Departamento de Desarrollo Tecnológico

Sistema Administración de Compras y Control Financiero

Descripción Código VB

Versión: 1.00

Junio, 2006

TABLA DE CONTENIDOS

1. Generalidades

Para el desarrollo del sistema se definieron algunos estándares, los cuales se explicarán de forma general para ayudar a una mayor comprensión del código que da origen al sistema denominado SACCFI.

1.1 Objeto de conexión: "objAdoconnection" referencia al objeto adodb, este objeto es referenciado luego de autentificarse en el sistema, siendo un objeto global al mismo.

1.2 Consultas

Todas las consultas deberán abrir un objeto Recordset, el cual contendrá los resultados de la misma.

Para abrir el objeto recordset, se deberá llamar al procedimiento **OpenRecordset** que está definido en el módulo de Conexiones, y que recibe como parámetros un string con la consulta y una referencia al objeto de Ado. Este procedimiento retornará el recordset con los datos obtenidos tras ejecutar la consulta.

Para las consultas se usará la sentencia definida para trabajar con la mayor funcionalidad de access "**Join", "Right Join" y "Left Join"**

1.3 Acciones

Todas las acciones de actualización, eliminación e inserción serán ejecutadas a través del método execute del objeto de conexión.

1.4 Definición de variables

Para la definición de las variables se usará el nombre completo del tipo de dato que se define, y no la abreviación que posee VB.

1.5 Formularios

Todo nombre de formulario comenzará con el prefijo Frm.
Los formularios poseen procedimientos y funciones comunes que siempre deberán ser definidos para los formularios que mantienen registros, estos son:

Función **RegModificado**, cuyo objetivo es verificar si el registro actual en pantalla sufrió alguna modificación. Si es así se preguntará al usuario si desea grabar los cambios dándole las opciones de almacenarlos, no almacenarlos o de cancelar la operación solicitada.

Función **SugiereCodigo**, cuyo objetivo es entregar el próximo código a insertar para un registro, de manera de llevar un correlativo.

Procedimiento **DesHabilitaControles**, cuyo objetivo es el de deshabilitar los controles del formulario según sean los privilegios que tenga el usuario actual del sistema.

Procedimiento **HabilitaControles,** cuyo objetivo es el de habilitar los controles del formulario según sean los privilegios del usuario o el bloqueo del registro que se ha rescatado para su visualización.

Procedimiento **Carga_Masiva**, cuyo objetivo es llamar en algunos casos al formulario frmCargaMasiva, el cual recibe un string con la consulta y entrega un

listado con los registros obtenidos para su selección. Existen formularios que no llaman a frmCargaMasiva debido a la cantidad de atributos que deberán ser mostrados como resultado, para estos formularios se generarán formularios propios, denominados frmCargaMasivaxxx.

Función **ValidaDatos**, cuyo objetivo será el de validar la completitud de los datos que se requieren para poder grabar un determinado registro.

Procedimiento **CalledMasiva**, cuyo objetivo es el de obtener las claves del registro seleccionado desde el formulario frmCargaMasiva, para su posterior carga en el formulario desde el cual se llamó.

Procedimiento **MueveRegistros**, cuyo objetivo será el de intentar el bloqueo del registro solicitado, si el bloqueo es efectuado de forma satisfactoria se mostrará el registro sin ninguna restricción en cuanto a la funcionalidad del formulario, de no ser así se llamará al procedimiento deshabilitacontroles, el cual deshabilitará la funcionalidad del formulario.

Procedimiento **LLena_Controles**, cuyo objetivo es el de llenar los controles del formulario con los datos obtenidos de la consulta de un determinado registro.

Procedimiento **Limpia_Controles**, cuyo objetivo es el de limpiar los controles del formulario.

Función **Eliminar,** cuyo objetivo es eliminar el registro actual de la base de datos, esta función retornara un valor booleano que indicará si el registro fue eliminado satisfactoriamente.

Función **Grabar**, cuyo objetivo es grabar los cambios realizados en el formulario, esta función retornara un valor booleano que indicará si el registro fue grabado satisfactoriamente.

Función **ExisteRegistro,** cuyo objetivo es determinar si existe el registro en la base de datos, esta función retornará un valor booleano dependiendo de la existencia del registro.

2. Módulos

Los módulos del sistema encapsulan funciones y procedimientos de uso común a todos los formularios del sistema, estos módulos son.

2.1 Globales.bas

Módulo en el que se declaran todas las variables globales al sistema.

2.2 Conexiones.bas

Módulo en el que se declaran las funciones y procedimientos de conexión del sistema, estas son:

Procedimiento **OpenBd**, cuyo objetivo es el de abrir la base de datos instanciando el objeto objAdoConnection.

Procedimiento **CloseBd**, cuyo objetivo es el de cerrar la conexión a la base de datos.

Procedimiento **OpenRecordset**, cuyo objetivo es el de abrir un recordset con una determinada consulta, los parámetros de este procedimiento son un string pasado por valor con la consulta y una referencia al objeto adodb pasado por referencia que contendrá los resultados.

Procedimiento **CloseRecordset**, encargado de cerrar un objeto recordset, el parámetro de este procedimiento será el recordset a cerrar.

Función **ExisteBd**, encargada de determinar la existencia de la base de datos.

2.3 Bloqueos.bas

Módulo que encapsula las funciones y procedimientos que manejan los bloqueos a los registros del sistema, estas funciones son:

Función **Bloquea_Registro**, cuyo objetivo es intentar insertar el registro que se quiere bloquear, si este registro se logra insertar la función retornará un valor 1, si no es así retornará un valor 2, los parámetros que recibe esta función son, el código de bloqueo y el código de usuario.

Función **Desbloquea_Registro,** cuyo objetivo es desbloquear el registro que había sido tomado por un determinado usuario eliminándolo de la base de datos, los parámetros de esta función son el código de bloqueo y el usuario.

Función **Desbloqueo_Automatico**, función encargada de desbloquear todos los registros que estaban tomados desde una determinada máquina, esta función recibe como parámetros la dirección MAC del equipo y el número de serie del disco duro.

2.4 Ini.bas

Módulo en el que se definen las apis que usará el sistema para rescatar los datos desde el archivo ini.

2.5 MacAddress.bas

Módulo en el que se definen las funciones y procedimientos que serán los encargados de extraer información del equipo conectado a la red, estas son:

Función **GetMacAddr**, cuyo objetivo será retornar la dirección Mac de la tarjeta de red del pc que se está conectando al sistema.

Procedimiento **Extrae_Info_HD**, su objetivo será extraer información del disco duro, como el número de serie y la etiqueta.

Procedimiento **GetComputerName**, cuyo objetivo es retornar el nombre del pc.

2.6 OrdenesCompra.bas

Módulo en el que se encapsulan las funciones y procedimientos que tienen relación con las órdenes de compra, estos son:

Función **NuevaVersionPlanCompra**, cuyo objetivo es entregar un string con la nueva versión que se generará del plan de compras.

Función **EntregaPlanVigente**, cuyo objetivo es de entregar el código del plan de compras vigente.

Función **ProdServEstaEnPlanCompra**, cuyo objetivo es determinar si el producto/servicio se encuentra en el plan de compras.

Función **AnularOrdenCompra**, cuyo objetivo es anular una determinada orden de compra.

2.7 Redondeo.bas

Módulo donde se redefinen las funciones usadas para redondear un valor numérico.

Función **Round,** encargada de entregar el valor redondeado, los parámetros de entrada de esta función son el valor a redondear y la cantidad de decimales.

Función **Replicate,** cuyo objetivo es generar un string con “n” cantidades de “x” caracteres, los parámetros de esta función son nChar(caracteres) y nCant(cantidad).

2.8 Stock.bas

Módulo que encapsula las funciones y procedimientos que usará el módulo de existencias, estas son:

Función **BuscaMaximo,** cuyo objetivo es retornar el valor más alto de determinada consulta agregándole 1.

Función **ValidaParametros,** encargada de validar los parámetros de entrada.

Función **GeneraMovimiento,** genera un movimiento de un determinado producto, según el parámetro de entrada que se le envíe, ya sea Ingreso, Salida o Ajuste. Esta función retornará un valor booleano dependiendo del éxito de la operación.

Función **CalculaStockIngresos**, cuyo objetivo es determinar el stock que ha ingresado de un determinado producto pasado como parámetro.

Función **CalculaStockPedidoNoEntregado**, cuyo objetivo es determinar el stock que ha sido pedido pero no entregado de un determinado producto pasado como parámetro.

Función **CalculaStockEntregado**, cuyo objetivo es determinar el stock que ha efectivamente ha sido entregado de un determinado producto pasado como parámetro.

Función **CalculaStockActual**, cuyo objetivo es determinar el stock actual en bodega de un determinado producto pasado como parámetro.

2.9 Valida_Rut.bas

Módulo que encapsula las funciones que operan con el Rut, estas funciones son:

Función **Val_ValiRut**, cuyo objetivo es determinar si un determinado rut es válido, esta función recibe como parámetros el rut y el dígito verificador y retornará un valor booleano dependiendo de la validez de los datos ingresados.

Función **CalculaDV**, encargada de calcular el dígito verificador de rut enviado como parámetro, retorna el dígito verificador calculado.

2.10 Utiles.bas

Este módulo encapsula las funciones y procedimientos que sirven como apoyo al sistema denominado “SACCFI”, en este módulo podemos encontrar funciones de validación, funciones de formateo de valores y funciones de fecha.

3. MDI

El formulario Mdi, es el formulario principal del proyecto, el cual contiene el menú, y las llamadas a los demás formularios que componen el sistema. Este formulario merece atención aparte ya que es el encargado de habilitar las opciones de menús que posee un determinado Rol dentro del sistema. De modo que si se desea agregar un nuevo formulario será acá donde ha de ser configurado.

4. Consideraciones de Migración

Ante una eventual migración del sistema a Oracle cabe señalar las siguientes consideraciones:

- Modificar el módulo conexiones.bas, para que el sistema se conecte a la nueva base de datos.

- Muchas tablas de la base de datos ADFIN.mdb deben estar contenidas en el modelo corporativo de la organización, por lo que sería necesario fusionarlas.

- Ya que en access no se pueden usar balanceadores de carga, tanto para las personas como para los documentos se generaron dos tablas, teniendo así:

TB_PERSONAS y TB_EMPRESAS
TB_DOCUMENTOS_COMPRA y TB_DOCUMENTOS_LEGALES

Las cuales pueden ser fusionadas ante la eventual migración.

- Para grabar una fecha en la base de datos el sistema usa la función fmtFecha, que se encuentra en el módulo Utiles.bas, y que es la encargada de llevar la fecha al estándar usado por la base de datos actual, por lo que ante una eventual migración sería necesario cambiarla.

- Los atributos de tipo Si/No existentes en access pueden ser reemplazados por un bit, aunque esto significaría el cambio en los módulos donde se ocupe ese atributo.

- Todas las consultas ocupan la sentencia JOIN, la cual no es soportada por todas las versiones de Oracle.

- La sentencia Is null cambiarla por la usada en Oracle.

- Los signos de concatenación “&” reemplazarlos por los signos “+”.