

PAUTAS DE DESARROLLO PARA PIS

con GeneXus

GeneXus Consulting S.A.



www.genexusconsulting.com

MONTEVIDEO - URUGUAY - Av. Italia 6201, Ed. Los Pinos, P1 - (598) 2601 2082

TABLA DE CONTENIDO

OBJETIVO	3
ALCANCE	3
DEFINICIONES REQUERIDAS PARA EL DESARROLLO	4
DISEÑO GRÁFICO.....	4
<i>Tamaño de las pantallas.....</i>	4
DETERMINAR TIPO DE CONTROL PARA CADA ENTIDAD.....	4
<i>Tipo de control en transacciones y filtros</i>	4
<i>Tamaño de campos de acuerdo al tipo de dato</i>	5
LOOK AND FEEL	6
<i>Theme.....</i>	6
PATRONES DE DISEÑO	7
PARÁMETROS DEL SISTEMA.....	7
NUMERADORES.....	7
<i>Source</i>	8
BUENAS PRÁCTICAS EN LA CODIFICACIÓN	9
FOR EACH	9
<i>Cuando utilizar ordenes.....</i>	9
<i>Control de flujo en el For Each.....</i>	9
<i>Uso de When None.....</i>	9
<i>Indentación del código</i>	9
USO DE LA TABLA EXTENDIDA.....	11
CORTE DE CONTROL	11
CALL VS. LINK	11
REGLA PARM	13
CALL VS. UDP VS. UDF	13
REFERENCIAS A IMÁGENES	14
VARIABLES DEFINIDAS.....	14
<i>Consideraciones para dominios.....</i>	15
<i>Consideraciones según tipo de datos</i>	15
<i>Character vs. VarChar vs. LongVarChar.....</i>	16
COMBO BOXES.....	17
<i>En Filtros.....</i>	17
<i>En Ingreso de datos.....</i>	17
CORRECTO USO DE SERVERNOW()	17
MANEJO DEL COMMIT.....	18
ENTITYSERVICES VS. WEB PANEL DESIGNER VS. DISEÑO MANUAL	18
MANEJO DE FILTROS.....	18
BUENAS PRÁCTICAS EN EL USO DEL TIPO DE DATOS	19
SDTs	19

INTRODUCCIÓN

OBJETIVO

El objetivo del documento es establecer pautas y mejores prácticas al momento de desarrollar aplicaciones con GeneXus.

ALCANCE

Esta versión tendrá como objetivo recapitular el conocimiento distribuido entre distintos documentos, sin generar nuevo conocimiento.

PAUTAS DE DESARROLLO CON GENEXUS

DEFINICIONES REQUERIDAS PARA EL DESARROLLO

Diseño Gráfico

Tamaño de las pantallas

Las aplicaciones se diseñan para ser usadas en pantallas de por lo menos 1024x768. En caso que el usuario tenga menos resolución ej. 800x600, se debe incorporar automáticamente barras de desplazamiento para facilitar la navegación.

El uso del Scroll horizontal es considerado excepcional y debe evitarse.

Se debe evitar el uso de fondos azules o verdes pues disminuyen y degradan la legibilidad. Usar un fondo rojo disminuye la legibilidad sin importar el color de las letras.

Determinar tipo de control para cada entidad

Tipo de control en transacciones y filtros

Hay diferentes maneras de mostrar una clave foránea, entre ellas se pueden utilizar Combo Boxes, Prompts, listas dinámicas, etc.

Por cada proyecto, se deberá definir una tabla en la cual diremos qué tipo de control usar para cada entidad (generalmente claves foráneas). Por ejemplo:

Entidad	Atributo	Control
EstadoTramite	EstadoTramiteId	Dynamic Combo Box
Persona	PersonalId	Prompt
...

Como regla general, las entidades que muestran más de 20 registros (aproximadamente) se muestran con Prompts que permiten filtrar o Suggest sobre descripciones. Por otro lado para las que tienen menos registros usamos los combos dinámicos, generalmente con Id y Descripción.

Intentar quitar la mayor cantidad de Prompts posibles para facilitar la entrada. Esto no significa sacarlos cuando el usuario debe visualizar mucha información para tomar una decisión.

No se utilizarán Suggest para cuando no existan índices únicos sobre las descripciones. Sin embargo, de ser necesario definir un atributo como fórmula redundante y concatenar su identificador + descripción. Luego utilizar el Suggest sobre dicha fórmula. Esto no hace más lento un sistema.

En los dominios, no se especifican Control Types Dinámicos; esto se hace en los atributos.

Los enumerados pueden ser Radio Buttons (Si son 2 o 3 valores y obligatorios) o el Combo definido por defecto. También puede ser un Listbox pero en la práctica no aporta valor ya que no se puede realizar una selección múltiple.

Los Booleans son CheckBox por defecto, se puede utilizar un combo en algún caso si justifica.

Tamaño de campos de acuerdo al tipo de dato

El tamaño de los campos se deberá definir por proyecto, ya que generalmente se presentan distintas necesidades de acuerdo a los requerimientos de cada uno. El mismo debe ser fijo, de acuerdo al tipo de datos y tamaño. Para ello llevaremos esa información en una tabla como la siguiente:

Entidad	Atributo	Tipo/Dominio	Dimensión
Persona	PersonaPrimerNombre	DescripcionCorta	Varchar(100)
PagoAporte	PagoAportelImporte	Importe	Numeric(12.2)
...

Los Varchars tienden a ocupar más de una línea, para que esto no suceda se modifican desde el dominio las siguientes propiedades:

- Autoresize = false
- Width = ancho en caracteres
- Height = alto en líneas, este se tiende a poner en 1

Look and Feel

Para todas las transacciones usar el Pattern Trn Form de las K2B Tools.

Theme

Antes de comenzar a desarrollar se recomienda definir e implementar en la KB el Look & Feel que tendrá la aplicación y definir el objeto Theme a usar en el proyecto. Al utilizarse las K2B Tools, hay un conjunto de objetos básicos que deben modificarse para obtener el diseño querido.

El tema por defecto es GenexusX. Este tema tiene ciertos nodos HTML que utiliza GeneXus en la generación de las aplicaciones. Estos nodos deben de estar en el Tema de las K2B Tools que se seleccione para la KB. Si no se copian estos nodos, se obtendrá un Look & Feel deficiente:

- PopupBorder
- PopupShadow

Utilizar el Theme **K2BModern**

PATRONES DE DISEÑO

Parámetros del sistema

Los parámetros de sistema se deben ingresar en una tabla Parámetros para la cual se especifican las siguientes columnas:

PARM CODIGO	PARM DESCRIPCION	PARM VALOR	PARM TIPO
Clave primaria, que identifica al parámetro. Ej: ApplicationLng	Descripción. Ej: Lenguaje de la Aplicación para usar en el CA.	Valor a utilizar en el Sistema, por ejemplo 'ESP'	Ejemplo: 'char' 'date' 'boolean' 'numeric' 'numericDecimal'

Para obtener un valor utilizar procedimientos de tipo:

ParametroObtenerFecha, ParametroObtenerNumero, ParametroObtenerTexto, etc que reciban un ParmCodigo y retornen un ParmValor del tipo de datos fecha, numérico, texto, etc.

Por cada parámetro, se debe crear un programa que encapsula la persistencia del mismo. Por ejemplo ObtenerFechaComienzoAportes que dentro llama a:

```
ParametroObtenerFecha.Call(!"FECHA_COMIENZO_APORTES")
```

Numeradores

Para manejar los identificadores de las distintas entidades del Sistema (con clave compuesta) se deberá contar con una tabla con la siguiente estructura:

NumeradorCodigo	NumeradorValor	NumeradorVigente	NumeradorFechaEliminacion
Almacena el nombre de la entidad a numerar	Almacena el último valor asignado	Para manejar vigencia (si corresponde)	Para manejar vigencia (si corresponde)

Se deben proveer de procedimientos que actualicen el valor para una entidad dada y retornen el valor a asignar en el identificador para utilizar en las inserciones. Por ejemplo:

```
&EntidadId = NumeradorObtenerProximoValor.Udp(!"ENTIDAD") if Insert on  
AfterValidate;
```

Source

```
for each  
    where NumeradorCodigo = &NumeradorCodigo  
    NumeradorValor = NumeradorValor + 1  
    &NumeradorNuevoValor = NumeradorValor  
when none  
    &NumeradorNuevoValor = 1  
    &NumeradorBC = new()  
    &NumeradorBC.NumeradorCodigo = &NumeradorCodigo  
    &NumeradorBC.NumeradorValor = &NumeradorNuevoValor  
    &NumeradorBC.NumeradorVigente = Vigente.Si  
    &NumeradorBC.Save()  
endfor  
commit
```

El procedimiento debe ejecutar en otra UTL por lo que se deben configurar las siguientes propiedades:

- Main program = True
- Execute in new LUW = True
- Commit on exit = No

BUENAS PRÁCTICAS EN LA CODIFICACIÓN

For each

Cuando utilizar ordenes

Definir el Order solamente cuando por el requerimiento hay que utilizar un orden específico. Ej: recorrer facturas por fecha.

En un For Each, si se está haciendo una suma Aggregate/Select, no se deben usar órdenes en los atributos que lo realizan. O sea, los atributos que participan en el Where, no van en el Order.

Control de flujo en el For Each

No escribir en el código del For Each condicionales If que pueden ser agregados a los filtros Where. O sea, siempre que se pueda debemos recorrer la cantidad de registros necesaria utilizando filtros. Esto aplica a menos que tengamos cierta lógica que debemos poner en el Else de la condición.

Uso de When None

Siempre que se pueda debemos usar el When None del For Each en lugar de programar usando Flags en caso de que el registro no cumpla con los filtros.

En caso de que sea un error el que no se recorra ningún registro, explicitarlo en el When None.

Indentación del código

Para que los For Each queden más claros y fáciles de identificar dentro de los eventos o del código en general, se recomienda que se escriban de la siguiente manera:

```
// Forma Incorrecta:
Event 'NuevoCli'
For Each
where CliCod = &CliCod
//Código
EndFor
EndEvent

// Forma correcta:
Event 'NuevoCli'
    For Each
        where CliCod = &CliCod
        //Código
    EndFor
EndEvent
```

Para que los filtros de los For Each queden más claros se recomienda tener un Where para cada condición y no utilizar AND.

```
//Forma incorrecta:
For Each
    where CliCod = &CliCod and CliStatus = &CliStatus and CliTipo = &CliTipo
    //Código
EndFor

// Forma correcta:
For Each
    where CliCod = &CliCod
    where CliStatus = &CliStatus
    where CliTipo = &CliTipo
    //Código
EndFor
```


Uso de la tabla extendida

Siempre que sea posible acceder los atributos por su tabla extendida. No escribir un For Each por cada tabla. Esto implica una mejora en la performance y ayuda a mantener el código más fácilmente.

Las actualizaciones a atributos de la tabla extendida de una transacción se deben mantener a través del BC de la tabla extendida. Lo mismo para inserciones, eliminaciones, etc.

Más información en:

<http://training.genexus.com/principal/ampliacion-general/conceptos-de-tabla-base-y-tabla-extendida?es>

<http://training.genexus.com/principal/ampliacion-general/comando-for-each-determinacion-de-tabla-base?es>

<http://wiki.genexus.com/commwiki/servlet/hwiki?Category%3ABusiness+Components>,

Corte de control

Un corte de control se escribe como For Each anidados que determinan la misma tabla base. El criterio de corte o agrupación se define mediante la cláusula Order.

Se utiliza para el caso en que tenemos que recorrer claramente un cabecal y sus líneas.

Más información en:

<http://training.genexus.com/principal/ampliacion-general/for-eachs-anidados-corte-de-control?es>

Call vs. Link

Como regla general diremos que al Link lo usaremos como función y no como comando.

Ejemplo:

```
// accion
```

```
If &permisos = 1
```

```
    txtPermisos.Link = HWebpanel.Link(&Parm1,&parm2)
```

```
EndIf
```

En otras palabras, si lo que se quiere hacer es redirigir al usuario a otra pantalla donde se va a hacer una tarea (siempre con su validación correspondiente), hacer lo de arriba.

Notar que en el Hwebpanel se deben validar los permisos nuevamente.

Pero si hay una lógica que dice si se va para un lado o para otro, usar Call.

Ejemplo:

```
// accion
```

Event 'Confirmar'

```
&permisos = PAutorizar.udp(&parm1,&parm2)
```

```
Do Case
```

```
Case &permisos = 1
```

```
Hwebpanel.Call()
```

```
Case &permisos = 2
```

```
Hpepe.call()
```

```
Case &permisos = 3
```

```
HOtraPantalla.Call()
```

```
EndCase
```

EndEvent

Debemos tener en cuenta de que el Link corta la ejecución.

Ejemplo:

```
// accion
```

Event 'Confirmar'

```
Hwebpanel.link()
```

```
Msg("hola")
```

EndEvent

En el ejemplo de arriba, el mensaje "hola" no va a desplegarse.

Regla Parm

Definir siempre los parámetros en las reglas indicando si son de entrada, salida o ambos ("in:", "out:", "inout:"). Los mismos deben ir en minúscula y pegados al parámetro.

Por ejemplo:

```
Parm(in:&EntradaConfirmarPago, out:&SalidaConfirmarPago);
```

Call vs. UDP vs. UDF

No usar llamadas a procedimientos con Udf() ya que esta forma se encuentra en desuso, usar sólo Udp().

Utilizar PName.Call() en lugar de Call(PName,?). Lo mismo para Udp.

Tampoco utilizar la forma NombreObjeto(), la misma la usaremos solamente para cuando referenciamos Data Providers. De esta manera logramos un código más fácil de leer e interpretar para los demás desarrolladores.

Si hay un solo parámetro de salida utilizaremos la forma:

```
&Variable = Procedimiento.Udp()
```

Si retorna más parámetros, usamos la forma:

```
Procedimiento.Call(&Var1,&Var2,&Var3)
```

Referencias a imágenes

Las imágenes deben de guardarse como objetos de tipo Image en la KB y no deben utilizarse links a imágenes externas. Tener imágenes como objetos de la base de conocimiento proporciona varios beneficios, entre ellos:

- Referencia cruzada: permite saber dónde se utilizan una imagen determinada.
- Las imágenes pueden depender de temas, idiomas o densidad permitiendo a las aplicaciones utilizar una u otra en función del contexto.
- Importación / Exportación de objetos con las imágenes que utilizan. Ya no hay necesidad de generar archivos zip adicionales con imágenes.

Variables definidas

Las variables utilizadas deben de estar basadas en dominios, las que hacen referencias a atributos deben de estar basadas en los mismos y tener el mismo nombre. Cuando son calificadores, escribir el calificador antes del nombre del atributo o dominio en el cuál se quiere basar la variable. De esta forma, al hacer clic secundario sobre la variable y elegir Add Variable del menú contextual, GeneXus define la variable correctamente.

Siempre hemos de definir explícitamente las variables que GeneXus crea como autodefinidas.

Utilizar nombres nemotécnicos para las variables que no correspondan a ningún atributo del sistema. Por ejemplo:

Se quiere cargar en una variable la existencia de un cliente.

- Forma correcta: &ExisteCliente
- Forma Incorrecta: &Flag

Eliminar de los objetos las variables que no se utilizan.

Consideraciones para dominios

Definir dominios para los atributos que almacenen una descripción.

Utilizar dominios basado en otros dominios (especialización de dominios). Ejemplo: Moneda (positivo y negativo) y MonedaPositiva (basado en Moneda con la restricción de positividad).

Al utilizar dominios enumerados, definir las siguientes propiedades:

- Nombre, Descripción y Valor para cada uno
- No utilizar el Empty Value en el enumerado. Se maneja con el Empty Item
- ATT.EnumerationDescription() retorna la descripción
- IsEmpty() retorna si está vacío el valor
- En las transacciones, si el atributo está definido como Nullable = no se controla solo que se agregue un valor, sino te lo deja vacío

Consideraciones según tipo de datos

Para datos de tipo Numérico se definen las siguientes propiedades:

- Largo
- Si es signado; por defecto no hay signo
- Auto numérico para identificadores
- Rango de valores:
 - 1:20 entre los dos valores
 - 30: más que
 - 1 2 3 4 exactamente estos valores
 - Si se pone un rango de valores, agregar mensaje de Error

CHARACTER VS. VARCHAR VS. LONGVARCHAR

La decisión de utilizar el tipo de datos Character(n), VarChar(n) o LongVarChar(n) depende de analizar los datos que serán almacenados en la columna, en cada caso. Como recomendaciones generales podemos decir lo siguiente:

- Usar Character(n) para todo lo de largo fijo, es decir, cuando sabemos que los datos a almacenar en la columna son siempre de largo n.
- Varchar(n) lo vamos a utilizar para todo lo de largo variable. Por ejemplo, descripciones y nombres, de los que no sabemos de antemano el largo que van a tener.

- Longvarchar se utiliza para Strings de largo variable pero que sabemos que pueden ser muy grandes. Por ejemplo para almacenar XML.

Referencias:

<http://wiki.genexus.com/commwiki/servlet/hwiki?VarChar+data+type>,

<http://wiki.genexus.com/commwiki/servlet/hwiki?LongVarChar+data+type>,

No utilizar el tipo Bitmap en la definición de atributos. Utilizar Blob o Image en su lugar. Definir las siguientes propiedades para el tipo Blob:

- FileType :Mime Type que se utilizara en el atributo, definirlo en el atributo.
- FileTypeAttribute: utilizar en el atributo, definirlo en el atributo
- FileNameAttribute: utilizar en el atributo, definirlo en el atributo

Debemos siempre definir qué es lo que se va a almacenar en el atributo, y solamente permitir la carga de ese tipo específico y rechazar la de cualquier otro.

Otros tipos de datos (extendidos), no utilizarlos en los dominios que definen atributos.

Combo boxes

Establecer en los combos la propiedad EmptyItem en true.

En Filtros

La descripción de los Empty Values en un combo box que oficie de filtro debe ser:

- “(Todos)” cuando tengo un filtro con condición “when not empty”
- “(Ninguno)” cuando el filtro no tiene la condición “when not empty”

En Ingreso de datos

La descripción de los Empty Values en un combo box de ingreso de datos debe ser:

- “(Seleccionar)” cuando no puede ser vacío el datos ingresado (Ej: hay una regla de error que dice “debe seleccionar el tipo de trámite”
- “(Ninguno)” cuando el dato ingresado puede ser vacío

Correcto uso de ServerNow()

No hacer llamadas innecesarias a la función ServerNow() o ServerDate() ya que esto genera accesos a la base de datos. Tener en cuenta especialmente si se encuentra dentro de una estructura iterativa.

Por ejemplo, en lugar de esto:

```
&Hora = Hour(ServerNow())
```

```
&Minuto = Minute(ServerNow())
```

```
&Fecha = ServerDate()
```

Hacer algo como:

```
&DateTime = ServerNow()
```

```
&Hora = & DateTime.Hour()
```

```
&Minuto = & DateTime.Minute()
```

```
&Fecha = &DateTime
```

Realizando una llamada solamente al comienzo ganamos en performance.

Manejo del Commit

Los procedimientos se definirán con la propiedad Commit on exit en No y se debe utilizar el comando Commit en el código para controlar la UTL, generalmente en el Web Panel o transacción llamador. Permite mayor control del momento en que se realiza el Commit y evita que se rompa la UTL sin querer por llamadas a procedimientos que realizan modificaciones y hacen Commit al salir (con la propiedad Commit on exit en yes).

Tener en cuenta que en las transacciones el Commit se hace al final de forma automática, no así en Web Panels y procedimientos Main.

También tener en cuenta que ninguna de las APIs de GXflow hace Commit, el mismo deberá ser controlado por el usuario.

EntityServices vs. Web Panel Designer vs. diseño manual

Si el requerimiento es un ABM se utiliza el patrón EntityServices de las K2BTools.

Cuando se requiera realizar una pantalla con funcionalidad que no sea ABM, utilizar el patrón WebPanel Designer de las K2BTools.

Solamente si precisamos añadir funcionalidad que no se pueda realizar con las K2BTools, diseñaremos manualmente el Web Panel. Validar siempre antes con el jefe de desarrollo.

Utilizar las K2BTools tiene la ventaja de facilitar y agilizar el desarrollo y uniformizar la aplicación.

Manejo de Filtros

Los filtros de ahora en adelante deberíamos tanto en el patrón como si los programamos a mano hacer lo siguiente:

1. Si es un filtro de tipo Character, concatenar los '%' en la variable y hacerlo con Like de forma tal que si la variable del filtro es &Nombre el filtro tendría la siguiente forma:

```
Nombre like '%' + &Nombre + '%' when not &Nombre.isEmpty();
```

Entonces si ingresan "Pérez" como valor del filtro me encontraría nombres del estilo "Alejandro Pérez".

2. Si es un filtro de fecha, deberíamos buscar por rangos. Las variables que pondríamos de filtro serían por ejemplo &FechaInicial y &FechaFinal y la condición quedaría:

```
Fecha >= &FechaInicial When not &FechaInicial.isEmpty();  
Fecha <= &FechaFinal When not &FechaFinal.isEmpty();
```

3. Si es un filtro numérico la condición debería ser =. Si tenemos la variable de filtro &Numero la condición quedaría:

```
Numero = &Numero When not &Numero.isEmpty();
```

(Salvo que ustedes consideren que se necesite un rango de búsqueda del valor numérico)

4. Filtros por fecha de vigencia. En el WHERE del FOR EACH contemplar los 2 casos para valores nulos y vacíos:

Ej.:

```
For Each  
  Where FechaIniVigen <= &Fecha  
  Where FechaFinVigen >= &Fecha OR FechaFinVigen.isEmpty() OR  
  FechaFinVigen.IsNull()  
EndFor
```

BUENAS PRÁCTICAS EN EL USO DEL TIPO DE DATOS

SDTs

No hacer SDTs que se llamen list. La raíz de un SDT no debe ser una colección. Definirla en la variable que se utilice como colección.

Los campos basados en atributos deben tener el atributo como tipo de datos.

El nombre de los SDTs no puede coincidir con el nombre objetos, atributos o dominios.