StartFrame Net Framework

Manual Programación

El objetivo de este manual es definir una serie de criterios que sirvan de base para un arquitecto, analista o programador que desee utilizar el producto. Podrá encontrar dentro de este manual una serie de convenciones adoptadas, guías paso a paso para utilizar el producto, mejores prácticas, entre otros aspectos igualmente importantes.

Se destaca que el lector de este manual debe tener un perfil de programador .Net.

iTSouth & CDT Consultores

Versión liberada el 01/10/2011

StartFrame Net Framework

Manual Programación

Índice General

CONVENCIONES DE DESARROLLO

Si bien no se trata de reglas formales, se sugiere tener presente las siguientes prácticas adoptadas por los mejores desarrolladores del mercado para la escritura de código:

- ☑ **Documentar** todos los bloques de código con títulos y comentarios, a fin de que cualquier programador que no sea el que escribió el código pueda interpretarlo perfectamente. Debe indicarse principalmente el objetivo del proceso, los parámetros pasados (indicando su función, posibles valores), y el valor de retorno de la rutina (si corresponde). Redactar los comentarios en un **lenguaje impersonal**. No sólo se comentarán los encabezados de los procesos, sino que dentro de los mismos deberá comentarse cada bloque de código para que se pueda determinar lo que se está haciendo. Siempre que sea posible, se comenzará el comentario con una identificación que ligue el bloque de código a un punto en particular de la especificación de diseño (si hubiera).
- ☑ **No escribir grandes bloques de código**, ya que es muy difícil realizar el seguimiento. Deberán utilizarse subrutinas para separar dicho código. Es conveniente que las subrutinas no dependan de variables u otros elementos que sean externos a la misma, para que pueda ser reutilizada. Estas subrutinas deben ser métodos (privados o públicos) dentro de la clase que se esté utilizando.
- ☑ Es conveniente **declarar explícitamente todas las variables** al comienzo de cada bloque de código. En dicha declaración, se indicará la finalidad de uso de la variable. Siempre debe tratar de utilizarse el menor alcance (*scope*) posible para todas las variables, evitando al máximo el uso de variables globales.
- ☑ En lo posible, **no superar el ancho máximo de la pantalla** con un comando. Usar el separador de líneas para seguir en el renglón siguiente y adentrado:

```
sSQL = "Select C.IdCliente, C.Nombre, C.Calle, C.Numero," & _
       "C.Localidad, C.IdProvinc " & _
       "From VACliente C " & _
       "Where C.IdCliente = 12300015"
```

- ☑ Todo programa realizado recién se considera terminado cuando está **probado** (ver CONTROL DE CALIDAD) y documentado en el diccionario de datos en línea del sistema.
- ☑ Deben respetarse al 100% las **convenciones de nombres** (ver más adelante) adoptadas para variables, formularios, tablas, campos, procedimientos almacenados, reportes, etc.
- ☑ Nunca dejar una **condición sin su correspondiente alternativa** de acción (IF sin ELSE, CASE sin CASE ELSE, etc.), aunque la posibilidad de que ello ocurra sea muy remota.
- ☑ Cada vez que se **agreguen controles en un formulario** controlar lo siguiente:
  - ❍ Tamaño y alineación
  - ❍ Orden de tabulación
  - ❍ Inclusión del campo en el diccionario (si corresponde)
- ☑ Para todos los **mensajes al usuario** (de error, validación, información, etc.), deberá tenerse en cuenta:
  - ❍ Emplear un lenguaje impersonal.
  - ❍ Capitalizar sólo la primera palabra de la oración.
  - ❍ Colocar signos de expresión, si corresponde, a ambos lados de la oración sin dejar espacios intermedios.
  - ❍ En caso de no haber signos de expresión, colocar punto final de la oración.
  - ❍ Siempre que sea posible, utilizar el archivo de recursos para los mensajes al usuario, creando una etiqueta prefijada (ver *Etiquetas del Archivo de Recursos*).

- La **forma de mostrar el mensaje** dependerá si se trata de un error (*ShowError*) o información general (*MsgBox*).

☑ Para las etiquetas (**labels**) que se relacionen directamente con algún campo, en lugar de llenar la propiedad *text* del label deberá asignarse a la etiqueta el nombre del campo vinculado con el prefijo *"lbl"* en su propiedad *name*.

☑ Siempre que se deje una tarea pendiente en algún bloque de código, deberá utilizarse la **lista de tareas** (*task list*) con un prefijo correspondiente a las iniciales del desarrollador, para realizar una anotación apropiada sobre el tema.

☑ Bajo ningún concepto deberán ingresarse **datos basura** para las pruebas, siempre deberán utilizarse datos reales, o al menos potencialmente reales.

Manejo y Control de Errores

El siguiente es un código estándar que representa una captura de error típica **(capa US desktop)**, donde atrapa el error y lo muestra en una forma estándar.

```
Try             'Trata de abrir la conexión
     objSqlCommand.Connection = objSqlConnection
     objSqlConnection.Open()

Catch objException As Exception         'Muestra el error
     Display.ShowError(objException)

Finally         'Cierra la conexión (si está abierta)
     If objSqlConnection.State = ConnectionState.Open Then
          objSqlConnection.Close()
     End If
End Try
```

El siguiente es un código estándar que representa una captura de error típica **(capas BR - DA)**, donde graba un log (en el *eventlog* de la aplicación, tanto en la base de datos como en el visor de sucesos) y luego propaga el error para que un componente de la capa US lo intercepte e informe al usuario.

```
Try             'Trata de abrir la conexión
     objSqlCommand.Connection = objSqlConnection
     objSqlConnection.Open()

Catch objException As Exception         'Captura el error
     Env.LogError("[PROCESO]", TERMINAL, OPERADOR, COD_ERROR, objException.Message)
     Throw objException

Finally         'Cierra la conexión (si está abierta)
     If objSqlConnection.State = ConnectionState.Open Then
          objSqlConnection.Close()
     End If
End Try
```

Creación de Objetos

Debido a la utilización de la tecnología Remoting[1] para la ejecución remota de objetos en entornos desktop, deberán respetarse ciertas reglas para la creación de dichos objetos.

```
'Dimensiona una variable del tipo Interfaz
Private objOperadores As IOperadores
'Instancia el objeto a través de Remoting
objOperadores = CType(RemotingHelper.GetObject(GetType(IOperadores)), IOperadores)
```

Lo anterior es aplicable únicamente cuando se utilizan objetos remotos (entre capas). Si se utilizan objetos locales (dentro de la misma capa, como es el caso del *code behing* en las aplicaciones Web), podrá usarse la sintaxis común de .Net.

Ahora bien, si se instancian objetos de BR desde US desktop, cabe destacar que el operador con el cual lo instancia es el operador del sistema (*ServerUsr*) ya que la capa servidora, usando *remoting*, es totalmente ciega en lo que respecta al llamador (se limita a responder un llamado sin saber a quién). Luego valida la seguridad con el operador actualmente logueado (*lnk_NombreOperador*: propiedad de *StartFrame.BR.ClaseBase*). Esta inteligencia la realizan operaciones en conjunto entre las clases base de US (desktop o web) y BR.

Por ende, si se requiere instanciar un objeto de BR desde el mismo BR, el programador deberá tomar las precauciones necesarias para identificarse correctamente. Esto podría hacerse, por ejemplo, ejecutando el método *SetOperadorLogueado* pasándole el valor del operador actualmente logueado (*lnk_NombreOperador*). Esto es válido si ambas clases (la que se llama y la llamadora heredan de *ClaseBase* o cualquiera de sus clases heredadas).

```
'Instancia una clase de negocios
Private objOperadores As New Operadores()
ObjOperadores.setOperadorLogueado(lnk_NombreOperador)
```

Teclas de Corte (Shortcuts)

La siguiente tabla enumera las teclas de corte que *StartFrame Net Framework* tiene reservadas para distintos formularios en la plantilla Windows desktop.

TECLA	FORMULARIO / CLASE	ACCIÓN
^C	Menú Edición	Copiar
^E	Menú Edición	Seleccionar todo
^P	Consulta, Abm	Imprimir
^U	Menú Edición	Deshacer
^V	Menú Edición	Pegar
^X	Menú Edición	Cortar
Alt+*nro*	Consulta, Abm	Se desplaza al número de TAB tipeado en *nro*
F1	Consulta, Abm, Listado, Menú	Ayuda
^F1	Menú Ayuda	Ayuda sobre el uso del sistema
Shift+F1	Menú Ayuda	Ayuda *"¿Qué es esto?"*
Alt+F4	Menú Archivo	Salir del módulo
F9	Grabar	Graba altas o modificaciones
^F9	Consulta, Abm	Búsqueda puntual

[1] Ver el **Manual Técnico** para mayores detalles sobre *Remoting*

F10	Abm	Nuevo registro
^F10	Consulta, Abm	Búsqueda global
F11	Abm	Modificar registro
F11	Listador	Impresión preliminar
^F11	Consulta, Abm	Registro anterior
F12	Abm	Eliminar registro
F12	Listador	Impresión a impresora
^F12	Consulta, Abm	Registro siguiente

CONVENCIONES DE NOMBRES

Se definen a continuación las reglas que deberán utilizarse para nombrar cada objeto dentro del sistema y su entorno. El objetivo es lograr una estandarización tal del sistema de modo que cualquier programador, contando con la especificación de diseño, pueda modificar cualquier programa, aunque no haya sido él quién lo confeccionó. Además, esto facilitará en gran manera las tareas de control de calidad. Sin embargo, podrían adoptarse otras convenciones totalmente diferentes a las expresadas a continuación. Lo importante es que exista alguna convención y cuente con el consenso de la mayoría del equipo de desarrollo.

Elementos de la Aplicación

Se entiende por elementos de la aplicación a todos aquellos objetos que se pueden individualizar e identificar con un nombre determinado, como ser: variables, matrices, punteros de objetos instanciados, controles de pantalla, formularios, reportes, etc. A continuación se dictarán las reglas utilizadas para nominar todos ellos.

Variables de Memoria

Si bien se partió de las convenciones propuestas por *Microsoft* para *Visual Studio*, se simplificaron para unos fines más convenientes y prácticos. La siguiente tabla muestra la estructura del nombre de una variable:

POSICIÓN	DESCRIPCIÓN
1er. Elemento	Únicamente aplicable si el alcance de la variable es global pero privada a nivel de clase. En tal caso, deberá llevar un guión bajo (_). Si dicha variable es pública, por convenciones de .Net, no podrá utilizarse dicho prefijo (_). En su defecto, se reemplazará por “m_”.
2do. Elemento	Indica el tipo de dato de la variable: • s: string • b: boolean • d: date, datetime • n: numérica de cualq. tipo (integer, double, etc.) • u: tipos definidos por el usuario • a: array obj: para objetos en forma genérica (ver • Objetos)
3er. Elemento en adelante	Nombre de la variable que no supere los 30 caracteres. Siempre que sea posible, pueden utilizarse los mismos mnemotécnicos que los recomendados para los nombres de los campos (ver Campos). Capitalizar cada palabra en el nombre de la variable.

Un ejemplo de un nombre de variable global privada del tipo string para el código del operador, puede ser *_sCd_Operador*.

Objetos

En los abms, debido a una razón técnica interna del funcionamiento automático de la interfaz de programación, los nombres de los controles de pantalla que están vinculados a campos de tablas, deben tener el mismo nombre que el campo con el cual se vincula.

En el caso del resto de los formularios, los nombres de los controles responden a su contenido o funcionalidad, con la anteposición de un prefijo que indica su tipo. Capitalizar cada palabra componente de un nombre para mayor claridad. La siguiente lista es sólo indicativa.

PREFIJO	TIPO DE OBJETO
	No corresponde prefijo para los objetos vinculados a campos de los abms del sistema
lbl	Label.
txt	Textbox de cualquier tipo.
cbo	Combo box de cualquier tipo y contenido.
lst	List box de cualquier tipo y contenido.
opt	Option button.
chk	Check box.
cmd	Command button de cualquier tipo, forma y contenido.
img	Image, picture y cualquier tipo de control gráfico.
spn	Spinner.
tab	Panel, tab, frame.
grd	Grid de cualquier tipo, forma, contenido y fabricante.
frm	Form.
mnu	Menú.
ds	Dataset
drd	Datareader
dv	Dataview
dt	Datatable
dr	Datarow
dc	Datacolumn
ex	Exeption
obj	Control de cualquier tipo que no entre en ninguna de las clasificaciones antes descriptas en esta tabla.

Formularios y páginas

Hay que hacer una salvedad al hablar de nombres de formularios en entornos Windows desktop o páginas en entornos Web. Primero hay que distinguir perfectamente la diferencia entre el nombre físico con el cual el formulario o página se guarda en disco, el cual coincide con el nombre con el cual Visual Studio lo reconoce, y por otro lado está el nombre lógico compuesto por un nombre de clase y el espacio de nombre (*namespace*) a la cual esa clase pertenece. Por otro lado, los espacios de nombre para las capas de usuario y negocios tienen una estructura muy similar.

Para los nombres físicos se utilizarán las siguientes reglas:

POSICIÓN	DESCRIPCIÓN
1er. Dígito	Indica el subsistema al que pertenece el formulario, según se registró en la tabla *wap_subsistemas* (por ej. *"W"* para utilitarios).
2do. Dígito	Indica el tipo de programa: • A: abm • C: consulta • L: listado • P: proceso

	• R: rutina
3er. Dígito en adelante	Nombre del formulario

En cuanto a los espacios de nombre para la capa de usuario, se utilizará el siguiente árbol:

NAMESPACE	DESCRIPCIÓN
1er. Elemento	Copyright del producto o la empresa. Por ej.: *StartFrame*
[2er. Elemento]	Opcionalmente, puede indicarse el nombre en clave del proyecto. Por ej.: *ERP*
3er. Elemento	Capa a la cual pertenece el formulario. Por ej.: *US o Web*
4to. Elemento	Módulo al cual pertenece en forma prioritaria. Ejemplo: *Utilitarios*
5to. Elemento	Tipo de programa, a saber: • *Abms* • *Consultas* • *Listados* • *Procesos* • *Rutinas*

Por último, la clase recibe el mismo nombre que el archivo físico del formulario, pero sin los mnemotécnicos. Ejemplo de un nombre completo: *StartFrame.US.Utilitarios.Abms.Operadores* o *StartFrame.ERP.US.Ventas.Abms.Clientes*

En cuanto a los espacios de nombre para la capa de negocios, se utilizará el siguiente árbol:

NAMESPACE	DESCRIPCIÓN
1er. Elemento	Copyright del producto o la empresa. Por ej.: *StartFrame*
[2er. Elemento]	Opcionalmente, puede indicarse el nombre en clave del proyecto. Por ej.: *StartFrame*
3er. Elemento	Capa a la cual pertenece el formulario. Por ej.: *BR o BR.Web*
4to. Elemento	Módulo al cual pertenece en forma prioritaria. Ejemplo: *Utilitarios*

Por último, la clase recibe el mismo nombre que el archivo físico del formulario, pero sin los mnemotécnicos, tal cual el nombre de la clase de la capa de usuario. Ejemplo de un nombre completo: *StartFrame.BR.Utilitarios.Operadores* o *StartFrame.ERP.BR.Ventas.Clientes*

Reportes

Los nombres de los reportes (en entornos Windows desktop) utilizarán las siguientes reglas:

POSICIÓN	DESCRIPCIÓN
1er. Elemento	Nombre del formulario listador con el cual se vincula el reporte.
2to. Elemento	Guión bajo (_) como separador de elementos.
3er. Elemento	Nombre que identifica al reporte en sí.

Un ejemplo de un nombre de reporte válido puede ser *wloperadores_porgrupo*.

Etiquetas del Archivo de Recursos

El archivo de recursos se utilizará para facilitar las traducciones del sistema para otras culturas. Por ello, es indispensable que se utilice para almacenar cualquier tipo de información de índole cultural: gráficos, mensajes de cualquier índole, textos en pantalla, etc. La siguiente tabla muestra las etiquetas utilizadas en dicho archivo.

POSICIÓN	CONTENIDO
1er. elemento	Tipo de etiqueta: • Sin prefijo si es una imagen genérica • I16: Icono de 16x16 pixeles • I32: Icono de 32x32 pixeles • LOGO: Logotipo de alguna clase • PANEL: Panel de controles • TIP: Tooltip • LBL: Label • MSG: Mensaje • CMD: Texto de un botón de comando • GRD: Encabezado de columna de grilla
2do. elemento	Guión bajo (_) separando los campos
3er. elemento	Opcionalmente, puede indicarse el tipo de programa que utiliza el recurso: • ABM: Abms • CON: Consultas • LIS: Listados • PRO: Procesos • RUT: Rutinas • MENU: Menú
4to. Elemento	Guión bajo (_) separando los campos
5to. Elemento	Identificación del recurso, en mayúsculas y separando cada palabra por guiones bajos. Siempre que corresponda, deberá coincidir con el nombre del campo asociado al control. Por ejemplo: *LBL_ABM_cd_operador* o *CMD_PRO_configurar_impresora* o *MSG_LIS_rango_obligatorio.*

Existen dos archivos de recursos:

- **Lnkfrmwrk.es-AR.resources:** Archivo de recursos del Framework. No debe tocarse, ya que se sobreescribe con cada actualización de dicho marco de trabajo. También puede existir un archivo idéntico, pero para la cultura neutral para español (.es.resources).
- **App.es-AR.resources:** Archivo de recursos de cada solución desarrollada con *StartFrame*. Este es el archivo en el que deben añadirse todas las etiquetas deseadas.

Ambos archivos de recursos compilan con el proyecto *Common* de cada aplicación.

Elementos del Motor de BD

Tanto para los nombres de las tablas como para los campos, se utilizan caracteres en minúsculas con un guión bajo (_) separando las palabras. No se optó por un esquema de nombres capitalizados para obtener una mayor compatibilidad con los diferentes motores con respecto al *case sensitive*.

Tablas

Las tablas respetan las siguientes reglas de nombres:

POSICIÓN	DESCRIPCIÓN
1er. Dígito	Indica el subsistema al que pertenece la tabla, según se registró en la tabla *wap_subsistemas* (por ej. *"w"* para utilitarios).
2do. Dígito	Indica el tipo de tabla: • a: abm / maestro • c: cabecera • l: líneas o detalle
3er. Dígito	Indica la forma de ingresar datos en la tabla: • d: directo (por el usuario) • p: proceso (por el sistema, como por ej. un log) • v: vista
4to. Dígito	Guión bajo (_) separando los mnemotécnicos del nombre de la tabla
5to. Dígito en adelante	Nombre de la tabla en plural (en minúsculas y separando las palabras con guiones bajos)

Un ejemplo de un nombre de tabla válido puede ser *wad_operadores*.

Relaciones

En cuanto a las relaciones, estas llevan el nombre encabezado por el texto *"FK_"* seguido del nombre de la tabla hija (extremo *muchos* de una relación), un guión bajo, y el nombre de la tabla padre (extremo *uno* de una relación).

Un ejemplo de un nombre de relación válido puede ser *FK_wap_tracking_wad_operadores*.

Índices

Los índices llevan nombres diferentes según su tipo:

TIPO DE ÍNDICE	NOMBRE
Clave primaria	Encabeza el nombre el texto *"PK_"* seguido por el nombre de la tabla. Ejemplo: *PK_wad_operadores*.
Clave externa	Encabeza el nombre el texto *"FK_"* seguido por el nombre del primer campo indexado (con algún diferenciador si hiciera falta). Ejemplo: *FK_cd_operador*.
Clave única	Encabeza el nombre el texto *"UN_"* seguido por el nombre del primer campo indexado (con algún diferenciador si hiciera falta). Ejemplo: *UN_nm_operador*.

Campos

Los campos respetan las siguientes reglas de nombres:

POSICIÓN	DESCRIPCIÓN
1ro. y 2do. Dígitos	Indica el tipo de contenido del campo, cuyo nombre debe leerse traduciendo este mnemotécnico (ej.: *tp_operacion* se leería *"tipo de operación"*). Los mnemotécnicos más comunes son los siguientes. Sin embargo, cualquier mnemotécnico deberá registrarse en el archivo de recursos sin ningún tipo de prefijo (sólo el mnemotécnico). Cada mnemotécnico debe contener tres tipos de expresiones a retornar (separadas por ;): larga, media y corta. Por ej., para el mnemo *ca*, deberá grabarse: *Cantidad de;Cantidad,Cant.* • ca: cantidad • cd: código • de: descripción • fe: fecha • hr: hora • im: importe • ls: lista • nm: nombre • nu: número • po: porcentaje • st: status (flag) • tp: tipo de • va: valor
3er. Dígito	Guión bajo (_) separando los mnemotécnicos del nombre del campo
4to. Dígito en adelante	Nombre del campo en singular (en minúsculas y separando las palabras con guiones bajos)

Procedimientos Almacenados

Los procedimientos almacenados respetan las siguientes reglas de nombres:

POSICIÓN	DESCRIPCIÓN
1er. Dígito	Indica el subsistema al que pertenece el procedimiento, según se registró en la tabla *wap_subsistemas* (por ej. *"w"* para utilitarios).
2do. Dígito	Indica el tipo de programa asociado al mismo: • a: abm • c: consulta • l: listado • p: proceso • r: rutina
3er. Dígito en adelante	Nombre del procedimiento (en minúsculas y separando las palabras con guiones bajos), generalmente y siempre que sea posible, deberá utilizarse el mismo nombre que el programa vinculado al procedimiento. En caso de haber más de un procedimiento relacionado al mismo programa, deberá utilizarse un nuevo guión bajo y a continuación deberá ingresarse alguna palabra que actúe como diferenciador e identificador de la naturaleza del procedimiento.

Un ejemplo de un nombre de procedimiento almacenado válido puede ser *waoperadores_alta* o *wp_obtener_usuarios.*

CONFECCIONES ESTÁNDAR

Una Nueva Solución

StartFrame está diseñado como una herramienta para desarrollar nuevas soluciones. Dado que el producto se instala como una extensión de Visual Studio, para crear una nueva solución tan sólo deberá seleccionar la opción *"Nuevo Proyecto"* del menú principal de Visual Studio y seleccionar el *template* deseado[2].

Existen diferentes tipos de plantillas que pueden utilizarse para crear soluciones con diferentes arquitecturas (desktop, web asp, servicio windows, servicio web, etc.).

Cambios del Aspecto Visual

Es posible cambiar el aspecto íntegro de cualquier solución basada en *StartFrame* siempre y cuando se respeten los tamaños de las imágenes. Caso contrario, deberán además ajustar manualmente los tamaños de los objetos de pantalla que las contienen.

En el caso de las plantillas Windows, todas las imágenes prediseñadas se encuentran en el archivo de recursos de la aplicación, ya sea en *lnkfwrklnk.es-AR.resources* o bien en *App.es-AR.resources*, el cual se puede editar directamente desde Visual Studio.

En el caso de las plantillas de solución Web, el aspecto visual proviene en gran medida del tema seleccionado, el cual puede modificarse o bien agregarse uno nuevo. Cada tema está compuesto por imágenes plantillas *CSS* y *Skins*.

Las imágenes que debería cambiar las soluciones basadas en la plantilla *FwrkSolution* son:

ETIQUETA	TAMAÑO APROXIMADO (PIXELS)		DESCRIPCIÓN
	ANCHO	ALTO	
APP_LOGO	150	92	Imagen del *Splash* y *About* del sistema. Se recomienda utilizar fondo blanco.
BACKGROUND_LOGO	95	58	Logo mostrado sobre la imagen de fondo de cada módulo. Suele utilizarse la misma imagen de App_logo pero con diferente tamaño.
BACKGROUND_IMAGEN	700	500	Imagen de fondo de cada módulo. Se recomienda utilizar un fondo blanco en una imagen muy tenue (marca de agua) y con bordes difusos.
MENU_LOGO	220	57	Imagen del selector de módulos Windows Desktop (con fondo metálico verde).
APP_SPLASH	455	350	Imagen de fondo de la pantalla de *Spash* y el *Acerca de*.
MENU_*	48	48	Íconos del selector de módulos. Corresponde a las imágenes referenciadas en el archivo MenuModulos.xml.

[2] Referirse al **Tutorial** que demuestra los pasos necesarios para realizar esta tarea de creación de una nueva solución o bien al **Manual Técnico** para detalles específicos.

I16_*	16	16	Íconos de cada opción de cada menú de módulo. Corresponde a las imágenes referenciadas en los archivos de menúes.

Otro aspecto que debería modificar en cada solución creada, son los valores de los siguientes parámetros (tabla *wap_parametros*):

VA_PARAMETRO	DESCRIPCIÓN
EMPRESA	Nombre de la empresa a nombre de la cual se emite la licencia. Este parámetro no puede editarse directamente desde el abm. Debe grabarse encriptado.
APPNAME	Nombre de la aplicación, el cual figura en el Splash y el About.
VERSION	Nombre de la versión de la aplicación, el cual figura en el Splash y el About.

Menúes

Esta parte del manual explica cómo agregar o editar un ítem de menú, e incluso cómo agregar un nuevo módulo con su propio menú. La operatoria varía de acuerdo a la arquitectura utilizada.

Interfaz Windows Desktop

A continuación se especifica el procedimiento para cualquiera de las arquitecturas Windows Desktop.

Agregado de un nuevo módulo

StartFrame, en su plantilla de solución para Windows Desktop, ofrece una barra tipo *Outlook* que permite ingresar los submódulos del sistema y agrupar los mismos en áreas o módulos más generales. Para ello, tan sólo deberá agregar las características del nuevo módulo en el archivo *MenuModulos.xml*, respetando los tags del xml y sus atributos[3]:

- **Name**: Nombre del módulo o submódulo, el cual figurará en el control de *OutLook* y en el título del menú una vez ejecutado.
- **Image**: Etiqueta del *TAG* que contiene la imagen dentro del archivo de recursos *App.resources*. Deberán tratarse de íconos de 48x48 en 256 colores.
- **Descrip**: Descripción de ayuda que aparecerá en la ventana de display.
- **MenuXml**: Nombre del archivo XML que contiene las opciones del submódulo.

```
<?xml version="1.0" encoding="UTF-8" ?>
<MainMenu>
   <Menu Name="Ventas">
      <Menu Name="Facturacion" Image="MENU_MOD1_ITEM1" Descrip="Módulo de facturación"
            MenuXml="MenuFact.xml"/>
      <Menu Name="Cobranzas" Image="MENU_MOD1_ITEM2" Descrip="Módulo de control de
            cuentas corrientes" MenuXml="XmlCobranzas.xml"/>
   </Menu>
   <Menu Name="Sueldos">
      <Menu Name="Liquidador" Image="MENU_MOD2_ITEM1" Descrip="Módulo de liquidación de
            sueldos" MenuXml="XmlSueldos.xml"/>
   </Menu>
</MainMenu>
```

[3] Puede utilizar los *code snippets* provistos por *StartFrame* para agregar una nueva entrada al menú.

Edición del menú

Para crear un nuevo menú, deberá buscar los menúes ya definidos y referenciados en el archivo *MenuModulos.xml*. La opción más sencilla es seleccionar alguno de los XML ya hechos y hacer un *copy & paste* para crear el nuevo archivo XML de menú.

Una vez obtenido el XML del menú, deberá ingresar la siguiente información por cada ítem de menú[4]:

- **Name**: nombre de la opción de menú. Puede utilizar el prefijo *&* para resaltar una tecla. También puede utilizar un guión medio simple para crear una línea divisoria.
- **Shortcut**: tecla de corte para la opción de menú. El formato para ingresarla es (por ejemplo): *CtrlA, AltC, ShiftD*.
- **Icon**: corresponde a la etiqueta dentro del archivo de recursos *App.resources* del ícono asociado a la opción de menú. Deberán tratarse de íconos de 16x16 en 256 colores.
- **Checked**: indica si la opción del menú es del tipo *check/uncheck*. Los valores válidos son *"True"* y *"False"*.
- **Enabled**: indica si debe mostrarse habilitada o no la opción de menú. Los valores válidos son *"True"* y *"False"*.
- **Mdichild**: indica si el formulario a llamar estará contenido dentro del MDI (como MDIchild) o no. Este parámetro es opcional, asumiendo por defecto el valor *"True"* (dentro del MDI). Los valores válidos son *"True"* y *"False"*.
- **Nforms**: indica si el formulario a llamar soporta múltiples instancias o no. En caso de que no las soporte, al llamar al formulario por segunda vez, se pasará a poner el foco en la instancia activa del mismo. Este parámetro es opcional, asumiendo por defecto el valor *"True"* (múltiples instancias permitidas). Los valores válidos son *"True"* y *"False"*. Este parámetro es sólo aplicable cuando *MdiChild* es igual a *"True"*.
- **Id**: nombre de la clase y el assembly que la contiene, tal cual se indica en el diccionario de datos de programas, en el campo *nm_clase*. Esta celda puede llegar a contener una determinada etiqueta, la cual podrá ser interceptada por el evento *OnItemSelected* para ejecutar algún programa externo o directamente incluir código para dicha opción.

```
Public Shared Function OnItemSelected(ByVal item As String, _
   ByVal tipoLlamado As TipoLlamado) As Boolean

   Select Case item.ToUpper.Substring(0, 5)
      Case "ID001"
         'Ejecuta un proceso externo
         Dim curPath As String = CurDir()
         Dim pathnameBcos As String
         Dim ProcID As Integer

         Try
            Pathname = Common.Env.GetConfigValue("pathProcZ") _
               & Common.Env.GetConfigValue("exeProcZ")
            If Not IO.File.Exists(pathnameBcos) Then
               MsgBox("No se encuentra instalado el proceso en esta PC.", _
                  MsgBoxStyle.OkOnly)
            Else
               If MsgBox("¿Confirma la ejecución?", MsgBoxStyle.YesNo, _
                  "Confirmación") = MsgBoxResult.Yes Then
                  ProcID = Shell(pathnameBcos, AppWinStyle.NormalFocus)
                  ChDir(curPath)
               End If
            End If
```

[4] Puede utilizar los *code snippets* provistos por *StartFrame* para agregar una nueva entrada al menú

```
        Catch ex As Exception
           MsgBox("Se produjo un error al ejecutar el proceso: " & _
              ex.Message, MsgBoxStyle.OkOnly)
        End Try

        'Impide que se ejecute la opción desde el menú
        Return True

   Case Else
      'Llamado estándar desde el menú
      Return False
   End Select
End Function
```

En cuanto al TAG principal (MainMenu), se pueden ingresar los siguientes atributos:

- **Name**: Nombre del módulo (título del formulario MDI)
- **Collapse**: Es un atributo opcional que permite ingresar *true* o *false* (opción por defecto). Para lograr que el *splitter* que oculta o muestra el menú de árbol aparezca colapsado, deberá ingresar *true* en este atributo.

```
<?xml version="1.0" encoding="UTF-8" ?>
<MainMenu Name="Módulo Testing" Collapse="false">
   <Menu Name="&amp;Tablas" Shortcut="" Icon="" Checked="False" Enabled="True" Id="">
      <Menu Name="&amp;Tablas de Uso Compartido" Shortcut="" Icon="I16_TABLASVARIASM"
            Checked="False" Enabled="True" Id="">
         <Menu Name="&amp;Paises y Provincias" Shortcut="" Icon="I16_PAISES"
               Checked="False" Enabled="True" Mdichild="True" Nforms="False"
               Id=" StartFrame.US.Utilitarios.Abms.Paises,US.utilitarios" />
         <Menu Name="&amp;Localidades y Calles" Shortcut="" Icon="I16_CALLES"
               Checked="False" Enabled="True" Mdichild="False"
               Id=" StartFrame.US.Utilitarios.Abms.Localidades,US.utilitarios" />
      </Menu>
   </Menu>
   <Menu Name="&amp;Procesos" Shortcut="" Icon="" Checked="False" Enabled="True" Id="">
      <Menu Name="&amp;Cambiar Parametros de Conexión a la Base de Datos" Shortcut=""
            Icon="I16_DB" Checked="False" Enabled="True"
            Id=" StartFrame.US.Utilitarios.Abms.Conexiondb,US.utilitarios" />
   </Menu>
</MainMenu>
```

Interfaz Web

A continuación se especifica el procedimiento para la arquitectura Web Application.

En dicha plantilla no existe el concepto de módulos como tal, sino que deberá crear una página principal con un selector de módulos que redireccione a los correspondiente subsitios o bien manejar este concepto dentro del mismo menú de la aplicación si no son módulos demasiado grandes.

Edición del menú

El menú contenido en esta plantilla está formado por un control del tipo *treeview* que se alimenta de un archivo del tipo *sitemap*. Para actualizar el contenido de los ítems de menú (submenúes y opciones finales) simplemente deberá actualizar el archivo *Web.sitemap*, el cual se encuentra en el directorio principal del sitio web:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="" title="Menú del Sitio" description="" >
    <siteMapNode url="" title="Seguridad"  description="Módulo de Seguridad del Sitio">
      <siteMapNode url="FwrkPages/Login.aspx" title="Login" description="Login/out" />
      <siteMapNode url="FwrkPages/Login_NewUsr.aspx" title="Nuevo Usuario"
         description="Registrar un nuevo usuario" />
      <siteMapNode url="" title="Auditoría"  description="Informes de auditoría">
        <siteMapNode url="ManagerPages/Lst_Tracking.aspx" title="Log de Accesos"
         description="Informes de auditoría del sistema" />
      </siteMapNode>
    </siteMapNode>
    <siteMapNode url="" title="Herramientas"  description="Módulo de Utilitarios">
      <siteMapNode url="ManagerPages/Abm_Reportes.aspx" title="Reportes"
         description="Generador automático de reportes" />
      <siteMapNode url="ManagerPages/Abm_Filtro.aspx" title="Filtros"
         description="Filtros a utilizar en los reportes generados" />
      <siteMapNode url="FwrkPages/Temas.aspx" title="Temas" description="Selec.Tema" />
    </siteMapNode>
  </siteMapNode>
</siteMap>
```

Abms

La siguiente documentación enumera sintéticamente los pasos a seguir para la confección de un ABM estándar. El objetivo de la misma es lograr una estandarización total en la programación de todo el sistema, de forma tal que sea imperceptible el paso de varios programadores diferentes, facilitar el mantenimiento y minimizar la posibilidad de errores.

Es conveniente comenzar por las capas inferiores (las que están más cerca de los datos) y luego ir subiendo de nivel. Por ello, la confección de un abm se divide en dos partes, las cuales se detallan a continuación. Se aclara que la siguiente es sólo una guía de los pasos a realizar, pero no es la única manera de realizarlos ni necesariamente el orden requerido. Cuando el programador tome experiencia con *StartFrame*, él mismo podrá optar por la manera más cómoda en que le resulte realizar estas tareas.

Las partes grisadas de los ejemplos de código están a modo de referencia para indicar la ubicación del código relacionado al ítem de referencia. Las partes resaltadas indican que deberá reemplazarse el contenido por el apropiado en cada caso.

En el caso de las validaciones, estas partes están sólo a modo de ejemplo, ya que no se puede especificar una validación genérica.

Por razones de espacio y claridad, algunas líneas de código debieron ser tipeadas en varios renglones.

Casi cualquier porción de código utilizada en esta sección puede ser creada mediante el uso de los *code snippets* incluidos en *StartFrame*.

Parte I: Clase BR

Sin importar el tipo de plantilla arquitectónica seleccionada, los componentes de reglas del negocio son genéricos y reutilizables desde todos los tipos de soluciones (Windows, Web, Servicios, etc.). Deben realizarse las siguientes tareas[5] en el componente de la capa de negocios[6]:

1 **Agregar un nuevo componente** (clase) en la capa de reglas de negocios que herede de *StartFrame.BR.Abm*. Dentro del mismo componente se pueden crear tantas clases como sea necesario (ver punto 4).

2 Realizar la **declaración de la clase** y modificar los valores resaltados del código. Asegurarse de no dejar código innecesario al efectuar este paso. En caso de tratarse de un componente para utilizar en forma local del lado del servidor, no es necesario que implemente ninguna interfaz.

```
Imports System.Data.Oledb
Imports Common.Env
Imports StartFrame.DA.Sql

Public Class UnidadesMonetarias              'Nombre de la clase
   Inherits Abm                              'Herencia
   Implements IUnidadesMonetarias            'Interfaz

   Sub New()
      'Modificar nombre de programa y tabla principal relacionada a la clase
      MyBase.New(Operador, Password, "waunidmon", "wad_unidades_monetarias")

      'Habilita el tracking para esta clase
      Me._Tracking = True

   End Sub
End Class
```

3 *[sólo para componentes accesibles desde el front-end Windows desktop]* Agregar una **entrada con el nombre de la clase en el archivo XML** de configuración.

```
<wellknown mode="Singleton" type="StartFrame.BR.Utilitarios.UnidadesMonetarias,BR"
objectUri="UnidadesMonetarias" />
```

4 Si son necesarias **nuevas clases con reglas de negocios específicas**, crear dentro del mismo componente tantas clases como entidades relacionadas existan para efectuar las validaciones correspondientes. Además, si se utilizan métodos que no son propios de la clase heredada, se debe referenciar la interfaz si desea que dichos métodos sean visibles desde el front-end Windows desktop a través de *Remoting*.

```
Public Class UnidadesMonetarias              'Nombre de la clase
   Inherits Abm                              'Herencia
   Implements IUnidadesMonetarias            'Referencia de interfaz
   Dim _cotizaciones As New Cotizaciones()   'Clase asociada (subclase)

   Sub New()
      'Modificar nombre de programa y tabla principal relacionada a la clase
      MyBase.New(Operador, Password, "waunidmon", "wad_unidades_monetarias")
   End Sub
End Class

Public Class Cotizaciones                    'Nombre de la clase
```

[5] Las que figuran con la viñeta en **video inverso** son obligatorias
[6] Ver **tutorial** para una ilustración paso a paso de las tareas a realizar

```
    Inherits Abm                                'Herencia

    Sub New()
        'Modificar nombre de programa y tabla principal relacionada a la clase
        MyBase.New(Operador, Password, "waunidmon", "wld_cotizaciones")
    End Sub
End Class
```

5 Para **crear una grilla (o tabla secundaria) administrada automáticamente por la clase ABM**, insertar el siguiente código y modificar los valores resaltados. Tener en cuenta que al existir una tabla de detalle, la grabación de datos debe efectuarse con ciertas reglas a fin de automatizarla (ver punto 8).

```
Public Class UnidadesMonetarias             'Nombre de la clase
    Inherits Abm                            'Herencia
    Implements IUnidadesMonetarias          'Referencia de interfáz

    Dim _cotizaciones As New Cotizaciones()         'Crea el objeto detalle

    Sub New()
        'Modificar nombre de programa y tabla principal relacionada a la clase
        MyBase.New(Operador, Password, "waunidmon", "wad_unidades_monetarias")

        'Grilla (agrega tablas de detalle al dataset)
        Dim arrayTablasDetalle As New ArrayList()
        arrayTablasDetalle.Add("wld_cotizaciones")
        MyBase.setTablasDetalle(arrayTablasDetalle)
    End Sub
End Class

Public Class Cotizaciones                   'Nombre de la clase
    Inherits Abm                            'Herencia

    Sub New()
        'Modificar nombre de programa y tabla principal relacionada a la clase
        MyBase.New(Operador, Password, "waunidmon", "wld_cotizaciones")
    End Sub
End Class
```

6 Para efectuar **validaciones manuales** sobre todos los registros, utilizar el evento *_validando*. En este evento, el *dataset* se encuentra cargado con todos los registros de la tabla relacionada. Usar el siguiente esquema y modificar los valores resaltados. En el caso de estar utilizando un frontend Web, es recomendable realizar estas validaciones del lado de la página con controles *validators* para lograr un mejor resultado.

```
Private Sub UnidadesMonetarias_Validando(ByRef cancela As Boolean, _
    ByRef paramDataTable As System.Data.DataTable) Handles MyBase.Validando

    Dim row As DataRow

    'Recorre todos los registros de la tabla
    For Each row In paramDataTable.Rows
        With row
            If .RowState <> DataRowState.Deleted Then

                'Valida duplicados por el dato nombre
                If Not IsDBNull(row("nm_unidad_monetaria")) Then
                    If CType(StartFrame.DA.Sql.Search(ConnectionString, "COUNT(*)", _
                    "wad_unidades_monetarias", "nm_unidad_monetaria = '" & _
                    CType(row("nm_unidad_monetaria"), String) & "'"), Integer) > 0 Then
                        .SetColumnError("nm_unidad_monetaria", _
                        "El nombre de unidad monetaria ingresada ya existe.")
                    End If
                End If
            End If
        End With
    Next
End Sub
```

7 Para efectuar **validaciones manuales más precisas** sobre cada registro en particular, existen eventos determinados: *_insertando*, *_modificando*, *_eliminando*, *_insertado*, *_modificado*, *_eliminado*. En estos eventos en lugar de un *dataset* contiendo todos los registros a modificar, se trabaja con el *datarow* procesado. Utilizar el método que corresponda para cada validación. En el caso de estar utilizando un frontend Web, es recomendable realizar estas validaciones del lado de la página con controles *validators* para lograr un mejor resultado.

```
Private Sub UnidadesMonetarias_Insertando(ByRef cancela As Boolean, ByRef row As
System.Data.DataRow) Handles MyBase.Insertando, MyBase.Modificando

   'Valida el primer dígito
   If row("nm_unidad_monetaria").ToString.Length > 0 Then
      Select Case row("nm_unidad_monetaria ").ToString.Substring(0, 1)
         Case "A", "B", "C"        'Todo Ok
         Case Else                 'Error
            row.SetColumnError("nm unidad monetaria", _
              "El primer digito debe ser ""A, B, C""." )
      End Select
   End If
End Sub
```

8 La **grabación de las tablas de detalle** del ABM debe hacerse en forma transaccionada llamando a cada objeto de negocios asociado a cada tabla/grilla a procesar (ver punto 5). Para esto es necesario hacer un overload del método *ActualizarDatos()*. En dicho método, deberá ponerse código con el siguiente esquema y modificar los valores resaltados.

```
Public Overloads Overrides Function ActualizarDatos(ByRef paramDataSet As _
System.Data.DataSet) As Boolean

   Dim tran As OleDbTransaction = Nothing

   Try
      InicializarConeccion()
      lnk_cn.Open()
      tran = _cn.BeginTransaction

      'Elimina las columnas no mapeadas
      Me.RemoveUnmappedCols(paramDataSet.Tables("wld_cotizaciones"))

      'Setea el operador actual de las subclases
      _cotizaciones.SetOperadorLogueado(_NombreOperador)

      'Inserta o modifica registros...
      If paramDataSet.Tables(_tablaPrincipal).GetChanges(DataRowState.Deleted) Is _
         Nothing Then

         'Actualiza tabla principal
         MyBase.ActualizarDatos(paramDataSet, lnk_cn, tran)
         'Actualiza tabla detalle
         _cotizaciones.ActualizarDatos(paramDataSet, lnk_cn, tran)

      Else  'Elimina registros...

         'Actualiza tabla detalle
         _cotizaciones.ActualizarDatos(paramDataSet, lnk_cn, tran)
         'Actualiza tabla principal
         MyBase.ActualizarDatos(paramDataSet, lnk_cn, tran)
      End If

      'Confirma o anula la transacción
      If paramDataSet.HasErrors Then
         tran.Rollback()
      Else
         tran.Commit()
      End If
```

```
    Catch e As Exception
        tran.Rollback()
        Throw New Exception(e.Message, e)

    Finally
        'Cierra la transacción y la conexión
        tran = Nothing
        lnk_cn.Close()
        lnk_cn = Nothing
    End Try

End Function
```

9 Para **retornar los registros de la grilla a la capa US**, deberá crearse un método que, en base a una condición (la cual lo vincula con la tabla de cabecera), retorne los mencionados registros en forma de *dataset*. A continuación se muestra un ejemplo sencillo (solo retornando las columnas de la tabla *bindeada*).

```
'Retorna los registros para la grilla de cotizaciones
Public Function getCotizaciones(ByVal cotizacionID As String)
                    As DataSet Implements IUnidadesMonetarias.getCotizaciones

        Return _cotizaciones.Buscar("cd_unidad_monetaria = '" & cotizacionID & "'")

End Function
```

Parte II: Clase US

La capa de presentación o interfaz con el usuario dependerá de la arquitectura de solución elegida (Windows, Web, etc.).

Interfaz Windows Desktop

A continuación se especifica el procedimiento para cualquiera de las arquitecturas Windows Desktop. Deben realizarse las siguientes tareas[7] en el formulario de la capa de usuario:

1 **Agregar la interfaz** para la clase en el proyecto IBR que corresponda, con el siguiente esquema, modificando los valores resaltados. Este *assembly* reside tanto en el server como en el cliente.

```
Public Interface IUnidadesMonetarias
   Inherits IAbm

   'Agregar las declaraciones de funciones que correspondan
   Function getCotizaciones(ByVal cotizacionID As String) As DataSet

End Interface
```

2 Agregar un **nuevo formulario** en la capa de usuario que herede de *StartFrame.US.Abm* o *StartFrame.US.AbmGrilla* y modificar su constructor a efectos de que el formulario pueda ser llamado en modo automático.

```
#Region " Windows Form Designer generated code "
   Public Sub New()
      Me.New(Modos.Normal)
   End Sub

   Public Sub New(ByVal modo As Modos)
      Me.New(modo, "")
   End Sub

   Public Sub New(ByVal modo As Modos, ByVal condicion As String)
      MyBase.New(modo, condicion)
      'This call is required by the Windows Form Designer.
      InitializeComponent()
      'Add any initialization after the InitializeComponent() call
   End Sub
```

3 Realizar la **declaración de la clase** con una variable privada global.

```
Imports StartFrame.BR.Interfaces  'Procedencia de las referencias utilizadas

Namespace Abms                                  'Namespace de la clase
   Public Class UnidadesMonetarias              'Nombre de la clase
      Inherits StartFrame.US.Abm                'Herencia
      Dim _unidadesMonetarias As IUnidadesMonetarias
      ------------------------------------
      Windows form designer generated code
      ------------------------------------
   End Class
End Namespace
```

[7] Las tareas que figuran con la viñeta en **fondo negro** son obligatorias.

4 **Vincular el formulario de US a la clase de BR** modificando el *Load* del formulario agregando el siguiente código y modificando los valores resaltados.

```
Private Sub UnidadesMonetarias_Load(ByVal sender As Object, _
   ByVal e As System.EventArgs) Handles MyBase.Load

   Try
    _unidadesMonetarias=CType(RemotingHelper.getObject(GetType(IUnidadesMonetarias)),
IUnidadesMonetarias)
      Me.ObjetoReglasNegocio = _unidadesMonetarias

   Catch ex As Exception
      StartFrame.Us.Display.ShowError(ex.message)
   End Try
End Sub
```

5 Agregar una **entrada con el nombre de objeto en el XML** *US.ini.exe.config*.

```
<wellknown type=" StartFrame.BR.Interfaces.IUnidadesMonetarias,IBR"
url="tcp://localhost:1234/UnidadesMonetarias" />
```

6 Si fuera necesario, **agregar páginas** y cambiarles el *Text* de cada una de ellas. Si existiera una sola página, el *text* va debe decir "Datos Generales". Si existieran múltiples páginas, la primera de ellas deberá tener el *Text* igual a *"1. Datos Generales"*, las siguientes deberán estar igualmente numeradas (con el número resaltado para utilizar como acceso rápido) y con un título apropiado a los controles que en ella se ubiquen.

7 Modificar el *Text* del **título** del formulario en el evento *Load*.

8 Pegar los **controles de pantalla** utilizando las clases de *LnkControls* y/o *US*. En cada control deberá cambiarse el *Text* y el *Name* del control (igual al nombre del campo vinculado) y el *Text* y *Name* de la etiqueta relacionada (igual al nombre del campo anteponiendo el prefijo *"lbl"*).

9 **Alinear y ordenar** los controles según la importancia. Se sugiere utilizar grilla de 8 x8.

10 Verificar el **llamado al programa en el menú** dentro del archivo XML del módulo que corresponda. Además, agregar la correspondiente entrada en el **diccionario de datos de programas y tablas** si es que estas no existen.

11 Si fuera necesario **restringir el acceso a un campo** se deberá escribir código en el evento *When* del control con el siguiente esquema, modificando los valores resaltados.

```
Private Sub [va_obs_When](ByVal sender As Object, ByVal e As _
System.ComponentModel.CancelEventArgs) Handles va_obs.When
   e.Cancel = True
End Sub
```

12 Para **reconfigurar propiedades u otros seteos iniciales** (como creación de listas), utilizar el evento *Clase_Load*. Las propiedades que pueden afectarse en este evento son:

a. **TablaBusquedas**: por defecto, para las búsquedas se utilizará la tabla principal, pero puede optarse por utilizar alguna vista que contenga más parámetros de localización. En este caso, es necesario que la tabla de búsquedas contenga todos los campos que contiene la tabla principal. Caso contrario, habrá que hacer un *override* del método *OnLoadDataset*. Para que todo esto sea aplicable, deberá realizarse un *override* en la clase de negocios del método *Buscar* para que tome en cuenta que debe retornar los datos requeridos por la *TablaBusquedas*. Esto último no sería necesario si se utilizan vistas actualizables con al menos todos los campos de la tabla principal.

```
Protected Overrides Sub OnLoadDataSet(ByVal ds As DataSet, _
                    ByVal condicion As String, ByVal clearDS As Boolean)
   'Limpia el dataset
   If clearDS Then
      _dataSet.Tables(_tablaPrincipal).Clear()
   End If
   If _tablaBusquedas <> String.Empty And _tablaBusquedas <> _tablaPrincipal _
         And _dataSet.Tables.IndexOf(_tablaBusquedas) <> -1 Then
      _dataSet.Tables(_tablaBusquedas).Clear()
   End If
   'Agrega los registros obtenidos en la búsqueda
   ds.Tables(0).TableName = _tablaPrincipal
   _dataSet.Merge(ds)
   If _tablaBusquedas <> String.Empty And _tablaBusquedas <> _tablaPrincipal _
      And _dataSet.Tables.IndexOf(_tablaBusquedas) <> -1 Then
      ds.Tables(0).TableName = _tablaBusquedas
      _dataSet.Merge(ds)
   End If
   'Acepta los cambios
   _dataSet.AcceptChanges()
End Sub
```

b. **FiltroGlobal**: se usa en caso de tener que aplicar un filtro global a todas las búsquedas del abm, independientemente de los parámetros de búsqueda ingresados por el usuario. Todos estas propiedades deben setearse antes de configurar el *ObjetoReglasNegocio*.

```
Private Sub UnidadesMonetarias_Load(ByVal sender As Object, ByVal e As _
System.EventArgs) Handles MyBase.Load

   Try
      'Configuración del abm
      Me.ColumnasNoVisiblesEnGrilla = "va_observaciones, fe_operacion"
      Me.CamposNoActualizables = "st_bloqueado, st_estado"
      Me.FiltroGlobal = "not st_bloqueado"
      Me.TablaBusquedas = "wav_operadores_activos"
      Me.OrdenBusquedas = "nm_apellido, nm_nombres"
      Me.CargaAutom = False

   _unidadesMonetarias=CType(RemotingHelper.getObject(GetType(IUnidadesMonetarias)),
IUnidadesMonetarias)
      Me.ObjetoReglasNegocio = _unidadesMonetarias

   Catch ex As Exception
      StartFrame.Us.Display.ShowError(ex.message)
   End Try
End Sub
```

c. **AltasOk**: al ponerla en *false* se inhabilita la posibilidad de realizar altas en el abm.

d. **BajasOk**: al ponerla en *false* se inhabilita la posibilidad de realizar bajas en el abm.

e. **ModifOk**: al ponerla en *false* se inhabilita la posibilidad de realizar modificaciones en el abm.

f. **MovimOk**: al ponerla en *false* se inhabilita la posibilidad de realizar movimientos internos entre registros del abm o consultor. Todas estas propiedades deben setearse después de asignarle valor al *ObjetoReglasNegocio* para que sobreescriba la seguridad estándar.

13 Para **destruir objetos externos** al manejo de la clase que pudieron crearse en el programa, deberá ponerse código en el evento *Clase_Dispose*.

14 En caso de tener que realizar algún **refrezco de los datos de pantalla** (como carga de datos de las grillas) durante el movimiento entre registros, podrá utilizarse para ello el evento *Clase_Display*. Tener en cuenta que hay que crear un método en la capa BR para que retorne todos los registros de la grilla en base a una determinada condición, la cual lo vincula con la tabla de cabecera.

```
Private Sub UnidadesMonetarias_Display() Handles MyBase.Display

   'Borra los datos de la grilla y actualiza sus datos
   _dataSet.Tables("wld cotizaciones").Clear()
   _dataSet.Merge(_unidadesMonetarias.getCotizaciones(cd_unidad_monetaria.Text))

End Sub
```

15 Para **trapear errores** y mostrarlos en un formato más amigable para el usuario, utilizar el siguiente evento. Se aclara que los errores por excepciones vienen en formato de *string*, en tanto que los demás vienen informados en el dataset.

```
'Intercepta los errores
Private Sub Paises_MostrarErrores(ByVal ds As System.Data.DataSet, _
         ByVal textoErrores As String, ByRef cancela As Boolean) _
         Handles MyBase.MostrarErrores

   Dim valRet As String = ""       'Texto a mostar
   Dim findError As Boolean = True 'Indica si se trapearon todos los errores

   Try
      If ds.HasErrors Then
         'Busca errores recorriendo el dataset
         Dim t As Integer                'Table
         Dim r As Integer                'Row
         Dim c As Integer                'Col
         Dim tabError As DataTable       'Tabla con el error
         Dim colErrors() As DataColumn   'Columnas con errores
         Dim textError As String         'Texto con el error original

         'Recorre cada tabla, registro y columna del dataset
         For t = 0 To ds.Tables.Count - 1
            tabError = ds.Tables(t)
            For r = 0 To tabError.Rows.Count - 1
               colErrors = tabError.Rows(r).GetColumnsInError()
               For c = 0 To colErrors.GetUpperBound(0)
                  'Error a trapear
                  textError=tabError.Rows(r).GetColumnError(colErrors(r).ColumnName)

                  'Trapeo de errores
                  If tabError.TableName = "wad_provincias" _
                        And colErrors(c).ColumnName = "cd_provincia" _
                        And textError Like "*valor ingresado ya existe*" Then
                     valRet &= "Grilla Provincias: Está intentando ingresar
                        información duplicada." & vbCrLf
                  ElseIf tabError.TableName = "wad_provincias" _
                        And colErrors(c).ColumnName = "nm_provincia" _
                        And textError Like "*nombre incompleto*" Then
                     valRet &= "Grilla Provincias: Debe ingresar el nombre de
                        la provincia." & vbCrLf
                  Else
                     'Algún error no fue trapeado
                     findError = False
                  End If
               Next
            Next
         Next
```

```
      Else

         'Busca los errores por excepciones
         If textoErrores Like "*valor ingresado ya existe*" Then
            valRet &= "Grilla Provincias: Está intentando ingresar información
                  duplicada." & vbCrLf
         ElseIf textoErrores Like "*nombre incompleto*" Then
            valRet &= "Grilla Provincias: Debe ingresar el nombre de la
                  provincia." & vbCrLf
         Else
            'Algún error no fue trapeado
            findError = False
         End If

      End If

      'Verifica si encontró errores
      If findError Then
         'Cancela el display de errores en la clase base
         cancela = True
         'Muestra los errores
         StartFrame.US.Display.MsgBox(valRet, "Errores detectados", _
            StartFrame.US.Display.MsgBoxTipos.msgError)
      End If

   Catch ex As Exception
      'Ignora los errores, porque serán mostrados por la clase base
   End Try
End Sub
```

Interfaz Web

A continuación se especifica el procedimiento para la arquitectura Web Application. Deben realizarse las siguientes tareas[8] en la página Web (algunas corresponden al *html* de la página *ASPX* y otras al *code behing* subyacente):

1 **Crear una nueva página del tipo *Web Content Form*** dentro del proyecto *WebSite* en la carpeta *UserPages*. En el evento *load* de la página, deberá ponerse código para relacionar a la misma con el objeto de negocios correspondiente y vincularla con una instancia de *AbmBase*.

```
Imports StartFrame.BR.Web.Utilitarios

Partial Public Class Abm_Pruebas
   Inherits System.Web.UI.Page

   Public masterpage As AbmBase

   Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _
         Handles Me.Load
      'Instancia de la clase masterpage AbmBase
      masterpage = CType(Page.Master, AbmBase)
      'Asigna los atributos del AbmBase
      masterpage.lblTitulo = "Registro de Pruebas Realizadas"
      'Instancia del objeto de reglas del negocio
      Dim obr As New BR.Web.Utilitarios.Abms.Pruebas
      'Inicialización de la página Abmbase enlazada al objeto de negocios
      masterpage.InicializarPagina(obr)
   End Sub

   Protected Overloads Sub OnPreInit() Handles Me.PreInit
      'Aplicacion de Tema
      If Not Session("Tema") Is Nothing Then
         Page.Theme = Session("Tema").ToString
      End If
   End Sub
End Class
```

2 Podrán utilizarse ciertas propiedades de la plantilla para **configurar el aspecto y funcionamiento** del abm, sobre todo si va a crear un abm automático y no uno con diseño explícito.

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _
        Handles Me.Load
   'Campos de filtado (si esta vacio los incluye a todos)
   Dim cf As String = "cd_prueba, fe_prueba, nu_prueba, nm_prueba, nm_programa, sadf"
   'Columnas de la grilla (si esta vacio las incluye a todas)
   Dim cg As String = "fe_prueba, nm_programa, nm_prueba"
   'Columnas del formulario de alta/modificacion (si esta vacio los incluye a todos)
   Dim ce As String = "cd_prueba, fe_prueba, nu_prueba, nm_prueba, nm_programa"

   'Alias de columnas (si esta vacio pone los nombres de los campos)
   Dim ac As New Hashtable
   ac.Add("cd_prueba", "Código")
   ac.Add("fe_prueba", "Fecha")
   ac.Add("nu_prueba", "N° Prueba")
   ac.Add("nm_prueba", "Detalle")
   ac.Add("nm_programa", "Funcionalidad")

   'Instancia de la clase masterpage AbmBase
   masterpage = CType(Page.Master, AbmBase)

   'Asigna los atributos del AbmBase
   masterpage.lblTitulo = "Registro de Pruebas Realizadas"
   masterpage.FiltrosBusqueda = cf
   masterpage.ColumnasGrilla = cg
```

[8] Las tareas que figuran con la viñeta en **fondo negro** son obligatorias.

```
   masterpage.CamposEdicion = ce
   masterpage.AliasCampos = ac

   'Instancia del objeto de reglas del negocio
   Dim obr As New BR.Web.Utilitarios.Abms.Pruebas
   'Inicialización de la página Abmbase enlazada al objeto de negocios
   masterpage.InicializarPagina(obr)
End Sub
```

3 En caso de desear crear una **página con diseño explícito** en el código HTML, simplemente debe evitar asignarle valor a la propiedad del *AbmBase CamposEdicion*. Luego, deberá crear en la página ASPX un contenedor (con **ID="ContentPrincipal"** y **ContentPlaceHolderID="FormularioEdicion"**). Dentro de dicho contenedor se debe diseñar el formulario de la página, respetando siempre los ID de los controles asignándoles el mismo nombre que el campo de la tabla con la cual se bindea.

```
<asp:Content ID="Content1" ContentPlaceHolderID="FormularioEdicion" runat="server">
   <asp:Panel ID="panelEdicion" runat="server">
      <asp:Table ID="Table1" runat="server">

         <!--Textbox-->
         <asp:TableRow>
            <asp:TableHeaderCell>
               <asp:Label ID="Label4" class="label" runat="server" Text="Caracteres:">
               </asp:Label>
            </asp:TableHeaderCell>
            <asp:TableCell>
               <asp:TextBox ID="ca_caracteres" runat="server" >
               </asp:TextBox>
            </asp:TableCell>
         </asp:TableRow>

         <!--Calendario-->
         <asp:TableRow>
            <asp:TableHeaderCell>
               <asp:Label ID="Label5" class="label" runat="server" Text="Mínimo:">
               </asp:Label>
            </asp:TableHeaderCell>
            <asp:TableCell>
               <asp:TextBox ID="va_minimo" runat="server">
               </asp:TextBox>
               <cc1:CalendarExtender ID="CalendarExtender1" runat="server"
                  TargetControlID="va_minimo" Format="dd-MM-yyyy"
                  PopupButtonID="/images/Calendar_scheduleHS.png" Enabled="false">
               </cc1:CalendarExtender>
            </asp:TableCell>
         </asp:TableRow>

         <!--Combobox-->
         <asp:TableRow>
            <asp:TableHeaderCell>
               <asp:Label ID="Label1" class="label" runat="server" Text="Nombre:" >
               </asp:Label>
            </asp:TableHeaderCell>
            <asp:TableCell>
               <asp:DropDownList id="nm_campo" runat="server" class="dropdownlist">
               </asp:DropDownList>
            </asp:TableCell>
         </asp:TableRow>

         <!—Radio button-->
         <asp:TableRow>
            <asp:TableHeaderCell>
               <asp:Label ID="Label2" class="label" runat="server" Text="Filtro:">
               </asp:Label>
            </asp:TableHeaderCell>
               <asp:TableCell>
               <asp:RadioButtonList ID="tp_filtro" runat="server">
                  <asp:ListItem  Selected="True" Value="U" Text="Unico"></asp:ListItem>
                  <asp:ListItem Value="D" Text="Desde-Hasta"></asp:ListItem>
```

```
                </asp:RadioButtonList>
            </asp:TableCell>
        </asp:TableRow>

    </asp:Table>
  </asp:Panel>
</asp:Content>
```

4 Existen varios eventos que podrán utilizarse en el caso de que desee realizar validaciones, asignar valores por defecto a los controles, completar datos antes de grabar, etc. En cualquier caso, deberá crearse un manejador para cada evento en cuestión (ver *documentación de ayuda online* para mayores detalles).

5 Si fuera necesario, podrá cear **más páginas del tipo *Web Content Form***. Para ello, deberá ingresar el siguiente código en el evento *load* de la página:

```
        'Cantidad de tabs a utilizar máximo 8
        Dim tp As Integer = 8
        'Asigna los atributos del AbmBase
        masterpage.TabsAdicionales = tp
        'Nombre de los tabs a utilizar
        masterpage.lblTabs = "Página 1"
        masterpage.lblTabs1 = "Página 2"
        masterpage.lblTabs2 = "Página 3"
        masterpage.lblTabs3 = "Página 4"
        masterpage.lblTabs4 = "Página 5"
        masterpage.lblTabs5 = "Página 6"
        masterpage.lblTabs6 = "Página 7"
        masterpage.lblTabs7 = "Página 8"
```

Luego en el código HTML de la página, se ingresarán porciones de código similares a:

```
<asp:Content ID="Content1" ContentPlaceHolderID="FormularioEdicion" runat="server">
        <asp:Panel ID="panelEdicion" runat="server">
                'Código HTML'
        </asp:Panel>
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="FormularioEdicion1" runat="server">
        <asp:Panel ID="panelEdicion1" runat="server">
                'Código HTML'
        </asp:Panel>
</asp:Content>

…

<asp:Content ID="Content9" ContentPlaceHolderID="FormularioEdicion8" runat="server">
        <asp:Panel ID="panelEdicion8" runat="server">
        'Código HTML'
        </asp:Panel>
</asp:Content>
```

Grillas (interfaz Windows Desktop)

Ya se mencionaron algunos aspectos relativos a las grillas[9] en el capítulo que describe los abms, sin embargo aquí se detalla todo lo relativo a estos controles, que suelen ser los más complicados de cualquier formulario Windows Desktop.

Creación

Para crear la grilla (tanto la grilla en sí como sus columnas), deberán utilizarse los controles provistos para tales fines. Sin embargo, cabe señalar que hay que crear las columnas y las características de la grilla mediante código y no mediante un generador o el seteo de propiedades en forma visual.

En el *Load* del formulario, hay que llamar a un procedimiento que se encargue de crear cada grilla.

```
Private Sub UnidadesMonetarias_Load(ByVal sender As Object, ByVal e As _
System.EventArgs) Handles MyBase.Load

   Try
      _unidadesMonetarias = _
      CType(RemotingHelper.getObject(GetType(IUnidadesMonetarias)), IUnidadesMonetarias)
      Me.ObjetoReglasNegocio = _unidadesMonetarias

      'Grillas
      InicializarGrillaCuentas()

   Catch ex As Exception
      StartFrame.Us.Display.ShowError(ex.message)
   End Try
End Sub
```

Ese procedimiento tendrá la siguiente estructura. Tener en cuenta que se muestran varias columnas para que pueda apreciarse la diferencia en la creación de los diferentes tipos de campos. Crear sólo las columnas que deben mostrarse en la grilla (no crear columnas ocultas). Tener muy presente que el nombre de la grilla debe ser igual al nombre de la tabla de detalle a la cual está bindeada (para que el abm maneje automáticamente su estado).

```
#Region "Declaraciones"
   'Controles Grilla Cuentas
   Private dg1_cd_proveedor As New DataGridLnkTextBoxColumn()
   Private dg1_cd_banco As New DataGridLnkTextBoxFKColumn("ccd_bancos", "cd_banco",
      "nm_banco")
   Private dg1_tp_cuenta_bancaria As New DataGridLnkComboBoxColumn( _
      DataSource:=_Proveedores.getCuentasBancarias.Tables(0), _
      DisplayMember:=1, ValueMember:=0)
   Private dg1_st_noalaorden As New DataGridLnkCheckBox()
   Private dg1_im_total As New DataGridLnkTextBoxNumericoColumn()
#End Region

#Region "Grilla Cuentas"
   'Creación de la grilla
   Private Sub InicializarGrillaCuentas()

      'Inicialización del modo gráfico
      Dim avgCharWidth As Integer
      With Graphics.FromHwnd(Me.Handle). _
         MeasureString(Text:="ABCDEFGHIJKLMNOPQRSTUVWXYZ", Font:=Me.Font)
         avgCharWidth = CInt(.Width / 26.0!)
      End With

      'Columnas
```

[9] El presente capítulo hace referencia al uso de la grilla nativa de .Net

```
Dim gridTableStyle As DataGridLnkTableStyle
gridTableStyle = New DataGridLnkTableStyle()
With gridTableStyle
   'Mapeo
   .MappingName = "kld_cuentas_proveedores"
   'Agrega las columnas
   With .GridColumnStyles

      'cd_proveedor (columna del tipo Textbox común)
      With Me.dg1_cd_proveedor
         .MappingName = "cd_proveedor"
         .TextBox.Name = .MappingName
         .HeaderText = Common.Display.FieldToText("cd_proveedor", _
            Common.Display.FormatoMnemotecnicos.CORTO)
         .Alignment = HorizontalAlignment.Left
         .Width = avgCharWidth * 15
         .NullText = ""
         .Obligatorio = True
      End With
      .Add(dg1_cd_proveedor)

      'cd_banco (columna del tipo TextboxFK)
      With Me.dg1_cd_banco
         .MappingName = "cd_banco"
         .TextBox.Name = .MappingName
         .HeaderText = Common.Display.FieldToText("cd_banco", _
            Common.Display.FormatoMnemotecnicos.CORTO)
         .Alignment = HorizontalAlignment.Left
         .Width = avgCharWidth * 10
         .NullText = ""
         .LvCamposRetorno = "nm_sucursal,tp_sucursal"
         .AbmModo = DataGridLnkTextBoxFKColumn.Modos.PermiteTodas
         .AbmForm = " StartFrame.US.Tesoreria.Abms.Bancos, US.tesoreria"
         .AbmWhere = "cd_banco = '?'"
      End With
      .Add(dg1_cd_banco)

      'tp_cuenta_bancaria (columna del tipo Combobox)
      With Me.dg1_tp_cuenta_bancaria
         .MappingName = "tp_cuenta_bancaria"
         .HeaderText = Common.Display.FieldToText("tp_cuenta_bancaria", _
            Common.Display.FormatoMnemotecnicos.CORTO)
         .Alignment = HorizontalAlignment.Left
         .Width = avgCharWidth * 15
         .NullText = ""
      End With
      .Add(dg1_tp_cuenta_bancaria)

      'st_noalaorden (columna del tipo Checkbox)
      With Me.dg1_st_noalaorden
         .MappingName = "st_noalaorden"
         .HeaderText = Common.Display.FieldToText("st_noalaorden", _
            Common.Display.FormatoMnemotecnicos.CORTO)
         .Alignment = HorizontalAlignment.Center
         .Width = avgCharWidth * 9
         .NullValue = False            'valor por defecto para este tipo de campos
         .AllowNull = False
      End With
      .Add(dg1_st_noalaorden)

      'im_total (columna del tipo TextBoxNumérico)
      With Me.dg1_im_total
         .MappingName = "im_total"
         .TextBox.Name = .MappingName
         .HeaderText = Common.Display.FieldToText("im_total", _
            Common.Display.FormatoMnemotecnicos.CORTO)
         .Alignment = HorizontalAlignment.Center
         .Width = avgCharWidth * 10
         .NullText = "0"
         .Format = "c"  'formato numérico
         .Minimo = 0
         .Maximo = 999999999
      End With
      .Add(dg1_im_total)
```

```
         End With
      End With

      'Configuración de la grilla
      With kld_cuentas_proveedores
         .TableStyles.Add(table:=gridTableStyle)
         .DataSource = _dataSet.Tables("kld_cuentas_proveedores")
      End With

   End Sub

#End Region
```

Refrezco de Datos

Para obtener los datos de la grilla y conseguir que se refrezquen con el movimiento entre registros, deberá agregarse en la **capa de negocios** el siguiente método (sin olvidar declarar su interfaz como corresponde). Tal vez sea necesario reemplazar el uso del método base *Buscar* por la ejecución de un *DA.Select*.

```
'Retorna los registros para la grilla de cuentas de proveedores
Public Function getCuentas(ByVal cuentaID As String) _
                  As DataSet Implements ICuentasProveedores.getCuentas

       Return _cuentas.Buscar("cd_cuenta = '" & cuentaID & "'")

End Function
```

En la **capa de usuario**, deberá llamarse al método anteriormente mencionado en el evento **Display()**.

```
'Al moverse entre registros
Private Sub Proveedores_Display() Handles MyBase.Display
   'Borra los datos de las grillas
   _dataSet.Tables("kld_cuentas_proveedores").Clear()
   'Actualiza los datos de las grillas
   _dataSet.Merge(_Proveedores.getCuentas(Me.cd_proveedor.Text))
End Sub
```

También deberá llamarse al evento *Display* después de eliminar y al iniciar la edición (en las altas), a efectos de que refresque la grilla.

```
'Después de eliminar
Private Sub Proveedores_Eliminado(ByVal ds As System.Data.DataSet) _
                                  Handles MyBase.Eliminado
   'Refrezca datos de la grilla
   Me.Proveedores_Display()
End Sub

'Al iniciar la edición
Private Sub Proveedores_Editando(ByRef cancela As Boolean) _
                                  Handles MyBase.Editando
   'Refrezca datos de la grilla
   If Me.Estado = Estados.Alta Then
      Me._Proveedores_Display()
   End If
End Sub
```

Además, deberán completarse los campos de la tabla que no se muestran en la grilla y son generalmente los que vinculan la grilla con la tabla padre del abm.

```
'Antes de grabar
Private Sub Proveedores_CompletarDatos(ByRef ds As System.Data.DataSet) _
                                       Handles MyBase.CompletarDatos
   'Completa los campos ocultos de las grillas
   Dim row As DataRow
   For Each row In ds.Tables("kld_cuentas_proveedores").Rows
      If row.RowState <> DataRowState.Deleted Then
         If IsDBNull(row.Item("cd_proveedor")) Then
            row.Item("cd_proveedor") = Me.cd_proveedor.Text.TrimEnd
         End If
      End If
   Next
End Sub
```

Validaciones locales

Pueden realizarse dos tipos de validaciones: cuando cambia de celda (columna o fila) y a nivel de la grilla. Estas validaciones deberán apuntar a temas de formato, ya que las validaciones importantes deben estar siempre en la capa de negocios.

Las validaciones a nivel de la grilla en realidad se realizan a nivel del formulario debido a que estas no se ejecutarían si el usuario nunca entra a la grilla. Para ello, se utiliza el evento **Form_Actualizando** que se dispara antes de enviar a grabar los datos a la capa de negocios.

```
'Validaciones
Private Sub Proveedores_Actualizando(ByRef cancela As Boolean) _
                                     Handles MyBase.Actualizando
   'Valida que exista un registro en la grilla de cuentas
   If _dataSet.Tables("kld_cuentas_proveedores").Rows.Count = 0 Then
      ErrorProvider1.SetError(kld_cuentas_proveedores, "Grilla Cuentas incompleta.")
      cancela = True
   End If

End Sub
```

Para validar a nivel de celda, deberá utilizarse el evento de la grilla **CurrentCellChanged**. Este evento se dispara cada vez que se posiciona en una nueva columna (al entrar a la nueva celda, no al salir de la anterior, por lo cual deberá utilizarse cierta lógica para saber no sólo dónde llegó sino de dónde venía). Las validaciones deben referirse a las celdas de la grilla y no a los campos de la tabla, ya que esta última no posee registros hasta que se cambia de fila, por lo tanto puede funcionar sólo en ocasiones. Para impedir que se edite una columna, se la puede definir como *ReadOnly*, pero cuando hay que evitar que se edite la columna en forma dinámica (dependiendo del valor de otros campos), debe utilizarse este evento. El mismo también se utiliza para realizar validaciones y asignar valores por defecto. Tener presente que cuando se posee una columna con un localizador habilitado, puede utilizarse *OnTextValidated* para optar por una alternativa a este último tipo de validación.

```
#Region "Declaraciones"
   Private grdMG_col_ant As Integer = 0
   Private grdMG_row_ant As Integer = 0
#EndRegion

'Al cambiar de celda
Private Sub wld_miembros_grupo_CurrentCellChanged(ByVal sender As Object, _
                                     ByVal e As System.EventArgs) _
                                     Handles wld_miembros_grupo.CurrentCellChanged

   'Variables de trabajo
   Dim cel As DataGridCell = wld_miembros_grupo.CurrentCell
   Dim col As Integer = cel.ColumnNumber
   Dim row As Integer = cel.RowNumber
   Dim tabla As DataTable = CType(wld_miembros_grupo.DataSource, DataTable)
```

```
    'No realiza validaciones si la grilla es de sólo lectura
    If Not wld_miembros_grupo.Enabled Or wld_miembros_grupo.ReadOnly Then
        Exit Sub
    End If

    Try
        'Columnas a validar
        Dim nCol_cd_operador As Integer = 0
        Dim nCol_nm_operador As Integer = 1
        Dim nCol_tp_operador As Integer = 2
        Dim nCol_po_pago As Integer = 3

        'Valida el ingreso a las celdas
        Dim bEdit As Boolean = (wld_miembros_grupo.Item(row, 0)="N/A")
        Me.col_im_neto.ReadOnly = bEdit
        Me.col_im_iva.ReadOnly = bEdit
        Me.col_im_total.ReadOnly = bEdit

        'Asigna un valor al campo descriptivo y saltea la celda
        If grdMG_col_ant = nCol_cd_operador _
              AndAlso Not IsNothing(col_cd_operador.LvRetorno) Then
           wld_miembros_grupo.Item(grdMG_row_ant, nCol_nm_operador) =
col_cd_operador.LvRetorno().Trim
        End If

        'Asigna un valor por defecto
        If grdMG_col_ant = nCol_tp_operador _
              AndAlso IsDBNull(wld_miembros_grupo.Item(grdMG_row_ant, nCol_po_pago)) Then
           wld_miembros_grupo.Item(grdMG_row_ant, nCol_po_pago) = 100
        End If

        'Permite sólo valores numéricos entre 1 y 100 para po_pago.
        If grdMG_col_ant = nCol_po_pago Then
           If wld_miembros_grupo.Item(grdMG_row_ant, nCol_po_pago) > 100 Then
              wld_miembros_grupo.Item(grdMG_row_ant, nCol_po_pago) = 100
              StartFrame.US.Display.MsgBox("Solo se permiten números de 1 a 100", _
                 "Error de Validación", StartFrame.US.Display.MsgBoxTipos.msgAlerta)
           End If
        End If

        'Si cambió de fila, verifica que pueda hacerlo, sino retorna
        If grdMG_row_ant <> row _
              AndAlso Not ValidaRow(grdMG_row_ant) Then
           wld_miembros_grupo.CurrentCell = New DataGridCell(grdMG_row_ant, grdMG_col_ant)
        End If

    Catch ex As Exception

    Finally
        'Guarda la fila y columna actual para saber de dónde venía en el próximo ingreso
        grdMG_row_ant = row
        grdMG_col_ant = col
    End Try
End Sub
```

Otra alternativa para realizar las validaciones más comunes en relación al ingreso de datos en las columnas que contengan el control *DataGridLnkTextBoxFKColumn*, es utilizar el evento **OnTextValidated**, propio del mencionado control y que se ejecuta una vez ingresado un valor válido en la mencionada columna. Para poder hacer esto, el control debe declararse a nivel del formulario y con la cláusula *withevents*.

```
'Al ingresar un código de operador
Private Sub dg1_cd_operador_OnTextValidated(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles dg1_cd_operador.OnTextValidated
   'Variables de trabajo
   Dim cel As DataGridCell = wld_miembros_grupo.CurrentCell
   Dim col As Integer = cel.ColumnNumber
   Dim row As Integer = cel.RowNumber

   Try
      'Columnas de la grilla
      Dim nCol_cd_operador As Integer = 0
      Dim nCol_nm_operador As Integer = 1

      'Valida el código del operador y carga su nombre
      If Not IsNothing(dg1_cd_operador.LvRetorno) Then
         wld_miembros_grupo.Item(row, nCol_nm_operador) = dg1_cd_operador.LvRetorno()
      End If
   Catch ex As Exception
   End Try
End Sub
```

En caso de necesitar realizar alguna validación en un control *DataGridLnkTextBoxFKColumn* previo al llamado del localizador, podrá utilizarse el evento **OnLocate**, propio del mencionado control y que se ejecuta antes de llamar a una ventana de apoyo. Para poder hacer esto, el control debe declararse a nivel del formulario y con la cláusula *withevents*.

```
'Al ingresar un código de operador
Private Sub dg1_cd_operador_OnLocate(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles dg1_cd_operador.OnLocate
   'Variables de trabajo
   Dim cel As DataGridCell = wld_miembros_grupo.CurrentCell
   Dim col As Integer = cel.ColumnNumber
   Dim row As Integer = cel.RowNumber

   Try
      'Columnas de la grilla
      Dim nCol_cd_operador As Integer = 0
      Dim nCol_nm_operador As Integer = 1

      'Valida el código del operador y carga su nombre
      If dg1_tp_operador.Textbox.Text = "supervisor" Then
         dg1_cd_operador.LvWhere = "st_supervisor=1"
      End If
   Catch ex As Exception
   End Try
End Sub
```

Validaciones especiales

Existen algunas validaciones que si bien son aplicables a cualquier tabla, son muy típicas de las grillas, por ello se describen en este capítulo.

Para validar que no se puedan **insertar o eliminar registros en una grilla**, deberá realizarse la validación en las reglas de negocios dentro del evento **Validando** de la clase que maneja la grilla.

```
Private Sub _grupoAccesos_Validando(ByRef cancela As Boolean, ByRef paramDataTable _
                                    As System.Data.DataTable) Handles MyBase.Validando
   'No deben eliminarse registros en esta grilla
   If Not paramDataTable.GetChanges(DataRowState.Deleted) Is Nothing Then
      cancela = True
      Throw New Exception("No se pueden eliminar registros en la grilla de accesos.")
   End If
   'No deben insertarse registros en esta grilla
   If Not paramDataTable.GetChanges(DataRowState.Added) Is Nothing Then
      cancela = True
      Throw New Exception("No se pueden insertar registros en la grilla de accesos.")
   End If
End Sub
```

Columnas no mapeadas

En caso de necesitar agregar columnas a la grilla que no se correspondan con ninguna columna de la tabla, deberá realizarse lo siguiente.

En primer lugar, deberá modificarse el método que realiza la **búsqueda de registros** de la capa de negocios que retorna los registros de la grilla para que retorne todas las columnas necesarias, ya sea con un *DA.Sql.Select* (como en el siguiente ejemplo) o bien ejecutando un stored procedure.

```
Public Function getMiembros(ByVal grupoID As String) As DataSet
                            Implements IGrupoOperador.getMiembros
   'Consulta para buscar los datos necesarios
   Dim ds As DataSet
   Dim sSelect As String = "mgr.cd_operador, ope.nm_operador"
   Dim sFrom As String = "wld_miembros_grupo mgr JOIN wad_operadores ope " _
                       & "ON mgr.cd_operador = ope.cd_operador"
   Dim sWhere As String = "mgr.cd_grupo_operadores = '" & grupoID & "'"
   ds = DA.Sql.Select(Common.Env.ConnectionString, sSelect, sFrom, sWhere)

   'Renombra la tabla para poder hacer un Merge
   ds.Tables(0).TableName = "wld_miembros_grupo"

   Return ds

End Function
```

Las columnas agregadas al *datatable* que no pertenezcan a la tabla a grabar, deberán ser removidas del *dataset* antes de que el mismo sea grabado en el método **ActualizarDatos**.

```
Public Overloads Overrides Function ActualizarDatos _
                            (ByRef paramDataSet As System.Data.DataSet) As Boolean
   Dim tran As OleDbTransaction = Nothing
   Try
      InicializarConeccion()
      _cn.Open()
      tran = _cn.BeginTransaction

      'Elimina las columnas no mapeadas
      paramDataSet.Tables("wld_miembros_grupo").Columns.Remove("nm_operador")

      'Si se esta eliminando la cabecera primero se borran los detalles
      If Not paramDataSet.Tables(_tablaPrincipal).GetChanges(DataRowState.Deleted) _
         Is Nothing Then
```

```
            _grupoMiembros.ActualizarDatos(paramDataSet, _cn, tran)
```

En la capa de usuario, la **definición de la columna** vinculada, se realizará normalmente (aunque será por lo general de sólo lectura), pero deberá declararse privado (utilizando *Private* en lugar de *Dim*) cualquier control de columna del cual desee posteriormente consultar sus propiedades. Típicamente, esto ocurrirá con los controles del tipo *DataGridLnkTextBoxFKColumn* que posean alguna columna vinculada.

```
Private dgTextBoxColumn2 As New DataGridLnkTextBoxFKColumn( _
            "wad_operadores", "cd_operador", "nm_operador")

'Grilla Cuentas
Private Sub InicializarGrillaMiembros()

   'Crea las columnas
   With gridTableStyle
      'Mapeo
      .MappingName = "kld_cuentas_proveedores"
      'Agrega las columnas
      With .GridColumnStyles
         'cd_operador (columna del tipo TextboxFK)
         With dgTextBoxColumn2
            .MappingName = "cd_operador"
            .HeaderText = Common.Display.FieldToText("cd_operador", _
               Common.Display.FormatoMnemotecnicos.CORTO)
            .Width = avgCharWidth * 10
            .LvCamposRetorno = "nm_operador"
            .AbmModo = DataGridLnkTextBoxFKColumn.Modos.NoPermiteOperaciones
         End With
         .Add(dgTextBoxColumn2)

         'nm_operador (columna asociada a cd_operador)
         With dgTextBoxColumn3
            .MappingName = "nm_operador"
            .HeaderText = Common.Display.FieldToText("nm_operador", _
               Common.Display.FormatoMnemotecnicos.CORTO)
            .Width = avgCharWidth * 30
            .ReadOnly = True  'Columna de sólo lectura
         End With
         .Add(dgTextBoxColumn3)
```

Deberá asignarse el valor a la columna asociada cuando cambie el valor de la columna principal. Esto puede hacerse en el evento **CurrentCellChanged**.

```
Private Sub grilla_CurrentCellChanged(ByVal sender As Object, _
                          ByVal e As DataColumnChangeEventArgs)
   ...

   'Completa el nombre del operador
   If grdMG_col_ant = nCol_cd_operador _
         AndAlso Not IsNothing(col_cd_operador.LvRetorno) Then
      wld_miembros_grupo.Item(grdMG_row_ant, nCol_cd_operador) =
col_cd_operador.LvRetorno().Trim
   End If

End Sub
```

En el evento **_editando** y en **_eliminado**, deberá llamar al evento **_display** para que aparezcan las columnas no mapeadas en la grilla.

```
Private Sub grilla_Eliminado(ByVal ds As System.Data.DataSet) _
                                    Handles MyBase.Eliminado
   'Refrezca datos de la grilla
   Me.Proveedores_Display()
```

```
End Sub

Private Sub grilla_Editando(ByRef cancela As Boolean) _
                                    Handles MyBase.Editando
   'Refrezca datos de la grilla
   If Me.Estado = Estados.Alta Then
      Me. Proveedores_Display()
   End If
End Sub
```

Validaciones Estándar

A continuación se especificará la manera de realizar las distintas operaciones (entre ellas las validaciones más comunes) que pueden especificarse para un Abm. Cada ejemplo mostrado no es la única forma de hacerlas, pero es una forma válida y sin errores que es recomendable adoptar como estándar.

Casi cualquier porción de código utilizada en esta sección puede ser creada mediante el uso de los *code snippets* incluidos en *StartFrame*.

Dato sin duplicados

Para esto es necesario crear en el motor de datos una restricción (no un índice) del tipo *Unique*.

Dato obligatorio

Este control se realiza generalmente en el motor de datos, seteando la propiedad *Null / Not Null* del campo correspondiente.

Integridad referencial

Configurar esta validación en el motor de base de datos, en la tabla correspondiente. A no ser que se indique lo contrario, siempre es aconsejable adoptar el criterio de *"actualización en cascada"*, independientemente del criterio adoptado para la eliminación. Tener en cuenta que por razones internas del motor, hay ocasiones en que no se permite establecer un criterio de actualización o eliminación en cascada. En tal caso, deberá realizarse la validación o eliminación (según corresponda) dentro de las reglas de negocio.

Dato predeterminado al grabar

Desde el diseño de la tabla en el motor de datos, ingresar los valores predeterminados para cada campo.

Si el campo no figura en el formulario/página o el valor por defecto sea de cálculo condicionado a otros campos, la validación debe realizarse colocando código en el evento *CompletarDatos.* Otra posibilidad es agregar un *Textbox* invisible y asignarle su valor en el evento *Actualizando*.

```
Private Sub Operadores_CompletarDatos(ByRef ds As System.Data.DataSet) Handles
MyBase.CompletarDatos

   'Completa el dato va_clave_acceso = cd_operador en el caso de un alta
   If Me.Estado = Estados.Alta Then
      Dim row As DataRow
      For Each row In ds.Tables("wad_operadores").Rows
         If row.RowState <> DataRowState.Deleted Then
            row.Item("va_clave_acceso") = EncryptString(row.Item("cd_operador"))
         End If
      Next
   End If

End Sub
```

Registro inborrable

La validación debe realizarse colocando código en el evento *Eliminando*. A modo de ejemplo, en el caso de la interfaz Windows Desktop, utilizar la siguiente estructura modificando los valores resaltados. Si el campo condicionante de esta validación figura en el formulario, tratar como cualquier error, caso contrario dirigir el error al campo clave que figure en pantalla.

```
Private Sub TipoTelefonos_Eliminando(ByRef cancela As Boolean, ByRef row As _
System.Data.DataRow) Handles MyBase.Eliminando

   'Valida que no elimine un registro marcado como de sistema
   If CType(row.Item("st_interno_sistema", DataRowVersion.Original), _
   Boolean) = True Then
      row.SetColumnError("tp_telefono", "No se puede eliminar el registro.")
   End If

End Sub
```

FK contra otra tabla

Configurar esta validación en la base de datos, en la tabla correspondiente, según lo especificado en la planilla de diseño. En la tabla del lado de *muchos* en la relación deberá crearse un índice tipo *FK* por los campos de la relación.

Rango de valores aceptables

Se puede ingresar esta validación en el evento Validando de las reglas del negocio.

```
Private Sub Programas_Validando(ByRef cancela As Boolean, ByRef paramDataTable As
System.Data.DataTable) Handles MyBase.Validando

   Dim row As DataRow

   For Each row In paramDataTable.Rows
      With row
         If row.RowState <> DataRowState.Deleted Then
            'Valida el nombre de programa
            If Not IsDBNull(row("nm_programa")) AndAlso row("nm_programa").ToString
            <> "" AndAlso row("nm_programa").ToString.Length > 1 Then
               'Asigna el tipo de programa que sale del segundo caracter del nombre
               row("tp_programa") = row("nm_programa").ToString.Substring(1, 1)
            Else
               row.SetColumnError("nm_programa", "El nombre del programa es inválido.")
            End If
         End If
      End With
   Next
End Sub
```

Datos por defecto

Como esto es necesario antes de iniciar una edición, el mejor lugar para hacerlo es el evento *Editando*.

```
'Al iniciar la edición
Private Sub Proveedores_Editando(ByRef cancela As Boolean) _
                                    Handles MyBase.Editando
   'Valores por defecto
   If Me.Estado = Estados.Alta Then
      Me.fe_alta.Value = Now()
   End If
End Sub
```

Campos del tipo identity

A este tipo de campo no puede asignársele ningún valor en particular, con lo cual no podrá editarse nunca, pero es posible que quiera visualizarse. En este último caso, agregar el control pero del tipo *ReadOnly*.

Controles del tipo filtro

Interfaz Windows Desktop

En caso de desear utilizar algunos filtros a modo de buscadores que al elegirlos refresquen la tabla principal, deberán ubicarse los mismos sin un bindeo directo a la tabla (en todo caso, asignarle valor al campo bindeado equivalente). Una forma de hacer esto es utilizando un control del tipo *LnkTextboxFK* en modo *Combo*, como en el siguiente ejemplo.

```
'Al seleccionar un valor para el filtro
Private Sub sel_operador_OnSelectedChanged(ByVal sender As Object, _
   ByVal e As System.EventArgs) Handles sel_operador.OnSelectedChanged
   Try
      'Rearma la grilla principal
      Dim filtro As String = CType(Me.sel_operador.cboDescription.SelectedValue, String)
      Me.RefrescarGrilla("cd_operador='" & filtro & "'")
      'Restablece los valores de pantalla que ya se habían cargado
      Me.cd_operador.Text = filtro
      Me.sel_cd_operador.cboDescription.SelectedValue = filtro
   Catch ex As Exception
      'Ignora errores por estar sin datos
   End Try
End Sub
```

Controles del tipo localizadores

Interfaz Windows Desktop

A diferencia de los controles del tipo filtro, se entiende por localizador a un control utilizado para posicionarse en un registro pero sin realizar una nueva búsqueda, sino tan solo moviéndose en los registros del *dataset* actual. Se asume que dicho posicionamiento se desea realizar a través de la evaluación de una serie de condiciones. Para hacer lo descripto, deberá utilizar un código similar al del siguiente ejemplo (para un *abmGrilla*).

```
'Al seleccionar un valor para el localizador
Private Sub sel_operador_OnSelectedChanged(ByVal sender As Object, _
   ByVal e As System.EventArgs) Handles sel_operador.OnSelectedChanged
   Try
      'Rearma la grilla principal
      Dim filtro As String = CType(Me.sel_operador.cboDescription.SelectedValue, String)
      Dim cols As New ArrayList
      Dim vals As New ArrayList
      'Condiciones
      cols.Add("cd_operador")
      vals.Add(filtro)
      cols.Add("nu_interno_telefono")
      vals.Add(0)
      'Busca el registro
      Me.MovePosition(cols, vals)
      'Marca el registro seleccionado en la grilla
      Me.grdTablaPrincipal.Select(Me.CurrentPosition)
      Me.grdTablaPrincipal.CurrentRowIndex = Me.CurrentPosition
   Catch ex As Exception
      'Ignora errores por estar sin datos
   End Try
End Sub
```

Consultores

Como en los capítulos anteriores, el objetivo del presente punto es lograr una estandarización total en la programación de todo el sistema, de forma tal que sea imperceptible el paso de varios programadores diferentes, facilitar el mantenimiento y minimizar la posibilidad de errores.

Casi cualquier porción de código utilizada en esta sección puede ser creada mediante el uso de los *code snippets* incluidos en *StartFrame*.

Interfaz Windows Desktop

A continuación se especifica el procedimiento para cualquiera de las arquitecturas Windows Desktop.

En este caso, para confeccionar un *Consultor*, deberán realizarse los mismos pasos que para realizar un *Abm*, obviando todos los puntos referentes a validaciones y grabaciones de datos. En resumen, deberá:

1. Agregar un nuevo componente
2. Declarar la clase
3. Agregar la entrada en el archivo de configuración
4. Agregar las nuevas clases (si hubiera)
5. Agregar las grillas (si hubiera) al dataset

Interfaz Web

En este caso, las consultas toman la forma de "listados" contenidos en tablas HTML (en realidad, grillas paginadas).

A continuación se especifica el procedimiento para la arquitectura Web Application. Deben realizarse las siguientes tareas[10] en la página Web (algunas corresponden al *html* de la página *ASPX* y otras al *code behing* subyacente):

1 **Agregar un nuevo componente** (clase) en la capa de reglas de negocios que herede de *StartFrame.BR.Abm o StartFrame.BR.Consulta*. Realizar la **declaración de la clase** y asegurarse de apuntar a la tabla o vista origen de la consulta a realizar.

```
Imports Common.Env
Imports StartFrame.DA.Sql

Public Class Operadores                    'Nombre de la clase
   Inherits Abm                            'Herencia

   Sub New()
      'Modificar nombre de programa y tabla principal relacionada a la clase
      MyBase.New(Operador, Password, "waoperadores", "wad_operadores")
   End Sub
End Class
```

2 **Crear un stored procedure** con el origen de datos. Tener en cuenta que se deben colocar como parámetros del SP los rangos a utilizar como filtros en la página web.

[10] Las tareas que figuran con la viñeta en **fondo negro** son obligatorias.

3 Utilizando la misma aplicación web como interfaz, **dar de alta los filtros a utilizar en la consulta** por medio del abm correspondiente. Registrar correctamente las propiedades de cada filtro (tipo de dato, longitud, valor por defecto) y las validaciones pertinentes.

4 Utilizando la misma aplicación web como interfaz, **dar de alta los reportes a ejecutar en la página** por medio del abm correspondiente. Tener presente que pueden ejecutarse múltiples reportes por cada página (cada reportes apunta a un stored procedure que le da origen). También deberán **asociarse los filtros** antes mencionados.

5 **Crear una nueva página del tipo *Web Content Form*** dentro del proyecto *WebSite* en la carpeta *UserPages*. En el evento *load* de la página, deberá ponerse código para relacionar a la misma con el objeto de negocios correspondiente y vincularla con una instancia de *ReporteBase*.

```
Partial Public Class Lst_Pruebas
   Inherits System.Web.UI.Page

   Public masterpage As Reporte

   Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _
         Handles Me.Load
      'Instancia de la clase masterpage ReporteBase
      masterpage = CType(Page.Master, Reporte)
      'Asigna los atributos a la clase base
      masterpage.lblTitulo = "Informe de Pruebas Realizadas"
      'Instancia del objeto de reglas del negocio
      Dim obr As New BR.Web.Utilitarios.Abms.Operadores
      'Inicialización de la página ReporteBase enlazada al objeto de negocios
      MasterPage.InicializarPagina(obr)
   End Sub

   Protected Overloads Sub OnPreInit() Handles Me.PreInit
      'Aplicacion de Tema
      If Not Session("Tema") Is Nothing Then
         Page.Theme = Session("Tema").ToString
      End If
   End Sub

End Class
```

6 Existen varios eventos que podrán utilizarse en el caso de que desee obtener el origen de datos desde otra fuente diferente a un store procedure o validar dicho set de datos. En cualquier caso, deberá crearse un manejador para cada evento en cuestión (ver *documentación de ayuda online* para mayores detalles).

Procesos, Diálogos y Wizards

Como en los capítulos anteriores, el objetivo del presente punto es lograr una estandarización total en la programación de todo el sistema, de forma tal que sea imperceptible el paso de varios programadores diferentes, facilitar el mantenimiento y minimizar la posibilidad de errores.

Casi cualquier porción de código utilizada en esta sección puede ser creada mediante el uso de los *code snippets* incluidos en *StartFrame*.

Interfaz Windows Desktop

A continuación se especifica el procedimiento para cualquiera de las arquitecturas Windows Desktop.

Un *Proceso* es el tipo de programa que más variedad de formatos puede llegar a tener. Por ello, hay que estudiar previamente la funcionalidad deseada para el proceso y determinar si se desea realizar un **proceso** tipo *batch*, un **diálogo** con el usuario o un **Wizard**. Dependiendo de la elección anterior, cambiará la clase base de la cual hay que heredar:

- **Dialogo**: Esta clase deberá usarse para funcionalidades que tengan como obtetivo dialogar con el usuario y realizar algún proceso de relativamente corto tiempo de ejecución (*liviano*). Un ejemplo de este tipo de programas puede ser un logueo con cambio de password.
- **Wizard**: Es el caso de los procesos que son relativamente *pesados* en cuanto a su ejecución y que requieren una cierta comunicación con el usuario un poco más compleja o interactiva. Esta es la opción más recomendable por lo amigable de su funcionalidad. Un ejemplo de este tipo de programas puede ser una generación masiva de facturas en base a ciertos datos de abonos o pedidos de facturación.
- **Proceso**: Deberá usarse para ejecutar procesos *batch pesados* (sino debería usarse la clase *Dialogo*), los cuales tengan poca o nula comunicación con el usuario (sino debería usarse la clase *Wizard*). Podría decirse que debe utilizarse esta clase cuando no sean apropiadas ninguna de las anteriores.

Diálogos

Para el uso de esta clase, se hará mención únicamente a la parte correspondiente a la capa de usuario Windows Desktop, ya que puede utilizar cualquier componente de negocios con el cual el diálogo podría enlazarse.

En líneas generales, puede decirse que deberán realizarse los mismos pasos[11] mencionados para Abms, con algunas salvedades, las cuales se detallan a continuación[12].

> Los ejemplos dados a continuación harán referencia al enlace del diálogo con un componente de negocios del tipo Abm, que es el uso más complejo que puede dársele a esta clase.

1 **Agregar la interfaz**: puede que no se requiera una nueva interfaz, ya que el diálogo suele utilizar un componente de negocios ya existente.

2 **Agregar el formulario**: en este caso, heredado de *StartFrame.US.Dialogo*.

3 **Declaración de la clase**: varía dependiendo del tipo de uso que se le quiera dar.

```
Imports StartFrame.BR.Interfaces  'Procedencia de las referencias utilizadas

Namespace Procesos                          'Namespace de la clase
   Public Class EvolucionOficina            'Nombre de la clase
      Inherits StartFrame.US.Dialogo        'Herencia
      Dim _evolucion As Ievolucion          'Componente de negocios

      'Para poder utilizar BR.Abm
      Protected _dataset As DataSet
      Protected _tablaPrincipal As String

      ------------------------------------
      Windows form designer generated code
      ------------------------------------
   End Class
End Namespace
```

4 **Vinculación a BR**: deberá vincularse el diálogo a algún componente de la capa BR a fin de poder controlar la seguridad.

```
Private Sub EvolucionOficina _Load(ByVal sender As Object, ByVal e As _
System.EventArgs) Handles MyBase.Load

   Try
      _Ievolucion = CType(RemotingHelper.getObject(GetType(Ievolucion)), Ievolucion)
      Me.ObjetoReglasNegocio = _Ievolucion

      'Para poder utilizar BR.Abm
      _tablaPrincipal = _evolucion.getTablaPrincipal
      _dataset = _evolucion.getDataSet

   Catch ex As Exception
      StartFrame.Us.Display.ShowError(ex.message)
```

[11] Ver **tutorial** para una ilustración paso a paso de las tareas a realizar
[12] Las que figuran con la viñeta en **video inverso** son obligatorias

```
    End Try
End Sub
```

5 **Entrada en *.config**: esto será necesario únicamente si el programa utiliza algún nuevo componente de negocios.

6 **Aspecto visual**: Deberán realizarse todas aquellas tareas que tengan que ver con el aspecto visual del programa (título, controles) y la entrada del mismo en el menú.

7 **Uso con BR.Abm**: A fin de que pueda enlazarse el diálogo de la capa US con un componente del tipo Abm de la capa BR (para poder utilizar toda su funcionalidad elemental), deberán crearse algunas funciones internas (obtenidas de las clases *ConsultaBase* o *Abm*), a saber: *SetErrors, dsErrorsToString, ClearErrors*.

8 **Evento Ejecutar**: En este evento, deberá ubicarse todo el código que ejecute el proceso para el cual se creó esta pantalla, ya que el mismo se dispara cuando el usuario confirma la operación.

Wizards

Para el uso de esta clase, se hará mención únicamente a la parte correspondiente a la capa de usuario, ya que puede utilizar cualquier componente de negocios con el cual el wizard podría enlazarse.

En líneas generales, su uso es muy similar a los Diálogos y puede decirse que deberán realizarse los mismos pasos[13] mencionados en *Abms*, con algunas salvedades, las cuales se detallan a continuación[14].

> Los ejemplos dados a continuación harán referencia al enlace del wizard con un componente de negocios del tipo ClaseBase, sin embargo, puede enlazarse con cualquier tipo de componente.

1 **Agregar la interfaz**: puede que no se requiera una nueva interfaz, ya que el diálogo suele utilizar un componente de negocios ya existente.

2 **Agregar el formulario**: en este caso, heredado de *StartFrame.US.Wizard*.

3 **Declaración de la clase**: varía dependiendo del tipo de uso que se le quiera dar.

```
Imports StartFrame.BR.Interfaces  'Procedencia de las referencias utilizadas

Namespace Procesos                          'Namespace de la clase
   Public Class GeneracionFacturas          'Nombre de la clase
      Inherits StartFrame.US.Wizard         'Herencia
      Dim _genfact As Igenfact              'Componente de negocios

      'Para poder utilizar BR.Abm
      Protected _dataset As DataSet
      Protected _tablaPrincipal As String

      ------------------------------------
      Windows form designer generated code
      ------------------------------------
   End Class
End Namespace
```

4 **Vinculación a BR**: deberá vincularse el diálogo a algún componente de la capa BR a fin de poder controlar la seguridad.

```
Private Sub GeneracionFacturas_Load(ByVal sender As Object, ByVal e As _
System.EventArgs) Handles MyBase.Load

   Try
      _Igenfact = CType(RemotingHelper.getObject(GetType(Igenfact)), Igenfact)
      Me.ObjetoReglasNegocio = _Igenfact

      'Para poder utilizar BR.Abm
      _tablaPrincipal = _genfact.getTablaPrincipal
      _dataSet = _genfact.getDataSet

   Catch ex As Exception
      StartFrame.Us.Display.ShowError(ex.message)
   End Try
End Sub
```

[13] Ver **tutorial** para una ilustración paso a paso de las tareas a realizar
[14] Las que figuran con la viñeta en **video inverso** son obligatorias

5 **Entrada en *.config**: esto será necesario únicamente si el programa utiliza algún nuevo componente de negocios.

6 **Aspecto visual**: Deberán realizarse todas aquellas tareas que tengan que ver con el aspecto visual del programa y acceso al programa, a saber:

a. Eliminar las páginas sobrantes ejecutando, en el evento Load, el método *RemovePage*. Siempre deberán eliminarse las páginas intermedias (nunca la primera ni última página), comenzando desde la penúltima.
b. Cambiar las etiquetas de las páginas que quedaron, los labels de las páginas de inicio, fin e intermedias.
c. Agregar una entrada en el menú.

7 **Evento CambioPagina**: Utilizar este evento para poner el código requerido cada vez que se acceda a una página diferente.

8 **Evento Ejecutar**: En este evento, deberá ubicarse todo el código que ejecute el proceso para el cual se creó esta pantalla, ya que el mismo se dispara cuando el usuario confirma la operación.

Listados

Los listados tienen diferente formato dependiendo de la arquitectura (no es lo mismo un reporte Windows que una consulta por Web) y del tipo de reporte (informe diario, resumen gráfico, formulario, documento, etc.). Cualquiera sea el caso, a continuación se describe un procedimiento estándar para su creación.

Interfaz Windows Desktop

A continuación se especifica el procedimiento para cualquiera de las arquitecturas Windows Desktop.

La siguiente documentación enumera sintéticamente los pasos a seguir[15] para la confección de un listado completo (formulario listador y reporte) estándar.

> Las partes grisadas de los ejemplos de código están a modo de referencia para indicar la ubicación del código relacionado al ítem de referencia. Las partes resaltadas indican que deberá reemplazarse el contenido por el apropiado en cada caso.
>
> En el caso de las validaciones, estas partes están sólo a modo de ejemplo, ya que no se puede especificar una validación genérica.
>
> Por razones de espacio, algunas líneas de código debieron ser tipeadas en varios renglones.
>
> Casi cualquier porción de código utilizada en esta sección puede ser creada mediante el uso de los *code snippets* incluidos en *StartFrame*.

Parte I: Listador

Deben realizarse las siguientes tareas[16] en el formulario de la capa de usuario:

1 **Agregar la interfáz** (si no existe) para la clase en *StartFrame.IBR.IAbm* con el siguiente esquema, modificando los valores resaltados.

```
Public Interface IRubros
   Inherits IAbm

   'Agregar las declaraciones de funciones que correspondan

End Interface
```

2 **Agregar un nuevo formulario heredado** en la capa de usuario desde *StartFrame.US.Listador*.

3 Realizar la **declaración de la clase**.

```
Imports StartFrame.BR.Interfaces  'Procedencia de las referencias utilizadas
                                          '...(especificar otras si es necesario)
```

[15] Ver **tutorial** para una ilustración paso a paso de las tareas a realizar
[16] Las que figuran con la viñeta en **video inverso** son obligatorias

```
Namespace StartFrame.US.Stock.Listados        'Namespace de la clase
   Public Class Rubros                        'Nombre de la clase
      Inherits StartFrame.US.Listador         'Herencia

      Dim _Rubros As IRubros
      ------------------------------------
      Windows form designer generated code
      ------------------------------------
   End Class
End Namespace
```

4 Modificar el *Text* del **título** del formulario.

5 Pegar los **controles de pantalla** utilizando la clase *StartFrame.LnkControls*. No es necesario colocar etiquetas de título para los rangos, ya que no son necesarias en modo diseño.

6 **Agregar los reportes a la colección y vincular los rangos para cada reporte** modificando el *OnLoadForm* del formulario agregando el siguiente código y modificando los valores resaltados. Tener en cuenta que estas instrucciones son *case sensitive*. El orden de los rangos debe ser el mismo aplicado en el *stored procedure*.

```
Private Sub Auditoria_OnLoadForm(ByRef reportCollection As Reports) _
   Handles MyBase.OnLoadForm

   Try
      'Crea los rangos
      Dim rpt As Report
      Dim rangoRubro As New Rango(1, "cd_rubro", cd_rubro)
      Dim rangoSubRubro As New Rango(2, "cd_subrubro", cd_subrubro)

      'Crea los reportes
      rpt = New Report(1, "TLRubros_rubros", "Rubros", "tlrubros_rubros",
"us.utilitarios.reportes")
      rpt.Rangos.Add(rangoRubro)
      rpt.Rangos.Add(rangoSubRubro)
      reportCollection.Add(rpt)

   Catch ex As Exception
      StartFrame.Us.Display.ShowError(ex.message)
   End Try
End Sub
```

7 **Vincular el formulario de US a la clase de BR** modificando el *OnLoadForm* del formulario agregando el siguiente código y modificando los valores resaltados. Ejemplo:

```
Private Sub Auditoria_OnLoadForm(ByRef reportCollection As Reports) Handles _
MyBase.OnLoadForm

Try
   'Agrega los reportes
   Dim rpt As Report
   Dim rangoRubro As New Rango(1, "cd_rubro", cd_rubro)

   rpt = New Report(1, "TLRubros_rubros", "Rubros", "tlrubros_rubros")
   rpt.Rangos.Add(rangoRubro)
   reportCollection.Add(rpt)

   'Referencia el objeto de negocios relacionado
   _auditoria = CType(RemotingHelper.getObject(GetType(IAuditoria)), IAuditoria)
   Me.ObjetoReglasNegocio = _auditoria
```

```
    Catch ex As Exception
       StartFrame.Us.Display.ShowError(ex.message)
    End Try
End Sub
```

8 Pueden realizarse **validaciones previas o con posterioridad a la carga de datos** para la impresión del formulario. Esto puede hacerse en los eventos *AntesDeCargarDatos y DespuesDeCargarDatos* del formulario. En ambos casos, podrá ponerse el parámetro *cancelar* en true para impedir que se prosiga con la ejecución del informe.

9 Puede ejecutarse **código antes y después de imprimir** en los eventos *OnPrintClick, OnPrintEnd, OnScreenClick,* etc.

Interfaz Web

1 **Agregar un nuevo formulario heredado** en la capa de usuario que herede de *Reporte.Master.*

2 **Agregar un nuevo componente** (clase) en la capa de reglas de negocios. Dentro del mismo componente se pueden crear tantas clases como sea necesario (ver punto 4).

```
Imports System.Data.Oledb
Imports Common.Env
Imports StartFrame.DA.Sql

Public Class UnidadesMonetarias              'Nombre de la clase
   Inherits Abm                              'Herencia
   Implements IUnidadesMonetarias            'Interfaz

   Sub New()
      'Modificar nombre de programa y tabla principal relacionada a la clase
      MyBase.New(Operador, Password, "waunidmon", "wad_unidades_monetarias")

      'Habilita el tracking para esta clase
      Me._Tracking = True

   End Sub
End Class
```

3 Realizar la **declaración de la clase**. En caso de tratarse de un componente para utilizar únicamente desde el front-end Web, no es necesario que implemente ninguna interfaz.

```
Imports StartFrame.BR.Web.Utilitarios

Partial Public Class reporte_Pruebas
   Inherits System.Web.UI.Page

   Public masterpage As Reporte

   Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _
         Handles Me.Load
      'Instancia de la clase masterpage AbmBase
      masterpage = CType(Page.Master, Reporte)
      'Asigna los atributos del AbmBase
      masterpage.lblTitulo = "Registro de Pruebas Realizadas"
      'Instancia del objeto de reglas del negocio
      Dim obr As New BR.Web.Utilitarios.Abms.Pruebas
      'Inicialización de la página Abmbase enlazada al objeto de negocios
      masterpage.InicializarPagina(obr)
   End Sub

   Protected Overloads Sub OnPreInit() Handles Me.PreInit
      'Aplicacion de Tema
      If Not Session("Tema") Is Nothing Then
         Page.Theme = Session("Tema").ToString
      End If
   End Sub
End Class
```

4 En SQL Server crear un *StoredProcedure,* y en el select agregar todos los campos de la tabla del listado.

```
CREATE PROCEDURE [dbo].[listcategent_log_res]
        -- Add the parameters for the stored procedure here
        @id_categ_entidad varchar,
        @id_perfil varchar,
        @descripcion varchar
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

    -- Insert statements for procedure here
        select s.id_categ_entidad, s.id_perfil, s.descripcion
        from categ_entidad s
        where s.id_categ_entidad BETWEEN '000' AND 'zzz' and
                s.id_perfil BETWEEN '000' AND 'zzz' and
                s.descripcion BETWEEN '000' AND 'zzz'
        ORDER BY s.id_categ_entidad
END
```

Parte II: Reporte

Deben realizarse las siguientes tareas[17] en el formulario de la capa de usuario:

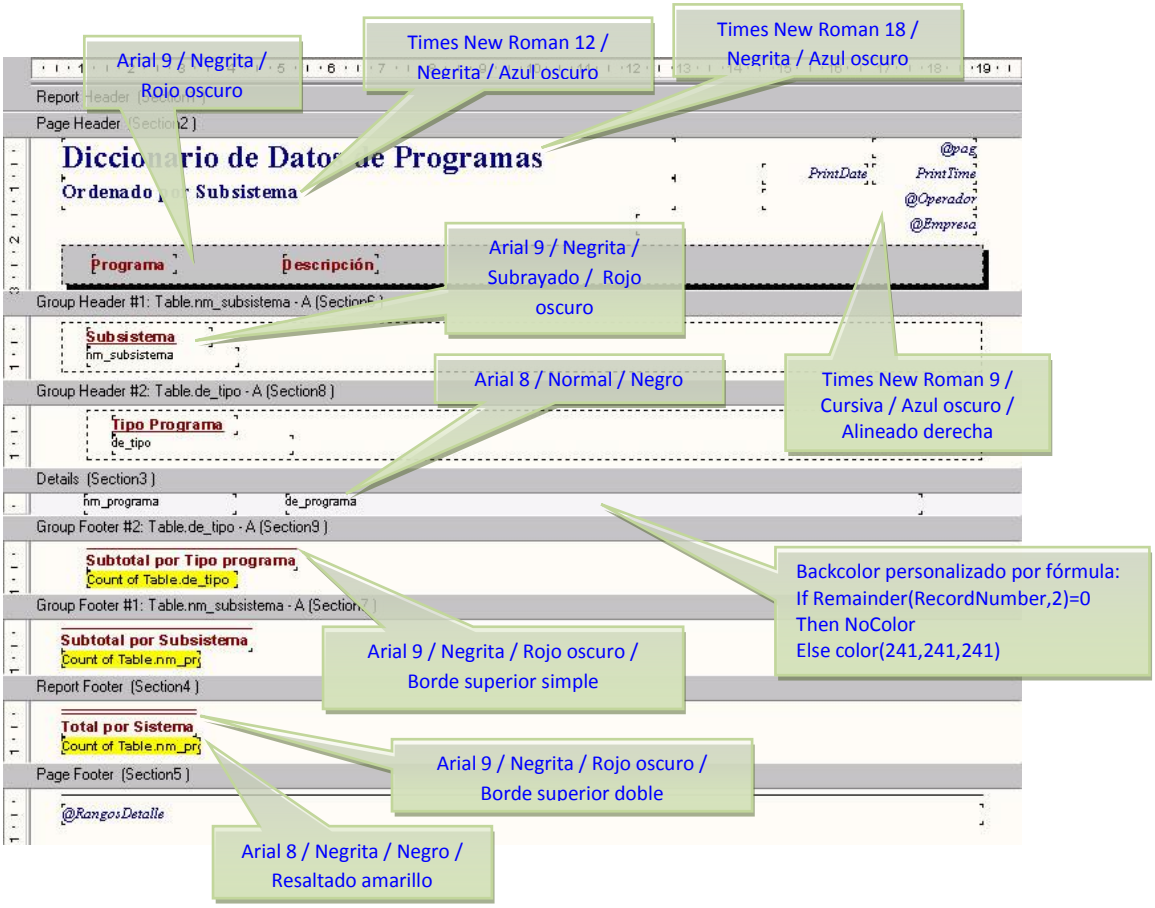
1 **Crear un stored procedure** con el origen de datos. Tener en cuenta que se deben colocar como parámetros del SP los rangos de pantalla del listador en el mismo orden de ubicación que tengan en el formulario.

2 **Generar un archivo XML** ejecutando la opción correspondiente del listador (desde el Sistema).

3 Utilizar el XML generado para **diseñar el reporte**. Para el diseño del reporte es válido copiar un reporte similar (ambos archivos) y guardarlo con el nombre correspondiente, para aprovechar el formato del mismo. Para cambiar el archivo XML vinculado al reporte, es necesario desvincular el anterior (log off) antes de seleccionar el nuevo. Este reporte puede crearse en el assembly indicado en su definición en el listador o bien en el assembly que contiene los reportes a medida del cliente (parámetro *REPORTSCTE*).

[17] Las que figuran con la viñeta en **video inverso** son obligatorias

4 El **formato del reporte** estandar se especifica más abajo a modo de sugerencia.



CONTROL DE CALIDAD

Con el fin de poder asegurar un determinado nivel de control de calidad funcional del sistema, previo al control operativo que el propietario o usuario líder de la aplicación deberá realizar, se **aconseja** proceder según se explica en este capítulo. Este procedimiento de control de calidad (QA) del código es de carácter orientativo y el mismo puede ser modificado sin afectar en modo alguno la programación realizada con *StartFrame*. Sin embargo, el mismo tiene en cuenta los diferentes aspectos del producto y las mejores prácticas en cuanto a evitar los errores u omisiones que se cometen con mayor frecuencia en este tipo de desarrollos. Cabe destacar que es posible realizar *unit test* dentro de *StartFrame* siguiendo las indicaciones del *Manual Técnico*.

Se destaca que este proceso de control de calidad es independiente de la arquitectura de la solución, pero en algunos casos serán aplicables ciertos controles que en otros casos serían irrelevantes o inaplicables.

Existen tres niveles de control de calidad funcional:

- ✓ **Controles básicos**: son llevados a cabo por el **programador** que desarrolló la funcionalidad. Consiste en una serie de controles fundamentales de nivel general que varían según el tipo de programa (ver más adelante).
- ✓ **Controles específicos**: son controles puntuales que se enumeran explícitamente en la planilla de especificación de diseño (si hubiera). Son realizados por un **programador en jefe**, **testeador** o **analista**. Son controles particulares de cada programa.
- ✓ **Control general**: es realizado por el **analista funcional** o el **líder del proyecto**. Consiste en un control general de todos los aspectos del programa y su vinculación con el entorno.

Controles Básicos

Antes de dar por finalizado el desarrollo de cualquiera de los tipos de programas mencionados, deberán realizarse una serie de controles, los cuales varían dependiendo de la arquitectura utilizada y del tipo de programa desarrollado. Dichos controles son de dos aspectos: visuales y funcionales. Los primeros apuntan a la interfaz en general. Los últimos se refieren al manejo básico de cada programa.

Es importante que se entienda que cada vez que se modifica un programa deberán realizarse nuevamente **todos los controles básicos**.

Abms

Para realizar estos controles es necesario cargar al menos tres registros con diferentes datos. Deberá controlarse que:

- ✓ Todos los controles estén alineados correctamente.
- ✓ El orden de tabulación sea el apropiado.
- ✓ El label de cada control (incluso las grillas) sea el apropiado según su tipo (clave, obligatorio, común).
- ✓ Presionando el botón de ayuda se llame a la documentación de ayuda en línea apropiada, la cual deberá estar completa y contener todos y cada uno de los controles en pantalla.
- ✓ Todos los campos vinculados (incluso en las grillas) posean ventana de apoyo y sus carteles

descriptivos relacionados, los cuales deberán actualizarse correctamente con el movimiento entre registros (incluyendo los registros de las grillas).

- ✓ No se pueda dar un alta con una clave en blanco.
- ✓ No se permita grabar con alguno de los campos obligatorios en blanco.
- ✓ Se permita grabar completando tan sólo la clave y los campos obligatorios.
- ✓ Se grabe correctamente, tanto en la tabla principal como en las vinculadas, tanto en altas y modificaciones como en bajas.

Listados

Para realizar estos controles es necesario que se cargue información en las tablas de origen de los informes. Es muy raro que se termine primero el informe y luego el Abm, sin embargo, en este último caso, deberán cargarse las tablas a mano con los registros que se consideren adecuados (consultar con el analista funcional).

En cuanto al **listador**, deberá controlarse que:

- ✓ Todos los controles estén alineados correctamente.
- ✓ El orden de tabulación sea el apropiado.
- ✓ Los diferentes rangos para cada reporte sean los que se indican en la especificación (ni más ni menos).
- ✓ El listador posea tantos reportes con tantos ordenamientos como se especificó.
- ✓ Presionando el botón de ayuda se llame a la documentación de ayuda en línea apropiada, la cual deberá estar completa y contener todos y cada uno de los rangos a utilizar y los reportes contenidos en el listador.
- ✓ Todos los campos vinculados posean ventana de apoyo aunque su ingreso sea libre y sin validación.
- ✓ Los valores por defecto de cada rango permitan listar todo el contenido de los reportes, de principio a fin de archivo.
- ✓ Los procesos de armado, especialmente los complicados, estén detalladamente documentados.

En cuanto a cada uno de los **reportes**, deberá controlarse que:

- ✓ Se haya respetado el formato general del informe (tipografía, colores, posiciones, etc.).
- ✓ Imprima información coherente con el origen de datos especificado, tanto en relación al contenido de los campos como a la cantidad de registros (esto es por si el SQL de armado retornó más registros de los debidos por una unión incorrecta con otra tabla).
- ✓ El orden de los registros sea el especificado.
- ✓ Los agrupamientos funcionen correctamente.
- ✓ Existan todos y cada uno de los campos estipulados en la especificación.
- ✓ Se hayan respetado todas las observaciones a los campos de cada reporte que se hicieron en la especificación.
- ✓ Los campos se encuentren alineados apropiadamente según su tipo de dato.
- ✓ Existan subtotales por cada agrupamiento y totales finales de los campos numéricos que expresen cantidades o importes.

- ✓ Cada uno de los rangos sean tomados en cuenta.

Consultas

Deberá controlarse que:

- ✓ Figuren en pantalla todos y cada uno de los controles determinados en la especificación.
- ✓ El orden de la información sea el especificado.
- ✓ Las grillas y los campos vinculados se actualicen apropiadamente con el movimiento entre registros.
- ✓ Los datos estén correctamente alineados.
- ✓ Presionando el botón de ayuda se llame a la documentación de ayuda en línea apropiada.

Procesos

Deberá controlarse que:

- ✓ Todos los controles estén alineados correctamente.
- ✓ El orden de tabulación sea el apropiado.
- ✓ Presionando el botón de ayuda se llame a la documentación de ayuda en línea apropiada.
- ✓ Todos los campos vinculados posean ventana de apoyo y sus carteles descriptivos relacionados.
- ✓ El proceso esté detalladamente documentado.
- ✓ No se permita confirmar el proceso con algún rango vacío (o inválido) que sea de carácter obligatorio.
- ✓ El proceso ponga carteles indicativos del estado de avance del mismo y no parezca que se colgó la máquina.
- ✓ El proceso termine correctamente y su resultado sea coherente con su finalidad, completando todas y cada uno de las tablas que correspondan.
- ✓ Se informen apropiadamente todos los errores ocurridos durante el proceso, con la información suficiente para que el operador pueda resolverlos.
- ✓ Se contemplen todas las validaciones posibles (incluso las que aparentemente no se puedan dar). En otras palabras, no dejar ningún *IF* sin su correspondiente *ELSE*.

Controles Avanzados

Estos controles figuran en la planilla de especificación de diseño. Los mismos, aunque se llaman avanzados, deben efectuarse con anterioridad a los controles básicos, ya que estos controles avanzados son algunas veces muy complejos y pueden ocasionar que los controles básicos que en un principio funcionan correctamente, dejen de hacerlo.

Prestar especial atención de no tildar con demasiada ligereza estos controles, sino asegurarse adecuadamente de su correcto funcionamiento.

Control General

Este control es ejercido por un analista funcional o bien por el líder del proyecto. Es muy importante realizar este trabajo a conciencia y tomarse su debido tiempo, ya que es el último control antes que el módulo llegue al usuario. El mecanismo de control varía dependiendo del tipo de programa y la arquitectura de la solución.

Abms

Debe contemplar:

- ✓ Aspecto visual en general: nombres de los labels, disposición de los controles, mensajes de estado, carteles relacionados.
- ✓ Aspecto funcional elemental: orden de tabulación, ventanas de apoyo, validaciones simples.
- ✓ Aspecto funcional avanzado: controles al azar de las validaciones especificadas en la planilla de diseño, manejo de las grillas.
- ✓ Aspecto funcional global: coherencia de la información con el contexto general del sistema.
- ✓ Ayuda en línea: verificación de su correcta vinculación y corrección o ampliación de la documentación desarrollada.

Listados

Debe contemplar:

- ✓ Existencia de todos los informes enumerados en la planilla.
- ✓ Rangos en general: todos y c/u de los indicados en la planilla según cada informe, valores por defecto, mensajes de estado, orden de tabulación, ventanas de apoyo.
- ✓ Aspecto funcional de cada reporte: que respete los rangos, que el orden sea el indicado, que los totales correspondan a campos apropiados, que los agrupamientos funcionen apropiadamente.
- ✓ Aspecto visual de cada reporte: títulos y encabezados, alineación, valores repetidos, campos numéricos, aspecto general.
- ✓ Ayuda en línea: verificación de su correcta vinculación y corrección o ampliación de la documentación desarrollada.

Consultas

Deberá contemplar:

- ✓ Disposición de los controles.
- ✓ Títulos de los labels.
- ✓ Actualización de los campos vinculados y las grillas.
- ✓ Ayuda en línea: verificación de su correcta vinculación y corrección o ampliación de la documentación desarrollada.

Procesos

Deberá contemplar:

- ✓ Disposición de los controles.
- ✓ Títulos de los labels.
- ✓ Ventanas de apoyo.
- ✓ Validaciones en línea.
- ✓ Ayuda en línea: verificación de su correcta vinculación y corrección o ampliación de la documentación desarrollada.
- ✓ Estructura interna del código del proceso.
- ✓ Lote de prueba para verificar su apropiado funcionamiento dentro del entorno.