

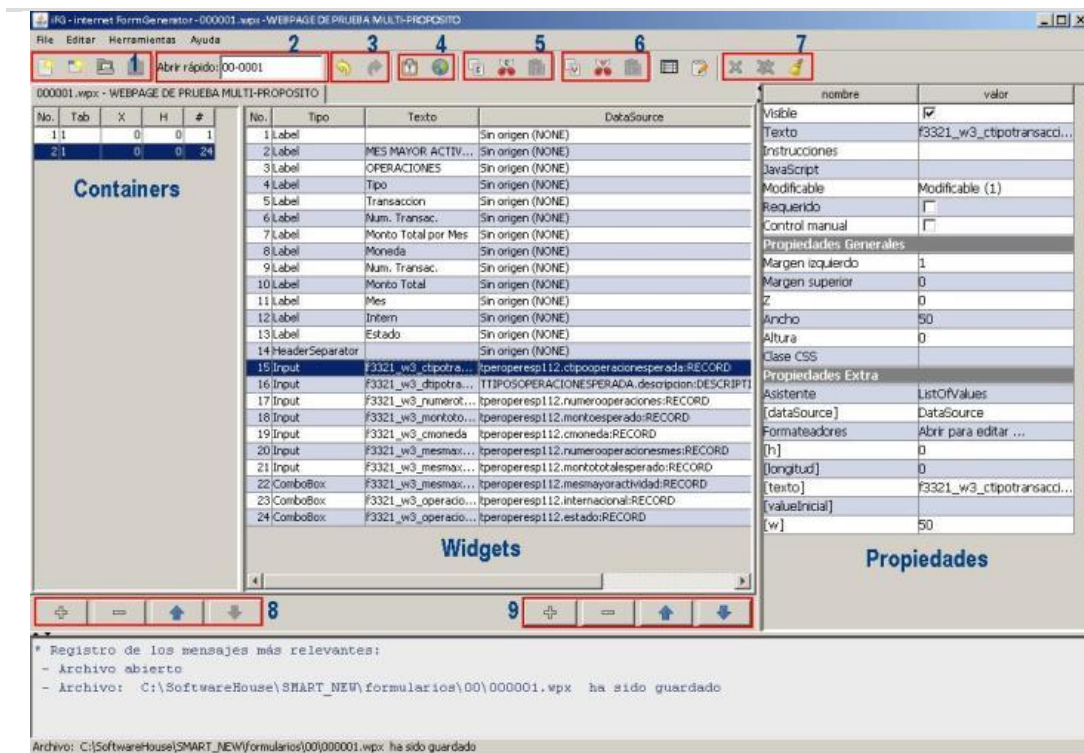
WEBPAGES-IDE

MANUAL BASICO DE USUARIO

WebPage

Es el formulario que agrupa los containers y sus widgets, que definen los elementos necesarios para formar una transacción FIT.

Vista Principal



Elementos

- 1: Nuevo/Wizard/Abrir/Guardar Formulario.
- 2: Abrir rápido formulario. (Se requiere especificar el [Directorio Base](#))
- 3: Deshacer/Rehacer
- 4: Previsualizar WebPage
- 5: Copiar/Cortar/Pegar [Container](#)
- 6: Copiar/Cortar/Pegar [Widget](#)

- 7: Cerrar/Cerrar todos/Limpiar WebPages
- 8: Agregar/Eliminar/Mover [Container](#)
- 9: Agregar/Eliminar/Mover [Widget](#)
- 10: Registro de eventos efectuados en el WebPages-Ide

Barra de Menú

File

Nuevo / Abrir / Recientes

Crear / Abrir o Utilizar webpages recientemente usados

Cerrar / Cerrar todo

Cerrar el webpage actual / Cerrar todos los webpages

Guardar / Guardar como

Guardar el actual webpage / Guardar el webpage con otra extensión

Salir

Cerrar el WebPage-Ide

Editar

Undo / Redo

Deshacer / Rehacer cambios efectuados en el WebPage

Widget / Container / WebPage

Abre la ventana de configuración de las propiedades de: Widget / Container / Webpage

Cálculos

Editar la sección de cálculos JS que se ejecutaran luego de cualquier acción en el WebPage

Resetear Ventana

Establecer la ventana bajo los parámetros iniciales. Borrar registro de eventos y tamaños.

Preferencias

Modificar la configuración de acceso de los WebPages y puerto para pre visualización.

Herramientas

Validar

Buscar errores de definición de elementos en todo el WebPage.

Fijar valor a columna / Agregar valor a columna X / Y / W / H

Establecer un valor fijo para los parámetros X, Y, W, H de cada widget dentro de un Container

Ajustar columna X

Cambiar la alineación X de todos los containers del Webpage

Limpiar estilos y valores de campos ocultos

Limpiar estilos CSS y valores de todos los widgets ocultos del WebPage

Ayuda

Ayuda

Muestra el documento de ayuda

Acerca de

Información del producto

Propiedades

Editar Propiedades

| nombre | valor |
|--------------------------------|---|
| Título | Ingreso Y Mantenimiento Provincias p... |
| Idioma | ES |
| Permite mantenimiento | <input checked="" type="checkbox"/> |
| Subsistema | 00 |
| Transacción | 0001 |
| Visualización | |
| Margen izquierdo | 0 |
| Margen superior | 0 |
| Estilo | |
| Funciones y Javascript inicial | |
| Comportamiento | |
| Paginación Multiregistro (F9) | Habilitada (H) |
| Control Cambio / Mant. | Cambio Automático (A) |
| Relaciones | |
| Referencias | Abrir para editar ... |
| Formularios adjuntos | Abrir para editar ... |

Cerrar

Titulo

Permite establecer el Titulo que identifica el WebPage. Este texto aparece en la parte superior central del WebPage.

Idioma

El idioma que manejara el WebPage. Por defecto ES (Español)

Permite Mantenimiento

Indica si el WebPage podrá Visualizar y Modificar datos de las tablas de la Base de Datos, o solamente será un Rebaje de consultas (no permite mantenimiento)

Subsistema

El numero del Subsistema bajo el que trabajara el WebPage.
Normalmente definido en la tabla
TSUBSISTEMATRANSACCIONES.

Transacción

El numero de la Transacción del Subsistema bajo la que trabajara el WebPage. Normalmente definido en la tabla
TSUBSISTEMATRANSACCIONES.

Margen Izquierdo

Ajuste de todas las propiedades X (margen izquierdo) de todos los containers del WebPage.

Margen Superior

Ajuste de todas las propiedades H (margen superior) de todos los containers del WebPage.

Estilo

Clase CSS de estilo del WebPage en general.

Funciones y JavaScript Inicial

El JavaScript que utilizara el WebPage para ser usado por cualquier subproceso de la transacción

Generalmente tiene 3 funciones pre establecidas:

preConsultar/posConsultar

Secuencia de instrucciones a ejecutarse previo/posterior a enviar una solicitud de Consulta (F7).

En caso de devolver algún error en esta secuencia de instrucciones, la consulta no se efectuara.

Se puede controlar validaciones previas a una consulta en esta sección, para ello, simplemente se devuelve un true en caso de permitir la acción o un false en caso contrario.

preMantener/posMantener

Secuencia de instrucciones a ejecutarse previo/posterior a enviar una solicitud de Mantenimiento (F12)

En caso de devolver algún error en esta secuencia de instrucciones, el mantenimiento no se efectuara.

Se puede controlar validaciones previas a un mantenimiento en esta sección, para ello, simplemente se devuelve un true en caso de permitir la acción o un false en caso contrario.

Código Libre

Secuencia de instrucción a ejecutarse justo después de cargarse completamente el formulario.

En caso de devolver algún error en esta secuencia de instrucciones, el formulario no se cargara completamente.

Paginación Multiregistro (F9)

Establecer si se puede paginar entre registros devueltos por una referencia. Si la paginación es condicionada, será controlada por un JS establecido en JS Inicial.

Control Cambio / Mantenimiento

Establece si el formulario puede hacer consultas y modificaciones simultáneamente o solo una de ellas.

Referencias

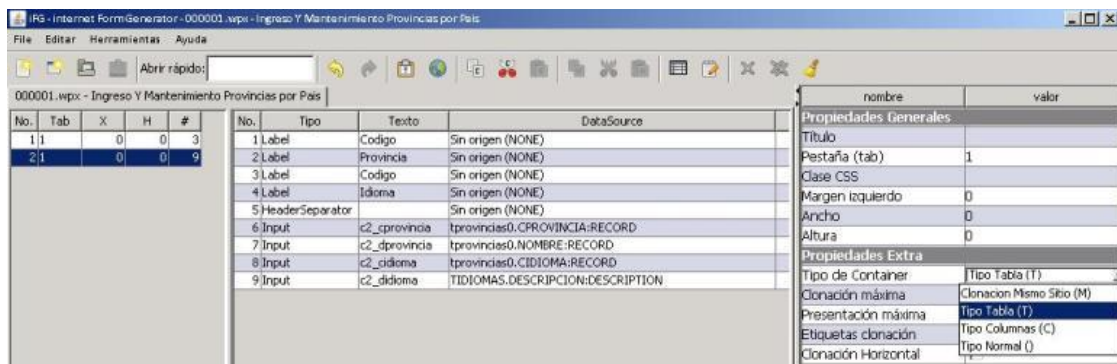
Permite crear/modificar/eliminar las [Referencias](#) a ser usadas por el WebPage

Formularios Adjuntos

Permite crear/modificar/eliminar los **Formularios Adjuntos** en el WebPage. Estos formularios se verán específicamente en el tab establecido en su definición.

Container

Como su palabra lo dice, es un contenedor de elementos (Widgets). Éstos se agrupan por un Tipo de Container establecido.



Propiedades

- **Título**

Permite establecer un Título que se visualizara sobre el container.

- **Pestana Tab**

El Tab dentro del webpage donde será visualizado el container

- **Clase CSS**

Clase CSS que establecerá el Estilo CSS del Container

- **Margen Izquierdo [X]**

Alineación (pixeles) horizontal del Container

- **Ancho [W]**

Ancho (píxeles) del container

- **Altura [H]**

Alto (píxeles) del container. Establecido por el espacio existente entre uno y otro Container.

- **Clonación Máxima**

En caso de querer clonar los elementos de un Container N veces, se utiliza esta opción para especificar cuantas veces se repitieran los widgets dentro del container. Tomar en cuenta que hay que establecer un Height [H] para el container lo suficientemente grande para separar la clonación de una fila de widgets con otra.

- **Presentación Máxima**

Aplica para poder visualizar únicamente M registros de los N registros clonados inicialmente.

- **Etiquetas Clonación**

Etiquetas Horizontales que se mostraran para diferenciar ciertas filas con otras.

- **Clonación Horizontal**

Indica si el container se clonara Horizontalmente.

Tipo de Container

Tipo de Agrupamiento de los Widgets dentro del Container.

Existen 4 tipos de Container:

Clonación Mismo Sitio

Los widgets se agruparan uno sobre otro. Estableciendo su ubicación mediante posición absoluta y sus scrolls serán editados por el usuario.

(Obsoleto)

Tabla

Los widgets se agruparan en un elemento tipo TABLE, que tendrá su Header y Detail. Ambos separados por un Widget tipo [HeaderSeparator](#)

Columnas

Los widgets se agruparan siguiendo un patrón de filas/columnas.

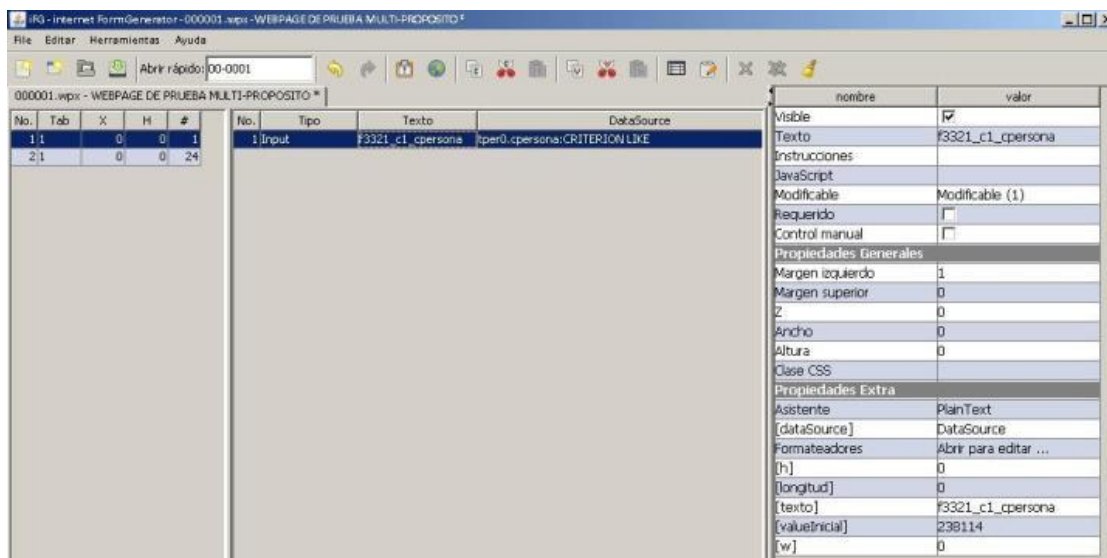
Separándose uno de otro, respecto al tamaño de sus widgets.

Normal

Los widgets podrán ser ubicados por posición absoluta. Si no se especifican sus posiciones, estos se apilaran en la posición 0,0.

Widget

Elementos web con los que el usuario puede interactuar para la manipulación de datos.



Propiedades Generales

- **Visible**

Establecer si el Widget podrá ser visible dentro del Container o no

- **Texto**

Si el Widget no es un label, esta propiedad guarda el nombre del Widget dentro del contexto. Caso contrario, se presentara como el caption de ese label.

- **Instrucciones**

ToolTipText indicando una breve descripción del widget.

- **JavaScript**

Funcionalidad adicional para el comportamiento del Widget. Se puede usar libremente JS en esta sección, pero siempre tomando en cuenta que esto debería ser como último recurso.

- **Modificable**

Establece si al Widget se le puede modificar o no, la información que tenga.

- **Requerido**

Establece si es un Widget requerido para algún proceso de consulta o mantenimiento. La propiedad requerido verifica que contenga por lo menos un caracter ingresado.

- **Control Manual**

Establece si el control de el widget debe ser manual o por un proceso interno de la aplicación.

- **Margen Izquierdo [X]**

Establece el valor de la propiedad X del Widget, que se comporta de diferentes maneras dependiendo del tipo de container. Toma el valor 0 por defecto.

Si el container es tipo **Normal**, esta propiedad establece el **pixel en X** donde se ubicara el widget

Si el container es tipo **Columnas**, esta propiedad establece la **columna** donde se ubicara el widget

Si el container es tipo **Tabla**, esta propiedad establece el **colspan** de dicha celda en la tabla

- **Margen Superior [Y]**

Establece el valor de la propiedad Y del Widget, que se comporta de diferentes maneras dependiendo del tipo de container. Toma el valor 0 por defecto

Si el container es tipo **Normal**, esta propiedad establece el **pixel en Y** donde se ubicara el widget

Si el container es tipo **Columnas**, esta propiedad establece la **fila** donde se ubicara el widget

Si el container es tipo **Tabla**, esta propiedad establece el **rowspan** de dicha celda en la tabla

- **Z**

Actualmente solamente tiene funcionalidad en containers tipo Tabla, logrando establecer la fila dentro del Header del Container en la que va a aparecer dicho widget. Con esto se logra Tabla con Headers multi-filas. Toma el valor 0 por defecto.

- **Ancho [W]**

Establece el ancho del widget. Si no se pone ningún valor, este toma un valor de 150px por defecto.

- **Altura [H]**

Establece el alto del widget. Si no se pone ningún valor, este toma un valor de 25px por defecto.

- **Clase CSS**

Establece la clase CSS del estilo que utilizara el Widget.

- **Asistente**

Establece un asistente para indicar la forma en cómo se ingresaran los datos a este widget. Valido preferiblemente para widgets de tipo Input (Input, InputLabel, TextArea). Véase [Asistentes](#) para más información.

- **Datasource**

Establece el origen de datos [DataSource] para este widget.

Véase [DataSources](#) para más información

- **Formateadores**

Establece uno o varios formateadores para el texto introducido en el widget.

Valido preferiblemente para widgets de tipo Input (Input, InputLabel, TextArea). Véase [Formateadores](#) para más información.

- **Longitud**

Establece el tamaño en número de caracteres para el widget. Valido preferiblemente para widgets de tipo Input (Input, InputLabel, TextArea).

- **ValueInicial**

Establece el valor por defecto que tendrá este widget al momento de cargar el webpage.

Las propiedades extras se definen por el tipo de widget.

Tipos de Widgets

Button

Propiedades

- **Etiqueta**

Establece la etiqueta que se muestra sobre el botón.

La funcionalidad que pueda disparar este widget, está controlada por el JavaScript que se le asigne en la propiedad JavaScript.

CheckBox

Propiedades

- **Etiqueta**

Establece la etiqueta que se muestra a un lado del CheckBox.

- **Lado**

Establece la posición donde se ubicara la etiqueta del CheckBox (Izquierda, Derecha).

- **Seleccionado Inicialmente**

Indica si el widget aparecerá seleccionado inicialmente al momento de cargar el webpage

- **Valor no Seleccionado**

Establece el valor que obtendrá el Widget en caso de no ser seleccionado (1,0)

Se asume que este widget va a estar relacionado en un datasource que contenga un campo booleano.

ComboBox

Propiedades

- **Valores Opciones (?, ?, ...)**

Establece los valores que tomara el widget al seleccionar la opción mostrada por el ComboBox.

- **Etiquetas Opciones (?, ?, ...)**

Establece las opciones que mostrara el ComboBox al desplegarse.

- **Incluir primera opción vacía**

Indica si se mostrara una opción en blanco al principio de las opciones del ComboBox.

- **Numero de Líneas**

Establece el numero de líneas disponibles para cada opción del ComboBox.

- **Selección Múltiple**

Establece si el widget permitirá la selección de múltiples opciones en el ComboBox.

El orden en que se definen las etiquetas debe ser el mismo en el que deseemos que se tomen los valores respectivos.

DeleteRecord

Agrega un CheckBox para seleccionar el registro vigente a caducar. La petición de caducar se enviara al momento de enviar una solicitud de mantenimiento.

Propiedades

- **Lista Alias**

Establece el nombre de la referencia de la que se enviara la solicitud de caducar el registro de la fila seleccionada actualmente.

Este tipo de widget se utiliza con mayor frecuencia para containers de widgets clonados. Su funcionalidad depende de las dependencias establecidas en la referencia base. Véase [Referencias](#), para más información.

HeaderSeparator

Establece la división entre el header y el detail de una Tabla. Su uso es exclusivo para containers tipo Tabla.

Todo aquel widget ubicado sobre éste pertenecerá al header de la tabla, mientras que todo widget que este debajo, será parte del detail.

Este es el único widget que no tiene propiedades, ya que su uso es solo para agrupar otros widgets.

Image

Permite agregar una imagen al container.

Propiedades

- **Extensión**

Establece la extensión de la imagen (.jpg, .gif, .png, ...).

- **Mostrar barra secuencia**

Indica si se visualizara la barra de secuencia de la imagen o no.

Este widget generalmente toma la imagen basándose en un datasource hacia un campo de tipo BLOB. Véase [DataSources](#), para más información.

Input

Widget tipo TextBox que permite ingresar datos a un input, pero de solo una línea.

InputLabel

Widget de tipo Label que permite asignar un DataSource

Label

Widget para mostrar Etiquetas

Este tipo de Widget no permite asignarle un DataSource. Para esto, mejor utilizar un InputLabel.

RadioButton

Propiedades

- **Etiqueta**

Establece la etiqueta que se muestra a un lado del RadioButton.

- **Lado**

Establece la posición donde se ubicara la etiqueta del RadioButton (Izquierda, Derecha).

- **Seleccionado Inicialmente**

Indica si el widget aparecerá seleccionado inicialmente al momento de cargar el webpage

- **Valor no Seleccionado**

Establece el valor que obtendrá el Widget en caso de no ser seleccionado (1,0)

Se asume que este widget va a estar relacionado en un datasource que contenga un campo booleano.

RemoteIFrame

Establece un IFrame remoto como una ventana a una página web externa a la aplicación

Su dirección de enlace la toma únicamente como un meta-dato de la base de datos mediante un Datasource: (TSUBSISTEMATransaccionesID.URL).

Square

Dibuja un cuadrado limitado por su posición X,Y y su dimensión W,H.

TabBar

Agrega un Barra de Tabs para navegar entre containers definidos por un Tab.

Propiedades

- **Ubicación de Pestanas**

Establece la ubicación de los tabs. Horizontal, Vertical.

- **Textos (?, ?, ...)**

Establece los textos que se visualizaran en cada Tab

- **Tabs (?, ?, ...)**

Establece los números de Tab que corresponden a cada Texto definido por Tab.

Este widget necesita tener una Altura (H) lo suficientemente alta para separar los tabs del resto de containers.

Cada container tiene una propiedad TAB, que es la referencia que usa este control para la navegación.

TextArea

Propiedades

- **Cortar Línea**

Indica si el ingreso del texto será Wrapped o no. Es decir, habrán saltos de línea automáticos al ingresar texto.

Este widget obtiene un Height de 60px por defecto.

Configuración

Directorio Base

Con esta configuración podremos especificar el directorio donde se almacenara todos los WebPages creados. Para posteriormente poder usarlos con la opción Abrir Rápido.

Para modificar esta configuración, se accede mediante la barra de menú. [Editar>Preferencias](#).



En la propiedad Directorio Base, se especifica el path completo donde se guardaran los WebPages

En la propiedad Puerto, se especifica el puerto que usara el explorador web por defecto, para previsualizar los webpages modificados.

Formularios Adjuntos

Permite adjuntar WebPages, como parte del WebPage actual. Estos formularios adjuntos podrán ser accedidos mediante tabs.

Position

Establece en que parte del WebPage aparecerá el formulario adjunto:

BEFORE: El WebPage adjunto aparecerá debajo del TabBar.

AFTER: El WebPage adjunto aparecerá sobre del TabBar.

FIXED: El WebPage adjunto aparecerá en un índice de contenedor fijo, definido por la propiedad Índice Container.

Subsistema

El subsistema que identifica al formulario a adjuntarse.

TabBase

El número del tab en el que aparecerá el formulario adjunto.

Transacción

La transacción del subsistema que identifica al formulario a adjuntarse.

REFERENCIAS

Creando Referencias

En las propiedades del WebPage, se pueden definir las referencias con las que trabaja la transacción.

En esta sección se agregan las n referencias con las que va a trabajar el WebPage. Hay que recalcar que esto es lo primero que se debe hacer antes de comenzar a desarrollar la Transacción.

- [^] [v] : Permite ordenar (arriba/abajo) los elementos.
- [+] [-] : Permite agregar / quitar un elemento.
- [...] : Permite modificar un elemento

Para agregar una Referencia

Click en [+]. Aparecerá un nuevo menú

En esta sección podemos definir el nombre de la tabla de la BDD y el alias con el que será referenciado dentro del formulario

Alias

Nombre (Alias) con el que se llamara a la tabla en el WebPage
Se recomienda que sea un nombre corto, pero entendible.

Tabla

Nombre de la Tabla de la base de datos.
No es sensible a mayúsculas. Pero se recomienda escribir todas las referencias de tabla en MAYUSCULAS

Dependencias

Dependencias que puede tener la tabla.
Se agregan para poder filtrar los datos mostrados por la referencia, o para hacer joins con otras tablas.

El resto de propiedades se contemplan en la capacitación previamente realizada.

Para agregar una Dependencia

Abrir el menú de edición de dependencias en el input respectivo.

La manipulación de dependencias en esta ventana es muy similar a la de referencias. Procederemos a crear un join con la tabla TPAISES igualándolos por el campo CIDIOMA. Esto se hace de la siguiente manera:



The screenshot shows a window titled "Editar Propiedades" with a close button (X) in the top right corner. It contains a table with two columns: "nombre" and "valor". The table has six rows of configuration data. Below the table is a "Cerrar" button.

| nombre | valor |
|-----------------|-----------------------|
| Tipo | CRITERION (CRITERION) |
| Campo | CPAIS |
| Comparador | LIKE |
| Alias Desde | tpaises0 |
| Campo Desde | CPAIS |
| Valor Inmediato | |

Cerrar

Tipo

Se especifica el tipo de dependencia. Esto ya esta explicado en [Dependencias](#).
En este caso escojemos el tipo CRITERION, pues vamos a copiar los criterios de Filtro que tenga la referencia tpaises0 y tomarlo como criterios para la consulta o mantenimiento de tprovincias0.

Campo

Se especifica el campo de la referencia recién creada, a la cual vamos a tomar como campo de comparación en el JOIN

Comparador

Se especifica el comparador para los campos del JOIN.

Estos comparadores pueden ser: LIKE, NOT LIKE, =, <>, IN('val1','val2',...), NOT IN('val1','val2',...)

En caso de usar IN o NOT IN, ya no se deben utilizar los campos siguientes.

Alias Desde

Se especifica el alias de la tabla desde la que vamos a hacer el JOIN. Esta será la tabla principal, y la referencia actual será el JOIN anidado.

Campo Desde

Se especifica el campo del alias desde, con el que vamos a efectuar la comparación.

Valor Inmediato

Un valor inmediato para usar con el comparador.

En caso de usar esta propiedad, ya no se deben usar las propiedades Alias Desde y Campo Desde.

Esta propiedad genera un criterio de filtro en vez de un JOIN.

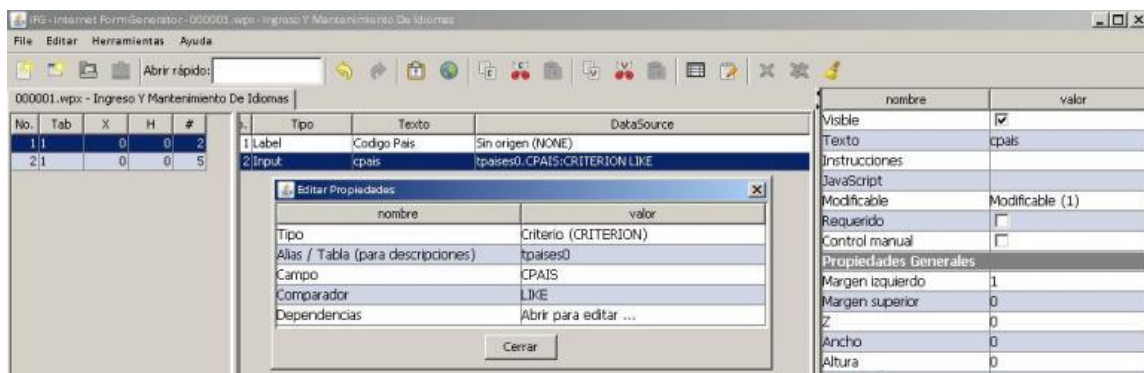
Se pueden especificar n dependencias. Esto tiene que ver con la lógica del query.

Usando Referencias

Para que cualquier WebPage funcione en el ambiente de la aplicación, debe tener al menos una referencia establecida. Y para agregarle funcionalidad, hay que llamar a los campos de sus referencias para poder trabajar con ellos. Esto se logra agregando [DataSources](#) a los Widgets.

Creando DataSources

Para poder establecer un DataSource a un Widget, hacemos click en la celda del DataSource en la sección de los widgets. Esto nos abrirá una ventana para configurar las opciones.



Tipo

El Tipo de DataSource que tendrá el Widget.

Alias / Tabla(para descripciones)

El alias establecido para la referencia creada en el grupo de referencias del webpage.

En caso de querer mostrar la descripción de un FK en la tabla utilizada, se escribe el nombre de la tabla (sin importar mayúsculas).

Es **indispensable** indicar las dependencias de esta tabla para devolver el registro único de la descripción del campo.

Campo

El campo de la tabla referenciada por el alias establecido en la opción anterior. En caso de ser una descripción, generalmente se usa el campo DESCRIPCION de la tabla, pero puede variar según la lógica del query. Es necesario especificar una(s) dependencia(s) para este tipo de campo. Véase [Trabajando con DataSources de tipo Descripción](#) para más información.

En caso de ser una imagen, se llama al campo IMAGEN de la tabla que es de tipo BLOB, el resto, se configura igual que un campo tipo DESCRIPTION.

Véase [Trabajando con DataSources de tipo Descripción](#) para más información.

Comparador

El comparador que utiliza el datasource en caso de ser tipo CRITERION, DESCRIPTION, CRITERION_DESCRIPTION.

Estos comparadores pueden ser: LIKE, NOT LIKE, =, <>, IN('val1','val2',...), NOT IN('val1','val2',...)

Dependencias

Las dependencias necesarias para el campo DESCRIPTION o CRITERION_DESCRIPTION.

Generalmente las dependencias hacen un INNER_JOIN de la tabla con el padre del FK para sacar su descripción, por tanto, hay que establecer el número necesario de dependencias para que devuelva un solo registro desde la tabla padre.

Trabajando con DataSources

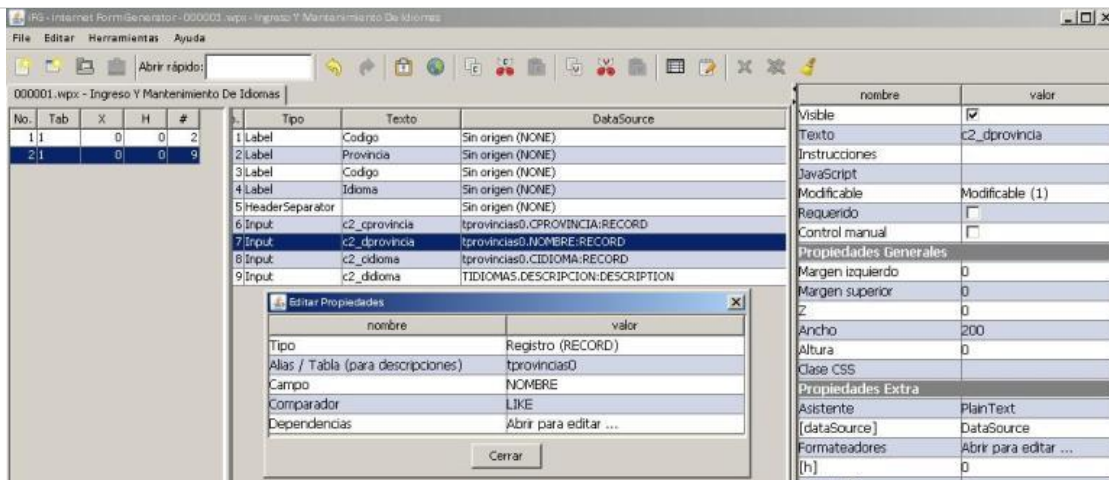
Para el ejemplo anterior, se agrego un DataSource al Widget 'cpais' para que el valor que tenga escrito, se tome como CRITERION al momento de ejecutar una consulta o mantenimiento.

Trabajando con el ejemplo de las [Referencias](#) creadas en el punto anterior, hicimos 2 referencias

tpaises0 ==> TPAISES: Es la referencia que obtendrá el campo de criterio puesto para el widget 'cpais'

tprovincias0 ==> TPROVINCIAS {Dependencia: CPAIS LIKE tpaises0.CPAIS} : Es la referencia que devolverá los registros de provincias para el CPAIS enviado como criterio en la referencia tpaises0.

DataSource de tipo Record

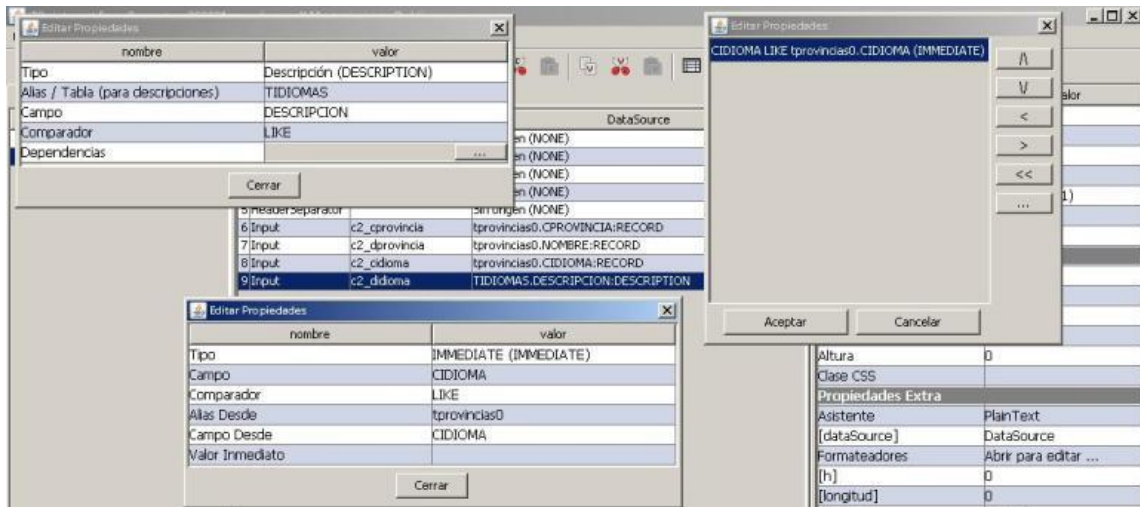


Los widgets agregados en el 2do container tendrán por tanto, datasources respectivos a cada campo de la tabla que queremos mostrar. Estos DataSources deben ser de tipo RECORD si queremos que sean mostrados así como también ser modificados en caso de algún mantenimiento.

DataSource de tipo Descripción

Para el caso especial del widget #9 'c2_didioma', donde queremos mostrar la descripción del idioma asociado a esta provincia, tenemos que establecer un DataSource tipo DESCRIPTION que saque la descripción de ese idioma, encontrada en la tabla TIDIOMAS.

Debemos especificar la dependencia requerida a esta tabla para poder sacar la descripción exacta.

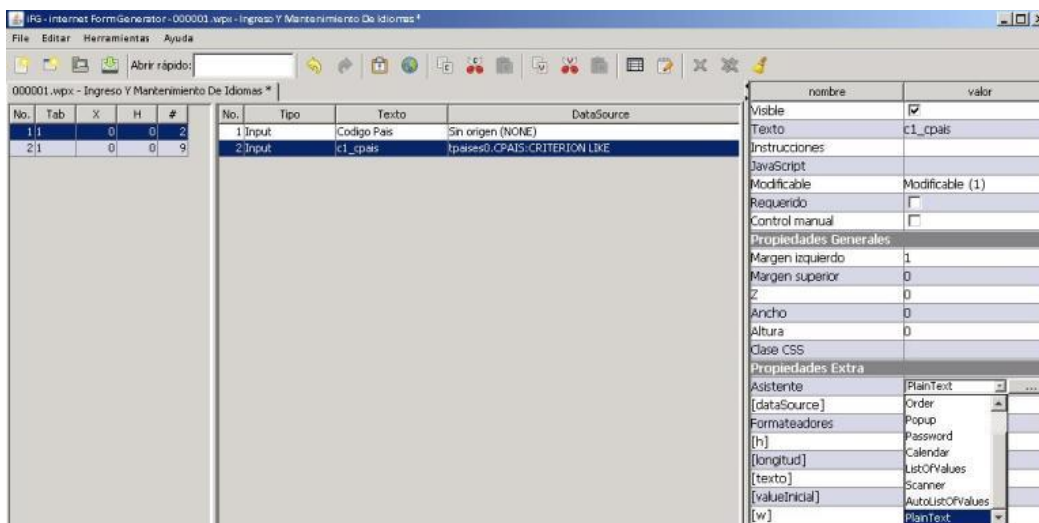


Como podremos observar, la dependencia es del campo CIDIOMA hacia su FK CIDIOMA que se encuentra en la referencia tprovincias0. El comparador en este caso es LIKE, aunque también puede ser (=, IN). Con esta configuración, mostraremos la descripción del CIDIOMA encontrado en la referencia tprovincias0, la cual servirá únicamente para mostrar este campo, ya que cualquier alteración a este registro, no afectara a ninguna tabla de la BDD.

Asistentes

Un asistente es una propiedad de los Widgets que permite establecer la forma en que serán ingresados los datos en un ellos.

Los asistentes aplican para casi todos los Widgets, sin embargo, su funcionalidad es exitosa en elementos en los que se puedan ingresar un valor. (Input, TextArea, InputLabel, CheckBox, .. etc)



Tipos de Asistentes

Una lista completa se la puede encontrar en los [fuentes](#).

Plain Text

Permite ingresar los datos como texto plano. El ingreso será directamente escrito por el usuario. Es el asistente que toman todos los widgets por default.

Long Text

Permite ingresar los datos como texto plano. El ingreso será a través de un input box desplegable que podrá almacenar registros de varias líneas.
El input box desplegable se muestra al hacer click en la flecha [>] ubicada al final del widget.

File

Permite ingresar los datos como un directorio del equipo local.
En caso que el webpage requiera subir un archivo, hay que configurar transacción en la BDD para que permita subir archivos.

Order

Permite ingresar los datos como un criterio de orden para datasources.
El ingreso de los criterios de orden aparece al hacer click en la imagen de orden ubicada al final del widget.
Existen: Orden Ascendente [/] Orden Descendente [V] Sin Orden [o].

PopUp

Permite ingresar los datos como un mensaje desplegable de información [PopUp].

Se pueden editar las opciones de este asistente mediante el botón de configuración [...]:

Tab

Establece el numero del Tab en el que aparecerá el PopUp de ingreso.

Mensaje

Establece el mensaje de ingreso para el PopUp.

Password

Permite ingresar los datos como texto de password.
El carácter de password es por defecto [*].

Calendar

Permite ingresar los datos como una fecha, mediante un dialogo desplegable de selección.

Se pueden editar las opciones de este asistente mediante el botón de configuración [...]:

DateFormatter

Establece el [formateador de fecha](#) que utilizara este asistente.

ShowIcon

Indica si queremos que se muestre el icono para desplegar el dialogo de selección de fecha.

En caso de no habilitar el icono, el dialogo se desplegara al hacer doble click sobre el widget.

Scanner

Permite ingresar los datos como un arreglo de bytes (mediante una imagen escaneada).

Para escanear, se procede a presionar el botón de Scan, posicionada al final del widget y luego seleccionar la opción Escanear.

ListOfValues

Permite ingresar los datos mediante un sub-webpage que muestre una consulta sencilla de alguna tabla de la BDD.

Este asistente requiere definir una(s) referencia(s) y sus campos respectivos a mostrar.

Se pueden editar las opciones de este asistente mediante el botón de configuración [...]:

Titulo

Mensaje PopUp que se mostrara al pasar el mouse sobre el widget.

Referencias

La(s) referencia(s) que utilizara la Lista de Valores. La definición de referencias es la misma que para el WebPage. Véase [Referencias](#) para más información.

Campos

Los campos de la(s) referencias que utilizara la Lista de Valores para mostrar el Query.


Para agregar/quitar/ordenar campos, se procede igual que cualquier dialogo de ingreso de elementos.

Los campos de la Lista de Valores se comportan como los widgets del webpage, la característica es que todos estos campos deben tener un

datasource **obligatorio** asociado. Véase [DataSources](#) para más información.

Propiedades

Las primeras propiedades de los campos son referentes al [DataSource](#) asociado. Se omitirá esta parte.



The screenshot shows a dialog box titled 'Editar Propiedades' with a table of properties. The table has two columns: 'nombre' and 'valor'. The properties listed are:

| nombre | valor |
|------------------------------------|-------------------------------------|
| Tipo | Registro (RECORD) |
| Alias / Tabla (para descripciones) | tpaisesLOV |
| Campo | CPAIS |
| Comparador | LIKE |
| Dependencias | Abrir para editar ... |
| Título | Codigo |
| Name Widget | c1_cpais |
| Presentar | <input checked="" type="checkbox"/> |
| Auto consultar | <input checked="" type="checkbox"/> |
| Ancho | 0 |
| Requerido | <input checked="" type="checkbox"/> |

At the bottom of the dialog is a 'Cerrar' button.

Titulo

El titulo que se mostrara sobre el campo en la Lista de Valores.

Name Widget

El widget (nombre) al que se le ingresara el valor escogido para este campo desde la Lista de Valores.

Presentar

Indica si se visualizara el campo dentro de la Lista de Valores o no. Cabe recalcar que, si el campo no se muestra, este utilizara el valor que tenga su Name Widget asociado, como criterio de filtro. Caso contrario no lo hará. Esto es útil para filtrar las consultas.

Auto Consultar

Indica si se auto consultan los registros desde la Lista de Valores, al momento que el Name Widget asociado pierda el foco.

Ancho

El ancho de la columna que mostrara el campo dentro de la Lista de Valores.

Requerido

Indica si este campo es requerido para hacer la consulta. Se utiliza en caso que el campo no se presente, caso contrario, no tiene uso.

PreQuery

Secuencia JavaScript que se ejecutara antes de mostrar la Lista de Valores. En esta sección se pueden hacer validaciones para mostrar o no el Asistente, así como también paso de valores a variables o filtros avanzados.

CallBack

Secuencia JavaScript que se ejecutara después de seleccionar un valor de la Lista de Valores. En esta sección se pueden hacer validaciones posteriores, así como también paso de valores a variables o llamar a otras funciones.

Subsistema

El subsistema que identifica a la Lista de Valores, generalmente es el 01.

Transacción

La transacción del subsistema que identifica a la Lista de Valores, generalmente es la 0003.

Versión

La versión de la transacción que identifica a la Lista de Valores, generalmente es la 01.

Existen casos en que se debe llamar a Listas de Valores generadas especialmente para que trabajen con alguna clase de control antes de hacer la consulta. En este caso, se deben llenar sus datos (Subsistema, Transacción, Versión) en los campos descritos anteriormente.

AutoListOfValues

Permite ingresar los datos mediante un sub-webpage que muestre una consulta sencilla de alguna tabla de la BDD.

A diferencia de ListOfValues, este asistente depende directamente que los datasources de sus widgets relacionados estén bien establecidos.

Ejemplo

Si queremos mostrar una Lista de Valores de los países de la tabla TPAISES, y que al seleccionar una opción, sus datos de CPAIS se guarde en el widget

[c1_cpais] y DESCRIPCION se guarde en [c1_dpais], pues deberíamos hacer lo siguiente:

1. Establecer para el widget [c1_cpais] un DataSource tipo RECORD a la tabla referenciada por el WebPage, hacia el campo CPAIS. Sin dependencias.
2. Establecer para el widget [c1_dpais] un DataSource tipo DESCRIPTION a la tabla TPAISES, para el campo DESCRIPCION con una dependencia hacia CPAIS desde cualquier referencia a nivel del webpage que también contenga el campo CPAIS.

Una vez hecho esto, ya debería funcionar la AutoListOfValues funcional para ambos widgets.

Este asistente no necesita configurar su referencia ni campos. Se recomienda usar este, antes de trabajar con una ListOfValues regular.

Link y Link Tab Bar

Permite llamar a un formulario, además permite pasar parámetros de un formulario a otro mediante la propiedad Valores y pasar objetos de JavaScripts mediante la propiedad Objetos js, dichos objetos se reciben en el formulario destino mediante la ejecución de la propiedad PosLink. Por ejemplo:

Objeto js:

```
[ c.$('campo1'), c.$V('campo2') ]
```

PosLink:

```
if (options.jsObject[1].length = 5) {  
    c.$('otroCampo').changeValue(options.jsObject[0][0].value);  
}
```

COMPORTAMIENTOS JS Y LINKTABBAR

Un formateador controla que la información ingresada en un Widget, cumpla con ciertas especificaciones de validación mientras que un behavior JS controla la forma en que se comportara el widget al momento de hacer click.

Tipos de Formateadores

IpFormatter

Controla que los datos ingresados en el Widget cumplan un formato de IP [#0-255.#0-255.#0-255.#0-255]

TextFormatter

Controla que los datos ingresados en el Widget cumplan un formato de texto establecido por una expresión regular.

Este formateador permite modificar su expresión, mediante la opción de configuración [...]:

Texto

Permite especificar el texto de expresión regular

NumberFormatter

Controla que los datos ingresados en el Widget cumplan un formato de numero establecido por una expresión regular.

Este formateador permite modificar su expresión, mediante la opción de configuración [...]:

Formato

Permite especificar el formato de expresión regular, por default es: [-#,###.00]

UpperCaseFormatter

Mantiene en formato mayúsculas, el texto ingresado en el Widget.

RegExFormatter

Controla que los datos ingresados en el Widget cumplan un formato establecido por una expresión regular compleja.

Este formateador permite modificar su expresión, mediante la opción de configuración [...]:

Formato

Permite especificar el formato de expresión regular.

DateFormatter

Controla que los datos ingresados en el Widget cumplan un formato de fecha establecido un formato de fecha regular.

Este formateador permite modificar su expresión, mediante la opción de configuración [...]:

Formato

Permite especificar el formato de fecha a controlar, por default: `TIMESTAMP`

Existen 4 tipos de formateadores de fecha:

TIME

Si necesitamos ingresar únicamente formatos de hora sencilla [HH:mm:ss].

DATE

Si necesitamos ingresar únicamente formatos de fecha sencilla [YYYY/MM/DD].

TIMESTAMP

Si necesitamos ingresar una fecha en formato TimeStamp [YYYY/MM/DD HH:mm:ss.ms Z+GMT].

DATETIME

Si necesitamos ingresar una fecha en formato Fecha-Hora [YYYY/MM/DD HH:mm:ss].

TIPS PARA ACCEDER A DATOS DE LA BDD

Paso a paso

Aquí se indicarán algunas estrategias para poder armar un WebPage que consulte de la manera que se requiere:

Desde el WebPage hasta el SQL

1. Definir las [referencias](#) que se van a usar en el formulario
2. Definir las [dependencias](#) entre las referencias y sus tipos
3. Crear campos en el formulario y definir su [datasource](#)

Desde el SQL hasta el WebPage

1. Definir cuantas consultas SQL se necesitan hacer
2. Definir para cada consulta las tablas, campos y joins necesarios
 - Para cada tabla consultada crear una [referencia](#)
 - Para cada join necesario crear una [dependencia](#) tipo IMMEDIATE
 - Para cada campo consultado crear un elemento el WebPage con un [datasource](#) adecuado
3. Para funcionalidad maestro-detalle usar [dependencias](#) con tipo DEFERRED

4. Si existen criterios que tienen que copiarse entre consultas usar [dependencias](#) con tipo CRITERION

Definiciones Especificas

Reference

Elemento que define una TABLA y un ALIAS a esa tabla. También contiene una lista de dependencias a otros alias (o a valores inmediatos). Es lo primero que se tiene que definir al crear un WebPage.

Ejemplo 1: Consulta y mantenimiento de provincias

Referencia 1:

```
alias=tprovincias1 table=TPROVINCIAS
```

Referencia 2:

```
alias=tpaises1      table=TPAISES dependencias=[ tprovincias1.CPAIS =  
CPAIS ]
```

Ejemplo 2: Consulta y mantenimiento de ciudades

Referencia 1:

```
alias=tciudades1    table=TCIUDADES
```

Referencia 2:

```
alias=tprovincias1 table=TPROVINCIAS dependencias=[ tciudades1.CPAIS =  
CPAIS, tciudades1.CPROVINCIA = CPROVINCIA ]
```

Referencia 3:

```
alias=tpaises1      table=TPAISES      dependencias=[ tciudades1.CPAIS =  
CPAIS ]
```

Dependency

Una dependencia define las condiciones del JOIN entre varias referencias o de un select anidado para el caso de DESCRIPCIONES. Las dependencias pueden ser de estos tipos:

Inmediatas (IMMEDIATE)

las que sirven para JOINS o selects internos

Diferidas (DEFERRED)

para maestro-detalle

Criterio (CRITERION)
para copiar valores de criterios entre tablas

Ejemplos:

Dependencia 1 en una referencia tpaíses1:

```
fromAlias=tciudades1 fromField=CPAIS field=CPAIS
```

Dependencia 2 en una referencia tprovincias1:

```
fromAlias=tciudades1 fromField=CPROVINCIA field=CPROVINCIA
```

Dependencia 3 en una descripción de TIDIOMAS.descripcion:

```
fromAlias=tciudades1 fromField=CIDIOMA field=CIDIOMA
```

DataSource

Un datasource (anteriormente OrigenDB) es el que se encarga de definir de que parte del query se obtendrá los valores. Cabe recalcar que ya no es un String si no un objeto. En varios lugares del editor y del generador aparece como un String solo como una guía para saber que contiene el objeto.

Tipos de DataSource

CRITERION

El valor que tenga el widget, será tomado como un criterio de filtro al momento de hacer una consulta o mantenimiento.

DESCRIPTION

Útil cuando queremos mostrar un registro como descripción de un FK que estamos consultando. El valor que tenga este widget al momento de hacer una consulta o mantenimiento no alterara en nada el proceso.
Este tipo de datasource **requiere** de alguna dependencia para mostrar la descripción del registro en la tabla actual.

NONE

Cuando no necesitamos que el widget obtenga valores de la BDD. Cabe recalcar que los valores que tenga este widget, se borrarán al momento de hacer una consulta o mantenimiento.
Todos los widgets tienen este tipo de datasource por defecto.

ORDER

Indica que el campo que hemos establecido en el datasource será tomado como criterio de orden
El orden será tomado en forma ascendente.

RECORD

En caso de un proceso de Consulta, este datasource muestra el registro que tiene ese campo para esa referencia en la tabla.

En caso de un proceso de Mantenimiento, este datasource guarda el valor que tenga el widget, sobre el registro que tiene ese campo para esa referencia en la tabla.

Cabe recalcar, que FIT no sobrescribe registros, en tal caso, caduca el registro anterior y crea uno nuevo con fecha vigente (Fhasta=2999-12-31)

CONTROL o CRITERION_CONTROL

El valor que tenga el widget, será enviado a una clase de control asociada al WebPage, para procesos especiales de consulta o mantenimiento.

Este tipo de datasource también se usa cuando no queremos que se pierda el valor que tenga este widget al momento de hacer una consulta o mantenimiento.

En este caso, todos los campos de control deben tener nombres (Campo) diferentes.

CRITERION_DESCRIPTION

Igual que el tipo DESCRIPCION, con la diferencia que el valor que tenga este widget, si será utilizado como criterio de filtro a su respectivo FK de la tabla del webpage.

Ejemplo: Consulta y mantenimiento de ciudades

Campo 1:

```
type=CRITERION alias=tciudades1 field=CPAIS
```

Campo 2:

```
type=CRITERION alias=tpaises1 field=DESCRIPCION
```

Campo 3:

```
type=CRITERION alias=tciudades1 field=CPROVINCIA
```

Campo 4:

```
type=ORDER alias=tpaises1 field=NOMBRE
```

Campo 5:

```
type=RECORD      alias=tciudades1 field=CCIUDAD
```

Campo 6:

```
type=RECORD      alias=tpaises1   field=SIGLAS
```

Campo 7:

```
type=RECORD      alias=tpaises1   field=NOMBRE
```

También se pueden definir campos tipo descripción para evitar tener que crear una referencia solo para agregar un campo que no va a ser insertado. Esto se logra de este modo:

Campo 1:

```
type=DESCRIPTION alias=TIDIOMAS field=DESCRIPCION  
dependencies=[ tciudades1.CPAIS = CPAIS ]
```

Además se pueden definir campos de control usando type=CONTROL. También no es necesario crear un alias para la tabla ID relacionada a cualquier tabla. Simplemente se setea el campo useTableId=true y el sistema se encarga de todo.

PROCESOS

Mantenimientos

Para este proceso se agrega un objeto Table al Detail por cada [referencia](#) que tenga el WebPage. Se agregan los registros con cambios y se eliminan las tablas vacías. El proceso interno es el mismo que se usa en Fit.

Consulta

De WebPage a Detail

El proceso que se ejecuta es el siguiente:

1. Se agrupa las [referencias](#) de acuerdo a las dependencias
2. Se crea un Detail
3. Para cada grupo de referencias se crea un Table en el Detail usando los datos de la referencia principal del grupo

4. Para cada referencia del grupo que no sea la principal se crea un Join dentro del objeto Table y se crea un Dependence por cada **dependencia** inmediata que exista en esa referencia.
5. Para cada **dependencia** diferida se crea un Dependence a nivel de la tabla.
6. Para cada **dependencia** criterio se copia el criterio correspondiente entre las tablas indicadas.

Proceso de agrupar referencias

Este proceso lo ejecuta la clase ArbolDependencias.

Grupo de referencias

Un grupo de referencias se convertirá posteriormente en una sola tabla en el Detail.

Referencia principal

Debe ser única **referencia** en el grupo. No tiene **dependencias** inmediatas a otras **referencias** y se usará como tabla base para la consulta. Puede tener **dependencias** diferidas.

Referencia secundaria

Es cualquier **referencia** que no sea principal. Se agrega al grupo de la **referencia** principal si existe una **dependencia** inmediata (o una cadena de dependencias inmediatas) entre esta y la principal.

De Detail a SQL

Antes de iniciar el proceso de generar el SQL se ejecutan las siguientes acciones:

- Se agrega criterio FHASTA solo **si no se especificó uno** para obtener solo registros activos. Si se requiere obtener registros no activos se debe definir este Criterion (con o sin valor) en el objeto Table.
- Se agrega criterio CIDIOMA y/o CIDIOMA_TEXTO si es necesario solo **si no se especificó uno** para obtener solo registros que tengan el idioma especificado en el header del Detail. Si se requieren obtener campos de todos los idiomas o de idiomas específicos hay que especificarlo con un Criterion (con el valor adecuado o sin valor) en el objeto Table.
- Se completan todos los PK en el Record de consulta en caso de faltar y se setea field.primaryKey="1" para estos campos
- Se agrega el campo de VERSIONCONTROL si es necesario en el Record de consulta
- Se encuentran campos que en realidad pertenecen a la tabla Id y se definen como field.type = FieldType.INNER_SELECT_ID

Se arma un query SQL por cada objeto Table que se envíe en el Detail de la siguiente forma:

```
SELECT {CAMPOS}
FROM {table.name} {table.alias} [JOIN {table.joins[n]} ...]
WHERE {CRITERIOS}
ORDER BY {CRITERIOS DE ORDEN}
```

Este query se limita a obtener table.requestedRecords si este valor es diferente de 0, si es = 0 se obtiene solo un valor.

Para obtener páginas posteriores hay que usar table.requestedPage.

Campos

En esta sección se agregan todos los campos enviados como Fields en el Detail.

```
{field.alias}.{field.name} {aliasConsulta}
```

En caso de enviar un Field que tenga type = FieldType.INNER_SELECT se insertará un select anidado de la siguiente forma:

```
(SELECT {field.name} FROM {field.alias} WHERE  
{CONDICIONES:field.dependencies}) {aliasConsulta}
```

Donde las condiciones se crean a partir de los objetos Dependence dentro del Field.

Joins

Esta sección se arma de la siguiente manera:

```
{join.name} {join.alias} ON {CONDICIONES:join.dependencies}
```

Condiciones

Si no tienen valor se arman de la siguiente manera:

```
{dependence.fromAlias}.{dependence.from} =  
{dependence.toAlias}.{dependence.to}
```

Si tiene valor se arma así:

```
{dependence.fromAlias}.{dependence.from} = {dependence.value}
```

Criterios

En esta sección se agregan todos los objeto tipo Criterium enviados en el objeto Table. Si tienen valor se agregan de la siguiente manera:

```
{criterion.alias}.{criterion.name} {criterion.condition}  
{criterion.value}
```

Si la condición es IS NULL o es IS NOT NULL se agregan así:

```
{criterion.alias}.{criterion.name} {criterion.condition}
```

Cabe recalcar que los criterions que no tienen valor no se agregan al query. Además se pueden agregar varios criterios del mismo campo con diferentes condiciones (como por ejemplo para obtener un rango de fechas).

Formulas en webpages

En todo widget que pueda tener valor (Input, InputLabel, Checkbox, etc) se puede definir el valor como una expresión usando un '=' como primer carácter, de la misma manera en que se hace en hojas de calculo. Al hacer esto el campo automáticamente se marcará como no modificable en el cliente y su valor será calculado continuamente.

Operaciones

Se pueden realizar operaciones matemáticas como:

- CAMPO1 + CAMPO2
- CAMPO1 - CAMPO2
- CAMPO1 * CAMPO2
- CAMPO1 / CAMPO2

También se pueden hacer operaciones booleanas:

- CAMPO1 < CAMPO2
- CAMPO1 != CAMPO2
- CAMPO1 >= CAMPO2
- CAMPO1 == CAMPO2
- etc

Referencias a campos

Para referenciar un campo solo se necesita usar el nombre del campo directamente.

Por ejemplo si ponemos a un campo F4Total el valor '=F4Valor' significa que el campo F4Total va a tomar el valor de F4Valor. Si este campo es un campo multiregistro va a tomar el valor del mismo registro que F4Total, lo que es equivalente en javascript a `F4Total[$D].value = F4Valor[$D].value` siendo \$D el registro del campo.

Referencias al detail

Para referenciar al detail se lo hace usando el nombre del campo en la clase java de [TransporteDB](#).

Ej:

- \$company
- \$originBranch

Strings

Se puede usar strings directamente. Para mas información ver [FormulaParser](#) y [FormulaTestCase](#). Ej:

- "ABC"
- 'abc\'def'

Otras funciones

- **IF(CONDICION; VALOR_1; VALOR2)**: Devuelve el valor indicado dependiendo de la condición
- **EMPTY(CAMPO1)**: Determina si un campo está vacío
- **NOT(EXPRESION)**: Devuelve el valor negado de la expresión
- **SUM(CAMPO1)**: Suma los valores de un multiregistro
- **POWER(CAMPO1, POTENCIA)**: Devuelve el valor de la potencia
- **ROUND(CAMPO1, DIGITOS)**: Devuelve el valor redondeado dado el número de dígitos
- **MAX(CAMPO1, CAMPO2)**: Devuelve el máximo entre 2 valores o un arreglo de datos.
- **AGE(CAMPO1)**: Calcula la edad dada una fecha en el campo
- **FULL_AGE(CAMPO1)**: Calcula la edad dada una fecha en el campo y la presenta con el formato "X años y Y meses"
- **NOW()**: Fecha actual con hora
- **TODAY()**: Fecha actual sin hora
- **DATEDIF(CAMPO1, CAMPO2, TIPO)**: Calcula la diferencia de fechas, ver [documentación](#)
- **YEAR(CAMPO1)**: Obtiene el año del campo
- **MONTH(CAMPO1)**: Obtiene el mes del campo
- **DAY(CAMPO1)**: Obtiene el día del campo
- **HOUR(CAMPO1)**: Obtiene la hora del campo
- **MINUTE(CAMPO1)**: Obtiene el minuto del campo
- **SECOND(CAMPO1)**: Obtiene el segundo del campo
- **DATE(VALOR_AÑO; VALOR_MES; VALOR_DIA)**: Crea un objeto tipo Date
- **TIME(VALOR_HORA; VALOR_MINUTO; VALOR_SEGUNDO)**: Crea un objeto tipo Date con la hora
- **CONCATENATE(VALOR1, VALOR2, ...)**: Junta los valores indicados en un solo string
- **TEXT(VALOR, FORMATO)**: Formatea el valor. El formato usado es el mismo de [SimpleDateFormatter](#) o de [NumberFormatter](#)

USO DE JAVASCRIPT

Advertencia: Es muy recomendado usar **Formulas**, **Asistentes** o **Comportamientos JS** en vez de usar directamente JavaScript en un formulario!

Acceso a elementos con c.\$

Para acceder a un elemento se puede hacer de esta manera:

```
c.$('nombreElemento', registro)
```

Asignación de valor a un campo

Para asegurarse de que un campo va a cambiar su valor correctamente y ejecutar todas las validaciones necesarias, formateo del campo, etc. es importante ejecutar esto:

```
c.$('nombreElemento').changeValue(nuevoValor);
```

Por ejemplo:

```
c.$('F2fechaActual').changeValue(new Date())
```

Obtener el valor de un campo

En cualquier elemento se puede llamar la siguiente función. Esto obtiene el valor ya sea numérico, fecha o string:

```
c.$V('nombreElemento')
```

Uso de números

Se debe asignar un formateador numerico al elemento y se pueden usar estos dos estilos:

```
c.$('nombreElemento').intValue  
c.$('nombreElemento').floatValue
```

El primero obtiene el valor entero del campo, el segundo el valor flotante.

Uso de fechas

Se debe asignar un formateador de fechas al elemento y se usa de la siguiente manera:

```
c.$('nombreElemento').dateValue
```

Habilitar o deshabilitar un campo

Para deshabilitar ejecutar lo siguiente:

```
c.$('nombreElemento').setDisabled(true);
```

Para habilitar:

```
c.$('nombreElemento').setDisabled(false);
```

Implementación

- El uso de javascript con relación a los widgets es transparente, es decir, el diseñador no debe saber ninguno de los siguientes mecanismos para poder usarlo.
 - Todo widget se compone de 2 componentes: el elemento visible (widget) y si es necesario, un elemento oculto para mantener el valor
 - Todos los eventos deben registrarse sobre el widget
 - Existe un mecanismo de sincronización de valores entre el widget y el oculto
- Consideraciones de usos historicos de javascript en los webpages**

Anteriormente, en algunos webpages se usaban ciertos javascripts que ahora se consideran deprecados. El soporte para estas formas de uso podría ser eliminado en un futuro. Anteriormente c.\$ devolvía el elemento oculto pero se redifinió para que devuelva el widget. Sin embargo **se sigue soportando** los casos de uso anteriores especificados a continuación:

- Se mapea las funciones **hide** y **show** desde los ocultos a los widgets.
- Sobre un elemento tipo checkbox o combobox se podía ejecutar lo siguiente:
 - **c.\$('elemento').checkbox|combobox**: Ahora se puede usar directamente c.\$('elemento') ya que devuelve el checkbox
 - **c.\$('elemento_checkbox|combobox')**: Ahora se puede usar directamente c.\$('elemento') ya que devuelve el checkbox
 - **c.\$('elemento').value**: Este valor toma el valor adecuado al sincronizarse con el oculto, anteriormente c.\$('elemento').checkbox.value devolvía siempre "on"
 - **c.\$('elemento').checked**: Este valor toma el valor adecuado al sincronizarse con el oculto
- Sobre otros elementos se podía ejecutar **c.\$('elemento').widget**. Ahora se puede usar directamente c.\$('elemento') ya que devuelve el widget

- Las llamadas a **c.\$('elemento').fireDOMEvent** se soportan directamente
- Las llamadas a **setDisabled** y otras funciones seguirán funcionando sobre los widgets.

CONSEJOS PARA ARMAR WEBPAGES

Para poder consultar

Para poder consultar es solo necesario crear en el formulario los campos que se van a necesitar presentar, no es necesario crear ningún campo oculto, ni siquiera del pk.

Para poder mantener

Para poder actualizar un registro existente no es necesario crear ningún campo oculto, ni siquiera del pk. Para poder insertar un nuevo registro es necesario que exista el pk de una referencia en una combinación de criterios y campos de esa referencia. En este caso al momento de insertar se dará prioridad al valor encontrado en un campo tipo RECORD antes que un CRITERION.

Mezcla de tipos de campos en containers

No es recomendable mezclar campos de criterios (CRITERION, CRITERION_DESCRIPTION, ORDER, CONTROL) con campos de registros (RECORD, DESCRIPTION) en el mismo container, esto seguramente causará problemas al consultar o mantener.

Campos con valores por defecto

NOTA: Esto ha sido mantenido solo para CIDIOMA y CIDIOMA_TEXTO, el resto está en discusión si se debe mantener o no.

Existen varios campos de las tablas que, si no les asociamos un valor (referenciar el datasource a un widget en el webpage) obtienen valores por defecto para la aplicación, los mas comunes son:

CPERSONA_COMPANIA = 2
CIDIOMA = ES
CIDIOMA_TEXTO = ES
CMONEDA = USD
CUSUARIO = [codigo de usuario de la sesion actual]
FHASTA = 2999-12-31 00:00:00.000

CSUCURSAL = [codigo de sucursal definido para la cpersona_compania]
COFICINA = [codigo de la oficina definida para la cpersona_compania]
CROL = [codigo del rol definido para el cusuario de la sesion actual]

En caso de referenciar algunos de estos campos como un datasource para algún widget, estos tomaran el valor enviado por el widget en vez de su valor por defecto. Estos campos no son necesarios agregarlos ni ponerlos como campos ocultos. Obtienen su valores por defecto sin necesidad de llamarlos.

Mantenimiento de PK's

En FIT, no esta permitido realizar mantenimiento a campos que son Primary Key de la tabla en referenciada. Esto muy seguramente devolverá un error:

EL REGISTRO FUE MODIFICADO POR OTRO USUARIO. POR FAVOR RECONSULTE.

En ese caso, hay que limitar la edición de este campo por todos los medios posibles:

- *Permitir el ingreso de este campo únicamente mediante ListOfValues, que tome como filtro el valor actual del widget, para que de esta manera, no permita seleccionar otra opción ademas de la que ya esta guardada en este registro*
- *El widget que muestre la información de este campo, debe ser NO MODIFICABLE*
- *Efectuar un proceso interno JS que impida el mantenimiento de un PK, o a su vez, que realice la caduacion y creacion del nuevo registro*

EJEMPLO DE ARMADA DE DETAILS Y SQLs

Consulta simple a una tabla

WebPage:

```
<?xml versión="1.0" encoding="UTF-8"?>
<webPage>
  <properties sub="00" tra="0001" type="MAN">
    <title>Ejemplo</title>
    <references>
      <reference alias="tpaises0" class="com.fitbank.webpages.data.Reference"
        table="TPAISES" />
    </references>
  </properties>
  <containers>
    <container clo="T" max="10" win="10">
      <widget class="com.fitbank.webpages.widgets.Label" tex="Siglas" />
      <widget class="com.fitbank.webpages.widgets.Label" tex="Descripción" />
      <widget class="com.fitbank.webpages.widgets.HeaderSeparator" />
    </container>
  </containers>
</webPage>
```

```

<widget class="com.fitbank.webpages.widgets.Input" w="100">
  <dataSource alias="tpaises0"
    class="com.fitbank.webpages.data.DataSource" field="siglas"
    type="RECORD" />
</widget>
<widget class="com.fitbank.webpages.widgets.Input" w="100">
  <dataSource alias="tpaises0"
    class="com.fitbank.webpages.data.DataSource" field="descripción"
    type="RECORD" />
</widget>
</container>
</containers>
</webPage>

```

Detail:

```

<FITBANK>
  <DET>
    <TBL alias="tpaises0" blq="0" financial="false" mpg="0" name="TPAISES"
      npg="1" nrg="10" ract="0" readonly="false" special="false">
      <REG numero="0">
        <CAM alias="tpaises0" name="siglas" pk="0" />
        <CAM alias="tpaises0" name="descripción" pk="0" />
      </REG>
    </TBL>
  </DET>
</FITBANK>

```

SQL:

```

SELECT
  tpaises0.siglas,
  tpaises0.descripcion
FROM TPAISES tpaises0
WHERE [FHASTA AND CIDIOMA]

```

Consulta simple a dos tablas

WebPage:

```

<?xml versión="1.0" encoding="UTF-8"?>
<webPage>

```

```

<properties sub="00" tra="0001" type="MAN">
  <title>Ejemplo</title>
  <references>
    <reference alias="tpaises0" class="com.fitbank.webpages.data.Reference"
      table="TPAISES" />
    <reference alias="tprovincias0" class="com.
fitbank.webpages.data.Reference"
      table="TPROVINCIAS">
  <dependencies>
    <item class="com. fitbank.webpages.data.Dependency" field="cpais"
      fromAlias="tpaises0" fromField="cpais" />
  </dependencies>
</reference>
</references>
</properties>
<containers>
  <container clo="T" max="10" win="10">
    <widget class="com.fitbank.webpages.widgets.Label" tex="Siglas" />
    <widget class="com.fitbank.webpages.widgets.Label" tex="Descripción" />
    <widget class="com.fitbank.webpages.widgets.Label" tex="Código" />
    <widget class="com.fitbank.webpages.widgets.Label" tex="Nombre" />
    <widget class="com.fitbank.webpages.widgets.HeaderSeparator" />
    <widget class="com.fitbank.webpages.widgets.Input" w="100">
      <dataSource alias="tpaises0"
        class="com.fitbank.webpages.data.DataSource" field="siglas"
        type="RECORD" />
    </widget>
    <widget class="com.fitbank.webpages.widgets.Input" w="100">
      <dataSource alias="tpaises0"
        class="com.fitbank.webpages.data.DataSource" field="descripción"
        type="RECORD" />
    </widget>
    <widget class="com.fitbank.webpages.widgets.Input" w="100">
      <dataSource alias="tprovincias0"
        class="com.fitbank.webpages.data.DataSource" field="cprovincia"
        type="RECORD" />
    </widget>
    <widget class="com.fitbank.webpages.widgets.Input" w="100">
      <dataSource alias="tprovincias0"
        class="com.fitbank.webpages.data.DataSource" field="nombre"
        type="RECORD" />
    </widget>
  </container>
</containers>

```

```
</webPage>
```

Detail:

```
<FITBANK>
  <DET>
    <TBL alias="tpaises0" blq="0" financiero="false" mpg="0" name="TPAISES"
      npg="1" nrg="10" ract="0" readonly="false" special="false">
      <JOIN alias="tprovincias0" name="TPROVINCIAS">
        <DEP aliasDesde="tpaises0" desde="cpais" hacia="cpais" />
      </JOIN>
      <REG numero="0">
        <CAM alias="tpaises0" name="siglas" pk="0" />
        <CAM alias="tpaises0" name="descripción" pk="0" />
        <CAM alias="tprovincias0" name="cprovincia" pk="0" />
        <CAM alias="tprovincias0" name="nombre" pk="0" />
      </REG>
    </TBL>
  </DET>
</FITBANK>
```

SQL:

```
SELECT
  tpaises0.siglas,
  tpaises0.descripcion,
  tprovincias0.cprovincia,
  tprovincias0.nombre
FROM TPAISES tpaises0 JOIN TPROVINCIAS tprovincias0 ON tpaises0.cpais =
tprovincias0.cpais
WHERE [FHASTA AND CIDIOMA]
```

Consulta maestro-detalle a dos tablas

WebPage:

```
<?xml versión="1.0" encoding="UTF-8"?>
<webPage>
  <properties sub="00" tra="0001" type="MAN">
    <title>Ejemplo</title>
    <references>
      <reference alias="tpaises0" class="com.fitbank.webpages.data.Reference">
```

```

        table="TPAISES" />
        <reference alias="tprovincias0"
class="com.fitbank.webpages.data.Reference"
        table="TPROVINCIAS">
        <dependencies>
            <item class="com.fitbank.webpages.data.Dependency" field="cpais"
                fromAlias="tpaises0" fromField="cpais" type="DEFERRED" />
        </dependencies>
        </reference>
    </references>
</properties>
<containers>
    <container clo="T" max="10" win="10">
        <widget class="com.fitbank.webpages.widgets.Label" tex="Siglas" />
        <widget class="com.fitbank.webpages.widgets.Label" tex="Descripción" />
        <widget class="com.fitbank.webpages.widgets.HeaderSeparator" />
        <widget class="com.fitbank.webpages.widgets.Input" w="100">
            <dataSource alias="tpaises0"
                class="com.fitbank.webpages.data.DataSource" field="siglas"
                type="RECORD" />
        </widget>
        <widget class="com.fitbank.webpages.widgets.Input" w="100">
            <dataSource alias="tpaises0"
                class="com.fitbank.webpages.data.DataSource" field="descripción"
                type="RECORD" />
        </widget>
    </container>
    <container clo="T" max="10" win="10">
        <widget class="com.fitbank.webpages.widgets.Label" tex="Código" />
        <widget class="com.fitbank.webpages.widgets.Label" tex="Nombre" />
        <widget class="com.fitbank.webpages.widgets.HeaderSeparator" />
        <widget class="com.fitbank.webpages.widgets.Input" w="100">
            <dataSource alias="tprovincias0"
                class="com.fitbank.webpages.data.DataSource" field="cprovincia"
                type="RECORD" />
        </widget>
        <widget class="com.fitbank.webpages.widgets.Input" w="100">
            <dataSource alias="tprovincias0"
                class="com.fitbank.webpages.data.DataSource" field="nombre"
                type="RECORD" />
        </widget>
    </container>
</containers>
</webPage>

```


Detail:

```
<FITBANK>
  <DET>
    <TBL alias="tpaises0" blq="0" financial="false" mpg="0" name="TPAISES"
      npg="1" nrg="10" ract="3" readonly="false" special="false">
      <REG numero="0">
        <CAM alias="tpaises0" name="siglas" pk="0" />
        <CAM alias="tpaises0" name="descripción" pk="0" />
      </REG>
    </TBL>
    <TBL alias="tprovincias0" blq="0" financial="false" mpg="0"
      name="TPROVINCIAS" npg="1" nrg="10" ract="0" readonly="false"
      special="false">
      <REG numero="0">
        <CAM alias="tprovincias0" name="cprovincia" pk="0" />
        <CAM alias="tprovincias0" name="nombre" pk="0" />
      </REG>
      <DEP aliasDesde="tpaises0" aliasHacia="tprovincias0" desde="cpais"
        hacia="cpais" />
    </TBL>
  </DET>
</FITBANK>
```

SQL:

```
SELECT tpaises0.siglas, tpaises0.descripcion FROM TPAISES tpaises0 WHERE
[FHASTA AND CIDIOMA]
SELECT tprovincias0.cprovincia, tprovincias0.nombre FROM TPROVINCIAS
tprovincias0 WHERE tprovincias0.cpais = [SELECCIONADO ARRIBA RACT] AND
[FHASTA AND CIDIOMA]
```

Consulta simple con una descripción

WebPage:

```
<?xml versión="1.0" encoding="UTF-8"?>
<webPage>
  <properties sub="00" tra="0001" type="MAN">
    <title>Ejemplo</title>
    <references>
      <reference alias="tprovincias0"
class="com.fitbank.webpages.data.Reference"
```

```

        table="TPROVINCIAS">
    </reference>
</references>
</properties>
<containers>
    <container clo="T" max="10" win="10">
        <widget class="com.fitbank.webpages.widgets.Label" tex="Código" />
        <widget class="com.fitbank.webpages.widgets.Label" tex="Nombre" />
        <widget class="com.fitbank.webpages.widgets.Label" tex="Descripción
País" />
        <widget class="com.fitbank.webpages.widgets.HeaderSeparator" />
        <widget class="com.fitbank.webpages.widgets.Input" w="100">
            <dataSource alias="tprovincias0"
                class="com.fitbank.webpages.data.DataSource" field="cprovincia"
                type="RECORD" />
        </widget>
        <widget class="com.fitbank.webpages.widgets.Input" w="100">
            <dataSource alias="tprovincias0"
                class="com.fitbank.webpages.data.DataSource" field="nombre"
                type="RECORD" />
        </widget>
        <widget class="com.fitbank.webpages.widgets.Input" w="100">
            <dataSource alias="TPAISES"
                class="com.fitbank.webpages.data.DataSource" field="descripción"
                type="DESCRIPTION">
            <dependencies>
                <item class="com.fitbank.webpages.data.Dependency"
                    field="cpais" fromAlias="tprovincias0" fromField="cpais" />
            </dependencies>
        </dataSource>
        </widget>
    </container>
</containers>
</webPage>

```

Detail:

```

<FITBANK>
    <DET>
        <TBL alias="tprovincias0" blq="0" financiero="false" mpg="0"
            name="TPROVINCIAS" npg="1" nrg="10" ract="0" readonly="false"
            special="false">
        <REG numero="0">
            <CAM alias="tprovincias0" name="cprovincia" pk="0" />

```

```

        <CAM alias="tprovincias0" name="nombre" pk="0" />
        <CAM alias="TPAISES" name="descripción" pk="0" tipo="INNER_SELECT">
            <DEP aliasDesde="tprovincias0" desde="cpais" hacia="cpais" />
        </CAM>
    </REG>
</TBL>
</DET>
</FITBANK>

```

SQL:

```

SELECT
    tprovincias0.cprovincia,
    tprovincias0.nombre,
    (SELECT a0.descripcion FROM TPAISES a0 WHERE a0.cpais =
tprovincias0.cpais AND [FHASTA AND CIDIOMA])
FROM TPROVINCIAS tprovincias0
WHERE [FHASTA AND CIDIOMA]

```

Consulta de un campo de una tabla ID

WebPage:

```

<?xml versión="1.0" encoding="UTF-8"?>
<webPage>
    <properties sub="00" tra="0001" type="MAN">
        <title>Ejemplo</title>
        <references>
            <reference alias="tpaises0" class="com.fitbank.webpages.data.Reference"
                table="TPAISES" />
        </references>
    </properties>
    <containers>
        <container clo="T" max="10" win="10">
            <widget class="com.fitbank.webpages.widgets.Label" tex="Siglas" />
            <widget class="com.fitbank.webpages.widgets.Label" tex="Riesgo" />
            <widget class="com.fitbank.webpages.widgets.HeaderSeparator" />
            <widget class="com.fitbank.webpages.widgets.Input" w="100">
                <dataSource alias="tpaises0"
                    class="com.fitbank.webpages.data.DataSource" field="siglas"
                    type="RECORD" />
            </widget>
            <widget class="com.fitbank.webpages.widgets.CheckBox" w="100">
                <dataSource alias="tpaises0"

```

```

        class="com.fitbank.webpages.data.DataSource" field="paisriesgo"
        type="RECORD" />
    </widget>
</container>
</containers>
</webPage>

```

Detail:

```

<FITBANK>
  <DET>
    <TBL alias="tpaises0" blq="0" financial="false" mpg="0" name="TPAISES"
      npg="1" nrg="10" ract="0" readonly="false" special="false">
      <REG numero="0">
        <CAM alias="tpaises0" name="siglas" pk="0" />
        <CAM alias="tpaises0" name="paisriesgo" pk="0" />
      </REG>
    </TBL>
  </DET>
</FITBANK>

```

SQL:

```

SELECT
    tpaises0.siglas,
    (SELECT a0.paisriesgo FROM TPAISESID a0 WHERE a0.cpais = tpaises0.cpais
AND [FHASTA AND CIDIOMA])
FROM TPAISES tpaises0
WHERE [FHASTA AND CIDIOMA]

```