Aprende ABAP de forma práctica

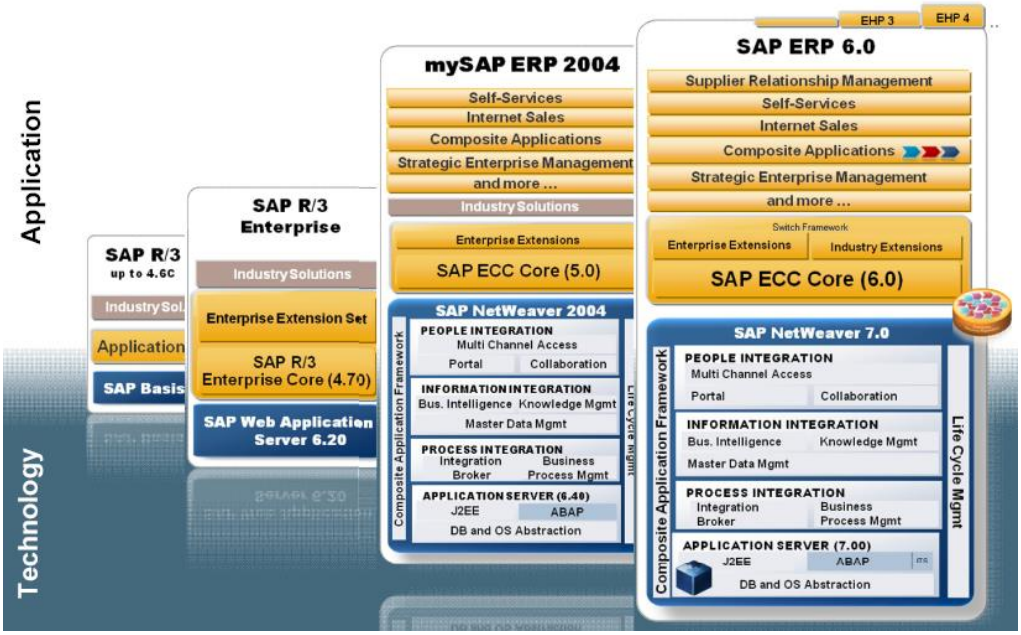
Una guía elemental para el aprendizaje del lenguaje de programación de SAP

CONTENIDO

I. Introducción

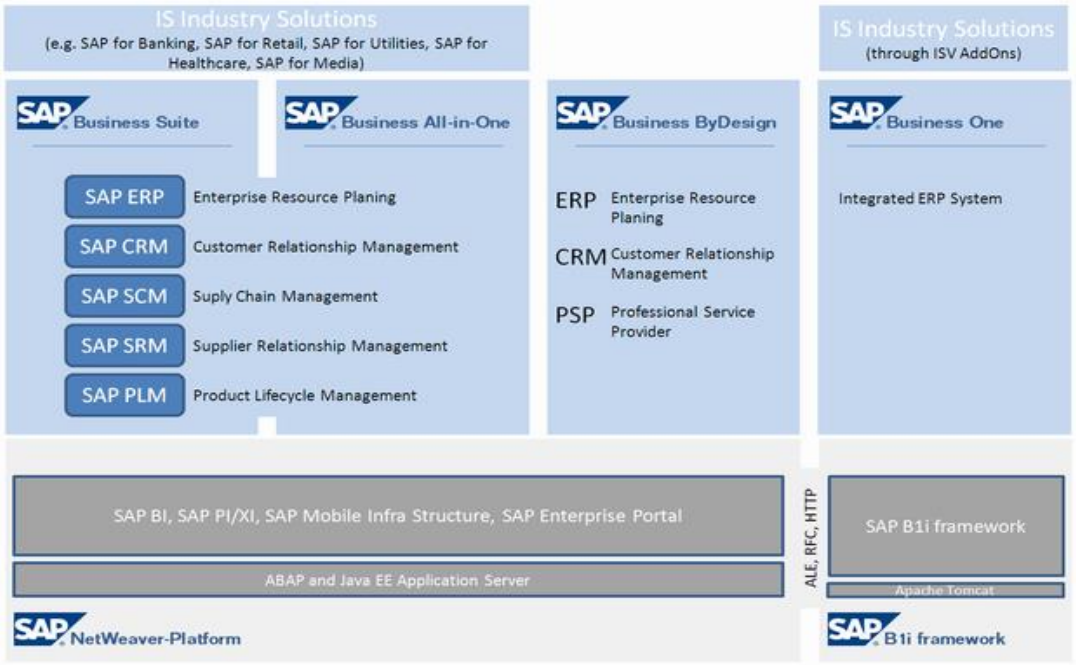
1. *Reseña de SAP.*

SAP ofrece modularidad, pudiendo instalar solamente aquellas funcionalidades que se quieren o desean gestionar en la empresa. Los módulos más comúnmente instalados son SD (Ventas y Distribución), HR (Recursos Humanos), FI (Finanzas) y PP (Plan de Producción). Además ofrece productos integrados con los anteriores para ventas por Internet (CRM), gestión de almacenes (WM), gestión documental, etc.
SAP ofrece además capacidad de adaptación a las necesidades de la empresa. Es totalmente configurable la información y el modo de gestionarla para adaptarla a nuestros procesos. Podemos incluir o excluir grupos de campos según nuestras necesidades, automatizar procesos en los que no deseamos intervención e incluso adaptar las pantallas a nuestros requisitos.
Además hay que considerar el equipo de Mantenimiento (OSS) al servicio de los clientes SAP para la detección y solución de errores.



2. *Soluciones SAP*

Debido a que no hay una solución genérica para todas las industrias, se crearon soluciones especiales para los procesos específicos de varias de ellas. Se implementa para robustecer y complementar el soporte que brinda una Solución SAP a un sector económico en particular.



3. *Módulos SAP.*

Las aplicaciones o módulos funcionales de SAP están divididos en áreas: financiera, logística, recursos humanos, que están altamente integrados. Además de las aplicaciones específicas de negocios, existen otros componentes especiales de SAP, llamados soluciones industriales (IS), que interactúan con los módulos estándar y que están orientados a industrias específicas. También existe un grupo de módulos o componentes cruzados de aplicación conocido como módulos CA.

Los módulos de aplicación de SAP incluyen cientos de procesos predefinidos de negocios, preparados para ser adaptados a negocios y a necesidades específicos de información de las empresas.

Módulos de SAP:

- Production Planning (PP).
- Sales and Distribution (SD).
- Financial Accounting (FI).
- Controlling (CO).
- Material Management (MM).
- Human Resources (HR).
- Quality Assurance (QA).
- Asset Management (AM).
- Plant Maintenance (PM).
- Project System (PS).
- Office and Communications (OC).

4. *Arquitectura de sistema (BASIS).*

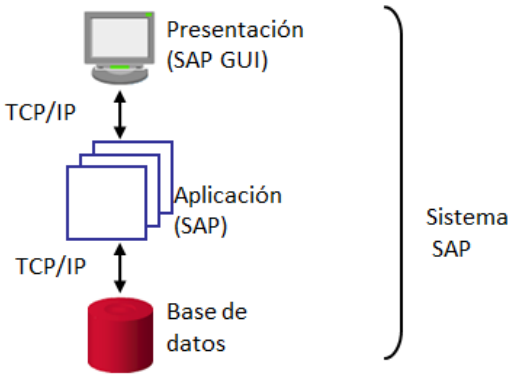
En el sistema SAP existen muchos sistemas, servicios, impresoras, usuarios, etc., que se necesitan distribuir, por ello SAP está basado en el concepto cliente/servidor, para poder compartir de manera óptima los servicios disponibles entre los clientes y los servidores para optimizar el rendimiento, la disponibilidad y el balanceo de las cargas del sistema. Para ello necesitamos utilizar los siguientes términos:

> Identificador de sistema SAP (SID).- Conocido como SID o SAPSID, es un código de tres letras o dígitos que identifica unívocamente a un sistema SAP.
> Mandante.- Se define como unidad independiente dentro del sistema SAP desde el punto de vista fiscal, legal y organizativo.
> Transacción.- Una transacción es una operación que permite al usuario realizar cambios mediante una secuencia de pasos relacionados de modo lógico. Todo el sistema SAP puede considerarse como un sistema de procesamiento de transacciones de negocio, porque todo el flujo de datos que fluye entre los módulos de las aplicaciones se ejecutan utilizando transacciones.
> Dispatcher.- Es el programa de control que gestiona el intercambio de información de los recursos de las aplicaciones SAP, funciona como un monitor de transacciones que recibe pantallas y datos, y los pasa a los correspondientes procesos de trabajo; un proceso de trabajo es un servicio ofrecido por un servidor y solicitado por un cliente.
> Instancia.- Una instancia SAP es una entidad administrativa que agrupa a varios componentes de SAP que proporciona uno o varios servicios.

Una de las características importantes y factor clave del éxito de SAP es que es un sistema abierto, lo cual significa que las aplicaciones SAP pueden trabajar sobre distintos sistemas operativos, diversos sistemas de bases de datos y protocolos de comunicación; este tipo de tecnología permite que los clientes SAP tengan un cierto grado de independencia de sus proveedores de hardware y base de datos. SAP es compatible con diversas plataformas de hardware y sistemas operativos, tales como la mayoría de las versiones UNIX, Windows NT, AS/400, OS/390, Linux, y con un gran número de interfaces gráficas de usuario como son, todas las versiones de Windows, Macintosh, OS/2, Motif, exploradores de Internet. A nivel base de datos, SAP es compatible con Oracle, Microsof SQL-Server, Informix, DBASE y variantes de IBM.

CAPAS/LAYERS: En SAP distingue tres tipos de servidores:

- **Servidor de base de datos**.- Contiene el sistema de base de datos para todos los servidores.
- **Servidor de aplicación**.- Ejecutan los servicios de SAP con un dispatcher; ejecuta los programas ABAP.
- **Servidor de presentación**.- Ejecuta la interfaz gráfica para el usuario (Front End) de SAP.



PANORAMA/LANDSCAPE: SAP recomienda idealmente la implementación de tres sistemas:

- **Sistema de desarrollo**.- Donde tiene lugar el desarrollo y parametrización.
- **Sistema de pruebas**.- Donde puede probarse previamente el trabajo, simulando operaciones.
- **Sistema productivo**.- Donde los usuarios finales trabajan con operaciones verdaderas y datos reales.

5. *Reseña de ABAP*

Advanced Business Aplication Programming 4th Generation

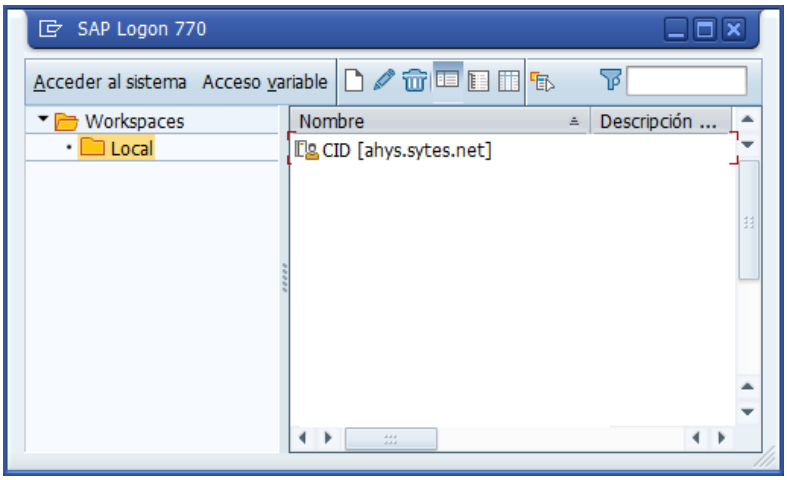
ABAP es un lenguaje de programación de 4a. Generación (4GL) orientado a cliente servidor tal como su definición específica, al desarrollo de aplicaciones de negocios.

Características

- Distribución de las aplicaciones entre distintos servidores.
- Soporte de interfaces gráficos comunes y estándar.
- Comunicación transparente con otros sistemas.
- Manejo transparente y abierto del sistema de gestión de la base de Datos.
- Comunicación con aplicaciones externas a través de RFC o de aplicaciones de PC.
- Todas las soluciones de negocio e industria de SAP están desarrolladas sobre servidores ABAP.

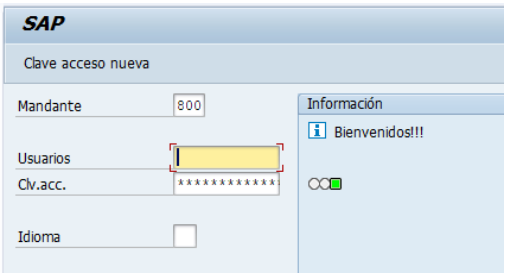
6. Acceso a SAP

Todo comienza con el *Logon*, un pequeño selector de servidores donde vamos a poder trabajar. En la figura vemos que nuestro *Logon* está preparado para trabajar en distintas instalaciones de SAP. En este programa, podemos crear, modificar y acceder a una conexión.
A partir de aquí y pulsando sobre la conexión deseada accedemos a SAP.



El acceso a SAP está compuesto por cuatro campos de entrada:

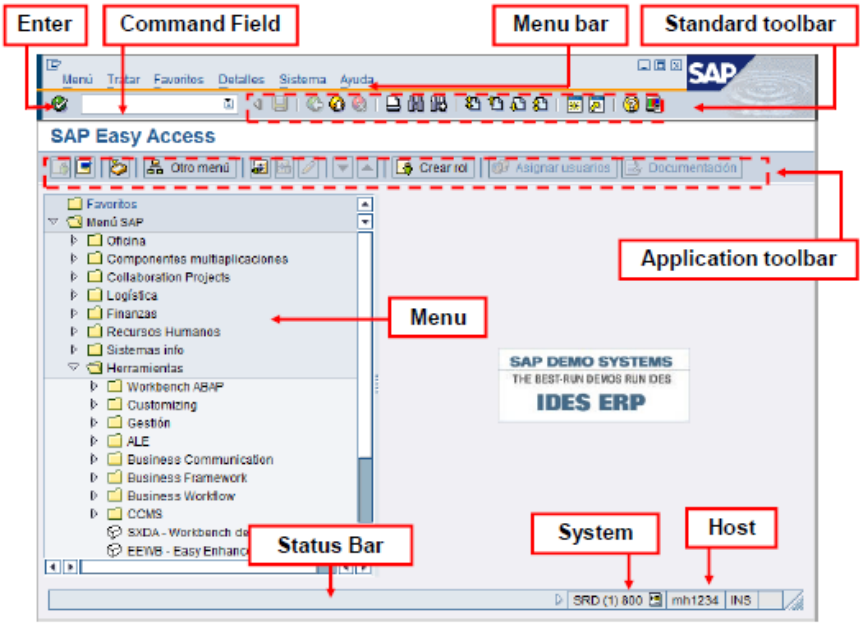
1. **Mandate (Client)** Identifica claramente con qué tipo de información vamos a trabajar.
2. **Usuario (User)** El usuario con el cuál vamos a acceder
3. **Clave (Password)** Palabra Clave relacionada con el usuario
4. **Idioma (Language)** Idioma para la Interfase



Elementos de la Interfase

(1) Barra de menú. En estos menús se muestran todas las opciones disponibles para la aplicación que se esté ejecutando.
(2) Línea de comandos. Aquí se ingresan los códigos de transacción**.**
(3) Barra de Titulo. Muestra el título de la aplicación que se esté ejecutando.
(4) Barra de estado. Muestra mensajes al momento de ejecución y el estado del sistema.
(5) Herramientas estándar.

(6) Herramientas de aplicación.



Transacción

Una transacción en un sistema de gestión, es un conjunto de órdenes que se ejecutan formando una unidad de trabajo, es decir, en forma indivisible o atómica. En SAP también se conoce como transacción al método abreviado de llamar a estas órdenes (los programas). En algunos casos son la única forma de llamarlos.

Modo

Un modo de trabajo es una ventana de SAP abierta. Solamente pueden abrirse 6 modos por cada Logon que se realice. Cada modo tiene su propia memoria de trabajo pero comparte una única zona de memoria por Logon, cada Logon es independiente entre sí.

Línea de comandos

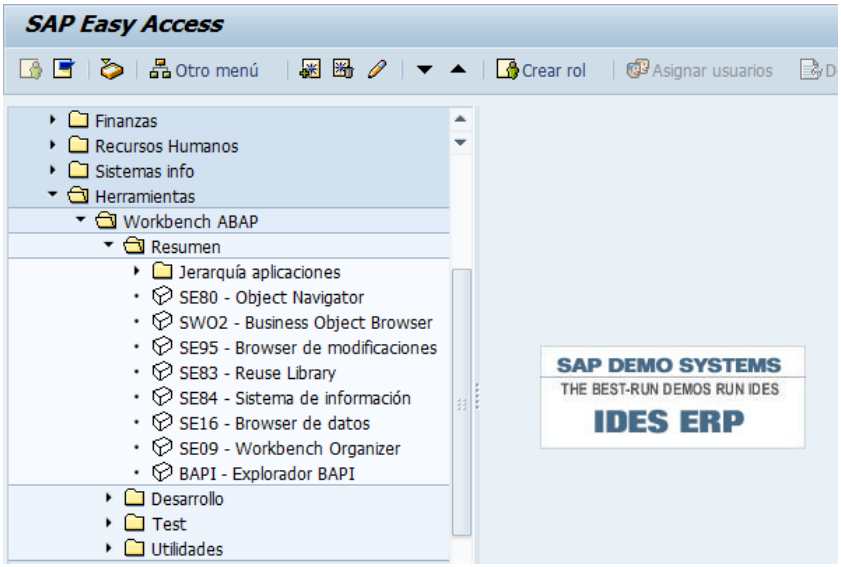
La casilla de texto superior izquierda, que se ubica en todas las pantallas de SAP, sirve para ingresar códigos que son comandos y transacciones.

- Los códigos de transacción se escriben directamente en esta casilla.
- El comando **/n** se usa para terminar una transacción en ejecución.
- El comando **/o** se usa para abrir un nuevo modo.
- Estos comandos se pueden combinar con los códigos de transacción por ejemplo /nse38 llama a la transacción se38 (editor ABAP) dentro del mismo modo y /ose38 llama a la transacción en un nuevo modo.
- Para ayuda sobre más comandos use la tecla F1 sobre esta casilla.

II. El ambiente de desarrollo

1. *ABAP Workbench*

El entorno de desarrollo para ABAP y sus componentes de programación están contenidos en el ABAP Workbench.
En la figura se pueden distinguir algunas herramientas del entorno de desarrollo de ABAP.



Como se puede ver varias las herramientas de desarrollo disponibles. A todas ellas se puede llegar desde el menú SAP. A algunas de ellas se pueden llegar navegando desde otras transacciones. La forma más rápida de acceder a las principales es mediante su código de transacción.

Browser (SE80) Con este sencillo navegador vamos a poder trabajar con todos y cada uno de los diferentes objetos que posee SAP: tablas, vistas, listados, informes, transacciones.
Editor ABAP (SE38) Editor de código ABAP.
Diccionario de datos (SE11) Mantenimiento de tablas, vistas, objetos de autorización, elementos de datos, etc.
Browser de datos (SE16/SE16n) Ver el contenido de tablas.
Biblioteca de funciones (SE37) Mantenimiento de módulos de funciones.
Generador de clases (SE24) Mantenimiento de clases

2. *RICEF*

Es un acrónimo usado en SAP. Estos describen 5 áreas de desarrollos. Estas letras se refieren a Reportes, Interfaces, Conversiones, Ampliaciones, Formularios.

R- Reports. Todos los objetos que tienen que ver con reportes, esto incluye simples reportes donde se usa la sentencia WRITE, Simple ALV, ALV Grid, ALV OO, etc.
I – Interfaces. Interfaces son desarrollos ALE/IDOCs. Esto incluye los desarrollos y la configuración de IDOC.

C – Conversion. Se refiere a programación BDC. Datos extraídos de sistemas Legacy en formato plano deben ser convertidos a formato SAP. Incluyen BDC, LSMW, BAPI, etc.
E – Enhancements. Las ampliaciones se refieren a User Exits, Customer Exits, BADI's etc.
F – Forms. Incluye SAP Smartforms, SAP Scripts.

Todas las anteriores implican el uso del lenguaje de programación ABAP. Con algunas diferencias por ejemplo el desarrollo de Formularios requiere un ABAP específico para SAP Script, el desarrollo de Pantallas requiere un ABAP específico para Dynpros, la definición y/o implementación de clases y métodos requiere de un ABAP específico Orientado a Objetos.

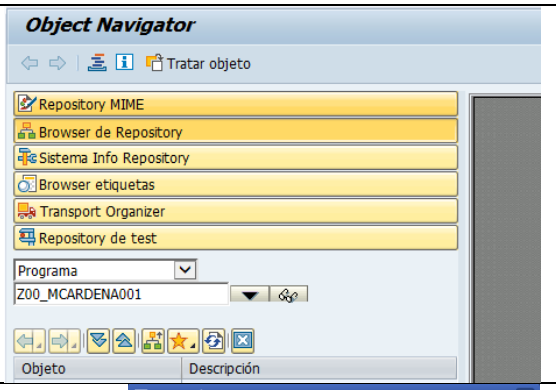
3. Componentes de Usuario (Y o Z)

Todos los componentes (programas, tablas, funciones, etc.) que necesitemos crear los vamos a crear con nombres que inicien con las letras Y o Z. De ahí que se use el término programas Z o tablas Z para referirse a programas y tablas creados por el usuario o cliente, y diferenciarlos de los programas y tablas ESTANDAR que son los creados por SAP que vienen incluidos en el sistema al momento de su instalación.

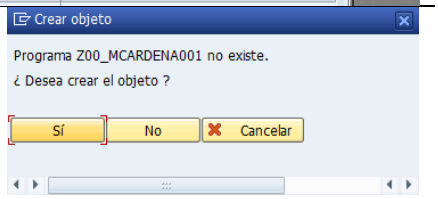
4. Primer programa

Vamos a seguir los pasos básicos para crear un nuevo programa. En este punto no analizamos todas las opciones posibles ni los parámetros o atributos disponibles. El objetivo es crear un nuevo programa de la forma más rápida y sencilla.

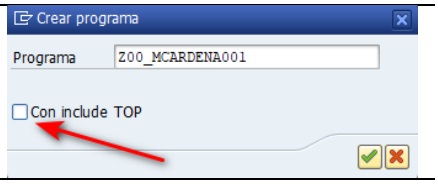
Usamos la transacción SE80. Aparece la pantalla inicial del navegador de objetos.



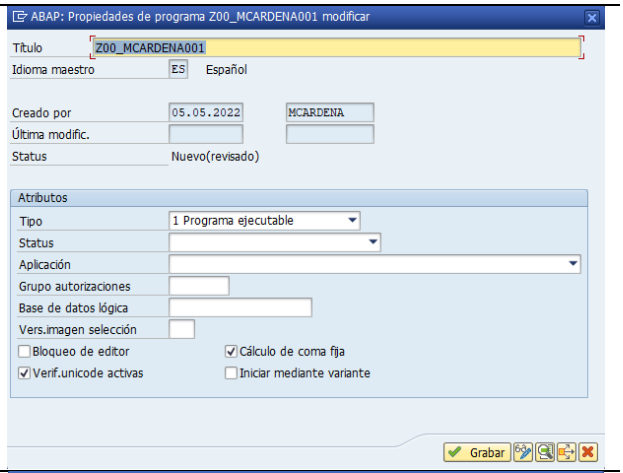
Elegimos programa y ponemos el nombre de nuestro programa. Damos ENTER y nos muestra el mensaje



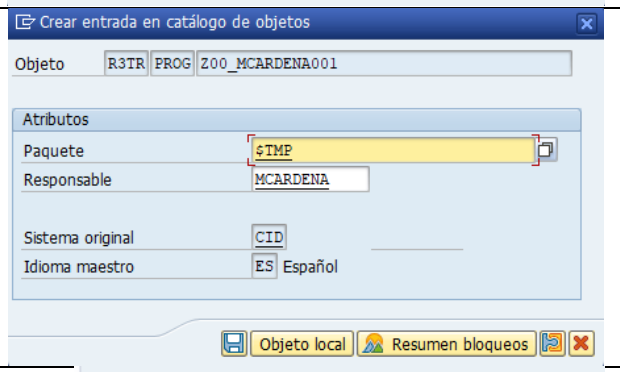
En la siguiente pantalla desmarcamos la casilla "Con include TOP"



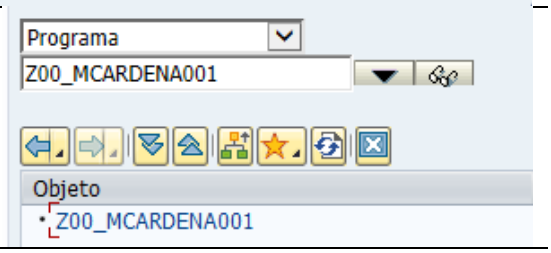
Aquí simplemente hacemos clic en el botón Grabar



Como nos aparece por defecto el paquete $TMP podemos dar clic en el icono de grabar o en el botón Objeto local.

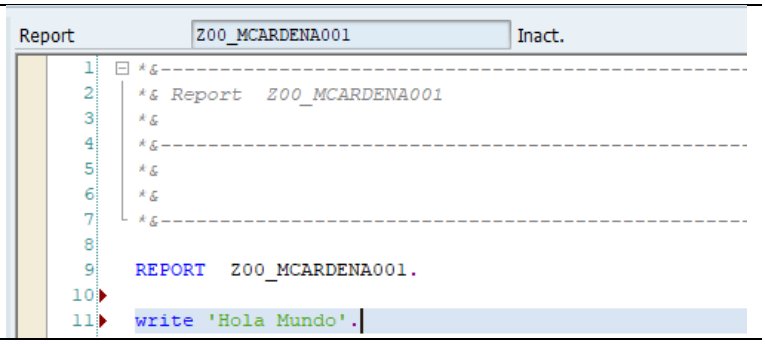


Ahora damos doble clic en el nombre del programa que se muestra en la lista de objetos.

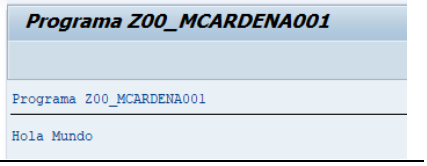


En este punto debemos identificar 4 iconos importantes en el menú:
1.- Visualizar/Modificar
2.- Verificar
3.- Activar
4.- Ejecutar

Hecho esto aparece el área del editor. Simplemente escribimos write 'hola mundo'.



Para ejecutar este programa usamos la tecla de función F8.
Muestra el siguiente resultado:

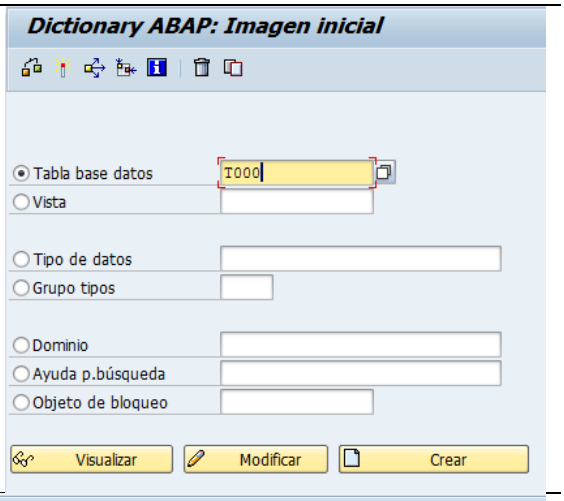


5. *Un vistazo al diccionario de datos y tablas*

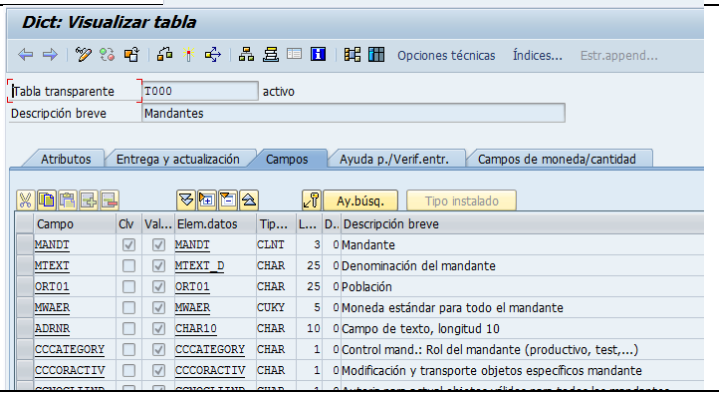
Para dar un vistazo a la BD desde SAP contamos principalmente con dos transacciones SE11 para ver el diccionario de datos y SE16 para ver el contenido de las tablas. En esta parte no analizamos el funcionamiento ni los detalles de estas transacciones. Simplemente nos servirá para reconocer donde y que datos se guardan y más adelante entender los ejemplos prácticos.

Ver estructura de Tabla (SE11)

Usamos la transacción SE11.
Aparece la pantalla inicial del diccionario de datos.
Elegimos la tabla T000 y damos clic a Visualizar.

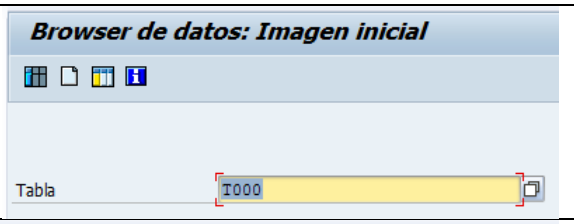


Y a continuación vemos la estructura de la tabla.

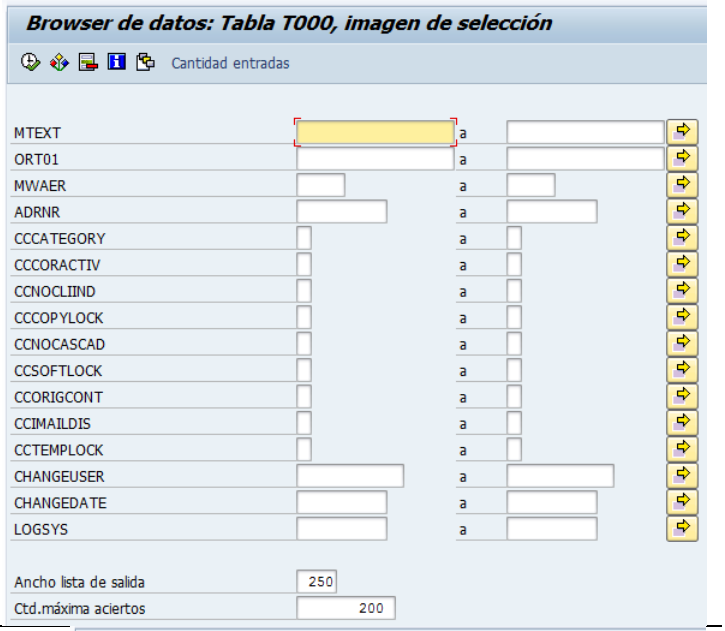


Ver contenido de Tabla (SE16)

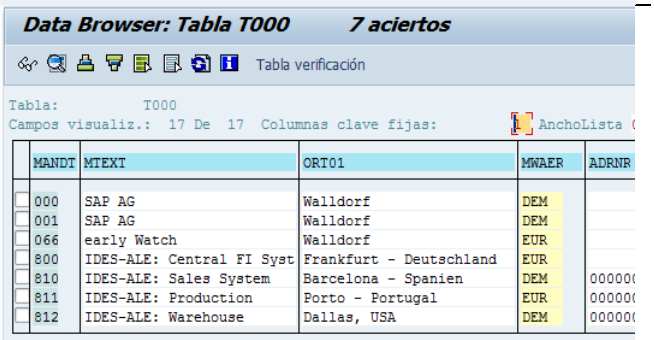
Usamos la transacción SE16.
Aparece la pantalla inicial del browser de datos.
Elegimos la tabla T000 y damos ENTER.



Nos muestra la Imagen de selección, donde ponemos restringir la búsqueda.
Si cambiar nada elegimos Ejecutar (F8)



Nos muestra el contenido de la tabal



6. *¿Donde encuentro ayuda?*

Existe mucha ayuda disponible:

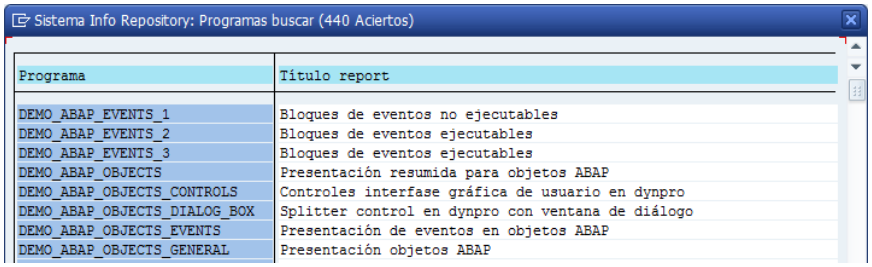
La página oficial de SAP:
http://help.sap.com/abapdocu_70/en/index.htm

El foro SAP Community Network Forums» ABAP Development
http://forums.sdn.sap.com/

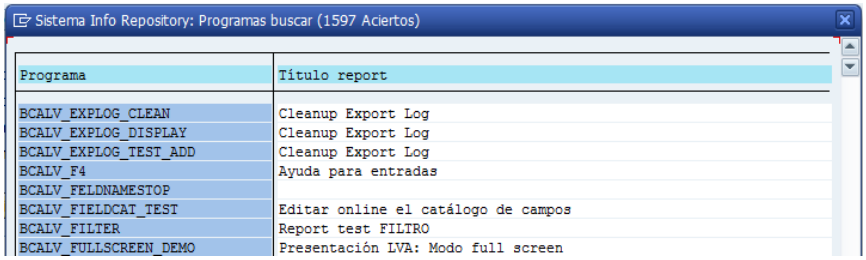
Dentro del mismo sistema las transacciones

- ABAPDOCU
- ABAPHELP
- BIBS
- DWDM

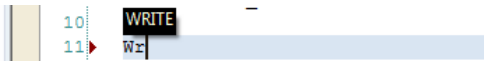
Los programas ejemplo *DEMO*



Los programas ejemplo BC*



El autocompletado en el editor ABAP

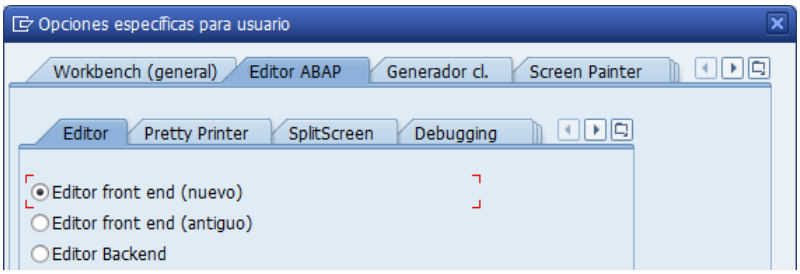


7. *Leer y depurar código ABAP*

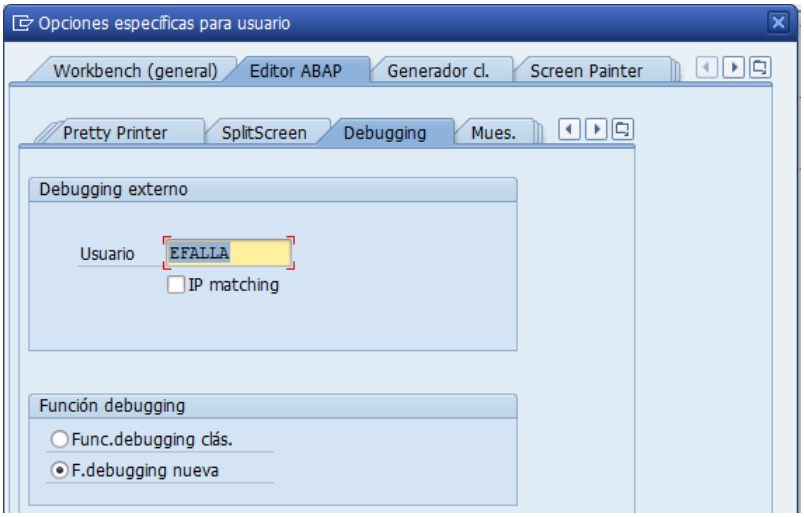
Para esta tarea vamos a usar las transacciones SE93, SE38, SE37 y SE80 para abrir unos tipos de programa ABAP. Existen otras transacciones para revisar código ABAP como el editor de clases, el editor de BADIS y otros.

Configurar el Editor ABAP
Empecemos por configurar y entender algo más del editor ABAP:

Dentro del Editor ABAP vamos al menú Utilidades>Opciones:
En la pestaña Editor ABAP > Editor, elegimos el Editor front end (nuevo):

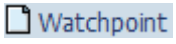


En la pestaña Editor ABAP > Debugging, elegimos el F.debugging nueva (nuevo):



Esto nos va a facilitar la escritura de código y las tareas de depuración de código (Debug).

DEPURAR PROGRAMA (DEBUG)
Para iniciar la depuración ponemos /h en la línea de comando.
Dentro de la pantalla de depuración identificamos las opciones:

- Paso a paso (F5) 
- Ejecutar (F6) 
- Retornar (F7) 
- Continuar (F8) 
- Crear breakpoint (F9) 
- Crear watchpoint (Shift+F4) 

Ejercicio aplicación 1

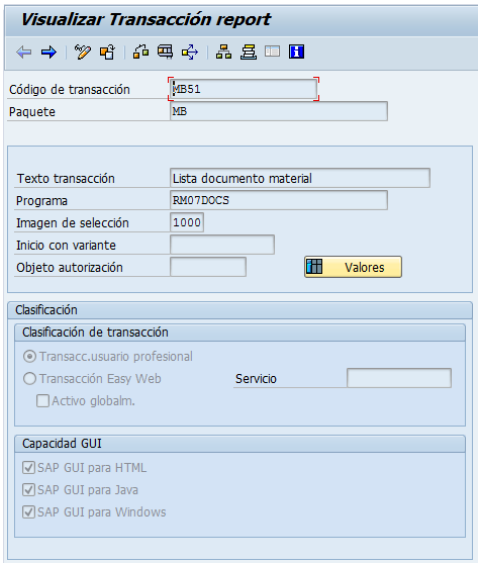
PROCESO: Lista documento material

Mediante este reporte obtendrá una lista de los documentos de material que se han contabilizado para un material o para varios materiales. Desde la lista de salida podrá visualizar un documento de material.

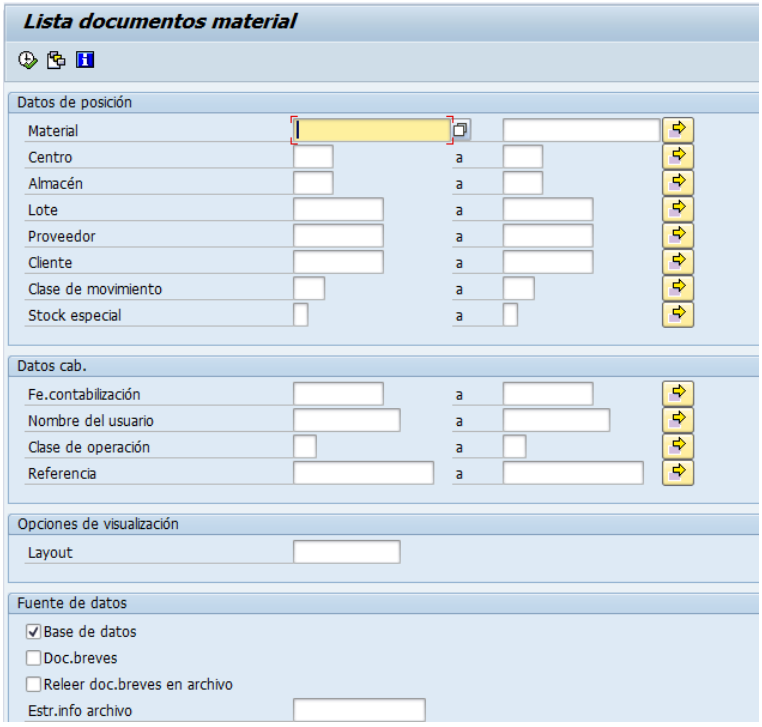
Transacción: MB51

Programa: RM07DOCS

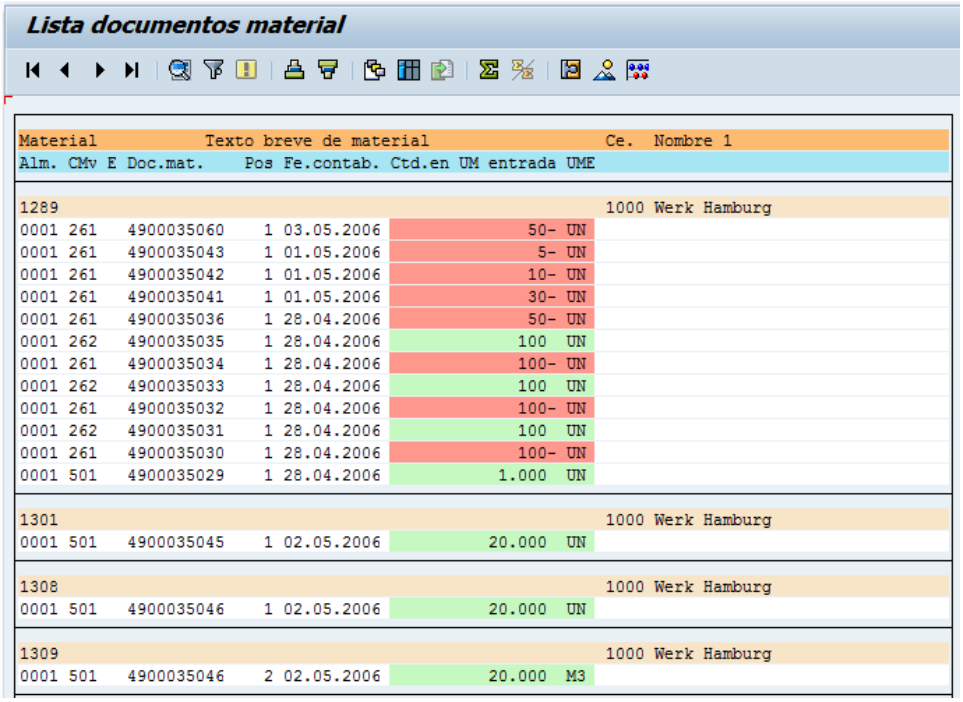
Con la transacción SE93 vemos la transacción y sus características:



Con la transacción MB51 vemos la pantalla inicial del programa:



Completamos como datos Centro = 1000 y Almacén = 0001.
Y al ejecutar tenemos como resultado:



Lista documentos material

Material	Texto breve de material					Ce.	Nombre 1
Alm.	CMv	E	Doc.mat.	Pos	Fe.contab.	Ctd.en UM entrada	UME		
1289								1000	Werk Hamburg
0001	261		4900035060	1	03.05.2006	50-	UN		
0001	261		4900035043	1	01.05.2006	5-	UN		
0001	261		4900035042	1	01.05.2006	10-	UN		
0001	261		4900035041	1	01.05.2006	30-	UN		
0001	261		4900035036	1	28.04.2006	50-	UN		
0001	262		4900035035	1	28.04.2006	100	UN		
0001	261		4900035034	1	28.04.2006	100-	UN		
0001	262		4900035033	1	28.04.2006	100	UN		
0001	261		4900035032	1	28.04.2006	100-	UN		
0001	262		4900035031	1	28.04.2006	100	UN		
0001	261		4900035030	1	28.04.2006	100-	UN		
0001	501		4900035029	1	28.04.2006	1.000	UN		
1301								1000	Werk Hamburg
0001	501		4900035045	1	02.05.2006	20.000	UN		
1308								1000	Werk Hamburg
0001	501		4900035046	1	02.05.2006	20.000	UN		
1309								1000	Werk Hamburg
0001	501		4900035046	2	02.05.2006	20.000	M3		

Cuestionario

(1) Mencione 10 de las tablas que consulta el programa (vía sentencia SELECT).

1 _____ 6 _____
2 _____ 7 _____
3 _____ 8 _____
4 _____ 9 _____
5 _____ 10 _____

(2) Verifique la descripción de la tabla en el diccionario de datos.

(3) Detenga en programa en la rutina output_list. Indique que valores se asignan a los campos:

alv_keyinfo-header01 = _____
alv_keyinfo-header02 = _____
alv_keyinfo-header03 = _____
alv_keyinfo-item01 = _____
alv_keyinfo-item02 = _____
alv_keyinfo-item03 = _____

III. Elementos básicos del lenguaje

1. *Comandos básicos y campos de sistema*

WRITE. Escribe textos, números o cualquier dato en pantalla. Actualmente se usa principalmente para dar formato a un texto antes de ingresarlo en una variable.
GET. Recupera valores, dependiendo de cómo se usa sirve para actualizar las variables de tiempo: GET TIME, la posición del cursor: GET CURSOR, una variable de memoria GET MEMORY ID.
SET. Establece valores, dependiendo de cómo se usa puede establecer el valor de una variable de memoria SET MEMORY ID.
MOVE. En la forma más básica tiene el mismo efecto que asignar un valor con el símbolo igual "=". Además permite la asignación en tipos de variables diferentes.
SY-SUBRC. Variable del sistema de tipo estructura que almacena una diversidad de datos como por ejemplo el ID de usuario, fechas, tiempos (hora), servidor.

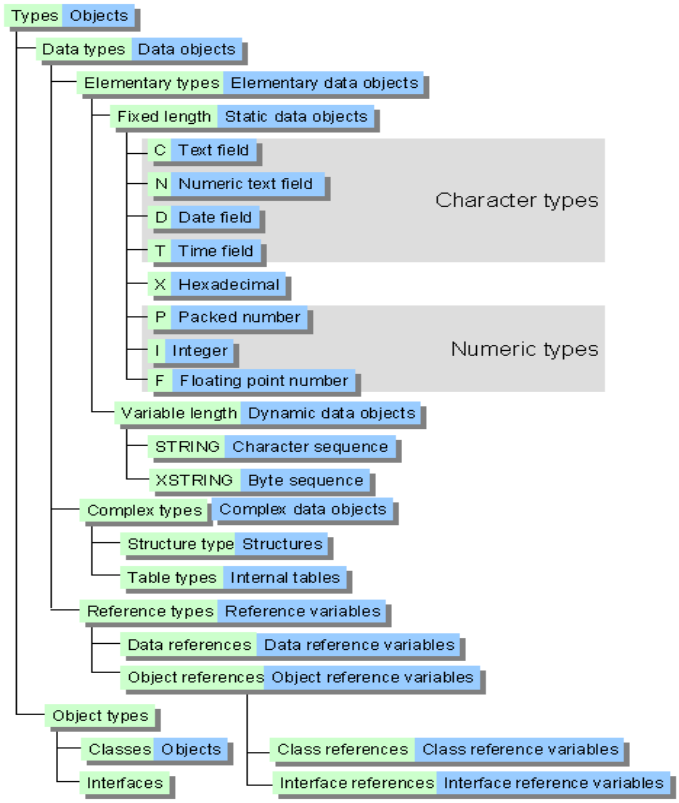
Ejemplo de aplicación

```
1   REPORT Z00_EFRCODE002 .
2
3   DATA:
4   * Date & time information variables
5   l_timestamp TYPE timestampl,
6   l_string(30) type c.
7
8   * Refresh all internal data & time fields like sy-uzeit. etc.
9   GET TIME.
10  * Get exact time stmp upo to nanoseconds and convert
11  * to string
12  GET TIME STAMP FIELD l_timestamp.
13  WRITE l_timestamp TIME ZONE sy-zonlo to l_string.
14  * Write information to screen
15  WRITE: / , / 'Date & Time Information',
16         / 'System Date: ', 30 sy-datum,
17         / 'System Time:', 30 sy-uzeit,
18         / 'Exact System Time Stamp: ', 30 l_string,
19         / 'Daylight Saving Time: ', 30 sy-dayst as checkbox input off,
20         / 'Local Date: ', 30 sy-datlo,
21         / 'Local Time: ', 30 sy-timlo,
22         / 'Local Time Zone: ', 30 sy-zonlo.
23  WRITE: / , / 'Login Information',
24         / 'Logical SAp System: ', 30 sy-sysid,
25         / 'Client:', 30 sy-mandt,
26         / 'User name: ', 30 sy-uname.
27  WRITE: / , / 'System & Program Information',
28         / 'SAP WebAS Name: ', 30 sy-host,
29         / 'Database Name', 30 sy-dbsys,
30         / 'Operating System Name: ', 30 sy-opsys,
31         / 'Program Transaction: ', 30 sy-tcode,
32         / 'Program Name: ', 30 sy-repid.
```

2. *Tipos de datos*

ABAP posee algunos tipos de datos elementales sobre los cuales se pueden crear otros muy complejos. De ser necesario, también pueden definirse tipos de datos dentro del programa mismo, de manera de ampliar el espectro de datos posibles de utilizar.

Como la mayoría de los lenguajes modernos, ABAP tiene un conjunto de tipos de datos primitivos sobre los cuales se crean los tipos de datos más complejos. Los tipos de datos predefinidos de ABAP:



En ABAP usamos los términos campos, estructura y variable para hacer referencia a variables teniendo en cuenta la dimensión de la misma.

- **CAMPO** valor individual.
- **ESTRUCTURA** arreglo lineal,
- **TABLA INTERNA** arreglo de 2 o mas dimensiones

Los tipos de datos pueden declararse en el Diccionario de Datos (transacción SE11), o a nivel de Programa. Si se declaran en el Diccionario de Datos están disponibles para todos los programas del sistema.

Los tipos se declaran en los programas con la sentencia TYPES.

```
1  REPORT z00_efrcode003.
2
3  TYPES:
4  ty_min_wage(10) TYPE p DECIMALS 5.
```

3. *Declaración de Variables en programa*

Todas las variables deben ser declaradas antes de ser utilizadas.

Declaración de Campos (valor individual)

Disponemos de las sentencias DATA, CONSTANTS, PARAMETERS y FIELD-SYMBOLS.
La forma de declarar una variables es: nombre, tipo, longitud (si aplica), modificadores adicionales (por ejemplo, número de decimales para los datos empaquetados) y opcionalmente el valor inicial.

```
1   REPORT z00_efrcode004.
2
3   DATA:
4   l_amounts  TYPE REF TO f,
5   l_consumer(30) TYPE c.
6
7   DATA:
8   l_creditor LIKE l_consumer.
9
10  CONSTANTS:
11  con_pi         TYPE f VALUE '3.14',
12  con_eacc_fld_fiscyeard TYPE string VALUE 'FISCYEAR'.
13
14  PARAMETERS:
15  p_date TYPE d OBLIGATORY,
16  p_save TYPE c AS CHECKBOX.
17
18  FIELD-SYMBOLS:
19  <l_amount> TYPE f,
20  <l_fiscalyear> TYPE string.
21
22  CREATE DATA l_amounts.
23    l_amounts->* = 1.
```

Usamos TYPE para definir a partir de tipo de dato, y LIKE a partir de otro campo.
Al usar la adición REF TO se crea una referencia al tipo de dato. Para utilizar la variable debe crearse un objeto DATA y asignársele un valor.

```
22  CREATE DATA l_amounts.
23    l_amounts->* = 1.
```

La sentencia PARAMETERS declara el campo al mismo tiempo que crea un objeto en *pantalla de selección*.
La sentencia FIELD-SYMBOL define el tipo de dato que ocupara un área de memoria (puntero).

Declaración de Estructuras (arreglo lineal)

Disponemos de las sentencias DATA.
La forma de declarar una estructura con la sentencia DATA es declarando cada uno de los componentes de la misma, entre las adiciones BEGIN OF y END OF.

```
1   REPORT z00_efrcode005.
2
3   DATA:
4   l_consumer(30) TYPE c.
5
6   DATA:
7     BEGIN OF acc,
8       amounts  TYPE f,
9       consumer(30) TYPE c,
10      creditor LIKE l_consumer,
11    END OF acc.
```

Tabla Interna (arreglo de 2 dimensiones)
Disponemos de las sentencias DATA, SELECT-OPTIONS.
La forma de declarar una tabla interna es a partir de la definición de una estructura.

```
1   REPORT z00_efrcode006.
2
3   DATA:
4   l_consumer(30) TYPE c.
5
6   DATA:
7     BEGIN OF st_acc,
8       amounts  TYPE f,
9       consumer(30) TYPE c,
10      creditor LIKE l_consumer,
11    END OF st_acc.
```

En ABAP se pueden declarar 3 tipos de tabla: STANDARD TABLE, SORTED TABLE y HASHED TABLE, poner STANDARD es opcional.

```
1   REPORT z00_efrcode007.
2
3   DATA:
4   l_consumer(30) TYPE c.
5
6   SELECT-OPTIONS: so_cons FOR l_consumer.
```

Con SELECT-OPTIONS declaramos un tipo particular de tabla (con los campos predefinidos: SIGN, OPTION, LOW, HIGH), que es usado como un rango de entrada y que adicionalmente define un elemento en pantalla de selección.

Cree un nuevo programa y copie el código que se muestra a continuación.

```
REPORT  z00_efrcode008.

PARAMETERS:
* Article data
p_ano(10) TYPE n OBLIGATORY,
p_aname(40) TYPE c OBLIGATORY,
p_aprice TYPE p DECIMALS 2 OBLIGATORY,
p_curr TYPE currencysap OBLIGATORY DEFAULT 'EUR',
p_aquant TYPE i OBLIGATORY DEFAULT 1,
* Tax
p_tax TYPE p DECIMALS 2 DEFAULT '19' OBLIGATORY,
* Terms of payment
p_cash TYPE c RADIOBUTTON GROUP 0001 DEFAULT 'X',
p_credit TYPE c RADIOBUTTON GROUP 0001,
p_months TYPE i OBLIGATORY DEFAULT '24'.

CONSTANTS:
* Interest per year in percent
con_annual_interest TYPE p DECIMALS 2 VALUE '6.75'.

DATA:
* Temporary data
l_net_amount TYPE p DECIMALS 2,
l_tax_factor TYPE f,
l_credit_amount TYPE p DECIMALS 2,
l_monthly_interest_factor TYPE f,
* Result data
l_monthly_vat_amount TYPE p DECIMALS 2,
l_monthly_amount TYPE p DECIMALS 2,
l_vat_amount TYPE p DECIMALS 2,
l_total_amount LIKE l_net_amount.

* Temporary calculations
l_net_amount = p_aprice * p_aquant.
l_tax_factor = p_tax / 100.
* Write article information to screen
WRITE: / , / 'Article information',
       / 'Article number: ', 30 p_ano,
       / 'Article name: ', 30 p_aname,
       / 'Article net price', 30 p_aprice, p_curr,
       / 'Quantity: ', 30 p_aquant.

WRITE: / , / 'Result'.
IF p_cash = 'X'.
* Calculate cash results
  l_vat_amount = l_net_amount * l_tax_factor.
  l_total_amount = l_net_amount + l_vat_amount.
* Write result to screen
  WRITE: / 'Total VAT amount: ', 30 l_vat_amount,
           p_curr,
         / 'Total amount: ', 30 l_total_amount, p_curr.
ELSE.
* Calculate interest results
  l_monthly_interest_factor = con_annual_interest / 100 / 12.
  l_credit_amount = l_net_amount + l_net_amount *
                    l_monthly_interest_factor * p_months.
  l_vat_amount = l_credit_amount * l_tax_factor.
  l_total_amount = l_credit_amount + l_vat_amount.
  l_monthly_vat_amount = l_vat_amount / p_months.
  l_monthly_amount = l_total_amount / p_months.
* Write results to screen
  WRITE: / 'Month: ', 30 p_months,
         / 'Monthly VAT amount: ' , 30 l_monthly_vat_amount, p_curr,
         / 'Monthly amount:', 30 l_monthly_amount, p_curr, ')',
         / '(Total amount: ', 30 l_total_amount, p_curr, ')'.
ENDIF.
```

4. *Rutinas y funciones*

Las rutinas y funciones son dos de las formas disponibles en ABAP para dar modularidad a los programas.
En general las rutinas *suelen* definirse para usarse internamente en un programa. Y las funciones se definen para que estén disponibles de manera global en el sistema.

RUTINAS

Para definir una subrutina:

```
*&---------------------------------------------------------------------*
*&      Form  nombre
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM nombre.

* codigo

ENDFORM.                    "nombre
```

Para invocar una subrutina:

```
PERFORM nombre.
```

Podemos definir una función con las adiciones USING, CHANGING y TABLES. En el nuevo enfoque de ABAP OO y apoyándonos del diccionario de datos podemos usar simplemente la adición USING.

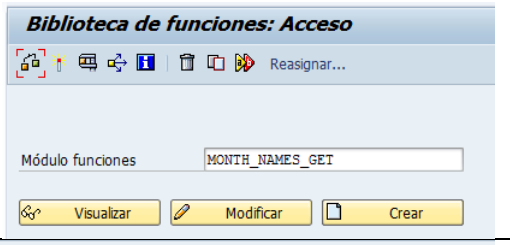
```
DATA: it_mara TYPE mara_tab.

PERFORM f_get_mat USING it_mara.

*&---------------------------------------------------------------------*
*&      Form  nombre
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM f_get_mat USING p_mara TYPE mara_tab.

* codigo

ENDFORM.                    "nombre
```

En este código estamos definiendo el parámetro p_mara como una tabla.

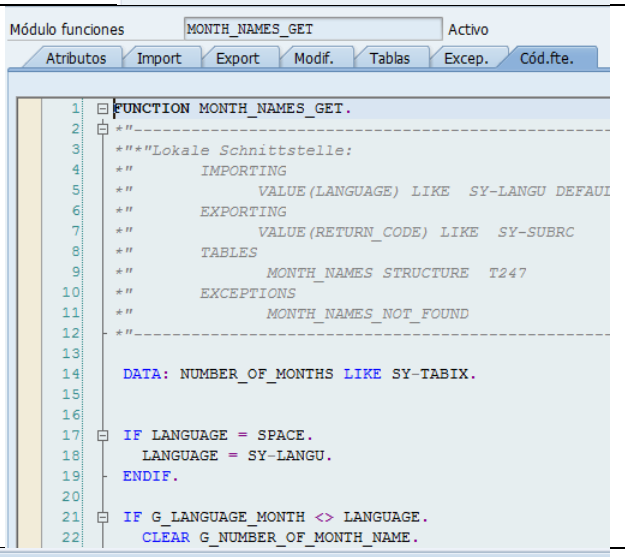
FUNCIONES

Para definir una función se usa la transacción SE37. Por el momento no vamos a definir una sino que vamos a utilizar las que ya existen para explicar su funcionamiento.

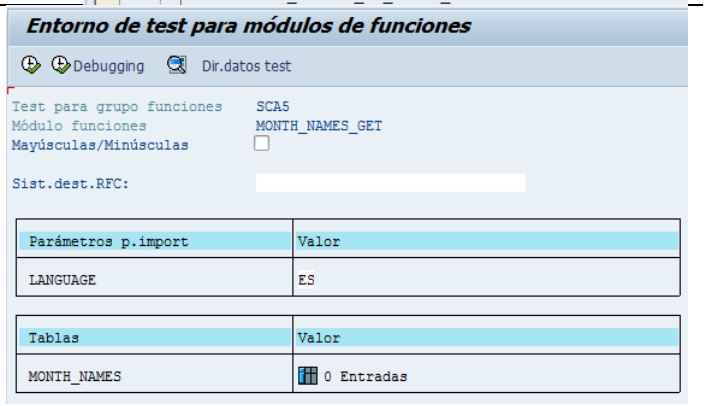
Usamos la transacción SE37. Aparece la pantalla inicial de la biblioteca de funciones.
Elegimos visualizar.



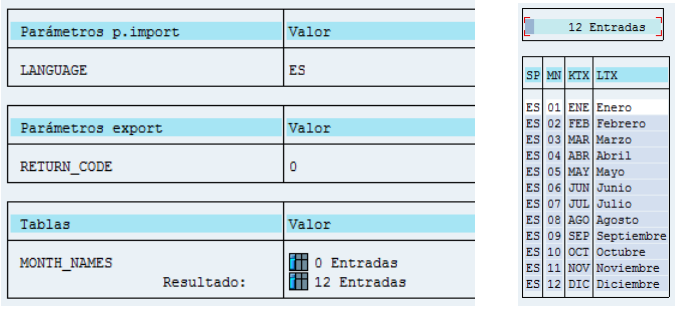
Vemos el entorno de desarrollo de las funciones. Y por defecto nos muestra la pestaña del código fuente (Editor ABAP)



Ejecutamos la función con el botón  (F8). Nos muestra la pantalla del Entorno de Test. La función que elegimos no tiene parámetros de entrada OBLIGATORIOS así que simplemente la ejecutamos.



Y el resultado es el llenado de la tabla MONTH_NAMES con 12 registros. Damos doble clic en el icono y vemos en contenido.



Vamos a repasar rápidamente las pestañas del entorno de desarrollo de funciones.

- **Atributos**. Propiedades y características de las funciones, datos de creación.
- **Import**. Parámetros de ENTRADA a la función desde la llamada.
- **Export**. Parámetros de SALIDA de la función hacia la llamada.
- **Modif**. Parámetros de ENTRADA/SALIDA modificados.
- **Tablas**. Parámetros de ENTRADA/SALIDA del tipo tabla
- **Excep**. Control de errores
- **Cód.fte** Código ABAP con la lógica del proceso

Para invocar esta subrutina:

```
CALL FUNCTION 'MONTH_NAMES_GET'
* EXPORTING
*   LANGUAGE                    = SY-LANGU
* IMPORTING
*   RETURN_CODE                 =
  TABLES
    month_names                 =
* EXCEPTIONS
*   MONTH_NAMES_NOT_FOUND       = 1
*   OTHERS                      = 2
          .
IF sy-subrc <> 0.
* MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
*         WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
ENDIF.
```

Si lo comparamos con la definición de la función:

```
FUNCTION MONTH_NAMES_GET.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"       IMPORTING
*"             VALUE(LANGUAGE) LIKE  SY-LANGU DEFAULT SY-LANGU
*"       EXPORTING
*"             VALUE(RETURN_CODE) LIKE  SY-SUBRC
*"       TABLES
*"              MONTH_NAMES STRUCTURE  T247
*"       EXCEPTIONS
*"              MONTH_NAMES_NOT_FOUND
*"----------------------------------------------------------------------
```

Debemos resaltar la relación cruzada entre parámetros IMPORT y EXPORT entre la definición y la llamada a la función.

5. *Flujo y condiciones*

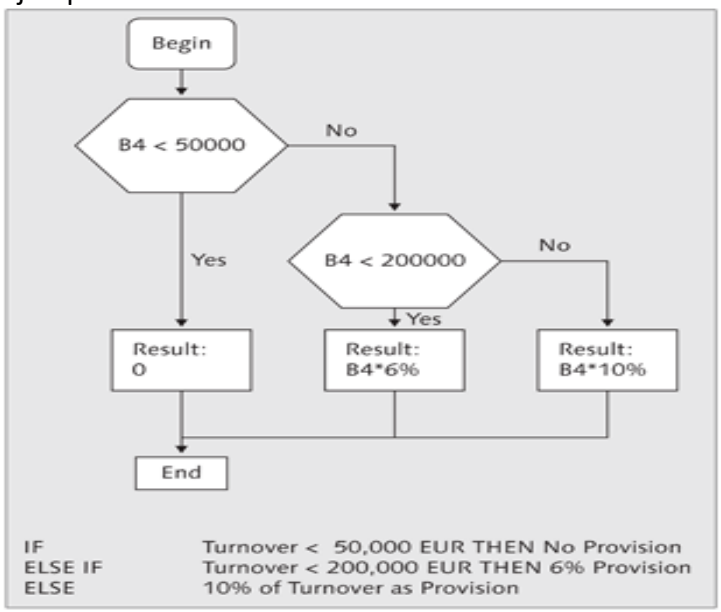
Estas sentencias son: IF y CASE.

SINTAXIS.

```
IF [condicion].

ENDIF.
```

```
CASE [campo].
  WHEN [valor1].
  WHEN [valor2].
  WHEN OTHERS.
ENDCASE.
```

Ejemplo:



En ABAP:

```
1   REPORT  z00_efrcode010.
2
3   PARAMETERS: b4 TYPE p.
4
5   DATA: provision TYPE p.
6
7   IF b4 < 50000.
8     provision = 0.
9   ELSEIF b4 < 200000.
10    provision = b4 * 6 / 100.
11  ELSE.
12    provision = b4 * 10 / 100.
13  ENDIF.
```

EXPRESIONES LÓGICAS Y OPERADORES:

Las expresiones lógicas típicas son de la forma: operando1 operador operando2, y devuelven un valor TRUE o FALSE. Las expresiones lógicas simples se pueden combinar con AND, OR y NOT, usando los parentesis adecuadamente (el primer nivel no es necesario).
Los tipos de operadores en ABAP son:

Descripción		Operador	
Igual	Equal	=	EQ
No igual	Not Equal	<> ><	NE
Mayor que	Greater than	>	GT
Mayor o igual	Greater or equal	>= =>	GE
Menor que	Less than	<	LT
Menor o igual	Less or equal	<= =<	LE

Para comparación de cadenas alfanuméricas:

Descripción		Operador
Contiene solo	Contains only	CO
Contiene cualquiera	Contains any	CA
Contiene la cadena	Contains string	CS
Contiene el modelo	Contains pattern	CP
No sólo contiene	Contains not only	CN
No contiene cualquiera	Contains not any	NA
No contiene ningún cordón	Contains no string	NS
No contiene ningún modelo	Contains no pattern	NP

Operaciones lógicas especiales:
Rango: usamos el operador BETWEEN.
Valor inicial: operador IS INITIAL.

Ejercicio aplicación 2

```
REPORT  zptb00_provision_calculator.
PARAMETERS:
* Start price and sales price
  p_astart TYPE p DECIMALS 2,
  p_asales TYPE p DECIMALS 2.
DATA:
* Net fee, net provision, cost and total cost
  l_fee TYPE p DECIMALS 2,
  l_provision TYPE p DECIMALS 2,
  l_net_cost TYPE p DECIMALS 2,

    l_total_cost TYPE p DECIMALS 2.
  * Calculate fee by start price
  IF p_astart <= 0.
    WRITE: /'Please enter a valid start price'.
    RETURN.
  ELSEIF p_astart <= 1.
    l_fee = '0.25'.
  ELSEIF p_astart <= '9.99'.
    l_fee = '0.40'.
  ELSEIF p_astart <= '24.99'.
    l_fee = '0.60'.
  ELSEIF p_astart <= '99.99'.
    l_fee = '1.20'.
  ELSE.
    l_fee = '2.40'.
  ENDIF.
  * Calculate provision by sales price
  IF p_asales < p_astart.
    WRITE: /'Please enter a valid sales price'.
    RETURN.
  ELSEIF p_asales <= 50.
    l_provision = p_asales * '0.04'.
  ELSEIF p_asales <= 500.
    l_provision = 2 + ( p_asales - 50 ) * '0.03'.
  ELSE.
    l_provision = '15.50' + ( p_asales - 500 ) * '0.015'.
  ENDIF.
  * Calculate total cost and write them to screen
  l_net_cost = l_fee + l_provision.
  l_total_cost = l_net_cost + ( l_net_cost * '0.16' ).
  WRITE: /'Start price  : ', p_astart,
         /'Sales price  : ', p_asales,
         /'Net cost     : ', l_net_cost,
         /'total cost   : ', l_total_cost.
```

6. *Bucles*

Estas sentencias son: DO y WHILE.

SINTAXIS.

```
DO [n] TIMES.

ENDDO.

WHILE [condicion].

ENDWHILE.
```

Sentencias de bifurcación: CONTINUE, CHECK, EXIT.

CONTINUE Salta a la vuelta siguiente del bucle.
CHECK Verifica una condición si no se cumple sale del bucle.
EXIT. Abandona el bucle de forma incondicional.

Ejemplo:

```
 PARAMETERS:
   p_invest TYPE p DECIMALS 2,
   p_profit TYPE p DECIMALS 2.
DATA:
   l_years TYPE i VALUE 0.
WHILE p_invest < p_profit.
   p_invest = p_invest * '1.05'.
   l_years = l_years + 1.
ENDWHILE.
WRITE: / 'You need ', l_
years, ' years to achieve this          profit.'.

WHILE sy-index <= 10.
   WRITE: / 'Hello for the ', sy-index, ''th time.'.
ENDWHILE.
```

Ejercicio aplicación 3

```
REPORT  zptb00_savings_calculator.
PARAMETERS:
* Savings target, monthly savings and yearly interest
* rate
  p_atargt TYPE p DECIMALS 2,
  p_mrate TYPE p DECIMALS 2,
  p_iperc TYPE p DECIMALS 2.
DATA:
* Monthly interest rate, current savings and months
  l_monthly_interest_rate TYPE f,
  l_savings TYPE p DECIMALS 2,
  l_months TYPE i.
* Calculate Months and savings
l_monthly_interest_rate = p_iperc / 12 / 100.
WRITE: /'Savings target : ', p_atargt,
       /'monthly amount : ', p_mrate,
       /'Interest rate %: ', p_iperc.
WRITE sy-uline AS LINE.
* WHILE Version
WHILE l_savings < p_atargt.
  l_savings = l_savings + ( l_savings *
  l_monthly_interest_rate ) + p_mrate.
  l_months = l_months + 1.
ENDWHILE.
WRITE: /'WHILE Statement',
       /'Months needed  : ', l_months,
       /'Saved amount   : ', l_savings.
* DO Version
CLEAR l_savings.
CLEAR l_months.
DO.
   l_savings = l_savings + ( l_savings *
   l_monthly_interest_rate ) + p_mrate.
   l_months = l_months + 1.
   IF l_savings >= p_atargt.
     EXIT.
   ENDIF.
 ENDDO.
 WRITE: /'DO Statement',
        /'Months needed  : ', l_months,
        /'Saved amount   : ', l_savings.
```

IV. Diccionario de datos

SAP es compatible como varios sistemas de base de datos (BD) como ORACLE, MS SQL y otros. Desde SAP incluso desde el entorno de desarrollo ABAP la base de datos que se utilice es *TRANSPARENTE,* esto es puede estarse trabajando con cualquier BD y el tratamiento es el mismo para crear nuevas tablas, leer y registrar entra en las tablas existentes.

En el diccionario de datos podemos definir los tipos de datos que vamos a tener de manera global en el sistema y las tablas que necesitemos adicionar para almacenar información que no contempla el sistema SAP de manera estándar.

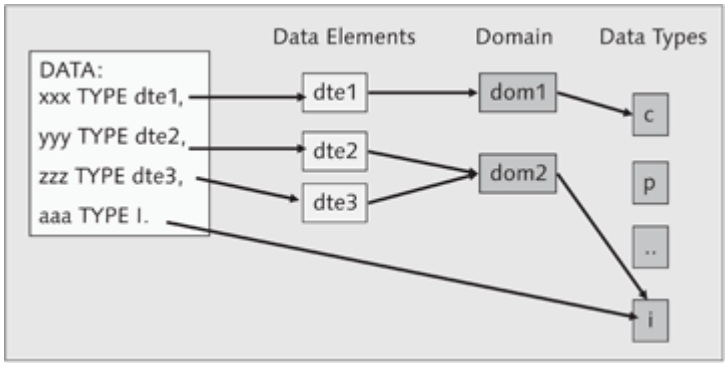
Revisaremos los conceptos de tipo de datos en el DDIC ABAP, desde el más sencillo al más complejo:

Tipo de datos en DDIC ABAP
El tipo de datos describe el formato de datos en la Interfase de usuario.
Si se utiliza un campo de tabla o de estructura o un elemento de datos en los programas ABAP, el tipo de datos adopta un formato utilizado por el procesador ABAP. Al crear una tabla en la base de datos, el tipo de datos de un campo de tabla se convierte en un formato de datos correspondiente al sistema de base de datos utilizado.

Dominio
Un dominio describe los atributos técnicos de un campo, tales como el tipo de datos o la cantidad de posiciones del campo. Un dominio define principalmente un ámbito de valores que describe los valores de datos válidos para los campos que hacen referencia al dominio.
Diferentes campos técnica y profesionalmente similares pueden resumirse mediante un dominio. Los campos que hagan referencia al mismo dominio se modificarán simultáneamente al modificarse el dominio. Con ello queda garantizada la consistencia de estos campos.

Elemento de datos
Un elemento de datos es un tipo elemental que describe los atributos de tipo (tipo de datos, longitud de campo y, en caso necesario, cantidad de decimales) y contiene información relevante para Dynpros (textos explicativos o Ayuda para campos) de objetos de datos Inestructurados (campos de tablas o de estructuras o variables).
Los campos de tablas o de estructuras con el mismo significado conceptual deberían remitir al mismo elemento de datos. Con ello se garantiza que estos campos siempre sean consistentes por lo que respecta a sus atributos.
En los programas ABAP puede remitirse a los elementos de datos con TYPE. Así, en un programa ABAP pueden definirse variables que tomen los atributos de tipo del elemento de datos.

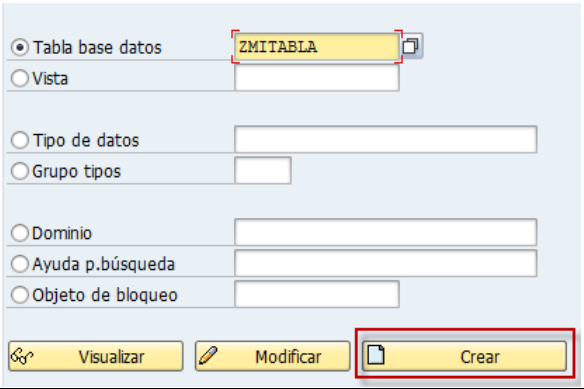
La jerarquía de Datos, Elemento de datos y Dominio:



CREANDO UNA TABLA EN DDIC

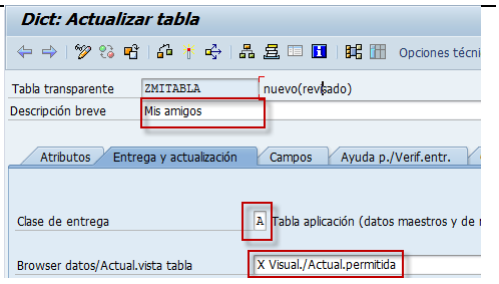
Vamos a crear una tabla sencilla para ilustrar cómo funciona el DDIC ABAP. Para este ejemplo ilustremos nuestro escenario:
La tabla se va a llamar ZMITABLA, va a tener 2 campos NOMBR, APELL, EDAD, SEXO. Vamos a definir Elementos de Datos ZNOMBR##, ZAPELL##, ZEDAD##, ZSEXO## y Dominios ZNOMBR##, ZAPELL##, ZEDAD##, ZSEXO##.

Usamos la transacción SE11. Aparece la pantalla inicial del DDIC ABAP.

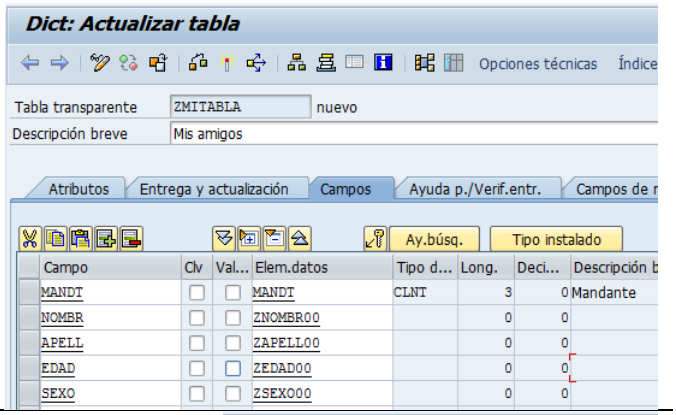


En cada uno de los siguientes pasos nos va a indicar que el componente no existe y si deseamos crearlo. Además al cambiar de pantalla nos pregunta si queremos guardar los cambios en todos los casos respondemos afirmativamente.

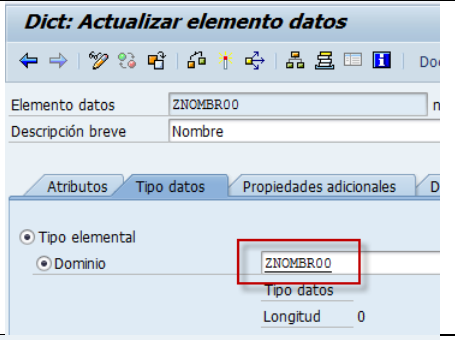
Completamos los tres datos obligatorios. Que son la descripción y los campos de la pestaña Entrega Actualización. Luego vamos a la pestaña Campos.



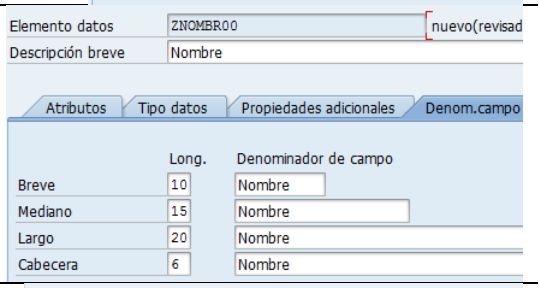
Completamos los campos y guardamos



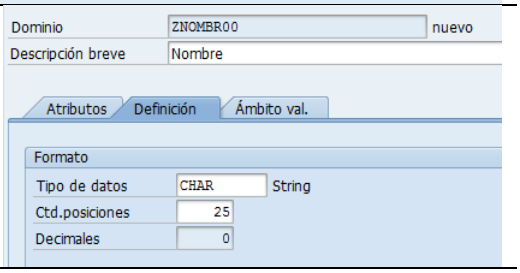
Le damos doble clic sobre ZNOMBR##, esto nos lleva a la pantalla de creación del elemento de datos.
En esta pantalla completamos la descripción y el Dominio ZNOMBR##.



Completamos los denominadores en la pestaña Denom.Campo.
Grabamos.
Regresamos a la pestaña tipo datos y le damos doble clic sobre el nombre del dominio que vamos a crear.



Nos abre la pantalla para crear el Dominio completamos el Tipo de dato y la cantidad de posiciones



Regresamos a la pantalla anterior hasta llegar a la definición de tabla

Completamos los demás campos con la variante de que para la edad le vamos a agregar rangos de edad y para el sexo los valores fijos H o M

Finalmente activamos todos los comentes

V. Tablas Internas y listados

Para procesar una lista de datos en ABAP usamos el concepto de Tabla Interna. Una tabla interna es una tabla que existe en tiempo de ejecución.

Recorremos el contenido de una tabla interna con la sentencia Loop

```
LOOP AT .

ENDLOOP.
```

Podemos llenar una tabla interna mediante la lógica del proceso mediante las sentencias APPEND o INSERT. O podemos llenarla directamente consultando la Base de Datos.

Llenar una tabla mediante APPEND:

```
3  DATA: BEGIN OF st_numeric,
4          number TYPE i,
5        END OF st_numeric.
6
7  DATA: it_numeric LIKE TABLE OF st_numeric WITH HEADER LINE.
8
9  DO 10 TIMES.
10   st_numeric-number = sy-index.
11   APPEND st_numeric TO it_numeric.
12 ENDDO.
13
14 LOOP AT it_numeric INTO st_numeric.
15   WRITE:/ st_numeric-number.
16 ENDLOOP.
```

El resultado de este código es:

```
ejemplo
-------
      1
      2
      3
      4
      5
      6
      7
      8
      9
     10
```

Con la sentencia INSERT debemos agregar además el índice de la posición donde queremos agregar el código. Cuando estamos haciendo un recorrido (Loop) a una tabla se asume el índice del ciclo.

```
15    st_numeric-number = '99'.
16    INSERT st_numeric INTO it_numeric INDEX 3.
17    INSERT st_numeric INTO it_numeric INDEX 5.
18    INSERT st_numeric INTO it_numeric INDEX 7.
19
20
21  LOOP AT it_numeric INTO st_numeric.
22      WRITE:/ st_numeric-number.
23  ENDLOOP.
```

También podemos llenar una tabla interno consultando la base de datos:

```
11    DATA: it_t000 TYPE TABLE OF t000 WITH HEADER LINE.
12
13    SELECT *
14      FROM t000
15      INTO TABLE it_t000.
16
17  LOOP AT it_t000.
18      WRITE:/
19      it_t000-mtext,
20      it_t000-ort01,
21      it_t000-mwaer,
22      it_t000-adrnr.
23  ENDLOOP.
```

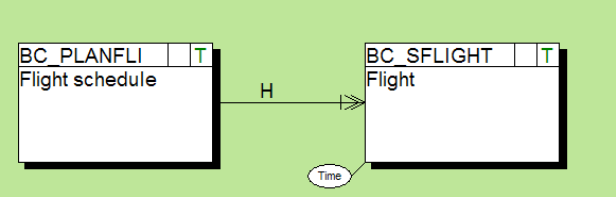
El resultado es:

```
ejm
------------------------------------------------------------------------------
SAP AG                    Walldorf                  DEM
SAP AG                    Walldorf                  DEM
early Watch               Walldorf                  EUR
IDES-ALE: Central FI Syst Frankfurt - Deutschland   EUR
IDES-ALE: Sales System    Barcelona - Spanien       DEM   0000000369
IDES-ALE: Production      Porto - Portugal          EUR   0000000370
IDES-ALE: Warehouse       Dallas, USA               DEM   0000000369
```

Ejercicio Aplicación 4

La práctica calificada consiste en construir un reporte de acuerdo a la especificación de desarrollo que estas adjuntando a continuación.

Para ver el modelo de datos usar la transacción SD11 y el data modeler BC_FLIGHT



En la especificación se esta incluyendo parámetros de selección para filtrar los datos y sentencias SQL para traer datos a tablas internas.
Una vez que los datos se tengan en una tabla interna se deben manipular con las sentencia LOOP AT.
Por otro lado para los quiebres usar las sentencias AT NEW, AT END . AT FIRST , AT LAST.
Para sumarizar importes en los quiebres usar la sentencia SUM.

Pantalla de selección

```
*---------------------------------------------
* Parámetros
*---------------------------------------------
SELECT-OPTIONS:
  s_carrid FOR sflight-carrid,
  s_connid FOR sflight-connid,
  s_fldate FOR sflight-fldate OBLIGATORY,
  s_curren FOR sflight-currency.
```

Cargar Datos

Cargar tabla interna con datos de tabla sflight filtrando los datos con los parámetros definidos.
Código Fuente:

```
  SELECT * INTO CORRESPONDING FIELDS OF TABLE t_reporte
  FROM sflight
  WHERE
    carrid IN s_carrid AND
    connid IN s_connid AND
    fldate IN s_fldate AND
    currency IN s_curren.
```

Cargar tabla interna con datos de tabla SPFLI.
Código Fuente:

```
  SELECT * INTO CORRESPONDING FIELDS OF TABLE t_spfli
  FROM spfli.
```

Cargar tabla interna con datos de tabla SCARR.
Código Fuente:

```
  SELECT * INTO CORRESPONDING FIELDS OF TABLE t_scarr
  FROM scarr.
```

Despliegue de datos en pantalla

Por cada registro de t_reporte procesar
Buscar "Airplane" en tabla interna t_scarr, e imprimir datos de "Airplane".
Código Fuente:

```
CLEAR: t_scarr.
READ TABLE t_scarr WITH KEY CARRID = t_reporte-CARRID.
```

Buscar "Flight schedule" en tabla interna t_SPFLI, e imprimir datos de "Flight schedule".
Código Fuente:

```
CLEAR: t_SPFLI.
READ TABLE t_SPFLI  WITH KEY
  CARRID = t_reporte-CARRID
  CONNID = t_reporte-CONNID.
```

Sumarizar campo PAYMENTSUM "Total of current bookings" por Airplane y Flight Number.

Resultado

```
                                                                                                    Page:     1
Program: Z00_EFRCODE011                                  Reporte de Vuelos                          Date: 27.10.2011
User   : EFALLA                                                                                     Time: 12:40:19

Airline Name: AA  American Airlines
  Flight Number: 0017 From: US  NEW YORK            JFK  To: US  SAN FRANCISCO        SFO  Departure: 11:00:00  Arrival: 14:01:00
    Flight Date    Price Curr   Max Capac Occu seats  Totalbookings
    14.06.2006    422,94  USD         385        373     192.099,55
    12.07.2006    422,94  USD         385        369     193.131,52
    09.08.2006    422,94  USD         385        367     189.206,66
    06.09.2006    422,94  USD         385        366     190.310,55
    04.10.2006    422,94  USD         385        373     194.442,64
    01.11.2006    422,94  USD         385        372     192.717,07
    29.11.2006    422,94  USD         385        372     194.776,73
    27.12.2006    422,94  USD         385        374     192.577,52
    24.01.2007    422,94  USD         385        119      62.413,28
    21.02.2007    422,94  USD         385         44      23.422,45
    21.03.2007    422,94  USD         385         52      26.573,33
    18.04.2007    422,94  USD         385         18       9.029,78
    16.05.2007    422,94  USD         385          4       1.607,16
    13.06.2007    422,94  USD         385         20      10.417,04
  Total Flight Number                                  1.672.725,28
  Flight Number: 0064 From: US  SAN FRANCISCO        SFO  To: US  NEW YORK            JFK  Departure: 09:00:00  Arrival: 17:21:00
    Flight Date    Price Curr   Max Capac Occu seats  Totalbookings
    16.06.2006    422,94  USD         280        271     134.803,77
    14.07.2006    422,94  USD         280        272     133.281,22
    11.08.2006    422,94  USD         280        267     133.251,63
    08.09.2006    422,94  USD         280        271     134.228,68
    06.10.2006    422,94  USD         280        267     130.807,03
```

Ingrese el código siguiente en un nuevo programa

```
*&---------------------------------------------------------------------*
*& Report  Z00_EFRCODE011                                              *
*&                                                                     *
*&---------------------------------------------------------------------*
*& Reporte Plan de Vuelos
*&---------------------------------------------------------------------*
REPORT  Z00_EFRCODE011 NO STANDARD PAGE HEADING .
*---------------------------------------------------------------------*
* Tables
*---------------------------------------------------------------------*
TABLES: sflight, spfli.
*---------------------------------------------------------------------*
* Datos
*---------------------------------------------------------------------*
DATA: zdbcnt LIKE sy-dbcnt.

*---------------------------------------------------------------------*
* Tablas Internas
*---------------------------------------------------------------------*
DATA: BEGIN OF t_reporte OCCURS 0,
        carrid TYPE sflight-carrid,
        connid TYPE sflight-connid,
        fldate TYPE sflight-fldate,
        price TYPE sflight-price,
        currency TYPE sflight-currency,
        seatsmax TYPE sflight-seatsmax,
        seatsocc TYPE sflight-seatsocc,
        paymentsum TYPE sflight-paymentsum,
      END OF t_reporte.

DATA: tmp_reporte LIKE t_reporte.

DATA: t_scarr LIKE scarr OCCURS 0 WITH HEADER LINE.

DATA: t_spfli LIKE spfli OCCURS 0 WITH HEADER LINE.

DATA: t_scurx LIKE scurx OCCURS 0 WITH HEADER LINE.


*---------------------------------------------------------------------*
* Parámetros
*---------------------------------------------------------------------*
SELECT-OPTIONS:
  s_carrid FOR sflight-carrid,
  s_connid FOR sflight-connid,
  s_fldate FOR sflight-fldate,
  s_curren FOR sflight-currency.

*---------------------------------------------------------------------*
* Proceso Principal
*---------------------------------------------------------------------*
START-OF-SELECTION.
  PERFORM cargar_datos.
  DESCRIBE TABLE t_reporte LINES zdbcnt.
  IF zdbcnt > 0.
    PERFORM imprimir_datos.
  ELSE.
    MESSAGE i555(bc_bor) WITH 'No hay datos'.
  ENDIF.

END-OF-SELECTION.
*&---------------------------------------------------------------------*
*&      Form  cargar_datos
*&---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
```

```
*  -->  p1        text
*  <--  p2        text
*---------------------------------------------------------------------*
FORM cargar_datos .
* Cargar tabla interna con sentencia SQL
  SELECT * INTO CORRESPONDING FIELDS OF TABLE t_reporte
  FROM sflight
  WHERE
    carrid IN s_carrid AND
    connid IN s_connid AND
    fldate IN s_fldate AND
    currency IN s_curren.

  SELECT * INTO CORRESPONDING FIELDS OF TABLE t_scarr
  FROM scarr.

  SELECT * INTO CORRESPONDING FIELDS OF TABLE t_spfli
  FROM spfli.

  SELECT * INTO CORRESPONDING FIELDS OF TABLE t_scurx
  FROM scurx.

ENDFORM.                    " cargar_datos
*&---------------------------------------------------------------------*
*&      Form  imprimir_datos
*&---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*---------------------------------------------------------------------*
FORM imprimir_datos .
  FORMAT RESET.
  LOOP AT t_reporte.
    tmp_reporte = t_reporte.
    AT NEW carrid.
* Buscar Datos Airline
      CLEAR: t_scarr.
      READ TABLE t_scarr WITH KEY carrid = tmp_reporte-carrid.
* Imprimir Datos Airline
      FORMAT COLOR 7.
      WRITE:/ 'Airline Name:',
              tmp_reporte-carrid,
              t_scarr-carrname.
      FORMAT RESET.
    ENDAT.

    AT NEW connid.
* Buscar Datos Flight schedule
      CLEAR: t_spfli.
      READ TABLE t_spfli  WITH KEY
        carrid = tmp_reporte-carrid
        connid = tmp_reporte-connid.
* Imprimir Datos de Flight Number
      FORMAT COLOR 7.
      WRITE:/3 'Flight Number:', tmp_reporte-connid,
            'From:', t_spfli-countryfr,
                     t_spfli-cityfrom,
                     t_spfli-airpfrom,
            ' To:',  t_spfli-countryto,
                     t_spfli-cityto,
                     t_spfli-airpto,
            ' Departure:', t_spfli-deptime,
            ' Arrival:',   t_spfli-arrtime.
* Imprimir encabezados
      FORMAT COLOR 1.
```

```
133         WRITE:
134          /5 'Flight Date',
135          18 '  Price',
136          26 'Curr',
137          33 'Max Capac',
138          43 'Occu seats ',
139          55 'Totalbookings'.
140         FORMAT RESET.
141
142       ENDAT.
143
144     * Inicio Procesar detalle
145       WRITE:/5  t_reporte-fldate,
146            (10) t_reporte-price,
147                 t_reporte-currency,
148                 t_reporte-seatsmax,
149                 t_reporte-seatsocc,
150            (14) t_reporte-paymentsum.
151     * Fin Procesar detalle
152
153       AT END OF connid.
154     * Sumarizar valores
155         SUM.
156     * Imprimir totales Flight Number
157         FORMAT COLOR 3.
158         WRITE: /3 'Total Flight Number', 55(14) t_reporte-paymentsum.
159         FORMAT RESET.
160       ENDAT.
161
162       AT END OF carrid.
163     * Sumarizar valores
164         SUM.
165     * Imprimir totales Airline
166         FORMAT COLOR 3.
167         WRITE: / 'Total Airline', 55(14) t_reporte-paymentsum.
168         FORMAT RESET.
169         SKIP.
170       ENDAT.
171
172       AT LAST.
173     * Sumarizar valores
174         SUM.
175     * Imprimir totales Generales
176         FORMAT COLOR 3.
177         WRITE: / 'TOTAL GENERAL', 55(14) t_reporte-paymentsum.
178         FORMAT RESET.
179       ENDAT.
180     ENDLOOP.
181
182   ENDFORM.                    " imprimir_datos
183
184   *---------------------------------------------------------------------*
185   * Top of Page
186   *---------------------------------------------------------------------*
187   TOP-OF-PAGE.
188   * Imprimir Datos Cabecera de Reporte
189     WRITE:/110 'Page:', sy-pagno.
190
191     WRITE:/    'Program:', sy-cprog,
192            60  'Reporte de Vuelos',
193            110 'Date:', sy-datum.
194
195     WRITE:/    'User   :', sy-uname,
196            110 'Time:',  sy-uzeit.
197     ULINE.
```

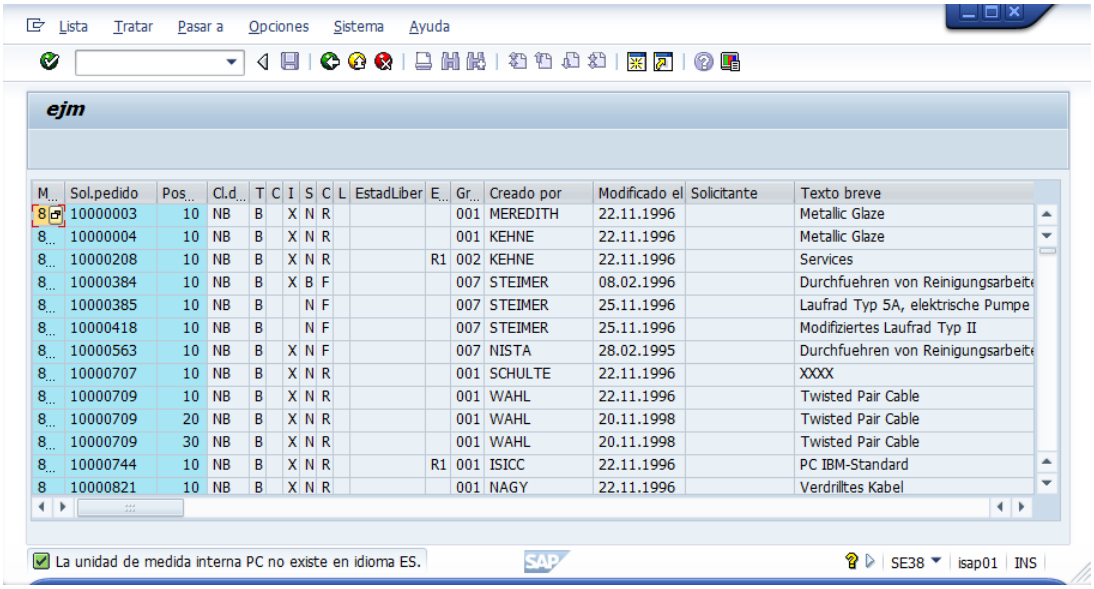
VI. Un ALV muy simple

ABAP List Viewer es un componente de programación muy poderosa que nos permite dar salida a reportes en un formato de hoja de cálculo con muchas funcionalidades que se crean automáticamente al llamar al ALV.

La forma más básica de usar este componente es usando la clase cl_salv_table. El siguiente código

```
1   REPORT  z00_efrcode014.
2
3   DATA it_eban TYPE TABLE OF eban.
4   DATA gt_outtab TYPE TABLE OF eban.
5   DATA gr_table TYPE REF TO cl_salv_table.
6
7   SELECT * INTO TABLE gt_outtab from eban.
8
9     cl_salv_table=>factory(
10       importing
11          r_salv_table = gr_table
12       changing
13          t_table = gt_outtab
14    ).
15
16  gr_table->display( ).
```

Nos despliega la pantalla:

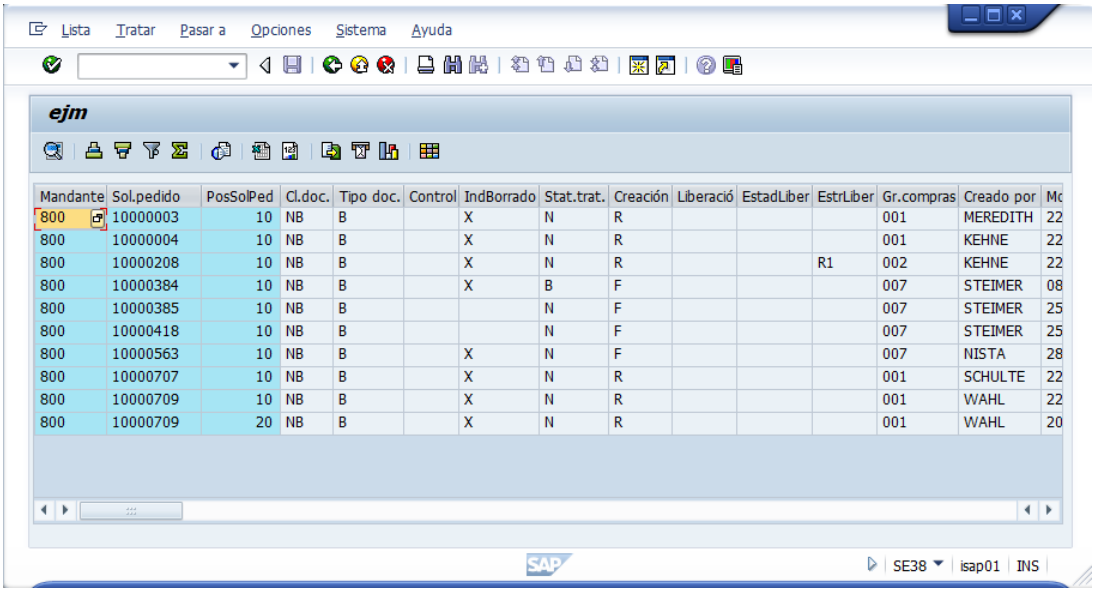


M...	Sol.pedido	Pos...	Cl.d...	T	C	I	S	C	L	EstadLiber	E...	Gr...	Creado por	Modificado el	Solicitante	Texto breve
8	10000003	10	NB	B		X	N	R				001	MEREDITH	22.11.1996		Metallic Glaze
8...	10000004	10	NB	B		X	N	R				001	KEHNE	22.11.1996		Metallic Glaze
8...	10000208	10	NB	B		X	N	R			R1	002	KEHNE	22.11.1996		Services
8...	10000384	10	NB	B		X	B	F				007	STEIMER	08.02.1996		Durchfuehren von Reinigungsarbeit
8...	10000385	10	NB	B			N	F				007	STEIMER	25.11.1996		Laufrad Typ 5A, elektrische Pumpe
8...	10000418	10	NB	B			N	F				007	STEIMER	25.11.1996		Modifiziertes Laufrad Typ II
8...	10000563	10	NB	B		X	N	F				007	NISTA	28.02.1995		Durchfuehren von Reinigungsarbeit
8...	10000707	10	NB	B		X	N	R				001	SCHULTE	22.11.1996		XXXX
8...	10000709	10	NB	B		X	N	R				001	WAHL	22.11.1996		Twisted Pair Cable
8...	10000709	20	NB	B		X	N	R				001	WAHL	20.11.1998		Twisted Pair Cable
8...	10000709	30	NB	B		X	N	R				001	WAHL	20.11.1998		Twisted Pair Cable
8...	10000744	10	NB	B		X	N	R			R1	001	ISICC	22.11.1996		PC IBM-Standard
8	10000821	10	NB	B		X	N	R				001	NAGY	22.11.1996		Verdrilltes Kabel

Afinando la programación podemos agregar botones y mejorar el despliegue:

```
REPORT  z00_efrcode014.

DATA gt_outtab TYPE TABLE OF eban.
DATA gr_table TYPE REF TO cl_salv_table.
DATA gr_functions TYPE REF TO cl_salv_functions.
DATA gt_columns  TYPE REF TO cl_salv_columns_table.

SELECT * INTO CORRESPONDING FIELDS OF TABLE gt_outtab
FROM eban UP TO 10 ROWS.

cl_salv_table=>factory(
   IMPORTING
      r_salv_table = gr_table
   CHANGING
   t_table = gt_outtab
).

gr_functions = gr_table->get_functions( ).
gr_functions->set_all( abap_true ).

CALL METHOD gr_table->get_columns
  RECEIVING
    value = gt_columns.

CALL METHOD gt_columns->set_optimize
  EXPORTING
    value = 'X'.

gr_table->display( ).
```

Nos muestra:



Mandante	Sol.pedido	PosSolPed	Cl.doc.	Tipo doc.	Control	IndBorrado	Stat.trat.	Creación	Liberació	EstadLiber	EstrLiber	Gr.compras	Creado por	Mc
800	10000003	10	NB	B		X	N	R				001	MEREDITH	22
800	10000004	10	NB	B		X	N	R				001	KEHNE	22
800	10000208	10	NB	B		X	N	R			R1	002	KEHNE	22
800	10000384	10	NB	B		X	B	F				007	STEIMER	08
800	10000385	10	NB	B			N	F				007	STEIMER	25
800	10000418	10	NB	B			N	F				007	STEIMER	25
800	10000563	10	NB	B		X	N	F				007	NISTA	28
800	10000707	10	NB	B		X	N	R				001	SCHULTE	22
800	10000709	10	NB	B		X	N	R				001	WAHL	22
800	10000709	20	NB	B		X	N	R				001	WAHL	20

Ejercicio aplicación 5

Una Interfase es un programa que lee un archivo plano y lo procesa en SAP. Generalmente se utiliza para un proceso repetitivo o procesar un lote de datos (BATCH).

Vamos a hacer un programa que cree usuarios a partir de un archivo plano. Entonces nuestro archivo debe tener los datos mínimos para poder crear estos usuarios.

Lo primero que debemos hacer es aprender a crear un usuario en SAP. Para eso vamos a crear usuarios BORRAR## de prueba. Usamos la transacción SU01, y descubrimos que los datos mínimos necesarios son User ID, Apellido y Clave.

Además ubicamos la función BAPI_USER_CREATE que nos ayudara a crear el usuario y la función GUI_UPLOAD que nos sirve para levantar los datos del archivo plano a una tabla interna.

Después de investigar un poco concluimos que podemos hacer una prueba con el siguiente código:

```
1   REPORT  z00_efrcode016.
2
3   PARAMETERS user LIKE bapibname.
4
5   DATA: logondata LIKE bapilogond,
6         address   LIKE bapiaddr3,
7         it_return TYPE STANDARD TABLE OF bapiret2,
8         pw LIKE bapipwd.
9
10  ADDRESS-LASTNAME = 'Apellido'.
11  PW = 'INICIO123'.
12
13  CALL FUNCTION 'BAPI_USER_CREATE'
14    EXPORTING
15      username  = user
16      logondata = logondata
17      password  = pw
18      address   = address
19    TABLES
20      return    = it_return.
21
22  CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'.
```

Completar el ejercicio.

VII. Open SQL

El Open SQL de ABAP es un subconjunto de sentencias SQL. Esto permite la compatibilidad con la variedad de Bases de Dato con las que trabaja.

Sentencias:

- SELECT,
- INSERT,
- UPDATE,
- MODIFY,
- DELETE,
- COMMIT WORK,
- ROLLBACK WORK.

Además de las variables del sistema:

SY-SUBRC: Código de retorno de una operación.
SY-DBCNT: Cantidad de registros afectados por la operación procesada.

Leer un solo registro
SELECT SINGLE * FROM <tab> WHERE <cond>.

Lectura Iterativa
Selección de un grupo de registros. No recomendable para grandes volúmenes de datos
SELECT * FROM <tab>
(WHERE <cond>).
ENDSELECT.

Lectura por bloque
SELECT * FROM <tab> INTO TABLE <intab> WHERE <cond>.
SELECT * FROM <tab> APPENDING TABLE <intab> (WHERE <cond>).

Opciones de WHERE
---- WHERE <campo>...
BETWEEN <var1> AND <var2>. Si <campo> está entre <var1> y <var2>.

LIKE <literal enmascarado>. Si <campo> cumple la máscara.
Se pueden utilizar los comodines:
‘_’ como carácter cualquiera.
‘%’ como una cadena de caracteres.

IN (<var1>, <var2>...). Si <campo> incluye valores <var1>, <var2>...

INSERT

La sentencia INSERT permite introducir registros sencillos o el contenido de una tabla interna en una base de datos SAP.

INSERT <tab>.

Grabará en la BDD el registro de cabecera. Por tanto previamente a esta instrucción moveremos los valores que queremos introducir sobre el área de trabajo de la tabla.

Si SY-SUBRC = 0 Registro insertado.

Si SY-SUBRC > 0 La clave del registro que queríamos insertar ya existía en la tabla.

También es posible introducir datos desde una tabla interna.

INSERT <tab> FROM TABLE <intab>.

Si SY-SUBRC = 0 Registros insertados.

Si existe algún registro en la base de datos con clave igual a algún registro de la tabla interna, se producirá un error de ejecución del programa.

UPDATE

La sentencia UPDATE permite modificar el contenido de uno o varios registros.

UPDATE <tab>.

Modifica el registro de la base de datos que está especificado en el registro de cabecera.

Si queremos modificar el contenido de más de un registro a la vez:

UPDATE <tab> SET <campo> = <valor> WHERE <cond>.

Con este UPDATE, todos los registros que cumplan <cond> modificarán el contenido del <campo> por <valor>.

También es posible utilizar la cláusula SET con:

<campo> = <campo> + <valor> o

<campo> = <campo> - <valor>

Es posible modificar registros desde una tabla interna:

UPDATE <tab> FROM TABLE <intab>.

Si el sistema no puede actualizar un registro, el proceso no finalizará sino que continuará con el siguiente registro.

Si SY-SUBRC = 0 Todos los registros modificados.

Si SY-SUBRC = 4 No todos los registros han sido modificados.

En SY-DBCNT Tendremos la cantidad de registros modificados.

MODIFY

La sentencia MODIFY se utilizará cuando no estemos seguros si utilizar un INSERT o un UPDATE. Es decir, cuando no sepamos con certeza si un registro existe o no, para modificarlo o añadirlo.

MODIFY <tab>.

MODIFY <tab> FROM TABLE <intab>.

En caso de que sepamos si existe o no un registro, por eficacia utilizaremos INSERTs o UPDATEs.

DELETE

Para realizar borrados de datos se aplica la sentencia DELETE.

DELETE <tab>.

Borrará el registro que especifiquemos en el área de trabajo.

Para borrar más de un registro (todos los que cumplan una cierta condición).

DELETE FROM<tab> WHERE <cond>.

Podemos borrar de BDD todos los registros de una tabla interna.

DELETE FROM <tab> FROM TABLE <intab>.

Si SY-SUBRC = 0 Todos los registros han sido borrados.

Si SY-SUBRC = 4 No todos los registros han sido borrados.

En SY-DBCNT Tendremos la cantidad de registros borrados.