MAG

MINISTERIO DE AGRICULTURA Y GANADERIA
COSTA RICA

Estándares de Programación

Estándares de Sistemas

Estos estándares deben considerarse como guías en las etapas de diseño de los sistemas.

Las técnicas efectivas de manejo y control de proyectos combinados con una participación activa de los usuarios y la utilización de metodologías estructuradas de desarrollo de sistemas, pueden minimizar riesgos de incumplimiento de fechas de actividades importantes, de gastos excesivos en relación a los costos estimados e insatisfacciones de los usuarios de los sistemas.

Arquitectura de Software

Comprende la definición y organización de los elementos arquitectónicos de una solución de software, sus interacciones y sus restricciones.

Dentro de las tareas a realizar dentro de la arquitectura se encuentran:
Definición de los diferentes componentes que integran la solución propuesta, analizando las diferentes interacciones que se presentan entre los mismos.
Análisis de componentes pre-existentes en el mercado, que pueden ser reutilizados en la arquitectura que se propone.
Análisis del modelo propuesto desde diferentes perspectivas, para validar características tales como performance, concurrencia y aspectos que deberá tener el sistema al ser implementado en redes de comunicaciones.
Definir el patrón arquitectónico que logre implementar adecuadamente las características funcionales y no funcionales definidas durante la toma de requerimientos.

Patrón Arquitectónico Arquitectura de 3 Capas

El patrón arquitectónico en 3 capas es una especialización del patrón “Aplicación en Capas”. Esto define que la solución debe de dividirse en tres capas lógicas: presentación, lógica de negocio y acceso a datos, con esto se busca especificar el conjunto de responsabilidades de cada una de las capas y los componentes que los conformarán.

Presentación: debe provee la interfaz para el usuario. Las tecnologías que se utilizan en .NET son Win Forms para aplicaciones de cliente inteligente y ASP.NET para aplicaciones web.

Lógica de Negocio: implementa la funcionalidad de la lógica de negocio. Esta capa está conformada de componentes desarrollados en cualquiera de los lenguajes de .NET

Acceso a Datos: provee el acceso a los repositorios de información persistente y/o sistemas externos. La tecnología utilizada para desarrollar esta capa es ADO.NET, se pueden también utilizar stored procedures y XML.

Nombres básicos para módulos de formulario y de clase

En lugar de utilizar tan sólo un nombre, como Clientes, podría preferir el identificar al formulario como frmClientes o a la clase como clsClientes. Por añadidura, tal como lo ve, la denominación incluye el uso de altas y bajas (mayúsculas y minúsculas) para identificar las diferentes palabras que conforman el nombre (en lo que se ha dado en llamar: notación camello). Es decir, es más fácil leer: frmVentasEnTotal, que: FRMVENTASENTOTAL o frmventasentotal (internamente), externa –

Con respecto al desarrollo de clases para los proyectos estas deben de ser creadas bajo el prefijo de “cls”, quedando de esta forma el nombre “clsConexion”. Dentro de estas clases deben de poseer una pequeña y breve explicación sobre cada uno de los procedimientos que la misma contenga.

Nombres básicos para los controles

Se utilizara el nombre del resultado o dato que se desplegará en el control. Es decir, si al leer alguna información de una base de datos ésta se desplegará en algún cuadro de texto (en la suposición de que el dato sea un apellido paterno), el cuadro se podría llamar apellidoPaterno, apPaterno, con el prefijo “txt”, quedando de esta forma “txtApellidoPaterno”.

Si se desea utilizar WebUserControl o Control de Usuario Web para configurar y crear controles programables y reutilizables se utilizará el nombre con el prefijo “wuc”, quedando en forma “wucControlAcceso”.
También se puede hacer uso de los archivos de recursos, para configurar los datos que sean necesarios, el mismo contendrá el prefijo ”rs” quedando de forma “rsRecurso”.

Nombres básicos para los módulos de código

Los módulos de código (también conocidos como módulos .aspx.vb, módulos de código plano, módulos Basic o tan sólo módulos). Los desarrolladores deben poder determinar el tipo de procedimientos que están contenidos en el módulo con tan sólo leer el nombre. Así, podría usar nombres como: modRutinasAPI_DelTechNet y modRutinasAPI_Biblioteca.

Nombres básicos para los procedimientos

Este es un punto importante en las tareas de definición de nombres. Sin embargo poco a poco se ha convertido en una buena norma el indicar una descripción precisa (y sin embargo corta) del procedimiento que se refiera. A estos nombres puede antecedérseles un prefijo sub o fnc para distinguir a los procedimientos Sub de los Function.

No obstante, se deberá anteceder con las letras do, get y set para indicar si un procedimiento tan sólo ejecutará una tarea (do), obtendrá información de alguna fuente (get) o la grabará (set). En algunos casos, do es sustituido por let, por lo que la ejecución de una tarea podría llamarse letMultiplicacion o doMultiplicacion.

Convenciones de nombres para objetos

Los objetos deben llevar nombres con un prefijo coherente que facilite la identificación del tipo de objeto. A continuación se ofrece una lista de convenciones recomendadas para algunos de los objetos permitidos por Visual Basic en general.

Prefijos sugeridos para controles

Tipo de control	Prefijo	Ejemplo
Panel 3D	Pnl	pnl_Grupo
Botón animado	Ani	ani_Buzon
Casilla de verificación	Chk	chk_SoloLectura
Cuadro combinado, cuadro de lista desplegable	Cmb	cmb_Ingles
Botón	Btn	btn_Ejemplo
LinkButton	Lnk	lnk_Ejemplo
HyperLink	Hyp	hyp_Ejemplo
DropDownList	Ddl	ddl_Ejemplo
Repeater	Rep	rep_Ejemplo
RadioButton	Rdo	rdo_Ejemplo
GroupBox	Grp	grp_Ejemplo
CheckedListBox	Clst	Clst_Ejemplo
DataTimePicker	Dtp	Dtp_Ejemplo
MonthCalendar	Cal	Cal_Ejemplo
Splitter	Spl	Spl_Ejemplo
DomainUpDown	Dup	Dup_Ejemplo

NumericUpDown	Nup	Nup_Ejemplo
TrackBar	Trk	Trk_Ejemplo
HelpProvider	Hlp	Hlp_Ejemplo
ToolTip	Tip	Tip_Ejemplo
ContextMenu	Cmnu	Cmnu_Ejemplo
Calendar	Cal	Cal_Ejemplo
AdRotator	Ad	Ad_Ejemplo
DataTimePicker	Dtp	Dtp_Ejemplo
NotifyIcon	Nic	Nic_Ejemplo
OpenFileDialog	Ofd	Oft_Ejemplo
SaveFileDialog	Sfd	Sfd_Ejemplo
FontDialog	Fd	Fd_Ejemplo
ColorDialog	Cd	Cd_Ejemplo
PrintDialog	Pd	Pd_Ejemplo
PrintPreviewDialog	Ppd	Ppd_Ejemplo
PrintPreviewControl	Ppc	Ppc-Ejemplo
ErrorProvider	Errp	Errp_Ejemplo
PrintDocument	Pdoc	Pdoc_Ejemplo
PageSetupDialog	Psd	Psd_Ejemplo
CrystalReportViewer	Crv	Crv_Ejemplo
Diálogo común	Dlg	Dlg_ArchivoAbrir
Comunicaciones	Com	Com_Fax
Control (dentro de procedimientos cuando no se conoce el tipo específico)	Ctr	Ctr_Activo
Control de datos	Dat	Dat_Biblio

Cuadro combinado enlazado a datos	Dbcmb	Dbcmb_Lenguaje
Cuadrícula enlazada a datos	Dbgrd	Dbgrd_ResultadoConsulta
Cuadro de lista enlazado a datos	Dblst	Dblst_TipoTarea
Cuadro de lista de directorios	Dir	Dir_Origen
Cuadro de lista de unidades	Drv	Drv-Destino
Cuadro de lista de archivos	Fil	Fil-Origen
Formulario	Frm	Frm_Entrada
Marco	Fra	Fra_Lenguaje
Medidor	Gau	Gau_Estado
Gráfico	Grf	Grf_Ingresos
Cuadrícula	Grd	Grd_Precios
Barra de desplazamiento horizontal	Hsb	hsbVolumen
Imagen (Image)	Img	imgIcono
Estado de tecla	Key	keyMayusculas
Etiqueta	Lbl	lblMsjAyuda
Línea	Lin	linVertical
Cuadro de lista	Lst	lstCódigosDePolítica
Mensaje MAPI	Mpm	mpmEnviarMsj
Sesión MAPI	Mps	mpsSesion
MCI	Mci	mciVideo
Formulario MDI secundario	Mdi	mdiNota
Menú	Mnu	mnuArchivoAbrir
MS Flex Grid	Fgd	fgdClientes
MS Tab	Mst	mstPrimero
Actives	Ole	oleHojaDeTrabajo

Esquema	Out	outDiagramaDeOrg
Pen BEdit	Bed	bedNombre
Pen Hedit	Hed	hedFirma
Trazo de pluma	Ink	inkMapa
Imagen (Picture)	Pic	picVGA
Clip de imagen	Clp	clpBarraDeHerramientas
Informe	Rpt	rptGananciasTrimestre1
Forma	Shp	shpCirculo
Cuadro de número	Spn	spnPaginas
Cuadro de texto	Txt	txtApellido
Cronómetro	Tmr	tmrAlarma
Arriba-abajo	Upd	updDireccion
Barra de desplazamiento vertical	Vsb	vsbVelocidad
Control deslizante	Sld	sldEscala
Lista de imágenes	Ist	ilstTodosLosIconos
Vista de árbol	Tre	treOrganizacion
Barra de herramientas	Tlb	tlbAcciones
TabStrip	Tab	tabOpciones
Barra de estado	Sta	staFechaHora
Lista	Lvw	lvwEncabezados
Barra de progreso	Pqb	pgbCargarArchivo
RichTextBox	Rtf	rtfInforme
GridView	Gdv	gdvMostrarDatos
DataList	Dl	dlCargarDatos
FormView	Fv	fvListas
SqlDataSource	Sds	sdsConexion
XmlDataSource	Xds	xdsXmlCarga

ReportViewer	Rw	rwReportes
RequeridFieldValidator	Rfv	rfvValidacion
RangeValidator	Rv	rvRangoValidacion
RegularExpressionValidator	Rev	revValidacionRegular
Login	Lg	lgLogin
LoginView	Lv	lvVistaUsuario

Componentes AJAX

Utilización de componentes ajax (acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML)), para el desarrollo de aplicaciones interactivas. Permitiendo que las aplicaciones que utilicen componentes ajax, se ejecuten en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

ASP.NET AJAX Control Toolkit

Tipo de control	**Prefijo**	**Ejemplo**
Accordion	acc	Acc_Ejemplo
AlwaysVisibleControl	ace	ace_Ejemplo
Animation	ae	ae_Ejemplo
AutoComplete	auc	auc_Ejemplo
Calendar	cal	cal_Ejemplo
CascadingDropDown	cdd	cdd_Ejemplo
CollapsiblePanel	Cpe	cpe_Ejemplo
ConfirmButton	Cbe	cbe_Ejemplo

DragPanel	dpe	dpe_Ejemplo
DropDown	dde	dde_Ejemplo
DropShadow	dse	dse_Ejemplo
DynamicPopulate	dp	dp_Ejemplo
FilteredTextBox	ftbe	ftbe_Ejemplo
HoverMenu	hme	hme_Ejemplo
ListSearch	lse	lse_Ejemplo
MaskedEdit	mee	mee_Ejemplo
ModalPopup	mpe	mpe_Ejemplo
MultiHandleSlider	mhse	mhse_Ejemplo
MutuallyExclusiveCheckBox	mecbe	mecbe_Ejemplo
NoBot	nbe	nbe_Ejemplo
NumericUpDown	nud	nud_Ejemplo
PagingBulletedList	pble	pble_Ejemplo
PasswordStrength	ps	ps_Ejemplo
PopupControl	popex	popex_Ejemplo
Rating	rat	rat_Ejemplo
ReorderList	rle	rle_Ejemplo
ResizableControl	rce	rce_Ejemplo
RoundedCorners	rcoe	rcoe_Ejemplo
Slider	sle	sle_Ejemplo
SlideShow	slse	slse_Ejemplo
Tabs	tabe	tabe_Ejemplo
TextboxWatermark	tbwe	tbwe_Ejemplo
ToggleButton	tbe	tbe_Ejemplo
UpdayePanelAnimation	upae	upae_Ejemplo
ValidatorCallout	PNReqE	PNReqE_Ejemplo

Objetos de ADO.NET

Aunque hay miles de objetos disponibles como parte de .NET, es probable que se use ADO.NET como parte de las aplicaciones, por lo tanto algunos estándares para nombrar los objetos de ADO.NET más comunes. A continuación se listan los prefijos que se utiliza:

Clase	Prefijo
DataSet	Ds
DataTable	Dt
DataView	Dv
DataRow	Drw
Connection*	Cnn
Command*	Cmd
DataAdapter*	Da
CommandBuilder*	Bld
DataReader*	Dr

Ejemplos: de declaración de los objetos ADO.net

```
Dim drEmps As New SqlDataReader()
Dim drCust As New SqlDataReader()
Dim dsEmps As DataSet
Dim dsCust As DataSet
```

Convenciones de nombres de constantes y variables

Las variables se deben definir siempre con el menor alcance posible. Las variables globales (públicas) pueden crear máquinas de estado enormemente complejas y hacer la lógica de una aplicación muy

difícil de entender. Las variables globales también hacen mucho más difícil mantener y volver a usar el código.

Además, los argumentos se deben pasar a los procedimientos Sub y Function mediante **ByVal**, a menos que sea necesario explícitamente cambiar el valor del argumento que se pasa.

Prefijos de alcance de variables

A medida que aumenta el tamaño del proyecto, también aumenta la utilidad de reconocer rápidamente el alcance de las variables. Esto se consigue al escribir un prefijo de alcance de una letra delante del tipo de prefijo propio, sin aumentar demasiado la longitud del nombre de las variables.

Alcance	**Prefijo**	**Ejemplo**
Global	G	gstrNombreUsuario
Nivel de módulo	M	mblnProgresoDelCálculo
Local del procedimiento	Ninguno	dblVelocidad
Público	P	pCantidadUsuario

Una variable tiene alcance global si se declara como **Public** en un módulo estándar o en un módulo de formulario. Una variable tiene alcance de *nivel de módulo* si se declara como **Private** en un módulo estándar o en un módulo de formulario, respectivamente.

Constantes

El cuerpo del nombre de las constantes se debe escribir en mayúsculas. Aunque las constantes estándar de Visual Basic.Net no incluyen información del alcance y el tipo de datos (estos en minúscula), los prefijos como i, s, g y m pueden ser muy útiles para entender el valor y el alcance de una constante. Para los nombres de constantes, se deben seguir las mismas normas que para las variables. Por ejemplo:

MINTMAXLISTAUSUARIO ' Límite de entradas máximas para
' la lista de usuarios (valor
' entero, local del módulo)
GSTRNUEVALINEA ' Carácter de nueva línea
'(cadena, global de la ' aplicación)

Variables

Declarar todas las variables ahorra tiempo de programación porque reduce el número de errores debidos a erratas (por ejemplo, aNombreUsuarioTmp frente a sNombreUsuarioTmp y frente a sNombreUsuarioTemp).
Las variables deben llevar un prefijo para indicar su tipo de dato. Opcionalmente, y en especial para programas largos, el prefijo se puede ampliar para indicar el alcance de la variable.

Tipos de datos de variables

Use los prefijos siguientes para indicar el tipo de datos de una variable.

Tipo de datos	**Prefijo**	**Ejemplo**
Bolean	B	b_Encontrado
Byte	By	by_DatosImagen
Objeto Collection	Col	col_Datos
Currency	Cur	cur_Ingresos
Date (Time)	Dt	dt_Inicio
Double	dbl	dbl _Tolerancia
Error	Err	Err_NúmDeOrden
Integer	I	i_Cantidad
Long	L	l_Distancia
Object	obj	obj_Activo

Single	sng	sng_Media
String	str	str_NombreF
Tipo definido por el usuario	udt	udt_Empleado
Variant	vnt	vnt_CheckSum
Short	srt	Srt_Value
Decimal	dec	Dec_Value
Char	chr	Chr_Letter

Nombres descriptivos de variables y procedimientos

El cuerpo de un nombre de variable o procedimiento se debe escribir en mayúsculas y minúsculas y debe tener la longitud necesaria para describir su funcionalidad. Además, los nombres de funciones deben empezar con un verbo, como IniciarNombreMatriz o CerrarDialogo.

Para nombres que se usen con frecuencia o para términos largos, se recomienda usar abreviaturas estándar para que los nombres tengan una longitud razonable. En general, los nombres de variables con más de 32 caracteres pueden ser difíciles de leer.

Cuando se usen abreviaturas, hay que asegurarse de que sean coherentes en toda la aplicación. Alternar aleatoriamente entre Cnt y Contar dentro de un proyecto provoca una confusión innecesaria. Por añadidura, aunque los nombres de variables y procedimientos en Visual Basic 3.0 y superiores soportan el uso de caracteres acentuados y eñes, estos no deben de ser utilizados.

Tipos definidos por el usuario

En un proyecto grande con muchos tipos definidos por el usuario, suele ser útil dar a cada uno de estos tipos un prefijo de tres caracteres sólo suyo. Si estos prefijos comienzan con "u", será fácil reconocerlos cuando se esté trabajando con tipos definidos por el usuario. Por ejemplo, "ucli" se podría usar como prefijo para las variables de un tipo Cliente definido por el usuario.

Convenciones de codificación estructurada

Además de las convenciones de nombres, las convenciones de codificación estructurada, como comentarios al código y sangrías coherentes, pueden mejorar mucho la legibilidad del código.

Convenciones de comentarios al código

Todos los procedimientos y funciones deben comenzar con un comentario breve que describa las características funcionales del procedimiento (qué hace). Esta descripción no debe describir los detalles de implementación (cómo lo hace), porque a veces cambian con el tiempo, lo que da como resultado un trabajo innecesario de mantenimiento de los comentarios o, lo que es peor, comentarios erróneos. El propio código y los comentarios de líneas necesarios describirán la implementación.

Los argumentos que se pasan a un procedimiento se deben describir cuando sus funciones no sean obvias y cuando el procedimiento espera que los argumentos estén en un intervalo específico. También hay que describir, al principio de cada procedimiento, los valores de retorno de funciones y las variables globales que modifica el procedimiento, en especial los modificados a través de argumentos de referencia.

Los bloques del comentario de encabezado del procedimiento deben incluir los siguientes encabezados de sección.

Encabezado de sección	**Descripción del comentario**
Finalidad	Lo que hace el procedimiento (no cómo lo hace).
Premisas	Lista de cada variable externa, control, archivo abierto o cualquier otro elemento que no sea obvio.
Efectos	Lista de cada variable externa, control o archivo afectado y el efecto que tiene (sólo si no es obvio).
Entradas	Todos los argumentos que puedan no ser obvios. Los argumentos se escriben en una línea aparte con comentarios de línea.
Resultados	Explicación de los valores devueltos por las funciones.

Se tiene que tomar en cuenta:

Cada declaración de variable importante debe incluir un comentario de línea que describa el uso de la variable que se está declarando.
Las variables, controles y procedimientos deben tener un nombre bastante claro para que los comentarios de línea sólo sean necesarios en los detalles de implementación complejos.

Dar formato al código

Como muchos programadores usan todavía pantallas VGA, hay que ajustarse al espacio de la pantalla en la medida de lo posible y hacer que el formato del código siga reflejando la estructura lógica y el anidamiento. Estos son algunos indicadores:

Los bloques anidados estándar, separados por tabuladores, deben llevar una sangría de dos espacios (predeterminado) como mínimo.
Los comentarios deben ser alineados todos iguales en la medida de lo posible
El comentario del esquema funcional de un procedimiento debe llevar una sangría de un espacio. Las instrucciones de nivel superior que siguen al comentario del esquema deben llevar una sangría de un tabulador, con cada bloque anidado separado por una sangría de un tabulador adicional. Por ejemplo en vb 6, .net 2002 y 2003:

```
'*************************************************
' Finalidad: Ubica el primer caso encontrado de un
' usuario especificado en la matriz ListaUsuario.
' Entradas:
' strListaUsuario(): lista de usuarios para buscar.
' strUsuarioDest: nombre del usuario buscado.
' Resultados: Índice del primer caso de rsUsuarioDest
```

```
' encontrado en la matriz rasListaUsuario.
' Si no se encuentra el usuario de destino, devuelve -1.
'*************************************************
Function intBuscarUsuario (strListaUsuario() As String, strUsuarioDest As _
String)As Integer
Dim i As Integer ' Contador de bucle.
Dim blnEncontrado As Integer ' Indicador de
' destino encontrado.
intBuscarUsuario = -1
i = 0
While i <= Ubound(strListaUsuario) and Not blnEncontrado
If strListaUsuario(i) = strUsuarioDest Then
blnEncontrado = True
intBuscarUsuario = i
End If
Wend
End Function
```

Si se utiliza la versión Visual Studio 2005 utilizar el formato preestablecido para la descripción de las funciones. Ejemplo:

```
''' <summary>
'''
''' </summary>
''' <param name="sender"></param>
''' <param name="e"></param>
''' <remarks></remarks>
Protected Sub btnCargar_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
btnCargar.Click
```

... codigo

End Sub

Operadores & y +

Se debe de utilizar el operador **&** para unir cadenas y el operador **+** cuando trabaje con valores numéricos. El uso del operador **+** para concatenar puede causar problemas cuando se opera sobre dos variables **Variant**. Por ejemplo:

vntVar1 = "10.01"

vntVar2 = 11

vntResultado = vntVar1 + vntVar2 'vntResultado = 21.01

vntResultaod = vntVar1 & vntVar2 'vntResultado = 10.0111

Crear cadenas para MsgBox, InputBox y consultas SQL

Cuando esté creando una cadena larga, use el guión bajo para la continuación de línea romper en múltiples líneas de código una línea lógica, de forma que pueda leer o depurar la cadena fácilmente. Esta técnica es especialmente útil cuando se muestra un cuadro de mensaje (**MsgBox**), un cuadro de entrada (**InputBox**), o cuando se crea una cadena SQL. Por ejemplo:

Dim Msj As String

Msj = "Esto es un párrafo que estará en un" _

& " cuadro de mensajes. El texto está separado en" _

& " varias líneas de código en el código de origen, " _

& "lo que facilita al programador la tarea de leer y depurar."

MsgBox Msj

Dim CTA As String

CTA = "SELECT *" _

```
& " FROM Título" _
& " WHERE [Fecha de publicación] > 1988"
ConsultaTitulos.SQL = CTA
```

Otros Aspectos:

Siempre al principio de los procesos que se sepa tomaran algún tiempo en ser ejecutados deberá colocarse el puntero del mouse en vbhourglass y al final del proceso regresar a vbnormal, esto con el fin de hacer saber al usuario de los sistemas que el programa está trabajando.

Nunca se deben de enviar nulos a los campos de las bases de datos

Siempre que una cadena o texto sea extraída de recordset o fields, deberá ser pasado por la función TRIM antes de ser enviada a lista, grids o textbox.

Siempre que existan lista o grids estos podrán ser ordenados por el usuario, con un simple click en el encabezado de la columna respectiva.

En caso de sea necesario mostrar los consecutivos a los usuarios, se creara el consecutivo justo antes de enviarlo a la base de datos y posteriormente se le mostrara al usuario, esto se debe hacer siempre y cuando el consecutivo NO sea auto numérico

En caso de enviar los registros por cadena de caracteres a la base de datos (construcción de sentencias SQL):

Las cajas de textos o textbox de tipo numérico nunca aceptaran letras, o signos especiales, ni comas, y solamente aceptaran un punto (.), como separador de decimales.

Al enviar datos numéricos a la base de datos, todos deberán ser formateados en su forma más básica (#####.##), sin separadores de miles y con dos decimales, se permiten excepciones en los casos en que sea necesario guardar mas decimales.

Recomendaciones:

Siempre que un dato sea extraído de un recordset o fields, se recomienda que sea examinado en busca de NULL.

En la medida de lo posible se recomienda no enviar datos a la base de datos con cadenas de longitud cero (‘’), por cuanto los usuarios podrían aducir olvido de digitación de esta información, en percance del sistema o de las consultas o datos resultantes

Apoyo apropiado del teclado

Para cualquier aplicación que requiera entrada de los datos cabeza-abajo, ésta es probablemente una regla buena para tener en cuenta. Todos las órdenes del menú necesitan tener un teclado accesible (con llaves mnemónicas o atajos del teclado), y todos las órdenes de sistema deben estar disponibles en el menú.

Generación de texto al usuario

Mensajes del texto, sobre todo informativo y diálogos de advertencia, necesitan ser llevados de forma apropiada y consistente. Aquí están algunas reglas:

Evite jerga técnica.

Limite los mensajes a 2-3 líneas.

Evite redacción que culpe al usuario.

Evite el uso de abreviaturas.

Los mensajes se alinearan a la Izquierda cuando sean de líneas múltiples.

Nunca, incluya faltas de ortografía, errores de gramática o pronunciación incorrecta.

Uso del conjunto de caracteres apropiado

A continuación se proporciona una guía simple del conjunto de caracteres y uso apropiado, adaptado ligeramente para la costumbre los proyectos.

Texto	Font (letra)
Title bars, menu text	Windows System font (10 point Sans Serif)

Controles, labels y captions	8 point bold Sans Serif
Campos de entrada de datos	8 point non-bold Sans Serif
Textos de los Status bar	10 point non-bold Sans Serif
Textos de los iconos	8 point non-bold Sans Serif

La interface con controles incoherentemente etiquetados (textos, combos, listas, option buttons, grupos, y otros) puede hacer tanto como cualquier otro factor para que el sistema sea rechazado por parte de los usuarios.

Ventanas y Diálogos

En la medida de lo posible se deben de utilizar los recursos proporcionados por la misma plataforma, tales como las ventanas de dialogo de guardar y abrir archivos ó de impresora.

Los mensajes (msgbox, messagebox) a pantalla, deben estar siempre titulados con el mismo caption de la ventana que los emite o en su defecto por una palabra o frase NO técnica que identifique al proceso o modulo, de fácil lectura por parte de los usuarios.

Menus

Su forma y colores deben de ser siempre consistentes en todos los sistemas, los ítem a seleccionar deben tener teclas de acceso rápido, en caso de utilizar iconos o imágenes en los menús estos deben ser consistentes, en todo el sistema y serán agregados tras previa discusión y puesta en estándares por el grupo de desarrolladores de la institución.

Para cualquiera de los tipos de menús, los textos deberán comenzar con un verbo, preferiblemente en infinitivo y que sea suficientemente descriptivo de la acción que realizara

El formato básico para los menús pop-up (emergente o contextual) , será como sigue:

El 1er Ítem será el titulo del menú, este debe ser escrito en mayúscula

Un separador

Ítems seleccionables (opciones)

Arquitectura de Desarrollo

Para el desarrollo de las nuevas aplicaciones de desarrollo se utilizará una arquitectura de 3 capas.

Capa Presentación: Provee la interfaz del usuario. Las tecnologías que se utilizan en .Net son win forms para aplicaciones de cliente inteligente y ASP.Net para aplicaciones web.

Capa Lógica de Negocios: implementa la funcionalidad de la lógica de negocio. Esta capa está compuesta por componentes desarrollados en cualquier lenguaje de .NET. Generalmente se utiliza COM+ (Enterprise Services) para brindar el soporte transaccional, escalabilidad, etc.

Capa Acceso a Datos: provee el acceso a los repositorios de información persistente y/o sistemas externos. La tecnología utilizada para desarrollar esta capa es ADO.NET, es común utilizar stored procedure y XML.

Capa Presentación

Componentes de Interfaz de Usuario (UIC)

En aplicaciones de cliente inteligente se utilizan los componentes del namespace System.Windows.Forms

En aplicaciones WEB, los componentes corresponden al namespace System.Web.UI

Componentes de Proceso de Interfaz de Usuario (UPC)

Componentes personalizados, frameworks de navegación, librerías de funciones comunes, que faciliten el desarrollo de la interfaz de usuario.

Capa Lógica de Negocios

Componentes de Negocios (BC)

Son los componentes que implementan las reglas de negocio.

Se implementan en clases .NET que puedan utilizar los servicios de COM+ y/o Remoting.

Componentes de Workflow de Negocios (BW)

Son componentes que representan las actividades de un proceso de negocio.

Se implementan las clases .NET en conjunto con algún motor de flujos de trabajo u orquestador de procesos.

Entidades del Negocio (BE)

Son contenedores de datos. Representan las entidades persistentes del negocio. Se transportan entre las capas.

Pueden implementarse en clases .NET, DataSets tipados o DataSets sin tipar.

Interfaces de Servicio (SI)

La lógica de negocios se expone mediante servicios. Cada servicios posee una interfaz, es decir un protocolo que indica cómo debe ser invocado.

Puede implementarse utilizando Web Services o interfaces .NET.

Capa de Acceso a Datos

Componentes de Accesos a Datos (DAC)

Abstraen a la lógica de negocios de los detalles específicos de la persistencia.

Minimizan el impacto en caso de cambio de motor de base de datos o la representación de los datos. Se implementan con ADO.NET y generalmente se construye un set de clases utilitarias para facilitar el desarrollo y abstraer la utilización de un proveedor específico (SQL Client, ORACLE Client, etc.)

Agentes de Servicio (SA)

Se encargan de manejar el acceso a aplicaciones y servicios externos.

Las tecnologías utilizadas son muy variadas y dependen de la aplicación externa (Sockets, Biztalk, Web Services.)

Aplicaciones Web

Aspectos básicos.

El desarrollo de aplicaciones web, se debe manejar, incluyendo el desarrollo de 3 capas, la estructura de elementos web como lo son:

- Manejar el contenido del Sitio como texto, imágenes, videos, sonidos, animaciones, archivos en sus carpetas específicas.
- Utilización de Hojas de estilo la cual se manejara una general para todos los sistemas y se modificará dependiendo de la aplicación, en diferentes clases o propiedades exclusivas para cada sistema en el archivo CSS. Permitiendo la estandarización para definir como se presentará cada elemento del contenido, el tipo de letra a definir, el formato y la posición de cada elemento del documento, y la presentación de las páginas.
- Estructura de las paginas basados en XHTML, paginas dinámicas aspx, con lenguaje de programación en .Net, archivo de lógica de programación aparte del archivo aspx. Ejemplo

que se cree el archivo .aspx asociado al master page y se cree el archivo aspx.vb para visual basic o aspx.cs par C#.

Para el manejo de las interfaces web, (Páginas web), deben de mantenerse la estructura en diseño, imágenes (en el caso de Banner), estilo de menús. Manejar un único template para las aplicaciones Web que se desarrollen en la institución, para la homogenización del Sitio web y se mantenga el mismo formato e interfaz para cualquier aplicación que se desarrolle.

Utilización de Paginas Principales

Una página principal es un archivo de ASP.NET con la extensión .master (por ejemplo, MySite.master) que tiene un diseño predefinido que puede incluir texto estático, elementos HTML y controles de servidor.

Las paginas principales nos permiten centralizar las funciones comunes de las páginas para que las actualizaciones puedan llevarse a cabo en un solo lugar. Además nos facilitan la creación de un conjunto de controles y código, y aplican los resultados en un conjunto de páginas. Por ejemplo, puede utilizar los controles en la página principal para crear un menú que se aplique a todas las páginas. Las páginas principales nos proporcionan un control más preciso sobre el diseño de la página final al permitir controlar el modo en que se representan los controles PlaceHolder, y un modelo de objetos que permite personalizar la página principal a partir de páginas de contenido individuales.

Haciendo así una página principal para el área administrativa o privada del sistema web que se esté desarrollando y una página principal para la parte pública o interactiva con los usuarios en general. Manteniendo siempre el template propio de la institución.

Manejo de Menús

Se debe mantener los menús acordes a la estructura utilizada de las páginas, en diseño, manejo, lenguaje de programación. Se debe de utilizar menús programables dentro de la aplicación, utilizando el lenguaje de programación en .net para realizar cambios, ajustes validaciones o asignación de opciones en el menú en tiempo de ejecución, ya sea mediante los componentes propios de .Net o componentes adicionales pero compatibles con .Net.

La ubicación de los menús se recomiendan que sean: Horizontales (Para opciones generales) por debajo del banner principal, y verticales (Para opciones específicas) al costado izquierdo de las páginas.