Wolters Kluwer

a3ERP

Guía: Introducción a3ERP ACTIVE X

Sumario

1.- ¿Qué es a3ERPACTIVEX? 3
2.- Instalación de a3ERPACTIVEX? 5
3.- Licencias a3ERPACTIVEX RUNTIME / a3ERP 5
4.- Organización de A3ERPACTIVEX 5
4.1.- Relación de objetos de a3ERPACTIVEX 6
5.- Enlace (Conexión) 7
5.1.- Necesidades de a3ERPACTIVEX para conectarse a una empresa en una estación 7
5.2.- Métodos y funciones del enlace 8
6.- Opción (llamada a opciones visuales) 11
7.- Listado (impresión de informes) 13
8.- Varios (consultas) 15
8.1.- Consultas según criterios de a3ERP 15
9.- Maestros 17
9.1.- Operaciones más importantes 19
9.1.1.- Asignar/leer valores a los campos de cada fichero maestro 19
9.1.2.-Crear registros nuevos 19
9.1.3.- Modificar registros existentes 20
9.1.4.- Recorrer registros de un maestro 20
10.- Documentos 20
11.- Apuntes 23
11.1.- Dos formas de trabajar con asientos 24
12.- Cartera 25
12.1.- Creación, modificación y borrado de vencimientos 26
12.2.- Realizar operaciones con vencimientos existentes 27
13.- Stocks 29
14.- Filtro 30
15.- Expediente 30
16.- Cobro/ Anulación parcial de remesas 31
17.- Enlace a3doc cloud 32

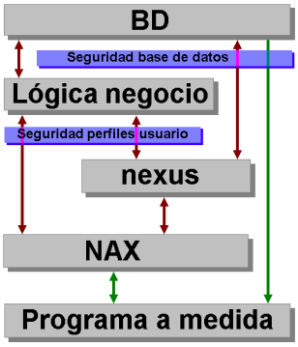
a3ERPACTIVEX

1.- ¿Qué es a3ERPACTIVEX?

Es una librería que permite a terceros conectarse con a3ERP.

Los objetivos de esta librería son los siguientes:

- Simplificar la tarea de utilizar la lógica de negocio de la aplicación a3ERP a través de una serie de objetos con métodos y propiedades que permiten manejar esa lógica.
- Poder llamar a cualquiera de las opciones visuales de a3ERP desde la aplicación desarrollada.
- Poder realizar cualquiera de los informes de a3ERP introduciendo los parámetros desde aplicaciones externas.
- Independizar la utilización de los objetos de la aplicación de su implementación, lo que permitirá, en el futuro, facilitar la migración a nuevas versiones de aplicaciones desarrolladas con a3ERPACTIVEX.
  Esta ventaja reducirá considerablemente el coste de mantenimiento de las aplicaciones desarrolladas con A3ERPACTIVEX tanto desde el punto de vista del cliente como desde el punto de vista del desarrollador. Igualmente minimizará los problemas de mantenimiento que podrían derivarse de las evoluciones de las distintas revisiones de a3ERP.
- Conseguir la sensación (por parte del usuario) de que está trabajando con una única aplicación y un único entorno.
- Garantizar la seguridad establecida por el administrador de a3ERP para aquellas aplicaciones que utilizan las opciones de a3ERP a través de A3ERPACTIVEX.
- Evitar el uso de ficheros intermedios (ASCII, XML, etc...) como sistema para conectarse con otras aplicaciones. En la mayoría de los casos es preferible la conexión en tiempo real que permite A3ERPACTIVEX. Este tipo de enlace permite que las dos aplicaciones se interrelacionen entre si lo que permitirá al usuario una reacción inmediata en el momento en que se produce el problema.
  En la figura que se presenta a continuación, puede verse el esquema que se utiliza desde una aplicación desarrollada a medida para acceder a los datos.
  Se establecen como únicos interlocutores con la aplicación a medida (las flechas verdes):



- Lectura de la base de datos directamente para acceder a los datos de la empresa (se recomienda utilizar este camino exclusivamente en modo lectura). El usuario (de SQL Server) que consulta la base de datos debe tener los permisos suficientes.
- Interacción con a3ERPACTIVEX para realizar las operaciones que interaccionan con la lógica de negocio (operaciones con maestros, documentos, contabilidad y cartera) y con a3ERP (para el enlace visual con cada opción o informe). En este caso, la seguridad es la que se ha establecido en el apartado de seguridad de a3ERP para cada usuario perteneciente a un perfil determinado.

Este manual pretende orientar al desarrollador sobre los principales objetos y su modo de empleo. Para ello, lo dividiremos en varios capítulos, que se detallan a continuación:

- **¿Cómo empezar?:** instalación de la librería.
- **Organización de a3ERPACTIVEX:** relación de objetos que forman a3ERPACTIVEX.
- **Enlace:** uso del objeto de conexión a los datos.
- **Enlace visual con a3ERP:** permite llamar a las opciones de a3ERP con parámetros que variarán su comportamiento.
- **Informes de a3ERP:** uso de los informes de a3ERP pasando parámetros desde el programa.
- **Consultas:** uso del objeto varios que permite realizar consultas sobre la base de datos.
- **Maestros:** ayuda a comprender el modo de utilizar los distintos ficheros maestros de la aplicación.
- **Selección:** permite presentar la pantalla de selección estándar de a3ERP, sin necesidad de crear un maestro.
- **Documentos:** permite conocer las primitivas para trabajar con documentos como facturas, albaranes, depósitos, pedidos, ofertas, regularizaciones, traspasos, expedientes, etc.
- **Apuntes:** objeto para trabajar con asientos contables.
- **Cartera:** permite mover la cartera de efectos (cobrar, pagar, etc.).
- **Stocks:** creación y modificación de inventarios, movimientos y reservas de stock.
- **Excepciones:** tratamiento de los errores reportados por a3ERPACTIVEX.
- **Cuotas:** objeto para trabajar con cuotas automáticas de facturas y albaranes.
- **CobroParcialRemesa**: permite cobrar parcialmente una remesa.
- **AnularCobroParcialRemesa**: permite anular el cobro de una remesa (cobrada parcialmente).

Las primitivas de a3ERPACTIVEX pueden dividirse en **cuatro grandes grupos**:

- **Enlace:** facilita procedimientos para conectarse a una empresa, de aquellas a las que puede accederse desde el ordenador en el que se está usando a3ERPACTIVEX.
- **Ejecución de las opciones visuales de a3ERP:** permite llamar a las opciones que se ejecutan desde el menú de a3ERP. Esto permite crear un entorno especial para una instalación y no ejecutar a3ERP.
- **Impresión de informes de a3ERP:** a3ERP dispone de sus informes ya definidos además de los informes que han sido modificados por el usuario. Estos informes constan de dos partes, la pantalla que solicita los filtros de la información que se desea listar y el listado como tal. La primera de estas tareas puede realizarse desde un programa externo y llamar a la segunda parte con los filtros deseados por el programador.
- **Operaciones con objetos de a3ERPACTIVEX:** Permite crear, borrar o modificar los distintos objetos que forman la lógica de negocio de la aplicación a3ERP. Entre otras caben destacar las operaciones que pueden realizarse sobre maestros, documentos (ofertas, pedidos, albaranes, facturas, etc.) apuntes, cartera, activos fijos y presupuestaria.

2.- Instalación de A3ERPACTIVEX?

Al instalar a3ERP, se instalará automáticamente el fichero a3ERPACTIVEX, y se registrará en la instalación. En caso de que no lo tenga registrado, puede ejecutar la opción automática, o bien registrarlo manualmente en el directorio de instalación de la parte cliente.

Normalmente, se instala de forma automática en el siguiente directorio: "**C:\Archivos de programa\ SIE\ a3ERP".**

En caso de tener que registrar a3ERPACTIVEX, la instrucción sería:

regsvr32 "C:\Program Files (x86)\A3\A3Erp\a3erpactivex.dll"

Una vez registrado a3ERPACTIVEX, deberá incorporar a la aplicación las funcionalidades, importando el a3ERPACTIVEX.

Para ello:

- **Aplicaciones en Delphi.** Puede hacerlo de dos formas:
  -. Haciendo un uses del A3ERPACTIVEX_TLB.pas en nuestro proyecto (ubicado en la carpeta interface de la instalación de a3ERP) para usuarios de delphi7.

  -. Importando la Type Library para inferiores a delphi7.
- **Aplicaciones en otros lenguajes:** importar el A3ERPACTIVEX según especificaciones del lenguaje (por ejemplo, en VB sería marcar una referencia).

3.- Licencias A3ERPACTIVEX RUNTIME / a3ERP

Si nuestro programa a medida va a llamar a pantallas de a3ERP como, por ejemplo, al abrir una ventana de selección de un maestro (cliente, proveedor, artículo, etc.), será necesario tener una licencia de a3ERP.
Si, por el contrario, nuestro programa no hace ninguna llamada 'visual', con la licencia de a3ERPACTIVEX Runtime ya será suficiente.

⚠ **A tener en cuenta:**
Para cualquier duda sobre tema de licencias, deberá consultar con el equipo comercial.

4.- Organización de A3ERPACTIVEX

A3ERPACTIVEX se organiza mediante objetos. Para cada objeto debe crearse una instancia en la aplicación que se desea conectar a través de a3ERPACTIVEX.
A partir de ese momento, se utilizará esa instancia para hacer referencia al objeto y poder utilizar sus propiedades y sus métodos.

4.1.- Relación de objetos de A3ERPACTIVEX

A continuación, se detallan todos los objetos que se pueden utilizar. En el siguiente capítulo, veremos en detalle cada uno de ellos junto con sus propiedades y métodos.

- **Enlace**: inicia y cierra la conexión con la base de datos de a3ERP.
- **Varios**: objeto con funciones varias destinadas a: obtención de precios, obtención de cambio entre monedas, obtención de identificadores únicos de a3ERP, obtención de cuentas y descripciones.
- **Maestro**: Artículos, Proveedores, Clientes, Cuentas, Bancos, Almacenes, Representantes, Transportistas, Provincias, Países, Tarifas, Idiomas, etc. Este objeto permite moverse por los ficheros mencionados, así como añadir nuevos y modificar los existentes.
- **Factura**: objeto que permite la creación, modificación y borrado de facturas.
- **Albarán**: objeto que permite la creación, modificación y borrado de albaranes.
- **Depósito**: objeto que permite la creación, modificación y borrado de depósitos.
- **Pedido**: objeto que permite la creación, modificación y borrado de pedidos.
- **Oferta**: objeto que permite la creación, modificación y borrado de ofertas (presupuestos).
- **Regularización**: objeto que permite la creación, modificación y borrado de albaranes de regularización.
- **Traspaso**: objeto que permite la creación, modificación y borrado de albaranes de traspaso entre almacenes.
- **Asiento**: objeto que permite la creación, modificación y borrado de asientos.
- **Cartera**: objeto que permite la creación, modificación y borrado de vencimientos en cartera.
- **OperacionesCartera**: objeto que permite realizar operaciones sobre los efectos en cartera a excepción de agrupaciones y remesas. Permite, por ejemplo, cobrar, pagar, devolver, recibir, etc.
- **Agrupación**: objeto que ayuda a gestionar las operaciones con las agrupaciones de efectos de cartera.
- **Remesa**: objeto que ayuda a gestionar las operaciones con las remesas de efectos de cartera.
- **Inventario**: objeto que permite la creación y modificación de inventarios.
- **MovimientoStock**: objeto que permite la creación, modificación y borrado de líneas que generan movimientos en el stock por documentos no controlados por a3ERP.
- **ReservaStock**: objeto que permite la creación, modificación y borrado de líneas que generan reservas en el stock por documentos no controlados por a3ERP.
- **Opción**: objeto que permite llamar a cualquier opción de la aplicación a3ERP.
- **Listado**: objeto que permite realizar informes pasando los parámetros deseados directamente.
- **OperacionesInmovilizado**: objeto que permite realizar operaciones sobre el inmovilizado, así como amortizar y repercutir las amortizaciones de las cuotas.
- **Presupuesto**: permite crear, borrar o modificar el contenido de la contabilidad presupuestaria.
- **Estructura**: objeto que permite la creación, modificación y borrado de estructuras.
- **OrdenProducción**: objeto que permite la creación, edición, borrado y evolución de las órdenes de producción.
- **Filtro**: objeto que permite la creación, modificación, y borrado de los filtros.
- **Expediente**: objeto que permite la creación, modificación y borrado de expedientes. También la facturación y anulación del mismo.

Ejemplo del uso de un objeto A3ERPACTIVEX en Delphi

```
A3ERPACTIVEX_TLB;
Function CambioEuro: Double;
var
a3ERPACTIVEXVarios: Varios;                    // Instancia del objeto Varios de
A3ERPACTIVEX
begin
a3ERPACTIVEXVarios := coVarios.Create;         // Creación del objeto
Result := a3ERPACTIVEXVarios.Cambio('DOLAR', '01/01/2002');   // Uso del objeto para obtener el
cambio
// de DOLAR a EURO en 2002
end;
```

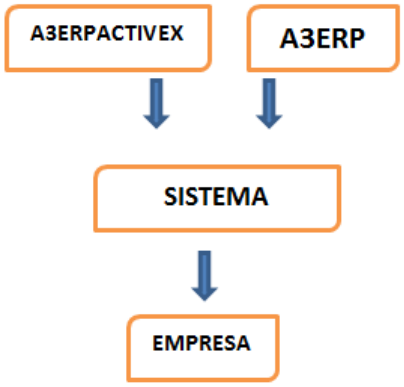
Ejemplo del uso de un objeto en Visual Basic

```
Dim a3ERPACTIVEXVarios As A3ERPACTIVEX.Varios                  // Instancia del objeto
Varios de A3ERPACTIVEX
Private Sub Cambio()
Cambio = a3ERPACTIVEXVarios.Cambio('DOLAR', '01/01/2002')     // Uso del objeto para obtener el
cambio
                                                               // de DOLAR a EURO en 2002
End Sub
```

5.- Enlace (Conexión)

5.1.- Necesidades de a3ERPACTIVEX para conectarse a una empresa en una estación

A3ERPACTIVEX necesita que a3ERP esté instalado en la estación, ya que comparten la instalación.



5.2.- Métodos y funciones del enlace

Métodos	Tipo	Descripción
Empresas	Función	Obtiene la lista de los nombres de las empresas de la instalación.
ParamConexion	Función	Obtiene los parámetros de conexión de una empresa determinada
Iniciar	Procedimiento	Inicia la conexión con una empresa indicada en un parámetro
SelecEmpresa	Función	Inicia la conexión con una empresa seleccionada por el propio usuario.
SelecEmpresaActiva	Función	Inicia la conexión en la empresa en la que se encuentra a3ERP. Creado para poder llamar a un programa que use a3ERPACTIVEX desde la barra de iconos de a3ERP.
SelecMultiEmpresa	Función	Activa el tipo contable (multiempresa) de los tipos permitidos por el usuario.
SelecMultiEmpresaActiva	Función	Activa el tipo contable (multiempresa) del registro de Windows, de los tipos permitidos por el usuario.
Acabar	Procedimiento	Termina la conexión
EliminarRecordatorioClave	Procedimiento	Elimina el flag de recordar la contraseña del registro de Windows.
EstacionActivada	Función	Retorna si el número de serie de PRODUCTO a3ERP está activado para la estación.
CerrarVentanas	Procedimiento	Cierra todas las ventanas abiertas de a3ERP
ActivarVentana	Procedimiento	Pone en primer plano la ventana activa de a3ERP
LoginUsuario	Función	Se utiliza para evitar la pantalla visual donde se piden el usuario y password de entrada en a3ERP, y retorna si dichos valores son correctos para entrar en la aplicación.

Siempre que se trabaje con a3ERPACTIVEX, previamente, debe haberse conectado a una empresa concreta. Para iniciar esta conexión (enlace) existen dos posibles sistemas:

- **Opción A**: que la aplicación desarrollada con a3ERPACTIVEX presente la relación de empresas existentes y el usuario se encargue de seleccionar la empresa con la que desea trabajar.
- **Opción B**: o bien, llamar a una función de a3ERPACTIVEX que sea la que se encargue de conectar con la última empresa con la que se ha conectado desde a3ERP.

En ambos casos, el resultado final es que a3ERPACTIVEX queda conectado con una empresa contra la que trabajarán todos los objetos que en adelante se utilicen.

A continuación, se muestran dos ejemplos sobre las dos formas de crear un enlace con una empresa elegida por el usuario.

Ejemplos en Delphi

INICIAR UN ENLACE (OPCIÓN A)

```
var
a3ERPACTIVEXEnlace: Enlace;
procedure Enlazar;
begin
a3ERPACTIVEXEnlace := coEnlace.Create;                    // Se crea el objeto para el enlace
// Saldrá la selección de empresas de a3ERP
if not a3ERPACTIVEXEnlace.SelecEmpresa then
Exit;
end;
```

INICIAR UN ENLACE (OPCIÓN B)

```
var
a3ERPACTIVEXEnlace: Enlace;
procedure Enlazar;
begin
a3ERPACTIVEXEnlace := coEnlace.Create;                    // Se crea el objeto para el enlace
// Se inicia el enlace en la empresa donde se esté con a3ERP
if not a3ERPACTIVEXEnlace.SelecEmpresaActiva then
Exit;
end;
```

Ejemplos en Visual Basic

```
Dim boaEnlace As A3ERPACTIVEX.Enlace
..
Set BoaEnlace = New A3ERPACTIVEX.Enlace
boaEnlace.SelecEmpresa
```

Ejemplo tratamiento de excepciones

Dentro de las operaciones que pueden ejecutarse en a3ERPACTIVEX pueden producirse errores.
No obstante, a través del enlace puede consultar el último error que se ha producido.

Para tratar correctamente las excepciones, debería consultarse si se ha producido algún error para cada operación que se realiza con a3ERPACTIVEX y, en caso de error, interrumpir el proceso.

Ejemplo de tratamiento de excepciones

```
var
a3ERPACTIVEXEnlace: Enlace;
a3ERPACTIVEXClientes: Maestro;
begin
a3ERPACTIVEXEnlace := coEnlace.Create;
a3ERPACTIVEXEnlace.Iniciar( 'Mi empresa', 'C:\archivos de programa\sie');

a3ERPACTIVEXClientes := coMaestro.Create;
a3ERPACTIVEXClientes.Iniciar( 'Clientes');
if  a3ERPACTIVEXEnlace.bError then begin
ShowMessage( 'Error: ' + a3ERPACTIVEXEnlace.sMensaje);
Exit;
end;
a3ERPACTIVEXClientes.Nuevo;
if  a3ERPACTIVEXEnlace.bError then begin
ShowMessage( 'Error: ' + a3ERPACTIVEXEnlace.sMensaje);
Exit;
end;
.
.
end;
```

Se recomienda la construcción de un procedimiento que realice estas comprobaciones:

```
Procedure a3ERPACTIVEXCheck;
begin
        if a3ERPACTIVEXEnlace.bError then begin
        raise Exception.Create( 'Error: ' + a3ERPACTIVEXEnlace.sMensaje);       // Rompe la ejecución del programa
        end;
end;
```

De esta forma la aplicación quedaría menos cargada de código:

```
var
        a3ERPACTIVEXEnlace: Enlace;
        a3ERPACTIVEXClientes: Maestro;
begin
        a3ERPACTIVEXEnlace := coEnlace.Create;
        a3ERPACTIVEXEnlace.Iniciar( 'Mi empresa', 'C:\archivos de programa\sie');

        a3ERPACTIVEXClientes := coMaestro.Create;
        a3ERPACTIVEXClientes.Iniciar( 'Clientes');
        a3ERPACTIVEXCheck;          // En caso de error rompe la ejecución del programa
        a3ERPACTIVEXClientes.Nuevo;
        a3ERPACTIVEXCheck;          // En caso de error rompe la ejecución del programa
        .
        .
end;
```

6.- Opción (llamada a opciones visuales)

Para ejecutar una opción debe conocerse el identificador único que la representa y los parámetros con que se utiliza. Se edita continuamente documentación en este sentido (al instalar a3ERP, en la carpeta ejemplos encontrarás el fichero OpcionesVisualesA3ERP con el listado de opciones, su identificador y sus posibles parámetros).

A continuación, veremos dos ejemplos del uso de este objeto. En el primer ejemplo no se utilizan parámetros y en el segundo sí.

Ejemplo 1: Mantenimiento de clientes sin parámetros

Igual que si se llamara desde el menú de a3ERP.

En Delphi

```
procedure MantenimientoClientes;
var
a3ERPACTIVEXOpcion: Opcion;
begin
a3ERPACTIVEXOpcion := coOpcion.Create;
a3ERPACTIVEXOpcion.Iniciar;
a3ERPACTIVEXOpcion.IdOpcion := 'manClientes';
a3ERPACTIVEXOpcion.Ejecutar;
a3ERPACTIVEXOpcion.Acabar;
end;
```

En Visual Basic

```
Private Sub btnEditarA3ERP_Click()
Dim A3ERPACTIVEXOpcion As A3ERPACTIVEX.Opcion
 Set A3ERPACTIVEXOpcion = New A3ERPACTIVEX.Opcion
   A3ERPACTIVEXOpcion.Iniciar
   A3ERPACTIVEXOpcion.IdOpcion = "manClientes"
   A3ERPACTIVEXOpcion.Ejecutar
   A3ERPACTIVEXOpcion.Acabar **
```

Ejemplo 2: Edición de la Ficha con Parámetros

De un cliente determinado.

En Delphi

```
procedure MantenimientoClientes( sCliente: string);
var
a3ERPACTIVEXOpcion: Opcion;
begin
a3ERPACTIVEXOpcion := coOpcion.Create;
a3ERPACTIVEXOpcion.Iniciar;
a3ERPACTIVEXOpcion.IdOpcion := ‘manClientes’;
a3ERPACTIVEXOpcion.AnadirParametro( ‘Accion’, ‘Edicion’);
a3ERPACTIVEXOpcion.AnadirParametro( ‘Codigo’, sCliente);
a3ERPACTIVEXOpcion.Ejecutar;
a3ERPACTIVEXOpcion.Acabar;
end;
```

En Visual Basic

```
Private Sub btnEditar_Click()
Dim A3ERPACTIVEXOpcion As A3ERPACTIVEX.Opcion
 Set A3ERPACTIVEXOpcion = New A3ERPACTIVEX.Opcion
 A3ERPACTIVEXOpcion.Iniciar
 A3ERPACTIVEXOpcion.IdOpcion = "manClientes"
 Call A3ERPACTIVEXOpcion.AnadirParametro("Accion", "Edicion")
 Call A3ERPACTIVEXOpcion.AnadirParametro("Codigo", edtCodigo.Text)
 A3ERPACTIVEXOpcion.Ejecutar
 A3ERPACTIVEXOpcion.Acabar
End Sub
```

7.- Listado (impresión de informes)

Para ejecutar una opción debe conocerse el identificador único que la representa y los parámetros con que se utiliza. Por ello, se edita continuamente la documentación en este sentido.

A continuación, veremos dos ejemplos del uso de este objeto, en el primero realizando la presentación preliminar de un listado de clientes y en el segundo generando un fichero en formato PDF con el contenido del mismo listado.

Ejemplo 1: Presentación preliminar del listado de clientes en Delphi

```
procedure ListadoClientes;
var
a3ERPACTIVEXListado: Listado;
begin
a3ERPACTIVEXListado := coListado.Create;
a3ERPACTIVEXListado.Iniciar;                    // Reservar los recursos
a3ERPACTIVEXListado.IdListado := 'lstCli';      // Identificador del listado
a3ERPACTIVEXListado.Destino := destPantalla;    // Destino del listado (presentación preliminar)
a3ERPACTIVEXListado.Copias := 1;                // Nº de copias
a3ERPACTIVEXListado.Impresora := 'Impresora local LaserJet 4200'; // Nombre de la impresora
a3ERPACTIVEXListado.Modelo := 'Listado con email';  // Nombre del modelo definido por el usuario
a3ERPACTIVEXListado.AnadirParametro( 'CodIni', '      1');  // Código inicial
a3ERPACTIVEXListado.AnadirParametro( 'CodFin', '99999999'); // Código final
a3ERPACTIVEXListado.Imprimir;                   // Imprimir
a3ERPACTIVEXListado.Acabar;                     // Liberar los recursos
end;
```

Ejemplo 2: Fichero PDF con el listado de clientes en Delphi

```
procedure PDFClientes;
var
a3ERPACTIVEXListado: Listado;
begin
a3ERPACTIVEXListado := coListado.Create;
a3ERPACTIVEXListado.Iniciar;                    // Reservar los recursos
a3ERPACTIVEXListado.IdListado := 'lstCli';      // Identificador del listado
a3ERPACTIVEXListado.Destino := destPDF;         // Destino del listado (fichero en formato PDF)
a3ERPACTIVEXListado.Fichero := 'C:\Listados \Clientes email.pdf'; // Fichero donde almacenar el listado
a3ERPACTIVEXListado.Copias := 1;                // Nº de copias
a3ERPACTIVEXListado.Impresora := 'Impresora local LaserJet 4200'; // Nombre de la impresora
a3ERPACTIVEXListado.Modelo := 'Listado con email';  // Nombre del modelo definido por el usuario
a3ERPACTIVEXListado.AnadirParametro( 'CodIni', '      1');  // Código inicial
```

```
a3ERPACTIVEXListado.AnadirParametro( ‘CodFin’, ‘99999999’); // Código final
a3ERPACTIVEXListado.Imprimir;                                  // Imprimir
a3ERPACTIVEXListado.Acabar;                                 // Liberar los recursos
end;
```

Ejemplo 3: Presentación preliminar del listado de clientes en VB.

```
Private Sub btnListado_Click()
Dim A3ERPACTIVEXListado As A3ERPACTIVEX.Listado
Set A3ERPACTIVEXListado = New A3ERPACTIVEX.Listado
A3ERPACTIVEXListado.Iniciar
A3ERPACTIVEXListado.IdListado = "LSTCLI"
Call A3ERPACTIVEXListado.AnadirParametro("CodIni", "      1")
Call A3ERPACTIVEXListado.AnadirParametro("CodFin", "99999999")
A3ERPACTIVEXListado.Imprimir
A3ERPACTIVEXListado.Acabar
End Sub
```

Ejemplo 4: Imprimir un efecto formato recibo en Delphi

```
procedure PDFRecibo;
var
  a3ERPACTIVEXListado: Listado;
begin
  A3ERPACTIVEXLis := CoListado.Create;
  A3ERPACTIVEXLis.Iniciar;
  a3ERPACTIVEXLis.IdListado := 'LSTIMPREFERE';
  a3ERPACTIVEXLis.AnadirParametro('NumCartera0', '6356');
  a3ERPACTIVEXLis.AnadirParametro('NumVen0', '1');
  a3ERPACTIVEXLis.AnadirParametro('Total', '0');
  a3ERPACTIVEXLis.AnadirParametro('Posicion', '1');
  a3ERPACTIVEXLis.AnadirParametro('FecExp', '07/12/2009');
  a3ERPACTIVEXLis.AnadirParametro('CobroPago', 'C');
  a3ERPACTIVEXLis.AnadirParametro('Portador', 'Prueba');
  a3ERPACTIVEXLis.Destino := destJPG;
  a3ERPACTIVEXLis.Fichero := 'C:\tmp\Prueba.jpg';
  A3ERPACTIVEXLis.Imprimir;
  A3ERPACTIVEXLis.Acabar;
End;
```

8.- Varios (consultas)

En este apartado veremos algunas de las funciones que pueden ayudarnos en la integración con a3ERP.
Los métodos que vamos a usar en este capítulo se encuentran en el objeto Varios y se detallan a continuación:

8.1.- Consultas según criterios de a3ERP

Métodos	Tipo	Descripción
ObtPrcVenta	Función	Obtiene el precio de venta.
ObtPrcCompra	Función	Obtiene el precio de compra.
ObtPrecioCoste	Función	Obtiene el precio de coste.
ObtPrecioMedio	Función	Obtiene el precio medio.
ObtComis	Función	Obtiene la comisión
Cambio	Función	Obtiene el cambio de una moneda.
CambioMon	Función	Obtiene el cambio entre dos monedas.
MonedaPrincipal	Función	retorna el código de la moneda principal
DescArticulo	Procedimiento	Devuelve la descripción y el texto de un artículo en un idioma.
ObtRepresentante	Función	Devuelve el representante de un cliente.
CuentaArtV	Función	Devuelve la cuenta del artículo de venta
CuentaArtC	Función	Devuelve la cuenta del artículo de compra
TotalLinea	Función	Devuelve la base imponible (en moneda) de la línea
FormatNum	Función	Devuelve el formato para n decimales (con miles)
FormatEdiNum	Función	Devuelve el formato para n decimales
Redondear	Función	Redondea un número a n decimales
Version	Propiedad	Contiene la versión instalada de a3ERP
Revision	Propiedad	Contiene la revisión instalada de a3ERP
Parche	Propiedad	Contiene el parche instalado de a3ERP
Vencimientos	Función	Crea un array con la información el conjunto de vencimientos con las condiciones especificadas.
LlamarVisual	Procedure	Muestra una pantalla visual de a3ERP

Funciones para recuperación de identificadores:

Métodos	Tipo	Descripción
SigId	Función	Devuelve el siguiente valor del identificador

Ejemplo 1: Consultas según criterios de a3ERP

```
Function CuentaDeVenta: string;
var
a3ERPACTIVEXVarios : Varios;
sCodArt, sCodCli: string
begin
a3ERPACTIVEXVarios := coVarios.Create;
sCodArt := '1';
sCodCli:= '1';
Result := a3ERPACTIVEXVarios.CuentaArtV(sCodArt, sCodCli);  // Cuenta de ventas dado un artículo y un cliente
end;
```

Ejemplo 2: Uso de los identificadores de a3ERP para obtener un identificador propio

En primer lugar, aclarar la utilización que un programador puede hacer del sistema de identificadores de a3ERP.

Los identificadores no son más que un sistema para obtener claves únicas con un sistema interno de a3ERP dado un nombre.

a3ERP utiliza sus propios identificadores (por ejemplo. para identificar una factura de venta de forma única se utiliza un identificador llamado IdFacV). Los identificadores que ya utiliza a3ERP es difícil que puedan interesar a un programa externo, pero el sistema como tal puede ser de utilidad en el caso en que se deseen tener identificadores únicos y consecutivos para alguna tarea.

En el ejemplo interesa conseguir un identificador único para cada cuota de un programa que estamos creando.

```
Function IdentificadorCuotas: Double; ( en Delphi)
var
a3ERPACTIVEXVarios : Varios;
Begin
a3ERPACTIVEXVarios := coVarios.Create;
Result := a3ERPACTIVEXVarios.SigId( 'Cuotas');
End;
Function Sub IdentificadorCuotas
Dim A3ERPACTIVEXVarios As A3ERPACTIVEX.Varios
Set A3ERPACTIVEXVarios = New A3ERPACTIVEX.Varios
Return( call A3ERPACTIVEXVarios.SigId( "Cuotas"))
```

Este fragmento de código garantiza al programador que cada vez que llame a esta función obtendrá un identificador único identificado con el texto Cuotas. El primer valor de este identificador será el 1 y éste se irá incrementando continuamente.

9.- Maestros

La mayoría de los ficheros maestros que utiliza a3ERP pueden ser gestionados desde este objeto.

Para ello, cuando se crea la instancia del objeto, hay que indicar el nombre del fichero al que desea asociarse la instancia, de la forma que se muestra en los siguientes ejemplos:

En Delphi

```
var
a3ERPACTIVEXClientes: Maestro;
begin
a3ERPACTIVEXClientes := coMaestro.Create;      // Crea la instancia del maestro de clientes
a3ERPACTIVEXClientes.Iniciar( 'Clientes');.     // Inicia el objeto especificando que se trata del maestro de
...                                            // clientes
...
a3ERPACTIVEXClientes.Acabar;                    // como siempre, destruir el objeto creado
end;
```

En Visual Basic

```
Dim a3ERPACTIVEXClientes As A3ERPACTIVEX.Maestro
Set a3ERPACTIVEXClientes = New A3ERPACTIVEX.Maestro
Call a3ERPACTIVEXClientes.Iniciar("Clientes")
...
...
a3ERPACTIVEXClientes.Acabar
```

El parámetro que se le pasa al procedimiento iniciar es el valor del parámetro en la siguiente tabla (si el nombre de la tabla coincide con el parámetro no se incluyen):

Nombre de tabla	Valor del parámetro
TipoIva	TIPOSIVA
RegIva	REGIMENESIVA
RegIva (solo soportados)	REGIMENESIVASOP
RegIva (solo repercutidos)	REGIMENESIVAREP
Proveed	PROVEEDORES
Articulo	ARTICULOS
DocuPago	DOCUMENTOSPAGO
FormaPag	FORMASPAGO
Almacen	ALMACENES
Represen	REPRESENTANTES
Transpor	TRANSPORTISTAS
Provinci	PROVINCIAS
CentrosC	CENTROSCOSTE
TiposCon	TIPOSCONTABLES
TarifaVe	TARIFASVENTA
PrcEsp	PRECIOSESPECIALES
DatosCon	DATOSCONTABLES
DomBanca (proveedores)	DOMBANCAC
DomBanca (clientes)	DOMBANCAV
RefCli	REFCLIENTES
RefPro	REFPROVEEDORES
FamiliaTalla	FAMILIATALLAHORIZONTAL
FamiliaTalla	FAMILIATALLAVERTICAL

LAS CLAVES FORÁNEAS

Las claves foráneas en a3ERP se guardan cuadradas a la derecha. Esto significa, que si el ancho de un código es de 8 dígitos y el valor que se introduce en él es el '3', el programa añade siete espacios delante para que el '3' quede a la derecha ('_____3'). Esto sólo se hace con los códigos numéricos ya que a los alfanuméricos no se añade ningún espacio.

Cuando se introduce un valor en un campo clave, el programa se encarga de cuadrarlo. De la misma forma, si se pasa un parámetro a una función que se refiere al valor de una clave, el programa también lo cuadra.

Ejemplo: a3ERPACTIVEXArticulo.Buscar('1') Esta función busca el artículo con código '_____1'.

EXCEPCIONES

Son excepciones a esta regla los valores de los parámetros de los objetos Opción y Listado que deben pasarse ya cuadrados dado que a priori se desconoce el contenido del parámetro.

9.1.- Operaciones más importantes

Las operaciones que pueden realizarse con un maestro son varias. A continuación, se detallan los usos más habituales:

9.1.1.- Asignar/ leer valores a los campos de cada fichero maestro

```
var
a3ERPACTIVEXArticulo: Maestro;
begin
a3ERPACTIVEXArticulo := coMaestro.Create;      // Crea la instancia del maestro
a3ERPACTIVEXArticulo.Iniciar( 'Articulos');.   // Inicia el objeto indicando que se trata de artículos.
if a3ERPACTIVEXArticulo.Buscar( '1') then      // Buscar el artículo a editar.
begin
  a3ERPACTIVEXArticulo.AsString[ 'DescArt'] := 'Descripción del artículo';
  a3ERPACTIVEXArticulo.AsFloat[ 'PrcCompra'] := 1500;
  a3ERPACTIVEXArticulo.AsFloat[ 'PrcVenta'] := a3ERPACTIVEXArticulo.AsFloat[ 'PrcCompra'] * 1.25;
end;
.
.
```

AsString, AsFloat, AsInteger y AsBoolean son funciones que sirven para asignar/leer cualquier campo del registro del maestro elegido.

9.1.2.-Crear registros nuevos

En la creación de registros nuevos, a3ERPACTIVEX aporta una de sus grandes ventajas, ya que el registro se carga con los valores por defecto y el programador debe limitarse a rellenar aquellos que conoce. Son campos obligatorios el código y la descripción. Asimismo, al guardarse el registro se realizan ciertas comprobaciones.

```
var
a3ERPACTIVEXArticulo: Maestro;
sCodigo: string;
begin
a3ERPACTIVEXArticulo := coMaestro.Create;      // Crea la instancia del maestro
a3ERPACTIVEXArticulo.Iniciar( 'Articulos');.   // Inicia el objeto indicando que se trata de de artículos
a3ERPACTIVEXArticulo.Nuevo;
sCodigo := a3ERPACTIVEXArticulo.NuevoCodigoNum;  // Busca el último código y le suma 1.
a3ERPACTIVEXArticulo.AsString[ 'CodArt'] := sCodigo;
a3ERPACTIVEXArticulo.AsString[ 'DescArt'] := 'Descripción del artículo';
a3ERPACTIVEXArticulo.AsFloat[ 'PrcCompra'] := 1500;
a3ERPACTIVEXArticulo.AsFloat[ 'PrcVenta'] := a3ERPACTIVEXArticulo.AsFloat[ 'PrcCompra'] * 1.25;
a3ERPACTIVEXArticulo.Guarda( False);            // False indica que se trata de un registro nuevo
                                                // True indicaría que se trata de una modificación
end;
```

9.1.3.- Modificar registros existentes

```
var
a3ERPACTIVEXArticulo: Maestro;
begin
a3ERPACTIVEXArticulo := coMaestro.Create;     // Crea la instancia del maestro de artículos
a3ERPACTIVEXArticulo.Iniciar( 'Articulos');.      // Inicia el objeto
if a3ERPACTIVEXArticulo.Buscar( ‘1’) then begin// Busca el cliente con el código 1
        a3ERPACTIVEXArticulo.Edita;
a3ERPACTIVEXArticulo.AsString[ ‘DescArt’] := ‘Descripción del artículo’;
        a3ERPACTIVEXArticulo.AsFloat[ ‘PrcCompra’] := 1500;
        a3ERPACTIVEXArticulo.AsFloat[ ‘PrcVenta’] := a3ERPACTIVEXArticulo.AsFloat[ ‘PrcCompra’] *
1.25;
        a3ERPACTIVEXArticulo.Guarda( True);   // True indica que se trata de un registro existente
end else
ShowMessage( ‘Registro no encontrado’);
end;
```

9.1.4.- Recorrer registros de un maestro

A3ERPACTIVEX ofrece las funciones necesarias para recorrer un fichero que son: BOF, EOF, Primero, Ultimo, Siguiente, Anterior, Buscar.

A continuación, se muestra un ejemplo sencillo del recorrido del fichero de artículos.

```
var
a3ERPACTIVEXArticulo: Maestro;
begin
a3ERPACTIVEXArticulo := coMaestro.Create;     // Crea la instancia del maestro de artículos
a3ERPACTIVEXArticulo.Iniciar( 'Articulos');.      // Inicia el objeto
a3ERPACTIVEXArticulo.Primero;                         // Se situa en el primer registro
while not a3ERPACTIVEXArticulo.EOF do begin  // Recorre la tabla mientras no llega al final
        …
        a3ERPACTIVEXArticulo.Siguiente;               // Se situa en el siguiente registro
end;
end;
```

10.- Documentos

Los documentos son algunos de los objetos más complejos que pueden tratarse con a3ERPACTIVEX. Con estos objetos pueden crearse los siguientes tipos de documentos en a3ERP:

- Facturas
- Albaranes
- Depósitos
- Pedidos
- Ofertas

- Regularizaciones
- Traspasos
- Inventarios
- Movimientos de stock
- Reservas de stock

Las operaciones que pueden realizarse con este tipo de objeto son: Añadir, Borrar y Modificar.
Para ello, es imprescindible crear e iniciar el objeto (con el procedimiento **Iniciar**). Al finalizar su uso hay que liberar los recursos (con el procedimiento **Acabar**).

```
var
a3ERPACTIVEXFactura: Factura;
begin
a3ERPACTIVEXFactura := coFactura.Create;      // Crea el documento factura
a3ERPACTIVEXFactura.Iniciar;                  // Inicia el objeto
.
a3ERPACTIVEXFactura.Acabar;                   // Libera los recursos internos del objeto
end;
```

Ejemplo en Delphi de la creación de una nueva factura

```
// Dar de alta una factura
Procedure UnaFactura;
Var
a3ERPACTIVEXFactura: Factura;
nFact: double;  //  Identifica a la factura que vamos a crear
Begin
// Crea el objeto
a3ERPACTIVEXFactura := coFactura.Create;
boaFctura.Iniciar;
// Comenzar una nueva factura
a3ERPACTIVEXFactura.Nuevo(  '30/03/2002',   // Fecha de la factura
'1',              // Código del cliente
False,            // Indica que es de venta (True indicaría que es de compra)
False,            // Indica que es de gestión (True indicaría que es contable)
True,             // Indica que se desean las repercusiones contables
True);            // Indica que se desean los vencimientos
// Cambiar algunos campos de la cabecera
a3ERPACTIVEXFactura.AsStringCab[ 'Tarifa'] := 'PVP';
// Añadir una línea en la factura
a3ERPACTIVEXFactura.NuevaLineaArt(  '1',      // Código del artículo20);        //  Unidades de la línea
a3ERPACTIVEXFactura.AsFloatLin[ 'PrcMoneda] := 1500;
a3ERPACTIVEXFactura.AnadirLinea;
// Genera definitivamente la factura. RETORNA UN IDENTIFICADOR
nFact := a3ERPACTIVEXFactura.Anade;
a3ERPACTIVEXFactura.Acabar;                   // Libera los recursos del objeto
End;  //  Fin del procedimiento
```

El identificador que retorna la función Anade (nFact), puede almacenarse donde desee el programador para más adelante utilizarlo, por ejemplo, para borrar esa factura, si es necesario.

Es importante tener en cuenta la gran parte del trabajo oculto que realiza a3ERPACTIVEX por nosotros. En este caso al guardar la factura:

1. Se asignan los valores por defecto dado el cliente y el artículo de la línea.
2. Se generan los vencimientos
3. Se generan los asientos contables
4. Se actualiza el stock de cada artículo en las líneas, dado el almacén de la cabecera.

Esto se hace de una forma totalmente oculta, y al programador que utiliza a3ERPACTIVEX le es absolutamente indiferente de qué forma se ha implementado internamente cada una de las operaciones.

En el futuro, puede que se modifique la implementación interna, pero para a3ERPACTIVEX las operaciones seguirán siendo las mismas.

Para realizar este borrado lo haremos como en la siguiente función **ejemplo**:

```
Procedure BorrarFactura( nIdFac:Double);
var
a3ERPACTIVEXFactura: Factura;
nFact: double;  //  Identifica a la factura que vamos a crear
Begin
// Crea el objeto
a3ERPACTIVEXFactura := coFactura.Create;
boaFctura.Iniciar;
// Borrar la factura con el IDENTIFICADOR pasado al procedimiento
a3ERPACTIVEXFactura.Borra( nIdFac, False);   // False indica que se trata de una factura de venta
a3ERPACTIVEXFactura.Acabar;
        end;

Procedure ModificarFactura( nIdFac:Double);
var
a3ERPACTIVEXFactura: Factura;
nFact: double;  //  Identifica a la factura que vamos a crear
Begin
// Crea el objeto
a3ERPACTIVEXFactura := coFactura.Create;
boaFctura.Iniciar;
// modifica  la factura con el IDENTIFICADOR pasado al procedimiento
a3ERPACTIVEXFactura.Modifica( nIdFac, False);   // False indica que se trata de una factura de venta
// Cambiar algunos campos de la cabecera
a3ERPACTIVEXFactura.AsStringCab[ 'Tarifa'] := 'PVP';

a3ERPACTIVEXFactura.Anade;
a3ERPACTIVEXFactura.Acabar;
end;
```

11.- Apuntes

El objeto Asiento nos permite generar apuntes en contabilidad, borrarlos y modificarlos.
Internamente acumula saldos y controla que no se produzcan errores de coherencia.
Un ejemplo claro de la utilización del objeto Asiento sería el que crearía los apuntes correspondientes a una factura.

```
Function AsientoFactura( sFecha, sTipoCont, sCtaCliente, sNumFac: string; nBase: Double): Double;
var
a3ERPACTIVEXAsiento: Asiento;
begin
// Crea el objeto asiento
a3ERPACTIVEXAsiento := coAsiento.Create;
a3ERPACTIVEXAsiento.Iniciar;
// En primer lugar se indican los datos de la cabecera del asiento que coinciden en todas las líneas
// 'N' indica que se trata de un asiento normal
a3ERPACTIVEXAsiento.Nuevo( sFecha, sTipoCont,'N');
// Otras posibilidades:
// 'A': Apertura, 'P': Explotación, 'T': Cierre
// A continuación se especifican los datos de cada una de las líneas
a3ERPACTIVEXAsiento.AApunte(        'EURO',                 // Moneda
                    '70000000',     // Cuenta
                    0,              // Importe debe en la moneda indicada
                    'De n/fra. ' + sNumFac,         // Descripción
                    nBase,          // Importe haber en la moneda indicada
                    '',             // Centro de coste
                    '');            // Texto
a3ERPACTIVEXAsiento.AApunte(        'EURO',                 // Moneda
                    '47700000',     // Cuenta
                    0,              // Importe debe en la moneda indicada
                    'De n/fra. ' + sNumFac,         // Descripción
                    nBase * 0.16,           // Importe haber en la moneda indicada
                    '',             // Centro de coste
                    '');            // Texto
a3ERPACTIVEXAsiento.AApunte(        'EURO',                 // Moneda
                    sCtaCliente,    // Cuenta
                    nBase * 1.16,   // Importe debe en la moneda indicada
                    'De n/fra. ' + sNumFac,         // Descripción
                    0,              // Importe haber en la moneda indicada
                    '',             // Centro de coste
                    '');            // Texto
        // Al acabar el asiento nos retorna el IDENTIFICADOR por si deseamos utilizarlo en el futuro
        //  para borrar/modificar este asiento.
Result := a3ERPACTIVEXAsiento.Anade;
a3ERPACTIVEXAsiento.Acabar;                         // Libera los recursos del objeto
end;
```

11.1.- Dos formas de trabajar con asientos

En la versión anterior a a3ERP (Euro 2000) se trabajaba con las primitivas que acabamos de ver para generar asientos, pero en a3ERP se han ampliado para introducir nuevas prestaciones, así como nuevos campos propios de la nueva versión de a3ERP.
A continuación veremos el **mismo ejemplo, pero con las nuevas funciones**:

```
Function AsientoFactura( sFecha, sTipoCont, sCtaCliente, sNumFac: string; nBase: Double): Double;
var
a3ERPACTIVEXAsiento: Asiento;
begin
// Crea el objeto asiento
a3ERPACTIVEXAsiento := coAsiento.Create;
a3ERPACTIVEXAsiento.Iniciar;

// En primer lugar se indican los datos de la cabecera del asiento que coinciden en todas las líneas
a3ERPACTIVEXAsiento.NuevoA3ERPACTIVEX(    'N',              // 'N' indica que se trata de un asiento
normal
                                          // Otras posibilidades:
                                          // 'A': Apertura, 'P': Explotación, 'T': Cierre
sTipoCont,      // Tipo contable
sDiario,        // Diario contable
sNumDoc,        // Nº documento
sFecha);        // Fecha
// A continuación se especifican los datos de cada una de las líneas
a3ERPACTIVEXAsiento.AApunteA3ERPACTIVEX(          'FV',          // Concepto
'De n/fra. ' + sNumFac,          // Descripción
'',             // Centro de coste 1
'',             // Centro de coste 2
'',             // Centro de coste 3
'EURO',                 // Moneda
'70000000',     // Cuenta
0,              // Importe debe en la moneda indicada
0,              // Importe debe en la moneda principal
nBase,          // Importe haber en la moneda indicada
nBase,          // Importe haber en la moneda principal
 '',            // Texto
sFecha);                // Fecha valor
a3ERPACTIVEXAsiento.AApunteA3ERPACTIVEX(          'FV',          // Concepto
'De n/fra. ' + sNumFac,          // Descripción
'',             // Centro de coste 1
'',             // Centro de coste 2
'',             // Centro de coste 3
'EURO',                 // Moneda
'47700000',     // Cuenta
0,              // Importe debe en la moneda indicada
0,              // Importe debe en la moneda principal
```

```
nBase * 0.16,    // Importe haber en la moneda indicada
nBase * 0.16,    // Importe haber en la moneda principal
 '',             // Texto
sFecha);                 // Fecha valor
a3ERPACTIVEXAsiento.AApunteA3ERPACTIVEX(        'FV',          // Concepto
'De n/fra. ' + sNumFac,          // Descripción
'',              // Centro de coste 1
'',              // Centro de coste 2
'',              // Centro de coste 3
'EURO',                  // Moneda
sCtaCliente,     // Cuenta
nBase * 1.16,    // Importe debe en la moneda indicada
nBase * 1.16,    // Importe debe en la moneda principal
0,               // Importe haber en la moneda indicada
0,               // Importe haber en la moneda principal
 '',             // Texto
sFecha);                 // Fecha valor
// Al acabar el asiento nos retorna el IDENTIFICADOR por si deseamos utilizarlo en el futuro
//  para borrar/modificar este asiento.
Result := a3ERPACTIVEXAsiento.Anade;
a3ERPACTIVEXAsiento.Acabar;                          // Libera los recursos del objeto
end;
```

El IDENTIFICADOR retornado por Anade podremos utilizarlo para borrar el asiento, como se hace en el siguiente ejemplo:

```
Procedure BorrarAsiento ( nAsiento: double);
var
a3ERPACTIVEXAsiento: Asiento;
begin
a3ERPACTIVEXAsiento := coAsiento.Create;      // Crea el objeto asiento
a3ERPACTIVEXAsiento.Iniciar;
a3ERPACTIVEXAsiento.Borra( nAsiento);              // Borra el asiento con el identificador indicado
a3ERPACTIVEXAsiento.Acabar;                  // Libera los recursos del objeto
end;
```

12.- Cartera

En cartera existen tres tipos de funciones:

- Creación, modificación y borrado de vencimientos.
- Realizar operaciones con vencimientos existentes
- Crear remesas y agrupaciones

12.1.- Creación, modificación y borrado de vencimientos

Un ejemplo claro de la utilización del objeto Cartera sería el que crearía los vencimientos correspondientes a una factura en dos plazos.

```
Function VencimientoFacturaVenta(sCodCli, sFechaFactura, sCodMon, sTipoCont, sSerie, sNumFac,
sFecha1, sFecha2: string; nImporte: Double): Double;
var
a3ERPACTIVEXCartera: Cartera;
begin
// Crea el objeto cartera
a3ERPACTIVEXCartera := coCartera.Create;
a3ERPACTIVEXCartera.Iniciar;
// En primer lugar se indican los datos de cabecera que son comunes para todos los vencimientos.
a3ERPACTIVEXCartera.Nuevo( sFecha, sCodCli, sCodMon, sTipoCont, sSerie, sNumFac, False);
// A continuación se indican los datos de cada uno de los vencimientos
a3ERPACTIVEXCartera.NuevoVen( sFecha1, Int( nImporte/2));
a3ERPACTIVEXCartera.AnadirVen;
a3ERPACTIVEXCartera.NuevoVenVen( sFecha2, nImporte - Int( nImporte/2));
a3ERPACTIVEXCartera.AnadirVen;
// Al acabar los cartera nos retorna el IDENTIFICADOR, por si deseamos utilizarlo en el futuro
//  para borrar/modificar este registro de cartera.
Result := a3ERPACTIVEXCartera.Anade;
a3ERPACTIVEXCartera.Acabar;
end;
```

El IDENTIFICADOR retornado por Anade podremos utilizarlo para borrar la cartera, como se hace en el siguiente ejemplo:

```
Procedure BorrarCartera( nCartera: double);
var
a3ERPACTIVEXCartera: Cartera;
begin
a3ERPACTIVEXCartera := coCartera.Create;       // Crea el objeto cartera
a3ERPACTIVEXCartera.Iniciar;
a3ERPACTIVEXCartera.Borra( nCartera);                    // Borra la cartera con el identificador indicado
a3ERPACTIVEXCartera.Acabar;
end;
```

Ejemplo 1: Ejemplo cómo modificar un vencimiento y añadir un nuevo vencimiento en cartera

```
A3ERPACTIVEXCartera := CoCartera.Create;
A3ERPACTIVEXCartera.Iniciar;
nNumCartera := 6398;
if A3ERPACTIVEXCartera.PuedeModificarse(nNumcartera) then
begin
A3ERPACTIVEXCartera.Editar(nNumcartera); // editamos el numero de cartera
A3ERPACTIVEXCartera.EditarVencimiento( 1); // editamos el vencimiento 1

A3ERPACTIVEXCartera.VAsFloat['IMPORTEMON'] := A3ERPACTIVEXCartera.VAsFloat['IMPORTEMON'] -
500;
A3ERPACTIVEXCartera.GuardarVencimiento; // al vencimiento 1 hemos restado 500€ y lo guardamos

// vamos a crear un nuevo vencimiento del 500€ con fecha 24/3/2010 para la cartera 6398
A3ERPACTIVEXCartera.AsFloat['IMPORTEMON'] := 500;
Nuevo := A3ERPACTIVEXCartera.AnadirVen;

A3ERPACTIVEXCartera.EditarVencimiento(Nuevo);
A3ERPACTIVEXCartera.VAsString['FECHA'] := '24/03/2010';
A3ERPACTIVEXCartera.GuardarVencimiento;

A3ERPACTIVEXCartera.Anade;
end;
A3ERPACTIVEXCartera.Acabar;
```

12.2.- Realizar operaciones con vencimientos existentes

Este conjunto de operaciones permite operar con los efectos pendientes de cada cliente/proveedor.
Las operaciones que pueden realizarse con a3ERPACTIVEX sobre la cartera son las siguientes:

Cobrar	Para realizar el cobro de efectos.
Pagar	Para realizar el pago de efectos.
Recibir	Para realizar la recepción de efectos.
CrearAnticipo	Para crear anticipos.
CrearAnticipoA3ERPACTIVEX	Para indicar que un importe de una factura se ha entregado a cuenta en un anticipo entregado anteriormente. Incluye el nº de documento contable.
Bloquear	Para bloquear un efecto y no permitir realizar con él más operaciones.
Enviar	Para marcar un efecto a un proveedor como enviado.
ActualizarRiesgo	Para actualizar el riesgo de un efecto al descuento cuyo vencimiento ha sido superado.
Devolver	Para indicar la devolución de un efecto cobrado indicando sus gastos.
ImputarGastos	Para imputar los gastos de una devolución a un cliente.
RecibirA3ERPACTIVEX	Ídem que **Recibir,** pero con más parámetros.

CobrarA3ERPACTIVEX	Ídem que **Cobrar,** pero con más parámetros.
PagarA3ERPACTIVEX	Ídem que **Pagar,** pero con más parámetros.
AnularCobro	Anular el cobro de un vencimiento.
AnularPago	Anular el pago de un vencimiento.
AnularRecepcion	Anular la recepción de un efecto.
AnularEnvio	Anular el envío de un efecto.
AnularBloqueo	Anular el bloqueo de un efecto.
AnularImputacion	Anular la imputación de gastos de una devolución al cliente.
AnularDevolucion	Anular una devolución.
AnularActualizacionRiesgo	Anular la actualización de riesgo de un efecto al descuento.
Ligar	Ligar un conjunto de efectos de cartera con una factura.
Desligar	Ligar un conjunto de efectos de cartera con una factura.
CobrarRemesa	Cobrar una remesa.
AnularCobroRemesa	Anular el cobro de una remesa.
PagarRemesa	Pagar una remesa (a proveedores).
AnularPagoRemesa	Anular el pago de una remesa (a proveedores).
ACuenta	Para consumir anticipos de cartera desde una factura.

A continuación, se muestran **ejemplos del uso de estas operaciones**:

Ejemplo 1: Cobro de un vencimiento

```
Procedure CobroVencimiento( nNumCart: double; nNumVen: integer; sCodBan: string);
var
a3ERPACTIVEXCartera: OperacionesCartera;
begin
a3ERPACTIVEXCartera := coOperacionesCartera.Create;
a3ERPACTIVEXCartera.Cobrar (   nNumCart,      // Nº de conjunto de cartera
nNumVen,       // Nº de vencimiento
True,          // Generar repercusiones contables
False,         // Al cobro (si fuera True sería al descuento)
'',            // Cuenta del banco
// (en blanco lo busca en el banco que le indicamos)
'',            // Cuenta de riesgo del banco
// (en blanco lo busca en el banco que le indicamos)
TODAY,         // Fecha
sCodBan);      // Código del banco
end;
```

13.- Stocks

Además de los movimientos propios efectuados por facturas, albaranes, regularizaciones, etc, se ha dotado a a3ERPACTIVEX de la posibilidad de generar movimientos de stock sin estar ligados con los documentos anteriores (por ejemplo, para un programa de producción externo que nos mueva el stock).

Ejemplo 1: Movimientos en stocks externos

```
var
a3ERPACTIVEXMovimientos: MovimientoStock;
begin

a3ERPACTIVEXMovimientos := coMovimientoStock.Create;
a3ERPACTIVEXMovimientos.Iniciar;                    // Reserva los recursos necesarios
a3ERPACTIVEXMovimientos.Nuevo(     '01/12/2003',   // Fecha
'2');              // Código del almacén

// Añadir movimiento de 10 unidades del artículo 1
a3ERPACTIVEXMovimientos.NuevaLineaArt(     '1',      // Código del artículo
10);      // Unidades del movimiento
a3ERPACTIVEXMovimientos.AsStringLin[ 'Motivo'] := 'Movimiento externo';
a3ERPACTIVEXMovimientos.AnadirLinea;

// Añadir movimiento de 20 unidades del artículo 2
a3ERPACTIVEXMovimientos.NuevaLineaArt(     '2',      // Código del artículo
20);      // Unidades del movimiento
a3ERPACTIVEXMovimientos.AsStringLin[ 'Motivo'] := 'Movimiento externo';
a3ERPACTIVEXMovimientos.AnadirLinea;

// Añadir movimiento de 30 unidades del artículo 3
a3ERPACTIVEXMovimientos.NuevaLineaArt(     '3',      // Código del artículo
30);      // Unidades del movimiento
a3ERPACTIVEXMovimientos.AsStringLin[ 'Motivo'] := 'Movimiento externo';
a3ERPACTIVEXMovimientos.AnadirLinea;
a3ERPACTIVEXMovimientos.Anade;                      // Genera los movimientos en la base de datos
a3ERPACTIVEXMovimientos.Acabar;                     // Libera los recursos utilizados
end;
```

14.- Filtro

Este objeto hace referencia a la pantalla visual de a3ERP de creación de filtros. Este objeto permite llamar a la pantalla visual o crearlo de forma interna mediante los eventos y propiedades públicas que tiene publicadas.

Ejemplo 1: Llamada a la pantalla de filtros

```
Var
A3ERPACTIVEXFiltro  : Filtro;
Begin
  A3ERPACTIVEXFiltro  := CoFiltro.Create;                    // Creamos el objeto
  A3ERPACTIVEXFiltro.Iniciar( 'CLIENTES');                   // Indicamos que queremos crear el filtro de clientes
  A3ERPACTIVEXFiltro.EditarVisual;                           // Llamamos a la pantalla visual
  sFiltro := A3ERPACTIVEXFiltro.ObtenerFiltro;               // Obtenemos el filtro de generado desde la pantalla visual
  A3ERPACTIVEXFiltro.Acabar;                                 // Libera los recursos utilizados
End;
```

15.- Expediente

Este objeto hace referencia a la pantalla visual de gestión de expedientes de a3ERP y permite realizar las mismas operaciones que a3ERP, mediante la implementación de los eventos y propiedades que tiene publicadas.

De esta forma, es posible crear líneas de expedientes, líneas de honorarios, líneas de provisiones y líneas de suplidos. Generando las repercusiones contables cuando proceda. También podemos realizar la impresión del mismo.

Ejemplo 1: Utilización genérica del objeto expedientes

Constantes a tener presente (ctHonorario, ctTrabajo, ctProvision, ctSuplido)

```
var
  A3ERPACTIVEXExpe:Expediente;
begin
    A3ERPACTIVEXExpe := CoExpediente.Create;                 // Creamos el objeto
    A3ERPACTIVEXExpe.Iniciar;                                // Iniciamos el objeto
    A3ERPACTIVEXExpe.Nuevo('1', '28/07/2006');               // Creamos la cabecera indicando cliente y fecha
    A3ERPACTIVEXExpe.AsStringCab['NOMEXP'] := 'Expediente realizado con a3ERPACTIVEX';  // Modificamos el nombre del expediente
    A3ERPACTIVEXExpe.NuevaLinea(ctTrabajo);                  // Creamos una nueva linea de trabajo (const ctTrabajo)
    A3ERPACTIVEXExpe.AsStringLin['CodArt'] := '1';           // Indicamos el artículo a tratar
    A3ERPACTIVEXExpe.AsStringLin['CodPer'] := '1';           // Indicamos la persona que lo realiza
    A3ERPACTIVEXExpe.AsFloatLin['Unidades'] := 2;            // Modifico las unidades por defecto a 2
    A3ERPACTIVEXExpe.AsFloatLin['PrcMoneda'] := 6;           // Modifico el precio por defecto a 6
    A3ERPACTIVEXExpe.AnadirLinea;                            // Guardamos la linea
```

```
  A3ERPACTIVEXExpe.ActivarFacturado := true;          // Activar marca para poder facturarlo
  A3ERPACTIVEXExpe.Facturar( A3ERPACTIVEXExpe.Anade, true);       // Guardo el expediente (Anade)
y lo facturo
  A3ERPACTIVEXExpe.ActivarFacturado := false;          // DesActivar marca para poder facturar
  A3ERPACTIVEXExpe.Acabar;
end;
```

16.- Cobro/ Anulación parcial de remesas

Estos dos objetos permiten realizar el cobro parcial de remesas y/o anulación de cobros (realizados a partir de cobros parciales de remesas).

Ejemplo 1: Utilización cobro parcial de remesas

```
var
CobroParcial:ICobroParcialRemesa;

begin
 CobroParcial := CoCobroParcialRemesa.Create;
 Screen.Cursor := crHourGlass;
 try
  CobroParcial.Iniciar(2);
  CobroParcial.AnadeEfecto(15231, 2);
  CobroParcial.AnadeEfecto(15232, 2);
  CobroParcial.ConRepercusiones := true;
  CobroParcial.UnAsiento := true;
  CobroParcial.AlDescuento := false;
  CobroParcial.CodigoBanco := '      1';
  CobroParcial.Fecha := DateToStr(Date);
  CobroParcial.FechaValor := DateToStr(Date);

  CobroParcial.Procesar(chkPedir.Checked);
 finally
  CobroParcial.Finalizar;
  CobroParcial := nil;
  Screen.Cursor := crDefault;
 end;
end;
```

Ejemplo 2: Utilización de anular cobro parcial de remesas

```
var
  query:TAdoQuery;
  CobroParcial: IAnularCobroParcialRemesa;
begin
  query:=TAdoQuery.create(nil);
  query.ConnectionString := boaEnlace.Conexion;
  query.SQL.Text := 'select IDOPEREMESA FROM __OPEREMESA WHERE IDREMESA = 2';
  query.open;
  CobroParcial := CoAnularCobroParcialRemesa.Create;
  Screen.Cursor := crHourGlass;
  try
    CobroParcial.Iniciar(2);
    while not query.Eof do begin
      CobroParcial.AnadeOperacion(query.FieldByName('IDOPEREMESA').AsInteger);
      query.Next;
    end;
    CobroParcial.Procesar;
  finally
    FreeAndNil(query);
    CobroParcial.Finalizar;
    CobroParcial := nil;
    Screen.Cursor := crDefault;
  end;
end;
```

17.- Enlace a3doc cloud

Este objeto permite operar y gestionar publicaciones en a3doc cloud a partir de su enlace con a3ERP. Como soporte a sus funcionalidades, se han creado **dos tipos Interface** para formatear el paso de parámetros y para recoger algunos valores retornados. Estos tipos son:

- **IAtributoA3DOC**
- **IAtributoA3DOCColeccion**

Interfaz IAtributoA3DOC: estructura que encapsula tres propiedades:

- **Índice: Entero.** Representa la posición del atributo en una lista de IAtributoA3DOC.
- **Descripción: String.** Representa la descripción del atributo.
- **Valor: String.** Representa el valor del atributo

Esta estructura tiene dos usos diferenciados:

1. **Recoger los documentos encontrados a partir de una búsqueda:** se devuelve una colección de IAtributoA3DOC en el que sus propiedades tienen el siguiente significado:
    - **Índice:** la posición del documento en la lista de documentos.
    - **Descripción:** Nombre y extensión del fichero.
    - **Valor:** Guid de identificador único del fichero. Necesario para visualizarlo o borrarlo.

2. **Recoger una lista de categorías de a3doc.** Mediante la función ObtenerCategorias, obtendremos una lista de todas las categorías para permitir seleccionar algunas al usuario. Se usa también como parámetro de la función EnviarDocumento para especificar en qué categorías debe publicarse un archivo.

    Sus propiedades tienen el siguiente significado:
    - **Índice:** la posición de la categoría en la lista de categorías.
    - **Descripción:** nombre y ruta de la categoría (como, por ejemplo, Ventas\Facturas)
    - **Valor:** identificador de la categoría.

Interfaz IAtributoA3DOCColeccion: estructura que representa una lista de objetos IAtributoA3DOC. Esta lista contiene una propiedad y dos métodos:

- **Propiedad Length: Entero.** Propiedad que nos devuelve el número de IAtributoA3DOC que contiene la lista.
- **Función GetAtributo(Indice: int): IAtributoA3DOC.** Función que nos devuelve el objeto IAtributoA3DOC que se encuentra en la posición especificada por parámetro. Se usa para explorar los valores obtenidos al recorrer la lista.
- **Procedimiento AgregarAtributo(Atributo: IAtributoA3DOC).** Procedimiento que agrega un IAtributoA3DOC a una lista en la última posición. Se usa para construir una lista IAtributoA3DOCColeccion.

Enumerado EntidadesA3DOCNAX: enumerado que referencia las diferentes tablas de entidadas soportadas por el EnlaceA3DOC. Los elementos del enumerado son los siguientes:

- **Enum** DocumentosVentasOferta **references 'CABEOFEV'**
- **Enum** DocumentosVentasPedido **references 'CABEPEDV'**
- **Enum** DocumentosVentasAlbaran **references 'CABEALBV'**
- **Enum** DocumentosVentasDeposito **references '__CABEDEPV'**
- **Enum** DocumentosVentasFactura **references 'CABEFACV'**
- **Enum** DocumentosComprasOferta **references 'CABEOFEC'**
- **Enum** DocumentosComprasPedido **references 'CABEPEDC'**
- **Enum** DocumentosComprasAlbaran **references 'CABEALBC'**
- **Enum** DocumentosComprasDeposito **references '__CABEDEPC'**
- **Enum** DocumentosComprasFactura **references 'CABEFACC'**
- **Enum** Expedientes **references '__CABEEXPE'**
- **Enum** AlbaranTraspaso **references 'CABETRAS'**
- **Enum** AlbaranRegularizacion **references 'CABEREGU'**

Objeto EnlaceA3DOC: los métodos y funciones del objeto son los siguientes:

- procedure Iniciar
- procedure Acabar
- function EnviarDocumento(entidad: EntidadesA3DOCNAX; identificadorEntidad: string; pathDocumento: string; categorias: IAtributoA3DOCColeccion): string
- function BorrarDocumento(identificadorEntidad: string): bool
- function VisualizarDocumento(identificadorEntidad: string): bool
- function ObtenerDocumentos(entidad: EntidadesA3DOCNAX; identificadorEntidad: string): IAtributoA3DOCColeccion
- function ObtenerCategorias: IAtributoA3DOCColeccion
- function NuevaColeccionAtributoA3DOC: IAtributoA3DOCColeccion
- function NuevoAtributoA3DOC: IAtributoA3DOC

Procedimiento Iniciar:
Método sin parámetros que inicia el objeto. Es necesario para poder ejecutar cualquier otro método del objeto.

Procedimiento Acabar:
Método sin parámetros que finaliza el objeto. Se debe ejecutar cuando existe un objeto EnlaceA3DOC iniciado con el que ya no se va a operar más.

Función EnviarDocumento(entidad: EntidadesA3DOCNAX; identificadorEntidad: string; pathDocumento: string; categorias: IAtributoA3DOCColeccion): string:
Función para publicar vinculo a un archivo asociado a un registro.
Recibe los siguientes parámetros:

- **Entidad:** enumerado EntidadesA3DOCNAX que referencia a que tipo de documento se va a crear el vínculo.
- **IdentificadorEntidad;** string que hace referencia al identificador de la entidad al que se va a vincular el documento.
- **PathDocumento:** string que hace referencia a la ruta y el nombre y extensión del fichero que se va a publicar.
- **Categorías:** objeto IAtributoA3DOCColeccion con las categorías en las que se va a publicar el archivo.
  Devuelve un string con el identificador único del documento publicado.

>> Ejemplo en C# de ENVIARDOCUMENTO

```
ENTIDADESA3DOCNAX ENTIDAD = ENTIDADESA3DOCNAX.DOCUMENTOSVENTASFACTURA;
STRING IDFACTURA = CREARFACTURA();

IATRIBUTOA3DOC ATRIBUTO = A3DOC.NUEVOATRIBUTOA3DOC();
ATRIBUTO.INDICE = 0;
ATRIBUTO.DESCRIPCION = "VENTAS\INFORMES";
ATRIBUTO.VALOR = "966";
IATRIBUTOA3DOCCOLECCION PRUEBAS = A3DOC.NUEVACOLECCIONATRIBUTOA3DOC();
PRUEBAS.AGREGARATRIBUTO(ATRIBUTO);
RETURN A3DOC.ENVIARDOCUMENTO(ENTIDAD, IDFACTURA, "C:\\TEMP\\PRUEBA1.TXT", PRUEBAS);
```

function BorrarDocumento(identificadorEntidad: string): bool

Función para eliminar el vínculo a un documento a partir de su identificador. Recibe los siguientes parámetros:

- **IdentificadorEntidad;** string que hace referencia al identificador único del documento que se va a borrar

Devuelve un verdadero en caso en que pueda encontrar y eliminar el documento. Devuelve falso en caso contrario.

>> Ejemplo en C# de BORRARDOCUMENTO

```
ENTIDADESA3DOCNAX ENTIDAD = ENTIDADESA3DOCNAX.EXPEDIENTES;
STRING IDEXPEDIENTE = GETNUMEROEXPEDIENTESELECCIONADO();
IATRIBUTOA3DOCCOLECCION ATR = A3DOC.OBTENERDOCUMENTOS(ENTIDAD, IDEXPEDIENTE);
BOOL BORRADO = FALSE;
FOR (INT X = 0; X < ATR.LENGTH; X++)
{
  IF (ATR.GETATRIBUTO(X).DESCRIPCION==DOCUMENTOSELECCIONADO.NOMBRE)
   BORRADO = A3DOC.BORRARDOCUMENTO(ATR.GETATRIBUTO(X).VALOR);
}
IF (BORRADO)
 MESSAGEBOX.SHOW("SE HA ELIMINADO LA FICHA SELECCIONADA");
```

function VisualizarDocumento(identificadorEntidad: string): bool

Función para abrir un documento vinculado en a3doc cloud a partir de su identificador único. El archivo se abrirá con la aplicación predeterminada para abrir ese tipo de archivo. Recibe los siguientes parámetros:

- **IdentificadorEntidad;** string que hace referencia al identificador único del documento que se va a visualizar.
  Devuelve un verdadero en caso en que pueda encontrar y eliminar el documento. Devuelve falso en caso contrario.

>> Ejemplo en C# de VISUALIZARDOCUMENTO

```
ENTIDADESA3DOCNAX ENTIDAD = ENTIDADESA3DOCNAX.EXPEDIENTES;
STRING IDEXPEDIENTE = GETNUMEROEXPEDIENTESELECCIONADO();
IATRIBUTOA3DOCCOLECCION ATR = A3DOC.OBTENERDOCUMENTOS(ENTIDAD, IDEXPEDIENTE);
BOOL ENCONTRADO = FALSE;
FOR (INT X = 0; X < ATR.LENGTH; X++)
{
  IF (ATR.GETATRIBUTO(X).DESCRIPCION==DOCUMENTOSELECCIONADO.NOMBRE)
   ENCONTRADO = A3DOC. VISUALIZARDOCUMENTO(ATR.GETATRIBUTO(X).VALOR);
}
```

function ObtenerDocumentos(entidad: EntidadesA3DOCNAX; identificadorEntidad: string): IAtributoA3DOCColeccion

Función para obtener un listado de los documentos vinculados a un registro. Recibe los siguientes parámetros:

- **Entidad:** enumerado EntidadesA3DOCNAX que referencia a que tipo de documento del registro.
- **IdentificadorEntidad**; string que hace referencia al identificador del registro en la entidad.

Devuleve una estructura IAtributosA3DOCColeccion con el listado de los documentos publicados asociados al registro. Estos registros tienen la estructura:

- **Índice:** la posición del documento en la lista de documentos.
- **Descripción:** nombre y extensión del fichero.
- **Valor:** guid de identificador único del fichero. Necesario para visualizarlo o borrarlo.

Ejemplo en C# de OBTENERDOCUMENTOS

```
ENTIDADESA3DOCNAX ENTIDAD = ENTIDADESA3DOCNAX.EXPEDIENTES;
STRING IDEXPEDIENTE = GETNUMEROEXPEDIENTESELECCIONADO();
IATRIBUTOA3DOCCOLECCION ATR = A3DOC.OBTENERDOCUMENTOS(ENTIDAD, IDEXPEDIENTE);
INT NUMERODOC;
STRING NOMBREDOC;
STRING IDDOC;
FOR (INT X = 0; X < ATR.LENGTH; X++)
{
        NUMERODOC = ATR.GETATRIBUTO(X).INDICE;
    NOMBREDOC = ATR.GETATRIBUTO(X).DESCRIPCION;
    IDDOC = ATR.GETATRIBUTO(X).VALOR;
        MESSAGEBOX.SHOW(FORMAT("DOCUMENTO: {0} CON ID {1}",NOMBREDOC, IDDOC));
}
```

function ObtenerCategorias: IAtributoA3DOCColeccion

Función que obtiene todas las categorías existentes en a3doc cloud. Su finalidad es obtener las categorías existentes para mostrarlas al usuario y permitir elegir al usuario o bien para a través de estos datos configurar en que categoría se vincula cada tipo de registro

Devuleve una estructura IAtributosA3DOCColeccion con el listado de las categorías. Estos registros tienen la estructura:

- **Índice:** la posición de la categoría en la lista.
- **Descripción:** nombre y ruta de la categoría. (Ejemplo: Ventas\Facturas).
- **Valor:** identificador de la categoría.

>> Ejemplo en C# de OBTENERCATEGORIAS

```
IATRIBUTOA3DOCCOLECCION ARBOLCATEGORIAS = A3DOC.OBTENERCATEGORIAS();
STRING LISTACATEGORIAS="";
IATRIBUTOA3DOC CATEGORIA;
FOR (INT X=0; X <= ARBOLCATEGORIAS.LENGTH - 1; X++)
{
 CATEGORIA = (IATRIBUTOA3DOC)ARBOLCATEGORIAS.GETATRIBUTO(X);
 LISTACATEGORIAS += CATEGORIA.DESCRIPCION + "-" + CATEGORIA.VALOR + "\N\R";
 IF ((X == 50) || (X == 100))
 {
  MESSAGEBOX.SHOW("LISTA DE CATEGORIAS:\N\R" +LISTACATEGORIAS);
  LISTACATEGORIAS = "";
 }
 IF (CATEGORIA.DESCRIPCION != NULL)
 {
  CHECKCATEGORIAS.ITEMS.ADD(CATEGORIA.DESCRIPCION);
 }

}
MESSAGEBOX.SHOW("LISTA DE CATEGORIAS:\N\R" + LISTACATEGORIAS);
```

function NuevaColeccionAtributoA3DOC: IAtributoA3DOCColeccion

Función que obtiene una lista de IAtributoA3DOC vacía. La finalidad del procedimiento es obtener un objeto vacío para poder llenarlo mediante la función AgregarAtributo

>> Ejemplo en C# de NUEVACOLECCIONATRIBUTOA3DOC

```
IATRIBUTOA3DOCCOLECCION CATEGORIAS = A3DOC.NUEVACOLECCIONATRIBUTOA3DOC();
```

function NuevoAtributoA3DOC: IAtributoA3DOC

Función que obtiene un objeto vació del tipo IAtributoA3DOC. La finalidad del procedimiento es obtener un objeto vacío para poder llenar sus propiedades y añadirlo a una lista del tipo IAtributoA3DOCColeccion

>> Ejemplo en C# de la NUEVACOLECCIONATRIBUTOA3DOC

```
 IATRIBUTOA3DOC ATRIBUTO = A3DOC.NUEVOATRIBUTOA3DOC();
 ATRIBUTO.INDICE = 0;
 ATRIBUTO.DESCRIPCION = "VENTAS\INFORMES";
 ATRIBUTO.VALOR = "966";
```