# Crear un ALV GRID usando ABAP orientado a objetos

SAP proporciona varias formas de realizar una tarea, crear un ALV GRID no es diferente. Aunque hay módulos de función simples disponibles para crear un ALV GRID, la metodología orientada a objetos es más preferida debido a sus propias ventajas que siempre se debaten en el mundo ABAP.

Para crear una pantalla con un ALV GRID utilizando la clase `CL_GUI_ALV_GRID`.

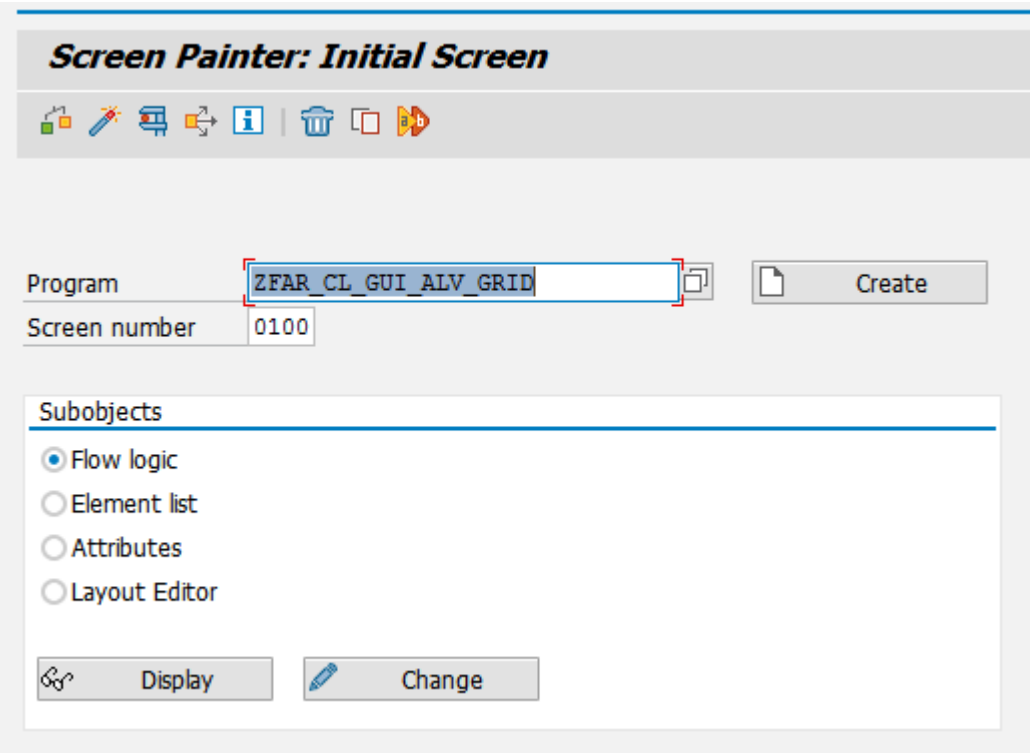
## Cómo vamos a hacer

Vamos a crear una pantalla con un contenedor personalizado colocado en ella. Cuando se ejecuta el programa, se creará una nueva instancia de un ALVGRID y se colocará en el contenedor personalizado de la pantalla.

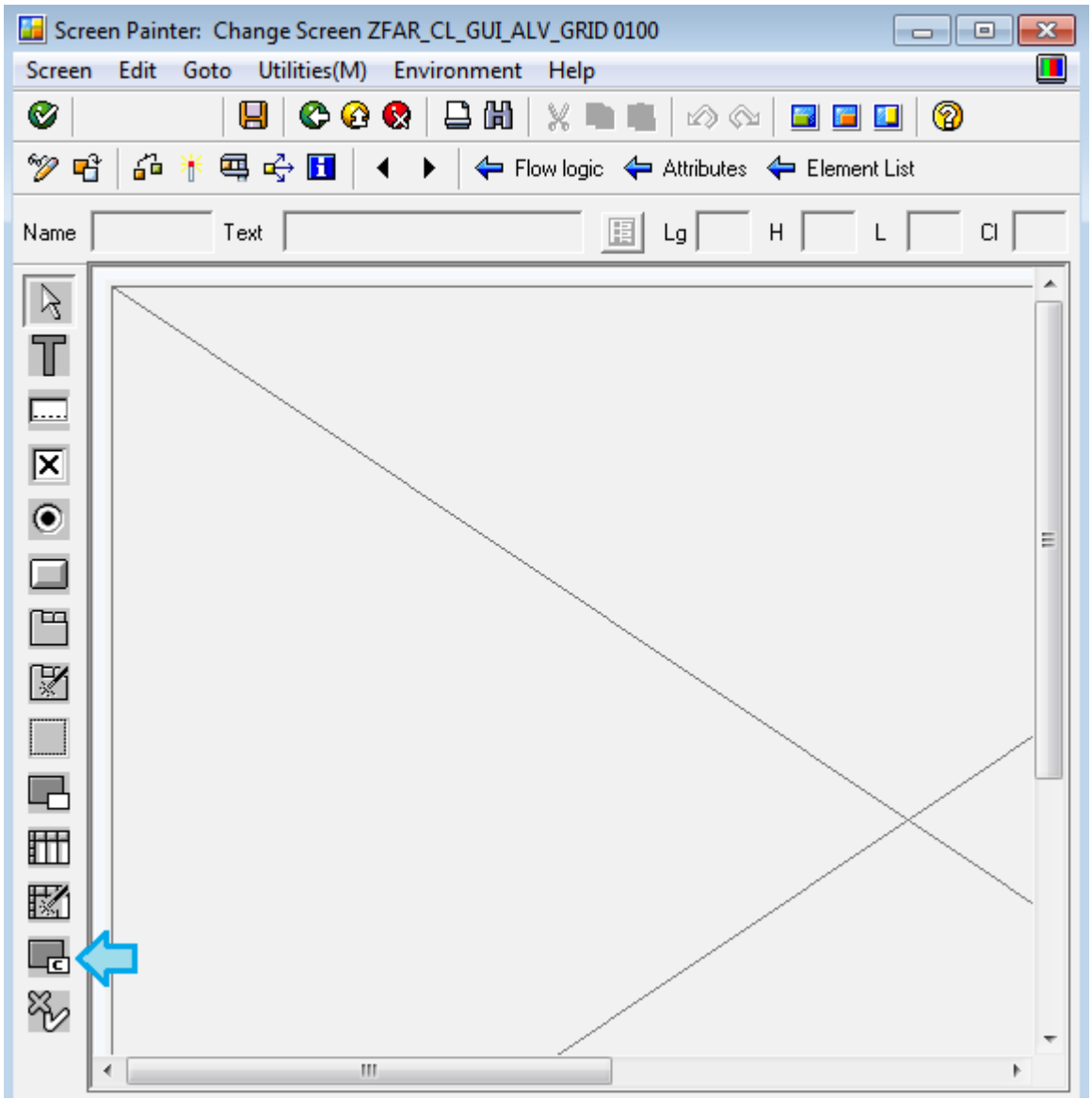
## Crea la pantalla

Antes de crear una pantalla, se necesita crear un programa para que posteriormente se pueda crear una pantalla correspondiente al programa. *Vaya* a *SE38* y cree un programa llamado `ZFAR_CL_GUI_ALV_GRID` (puede elegir sus propios nombres de programa).

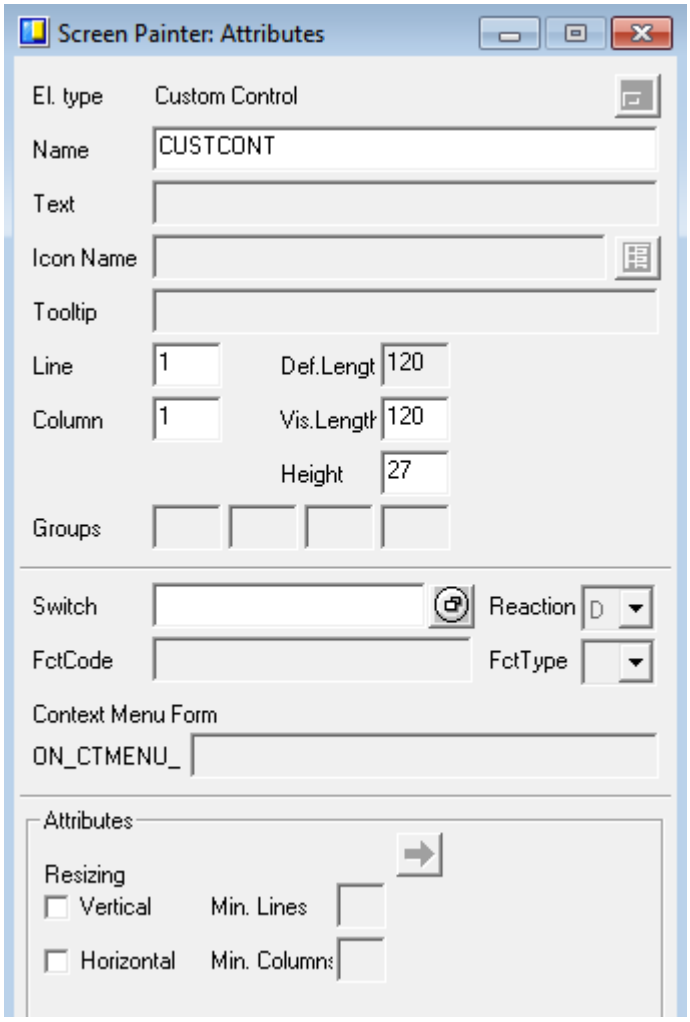
Ahora vamos a crear una pantalla. Vaya a SE51 y cree una pantalla con número de pantalla `0100`.



Presione el botón Layout en la barra de herramientas para abrir el diseñador de pantalla.




Aparecerá Screen Painter y deberá dibujar un contenedor personalizado en la pantalla. Haga doble clic en el contenedor que acaba de dibujar ahora y configure el nombre del contenedor como `CUSTCONT` en la ventana de propiedades que aparece como se muestra a continuación.




Ahora la pantalla está lista, pero debe especificar qué código se debe ejecutar durante los eventos de PBO y PAI de la pantalla. Cierre el Screen Painter y seleccione la pestaña Flow Logic en la sección anterior donde verá los eventos de PBO y PAI.

Debajo de esos eventos hay líneas comentadas, que debes descomentar ahora. Después de descomentar, haga doble clic para crear los módulos en el programa. Asegúrese de haber creado ambos módulos en el programa principal que creamos al principio.

Ahora su código debería verse así

```
REPORT zfar_cl_gui_alv_grid.

DATA: g_alv   TYPE REF TO cl_gui_alv_grid,
      g_cust  TYPE REF TO cl_gui_custom_container,
      gt_kna1 TYPE TABLE OF kna1,
      gt_fcat TYPE lvc_t_fcat,
      gs_fcat TYPE lvc_s_fcat,
      okcode  TYPE sy-ucomm.

START-OF-SELECTION.
```

```
  SET SCREEN 0100.
END-OF-SELECTION.
```

# Hora de codificar

Ahora vamos a crear un objeto para la clase CL_GUI_ALV_GRID y hacer que la cuadrícula aparezca en el contenedor personalizado que creamos en la pantalla con algunos datos de la tabla KNA1. Entonces vamos a tener las siguientes variables y referencias en el programa.

```
DATA: "Referencia a la instancia de ALV Grid

      g_alv   TYPE REF TO cl_gui_alv_grid,

      "Referencia al contenedor que colocamos en la screen

      g_cust  TYPE REF TO cl_gui_custom_container,

      "Tabla interna para mostrar los datos

      gt_kna1 TYPE TABLE OF kna1,

      "Tabla interna para el catálogo de campos del ALV Grid

      gt_fcat TYPE lvc_t_fcat,

      "Área de trabajo para catálogo de campo

      gs_fcat TYPE lvc_s_fcat,


      okcode  TYPE sy-ucomm.
```

Tan pronto como iniciemos el programa, necesitamos que aparezca la pantalla que creamos. Y por lo tanto...

```
START-OF-SELECTION.
  SET SCREEN 0100.
END-OF-SELECTION.
```

Entonces cuando la pantalla va a aparecer:

- Necesitamos cargar los datos necesarios
- Crear objeto para contenedor personalizado y cuadrícula ALV
- Prepare el catálogo de campo para ALV Grid
- Mostrar el ALV Grid

# Cargar los datos

```
FORM load_data.

  SELECT * FROM kna1
    INTO CORRESPONDING FIELDS OF TABLE gt_kna1
      UP TO 10 ROWS.

ENDFORM
```

# Preparar el catálogo de campo

```
FORM prepare_fcat.

  DEFINE add_fcat.

    CLEAR gs_fcat.

    gs_fcat-col_pos   = &1.
    gs_fcat-fieldname = &2.
    gs_fcat-coltext   = &3.
    gs_fcat-outputlen = &4.

    APPEND gs_fcat TO gt_fcat.

  END-OF-DEFINITION.

  add_fcat:
     1 'KUNNR' 'Customer No.' 15,
     2 'LAND1' 'Country'      5,
     3 'NAME1' 'Name'         30,
     4 'ORT01' 'City'         20.

ENDFORM.
```

Personalmente prefiero las macros para hacer este tipo de trabajo repetido, por lo que es fácil de codificar y el código se ve muy claro y comprensible. Para decir aproximadamente lo que estoy haciendo en este código, he agregado cuatro registros a gt_fcat con una configuración de catálogo de campo muy simple, para mostrar cuatro columnas en la ALV Grid.

# Poniendo todo junto

Hemos escrito dos FORMs para que podamos llamarlos cuando sea necesario. Ahora todo este proceso tiene que suceder antes de que la pantalla aparezca en la salida. Y entonces tenemos que colocar el código en * PBO.

Pero PBO se ejecuta una y otra vez cada vez que se lleva a cabo una acción en la pantalla, mientras que queremos cargar y mostrar la tabla solo la primera vez que se carga la pantalla. La solución es simple, haremos todo este proceso solo cuando g_alv sea inicial, y después de la primera ejecución g_alv no será inicial, por lo que el código se ejecutará solo una vez.

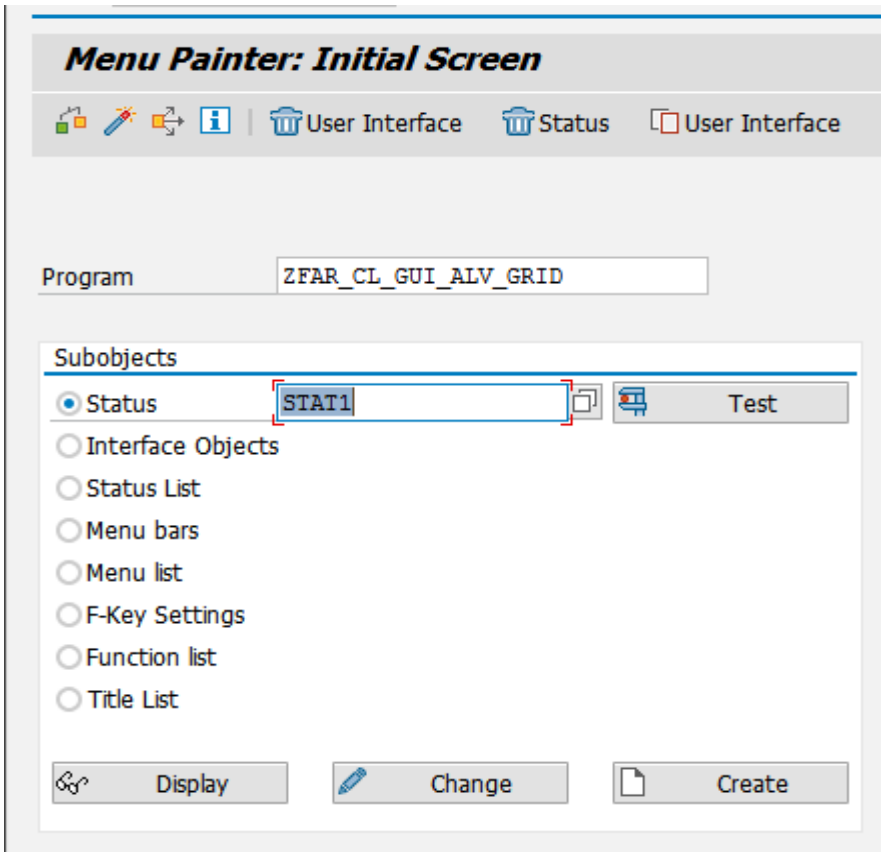
```
MODULE status_0100 OUTPUT.
```

```
  SET PF-STATUS 'STAT1'.

  IF g_cust IS INITIAL.

    CREATE OBJECT g_cust
      EXPORTING
        container_name              = 'CUSTCONT'
      EXCEPTIONS
        cntl_error                  = 1
        cntl_system_error           = 2
        create_error                = 3
        lifetime_error              = 4
        lifetime_dynpro_dynpro_link = 5
        OTHERS                      = 6.

    IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
                 WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
    ENDIF.

    CREATE OBJECT g_alv
      EXPORTING
        i_parent          = g_cust
      EXCEPTIONS
        error_cntl_create = 1
        error_cntl_init   = 2
        error_cntl_link   = 3
        error_dp_create   = 4
        OTHERS            = 5.

    IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
                 WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
    ENDIF.

    PERFORM load_data.
    PERFORM prepare_fcat.

    CALL METHOD g_alv->set_table_for_first_display
      CHANGING
        it_outtab                     = gt_kna1
        it_fieldcatalog               = gt_fcat
      EXCEPTIONS
        invalid_parameter_combination = 1
        program_error                 = 2
        too_many_lines                = 3
        OTHERS                        = 4.

    IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
                 WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
    ENDIF.

  ENDIF.

ENDMODULE.
```

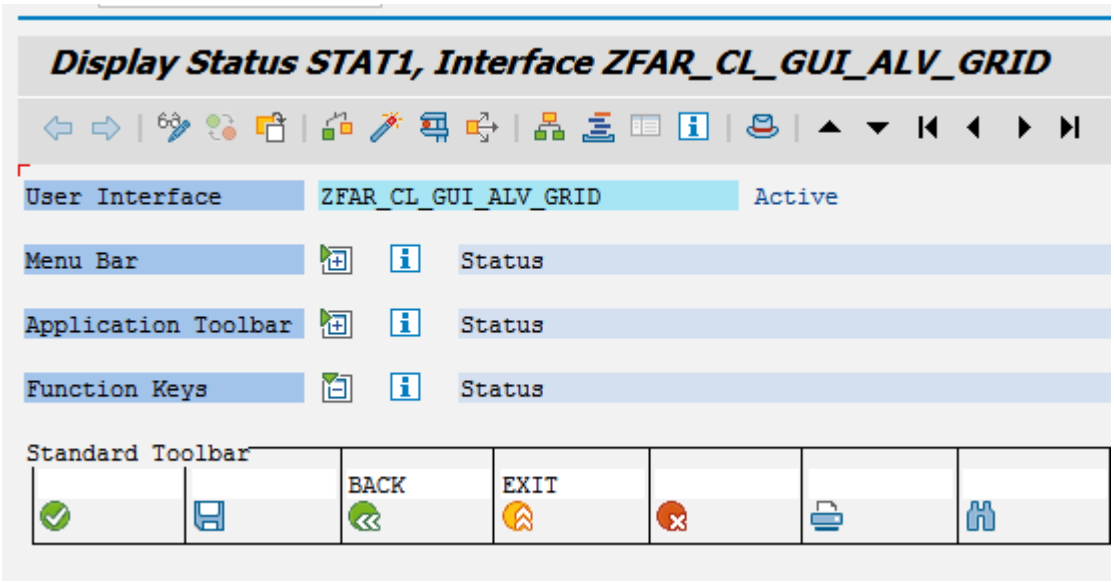
# Crear un Status GUI

Ahora debe crear un Status para nuestra pantalla, de modo que pueda presionar el botón Atrás y salir de la pantalla de salida.

Vaya a SE41 y cree un nuevo estado con el nombre `STAT1`.



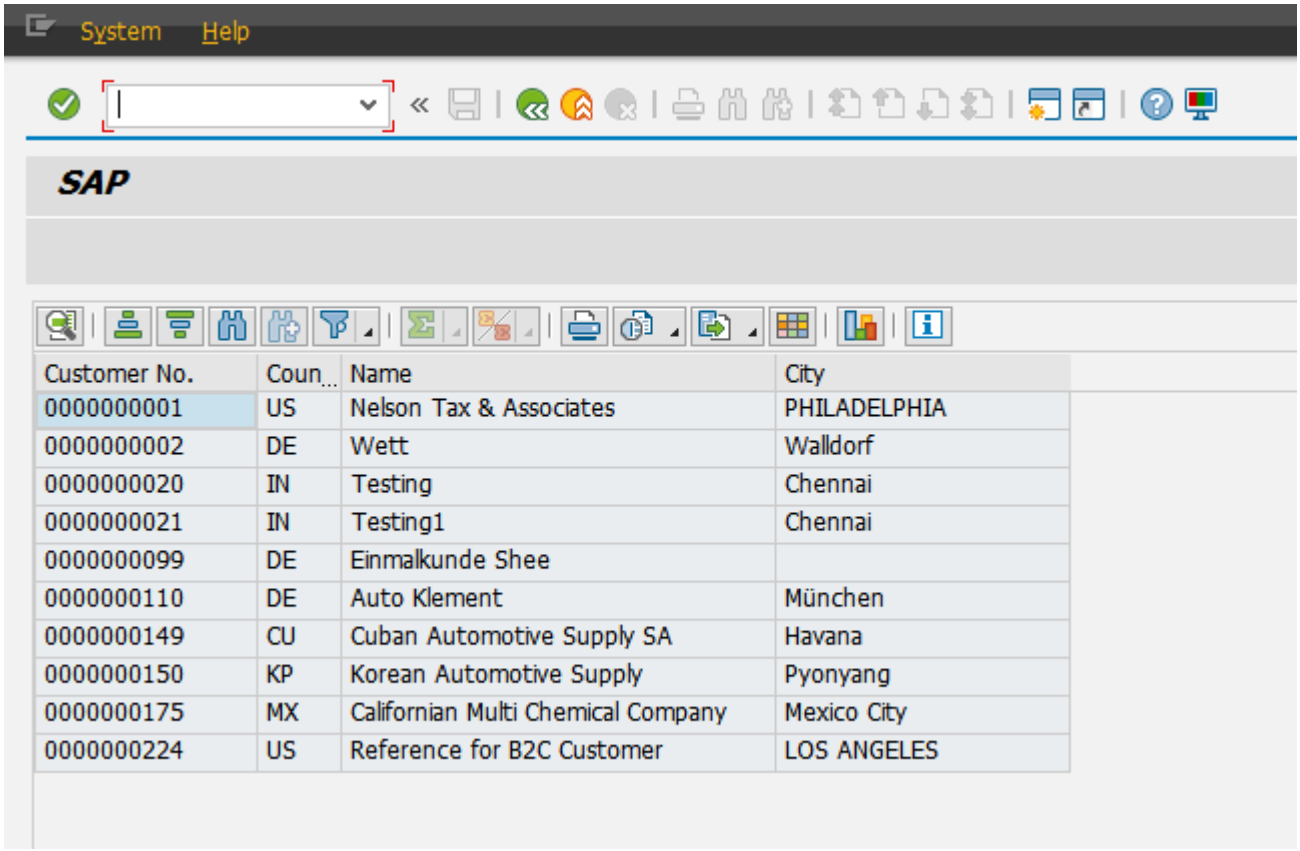

Y en la sección Teclas de función, agregue `BACK` y `EXIT` como se muestra a continuación.




BACK y EXIT deben manejarse en el evento PAI de la siguiente manera.

```
MODULE user_command_0100 INPUT.

  CASE okcode.
    WHEN 'BACK'.
      LEAVE TO SCREEN 0.
    WHEN 'EXIT'.
      LEAVE PROGRAM.
  ENDCASE.

ENDMODULE.
```

Ese es el final. Ahora podrá obtener la salida que queríamos.




| Customer No. | Coun... | Name | City |
|---|---|---|---|
| 0000000001 | US | Nelson Tax & Associates | PHILADELPHIA |
| 0000000002 | DE | Wett | Walldorf |
| 0000000020 | IN | Testing | Chennai |
| 0000000021 | IN | Testing1 | Chennai |
| 0000000099 | DE | Einmalkunde Shee | |
| 0000000110 | DE | Auto Klement | München |
| 0000000149 | CU | Cuban Automotive Supply SA | Havana |
| 0000000150 | KP | Korean Automotive Supply | Pyonyang |
| 0000000175 | MX | Californian Multi Chemical Company | Mexico City |
| 0000000224 | US | Reference for B2C Customer | LOS ANGELES |

Aquí está el programa completo.

```
REPORT zfar_cl_gui_alv_grid.

DATA: g_alv   TYPE REF TO cl_gui_alv_grid,
      g_cust  TYPE REF TO cl_gui_custom_container,
      gt_kna1 TYPE TABLE OF kna1,
      gt_fcat TYPE lvc_t_fcat,
      gs_fcat TYPE lvc_s_fcat,
      okcode  TYPE sy-ucomm.



START-OF-SELECTION.

  SET SCREEN 0100.

END-OF-SELECTION.


*&---------------------------------------------------------------------*
*&      Module  STATUS_0100  OUTPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE status_0100 OUTPUT.
```

```
SET PF-STATUS 'STAT1'.

IF g_cust IS INITIAL.

  CREATE OBJECT g_cust
    EXPORTING
      container_name              = 'CUSTCONT'
    EXCEPTIONS
      cntl_error                  = 1
      cntl_system_error           = 2
      create_error                = 3
      lifetime_error              = 4
      lifetime_dynpro_dynpro_link = 5
      OTHERS                      = 6.

  IF sy-subrc <> 0.


    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
               WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.


  ENDIF.

  CREATE OBJECT g_alv
    EXPORTING
      i_parent          = g_cust
    EXCEPTIONS
      error_cntl_create = 1
      error_cntl_init   = 2
      error_cntl_link   = 3
      error_dp_create   = 4
      OTHERS            = 5.

  IF sy-subrc <> 0.


    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
               WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.


  ENDIF.

  PERFORM load_data.


  PERFORM prepare_fcat.

  CALL METHOD g_alv->set_table_for_first_display
    CHANGING
      it_outtab                     = gt_kna1
      it_fieldcatalog               = gt_fcat
    EXCEPTIONS
      invalid_parameter_combination = 1
      program_error                 = 2
      too_many_lines                = 3
      OTHERS                        = 4.

  IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
               WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.
```

```
  ENDIF.

ENDMODULE.
*&---------------------------------------------------------------------*
*&      Form  LOAD_DATA
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM load_data.

  SELECT * FROM kna1
    INTO CORRESPONDING FIELDS OF TABLE gt_kna1
      UP TO 10 ROWS.

ENDFORM.
*&---------------------------------------------------------------------*
*&      Form  PREPARE_FCAT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM prepare_fcat.

  DEFINE add_fcat.

    CLEAR gs_fcat.

    gs_fcat-col_pos   = &1.
    gs_fcat-fieldname = &2.
    gs_fcat-coltext   = &3.
    gs_fcat-outputlen = &4.

    APPEND gs_fcat TO gt_fcat.

  END-OF-DEFINITION.

  add_fcat:
     1 'KUNNR' 'Customer No.' 15,
     2 'LAND1' 'Country'      5,
     3 'NAME1' 'Name'         30,
     4 'ORT01' 'City'         20.

ENDFORM.


*&---------------------------------------------------------------------*
*&      Module  USER_COMMAND_0100  INPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE user_command_0100 INPUT.

  CASE okcode.
    WHEN 'BACK' OR 'CANCEL'.
      LEAVE TO SCREEN 0.
    WHEN 'EXIT'.
      LEAVE PROGRAM.
  ENDCASE.

ENDMODULE.
```