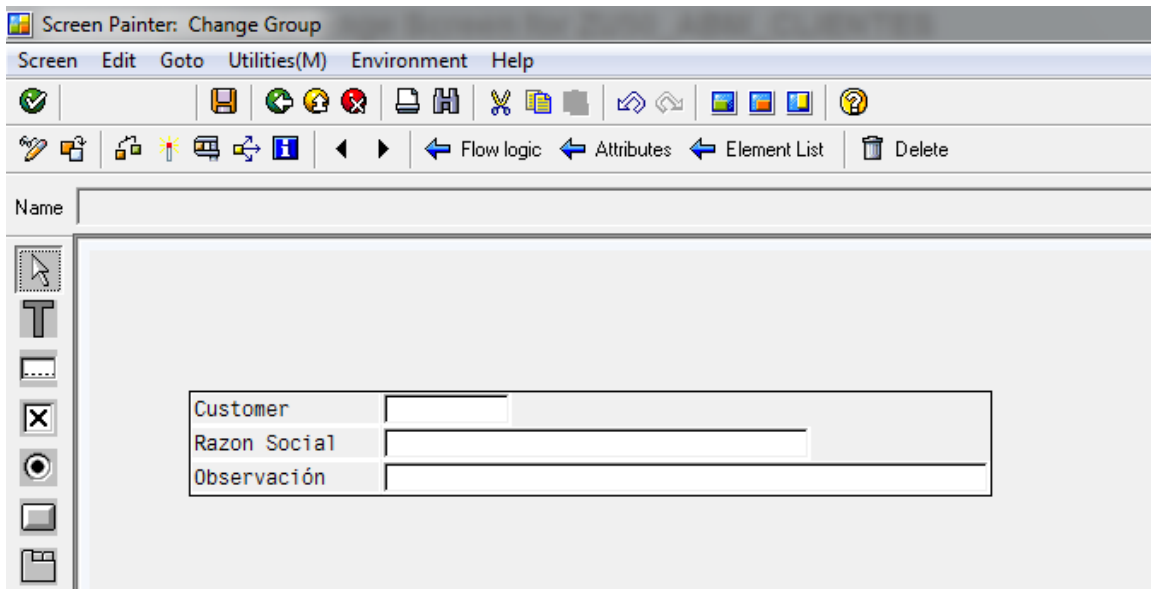
En Element List, seguimos usando OKCODE como código de salida.



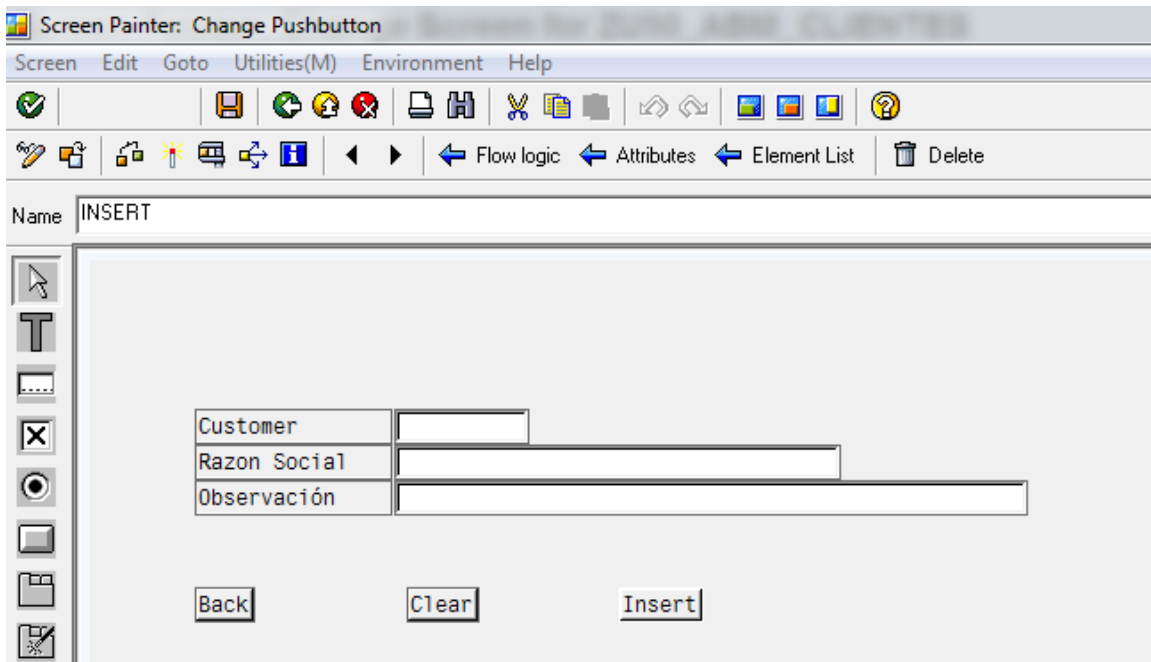
y en el flow logic de esta screen aun no tenemos nada. Vamos a definir el Layout de la screen  donde pondremos los datos del cliente y le vamos a agregar campos de Input/Output y sus textos:



Luego ubicamos esos campos dentro de la Screen 0200 seleccionados de la tabla ZCLIENTES:



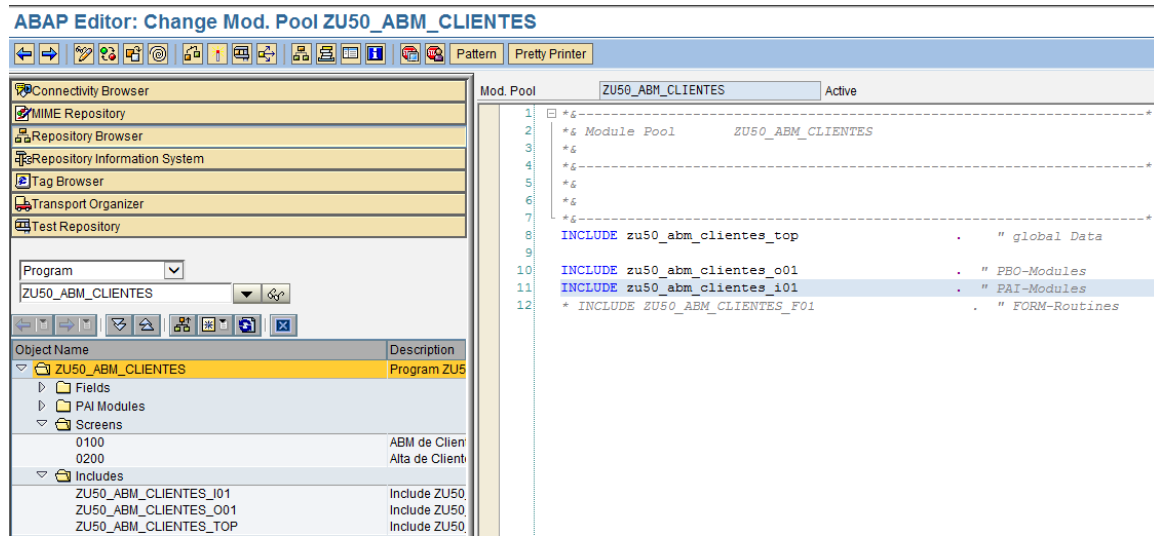
Ahora le agregamos por último 3 BOTONES, BACK, CLEAR E INSERT y le asignamos esos nombres como código de salida a cada uno.



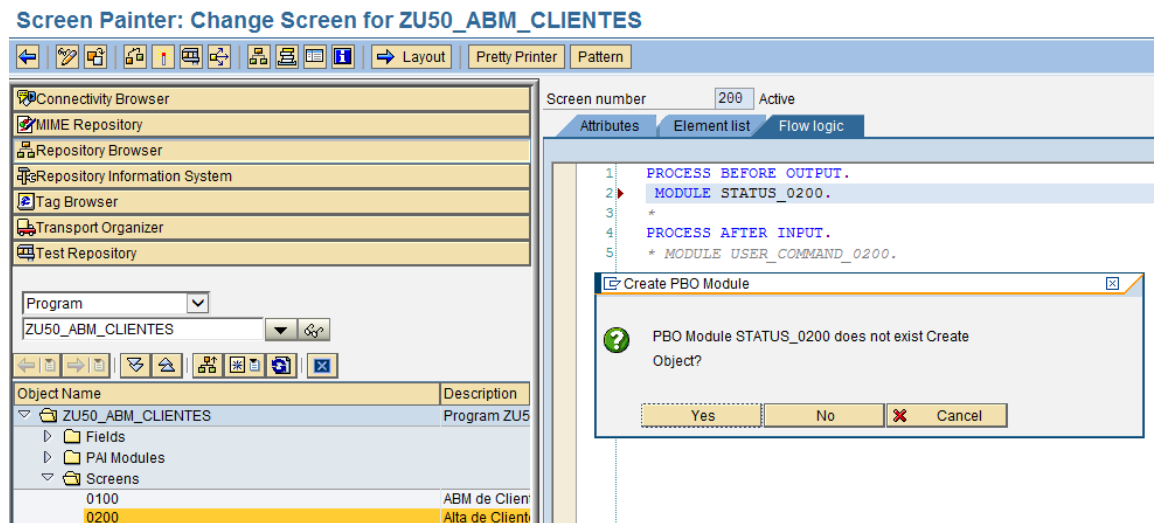
Bueno ya tenemos el Layout, ahora tenemos que programar un poco, la idea es que desde la screen 0100 se vaya a la screen 0200, se muestre apenas se carga la 0200, el ultimo cliente ingresado y que luego se pueda hacer un CLEAR y un INSERT (de un nuevo cliente).

Para que este el último cliente cargado cuando se vaya a la 0200, se deberá utilizar el PBO, que es lo que se ejecuta antes de cargar la 0200, luego el CLEAR y INSERT lo manejaremos en el PAI (process after input).

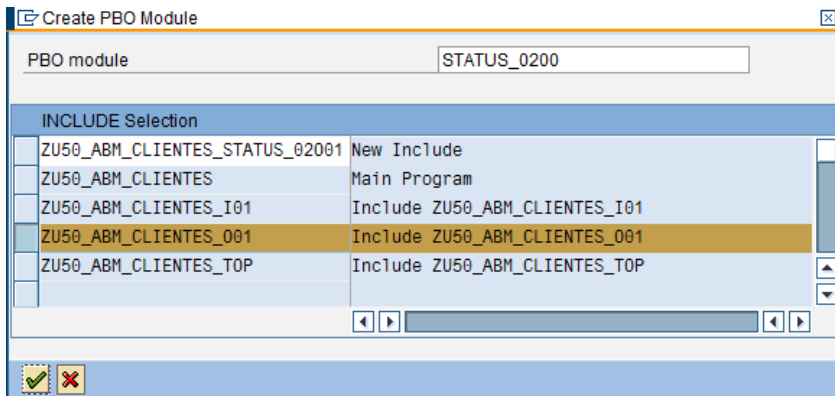
Primero se debe activar el INCLUDE ZU50\_ABM\_CLIENTES\_O01




Luego se debe crear el MODULE STATUS\_0200:

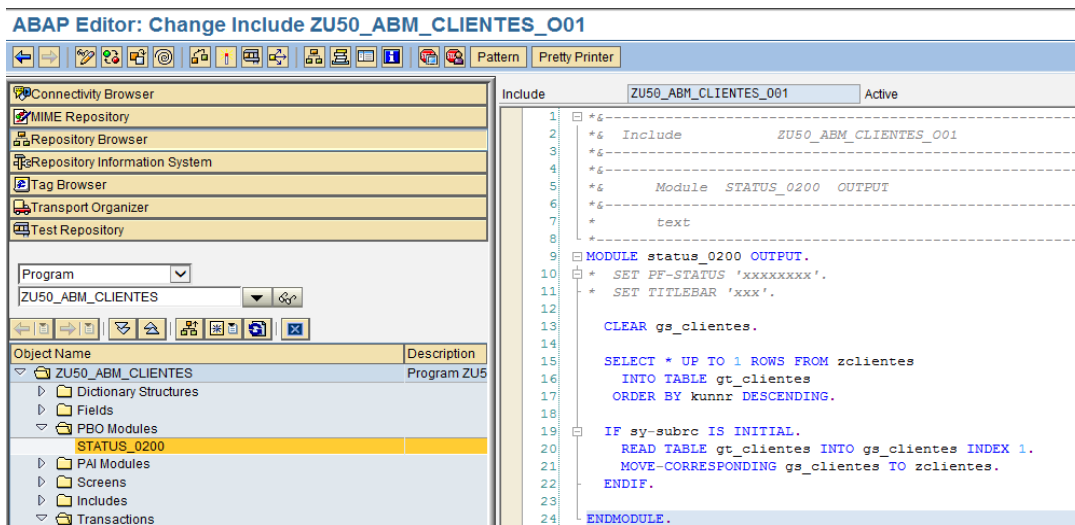


le Damos YES, y seleccionamos el INCLUDE ZU50\_ABM\_CLIENTES\_O01 previamente creado:

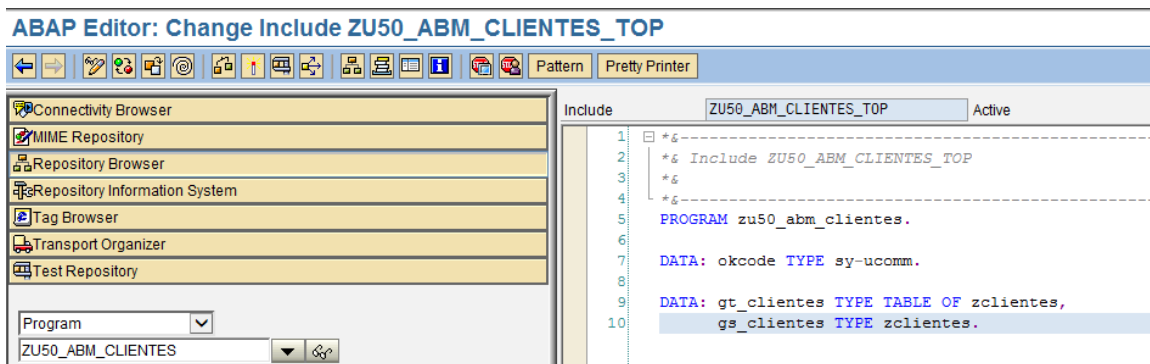


Click en 

Hacemos un select up to 1 ROWS \* (solo tomara el primero de los seleccionados en la tabla interna) ordenandolo descendientemente para tomar el ultimo registro, luego leemos la tabla interna (con indice 1, ya que sabemos que solo traera un registro) y lo guardamos en la Work Área y luego activamos, pero antes de activar debemos declarar la tabla interna y la working area como TABLE, lo hacemos en el TOP.

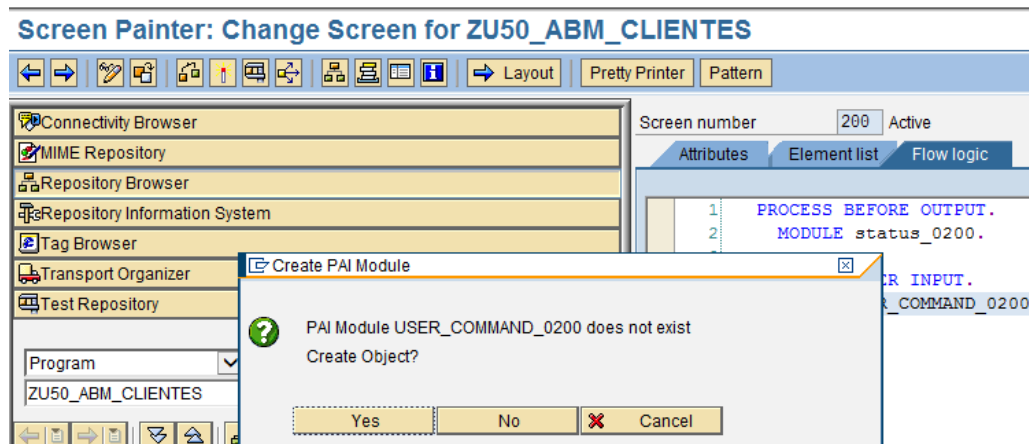


Declaramos la tabla interna GT\_CLIENTES y la working area GS\_CLIENTE en el TOP.

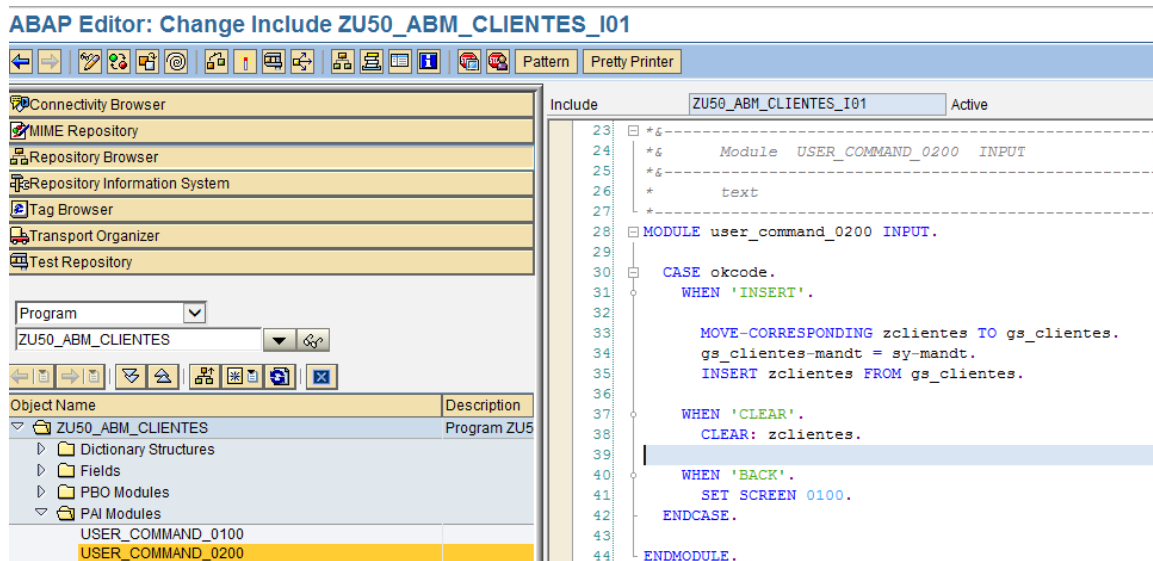
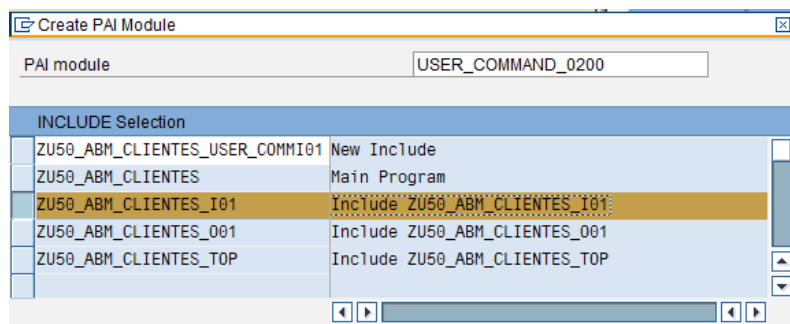


**Recordemos:** Luego hay que activar todo lo que esta en Azul para poder ejecutar el programa de dialogo. De todos modos no podemos probarlo debido a que no tenemos ningún cliente ingresado como para que nos muestre el último, así que programemos el alta de un cliente.

Ya que esta ingresamos el código para los 3 botones, por lo que debemos crear el MODULE USER\_COMMAND\_0200.



Luego seleccionamos el INCLUDE ZU50\_ABM\_CLIENTES\_I01:



Activamos todo, y ejecutamos la transacción ZU50\_CLI para probar lo programado.



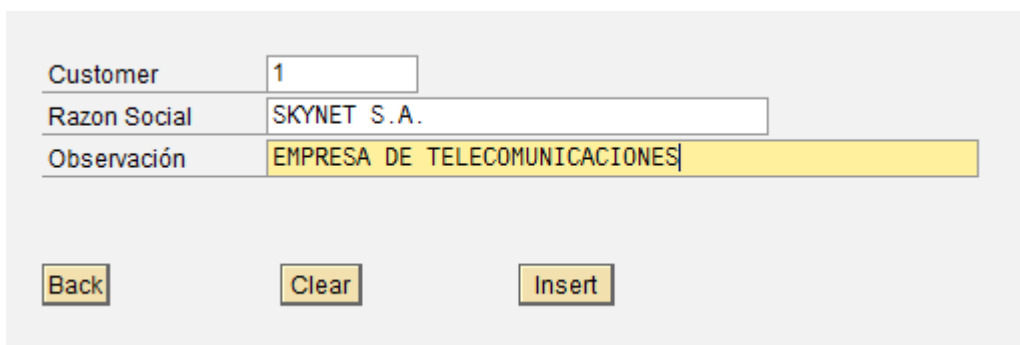
A menu with three yellow buttons stacked vertically on a light gray background. The buttons are labeled "Alta de Cliente", "Modificación de Cliente", and "Baja de Cliente".

Vamos a Alta de Cliente



A form for adding a new client. It has three input fields: "Customer" (a small box), "Razon Social" (a medium box), and "Observación" (a long box). Below the fields are three buttons: "Back", "Clear", and "Insert".

Ahora ingresamos un cliente cualquiera.



The same form as before, but now with data entered. The "Customer" field contains "1", the "Razon Social" field contains "SKYNET S.A.", and the "Observación" field contains "EMPRESA DE TELECOMUNICACIONES". The "Back", "Clear", and "Insert" buttons are still at the bottom.

al darle INSERT, no notamos mucho la diferencia.

Podemos abrir otra sesión y chequear desde la SE11 si efectivamente la tabla ZCLIENTES tiene el cliente recién ingresado por nosotros.



### ABAP Dictionary: Initial Screen

☒ Database table
 ☐ View
 ☐ Data type
 ☐ Type Group
 ☐ Domain
 ☐ Search help
 ☐ Lock object

ZCLIENTES

le damos DISPLAY

### Dictionary: Display Table

Transparent Table: ZCLIENTES Active
   
 Short Description: Maestro de Clientes

Field	Key	Initi...	Data element	Data Type	Length	Decim...	Short Description
MANDT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT	3	0	Client
KUNNR	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	KUNNR	CHAR	10	0	Customer Number
RAZON	<input type="checkbox"/>	<input type="checkbox"/>	ZRAZON S	CHAR	35	0	Razon Social
OBSERVACIONES	<input type="checkbox"/>	<input type="checkbox"/>	ZOBSCLI	CHAR	50	0	Observación de un cliente

y luego , para visualizar el contenido de la tabla. Le damos 

### Data Browser: Table ZCLIENTES: Selection Screen

KUNNR  to 
  
 RAZON  to 
  
 OBSERVACIONES  to 
  
 Width of Output List 
  
 Maximum No. of Hits

y vemos efectivamente que el registro se encuentra en la tabla.

Table Entry Edit Goto Settings Utilities Environment System Help				
Data Browser: Table ZCLIENTES Select Entries 1				
Table: ZCLIENTES Displayed Fields: 4 of 4 Fixed Columns: 2 List Width 0250				
MANDT	KUNNR	RAZON	OBSERVACIONES	
600	0000000001	SKYNET S.A.	EMPRESA DE TELECOMUNICACIONES	

Volvamos a la aplicación, probamos el botón CLEAR y sin embargo no deja en blanco los campos.

Esto se debe a que al presionar CLEAR, sale por el PAI, realiza el clear pero luego como vuelve a la misma 0200 pasa nuevamente por el PBO de la 0200, y el PBO le dice que seleccione el último registro de ZCLIENTES.

```

Include      ZU50_ABM_CLIENTES_001      Active
1  *-----
2  *& Include      ZU50_ABM_CLIENTES_001
3  *&-----
4  *&-----
5  *&      Module  STATUS_0200  OUTPUT
6  *&-----
7  *      text
8  *-----
9  MODULE status_0200 OUTPUT.
10 * SET PF-STATUS 'xxxxxxxx'.
11 * SET TITLEBAR 'xxx'.
12
13 CLEAR gs_clientes.
14
15 SELECT * UP TO 1 ROWS FROM zclientes
16 INTO TABLE gt_clientes
17 ORDER BY kunnr DESCENDING.
18
19 IF sy-subrc IS INITIAL.
20 READ TABLE gt_clientes INTO gs_clientes INDEX 1.
21 MOVE-CORRESPONDING gs_clientes TO zclientes.
22 ENDIF.
23
24 ENDMODULE.

```

para evitar esto, agreguemos este código:

Include	ZU50_ABM_CLIENTES_001	Active
1	*-----	
2	* Include               ZU50_ABM_CLIENTES_001	
3	*-----	
4	*-----	
5	*       Module   STATUS_0200   OUTPUT	
6	*-----	
7	*       text	
8	*-----	
9	MODULE status_0200 OUTPUT.	
10	*   SET PF-STATUS 'xxxxxxx'.	
11	*   SET TITLEBAR 'xxx'.	
12		
13	CLEAR gs_clientes.	
14		
15	IF okcode NE 'CLEAR'.	
16		
17	SELECT * UP TO 1 ROWS FROM zclientes	
18	INTO TABLE gt_clientes	
19	ORDER BY kunnr DESCENDING.	
20		
21	IF sy-subrc IS INITIAL.	
22	READ TABLE gt_clientes INTO gs_clientes INDEX 1.	
23	MOVE-CORRESPONDING gs_clientes TO zclientes.	
24	ENDIF.	
25		
26	ENDIF.	
27		
28	ENDMODULE.	

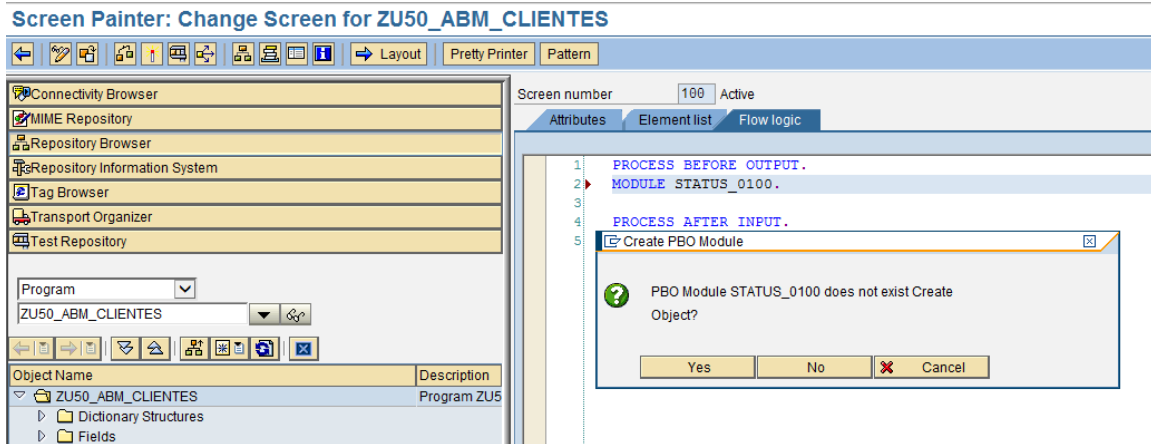
Si activamos y probamos de presionar el botón CLEAR y notaremos la diferencia. (Como el okcode lo definimos en el TOP, este se queda con el código de salida del último botón presionado).

Customer	<input style="width: 95%;" type="text"/>	
Razon Social	<input style="width: 100%;" type="text"/>	
Observación	<input style="width: 100%;" type="text"/>	

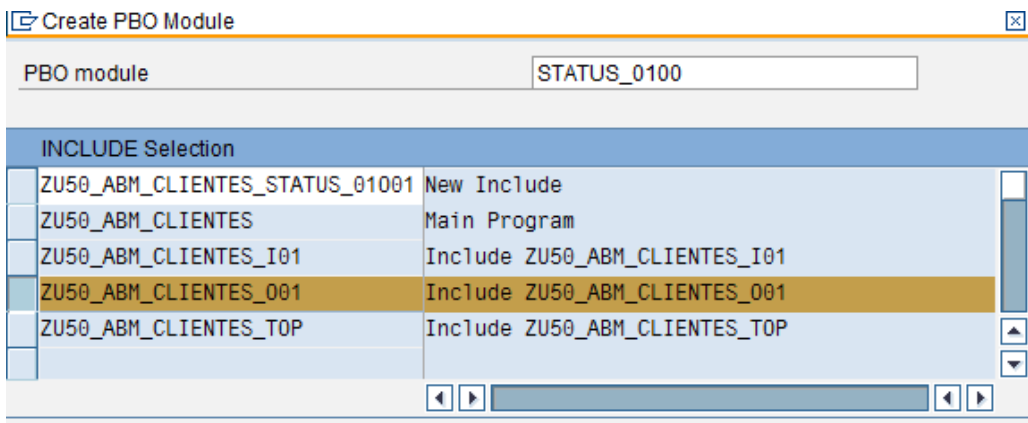
Back
Clear
Insert

Podemos también crear Status GUI para las Screens, en este caso vamos a crear un status para la Screen 0100 o sea para la primer pantalla:

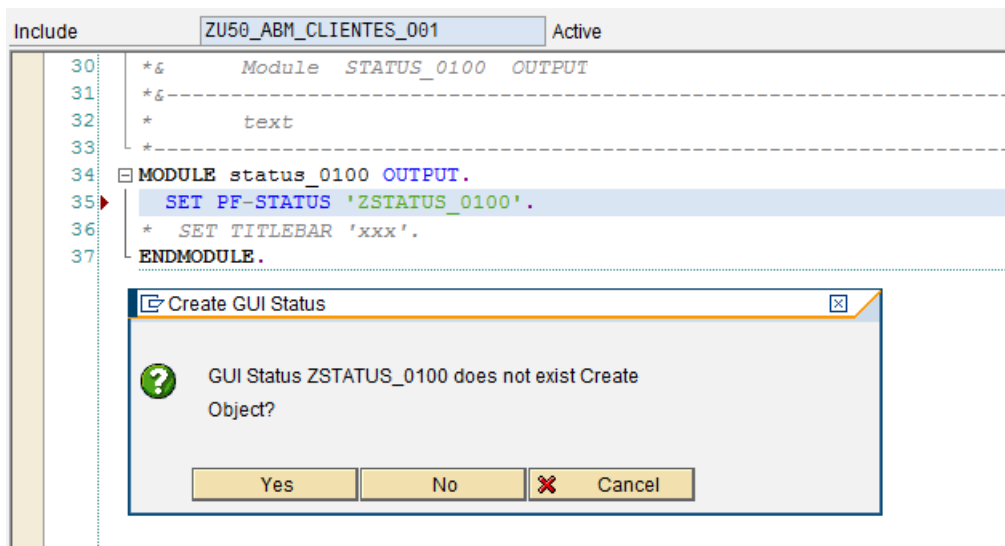
Crear el MODULE STATUS\_0100 y luego le damos a botón YES:



Seleccionamos el INCLUDE ZU50\_ABM\_CLIENTES\_O01 para incluir el MODULE



Luego creamos el Status de la siguiente manera y le damos a botón YES:



Le asignamos una Descripción al Status GUI que vamos a crear:

Create Status

Program: ZU50\_ABM\_CLIENTES

Status: ZSTATUS\_0100

Status Attributes

Short Text: Status para Screen 0100

Status type

☒ Normal Screen

☐ Dialog Box

☐ Context Menu

OK, Cancel, Help icons

Habilitamos los siguientes Botones en la barra de herramientas de la pantalla:

Maintain Status ZSTATUS\_0100 of Interface ZU50\_ABM\_CLIENTES

User Interface: ZU50\_ABM\_CLIENTES Active(revised)

Menu Bar: Status para Screen 0100

Application Toolbar: Status para Screen 0100

Function Keys: Status para Screen 0100

Standard Toolbar: BACK, EXIT, CANCEL (highlighted)

Recommended Function Key Settings:

F2	<..>	Choose
F9	<..>	Select
Shift-F2	<..>	Delete
Shift-F4	<..>	Save without check
Shift-F5	<..>	Other <object>

Freely Assigned Function Keys:

F5

F6

Siempre activamos todo lo que vayamos creando, ahora codeamos la parte del Module USER\_COMMAND\_0100 para darles la funcionalidad a los nuevos botones:

```

Include      ZU50_ABM_CLIENTES_I01      Active

4  *-----
5  *      Module  USER_COMMAND_0100  INPUT
6  *-----
7  *      text
8  *-----
9  MODULE user_command_0100 INPUT.
10
11  CASE okcode.
12      WHEN 'ALCLI'.
13          SET SCREEN 0200.
14      WHEN 'MODCLI'.
15          SET SCREEN 0300.
16      WHEN 'BAJACLI'.
17          SET SCREEN 0400.
18      WHEN 'BACK' OR 'EXIT' OR 'CANCEL'.
19          LEAVE PROGRAM.
20  ENDCASE.
21
22  ENDMODULE.

```

Ejercicios para Hacer:

Debe realizar la funcionalidad para los botones de Modificación de Cliente y Baja de Cliente, con sus respectivas Screens 0300 y 0400 donde en cada una de ellas se debe agregar los botones Back, Clear y Baja y Modificar.