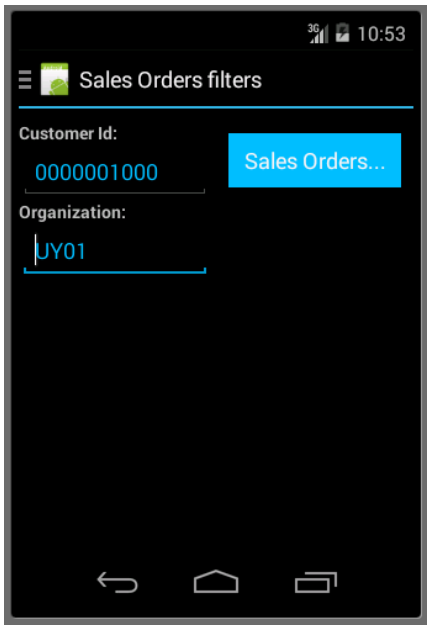
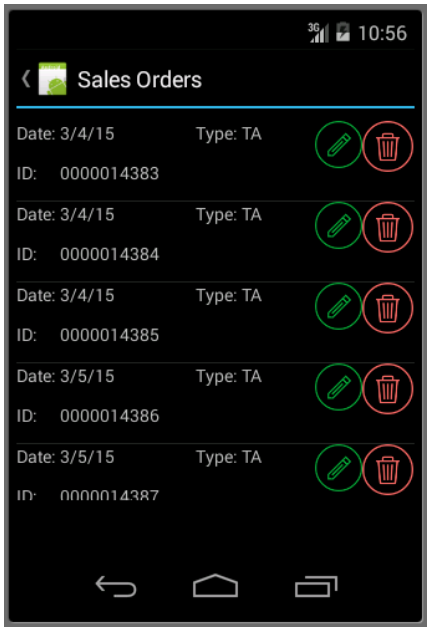
Cómo ingresar un pedido de venta en el SAP ERP desde la aplicación móvil creada con GeneXus

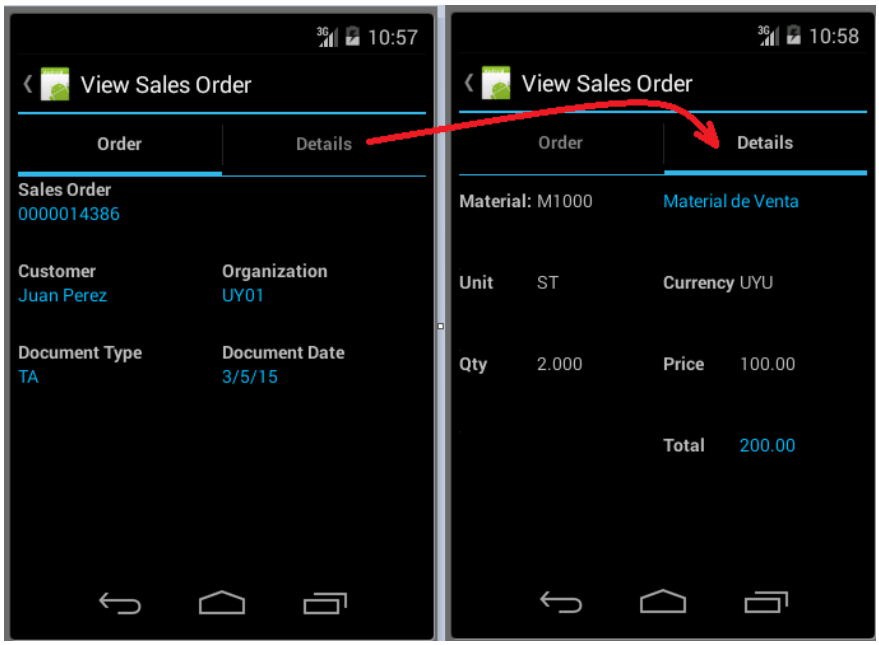
Partiremos de una aplicación en la que, además de poder trabajar con la información de los clientes y materiales, agregamos la posibilidad de, ingresando un cliente y organización (con estos valores por defecto, pero que pueden ser modificados por el usuario):



ver todos sus pedidos de venta (es decir, Trabajar con Sales Orders):

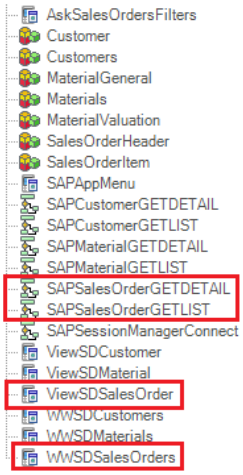


y eligiendo uno de ellos, ver su detalle:



(la información del cabezal de la Orden y sus líneas – aquí solamente una, pues el pedido se hace para un único material … si hubiera más, aparecerían aquí abajo).

La implementación de estas pantallas no difiere demasiado de las que ya vimos para los materiales y los clientes:



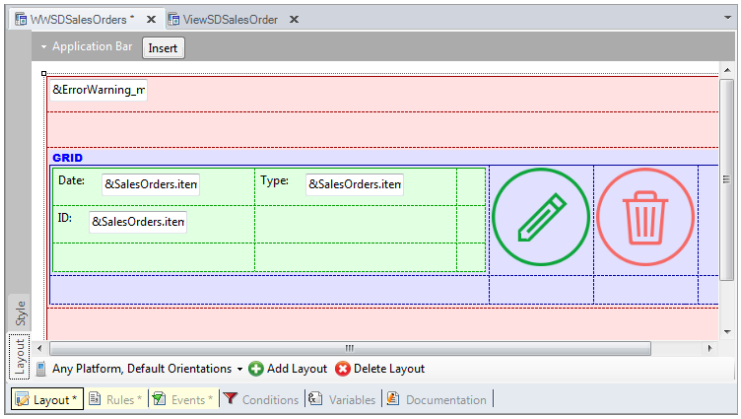
por lo que no entraremos en detalles.

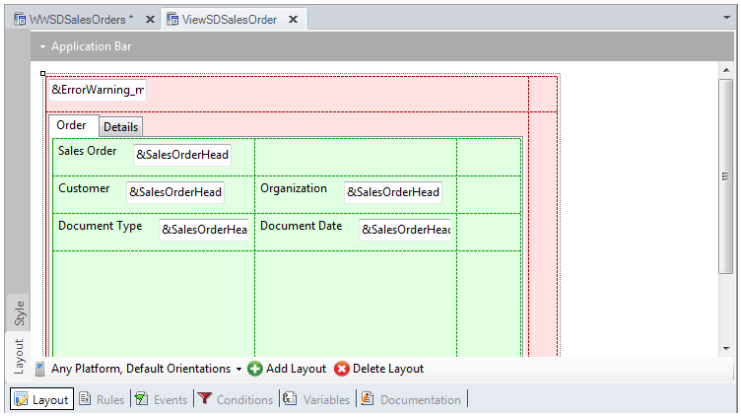
Aquí tenemos el Work With, con un grid que tiene como Default Action el evento “ViewOrder”, que invoca al panel de View, donde se muestra en dos solapas la información del pedido elegido.

(Puede importarse el xpz adjunto, SalesOrders.xpz, para incorporar estos objetos a su Knowledge base sin tener que desarrollarlos de cero. Deberá existir en su KB el procedmiento SAPSessionManagerConnect que realiza la conexión al SAP ERP. De lo contrario fallará la importación. El xpz SAPSessionManagerConnect contiene ese procedmiento pero con los datos de conexión vacíos. Deberá especificar allí los suyos.

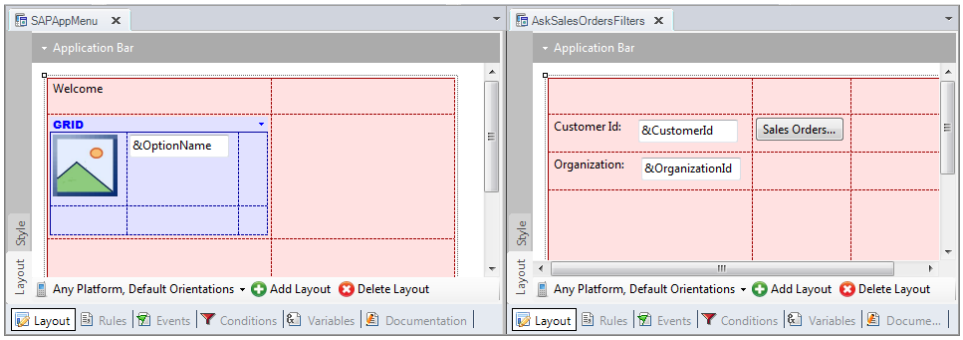
Si quiere importar en una KB vacía todos los objetos desarrollados hasta el momento, acceda al archivo KBStatePreCreateSalesOrder.xpz. Deberá abrirl el procedimiento

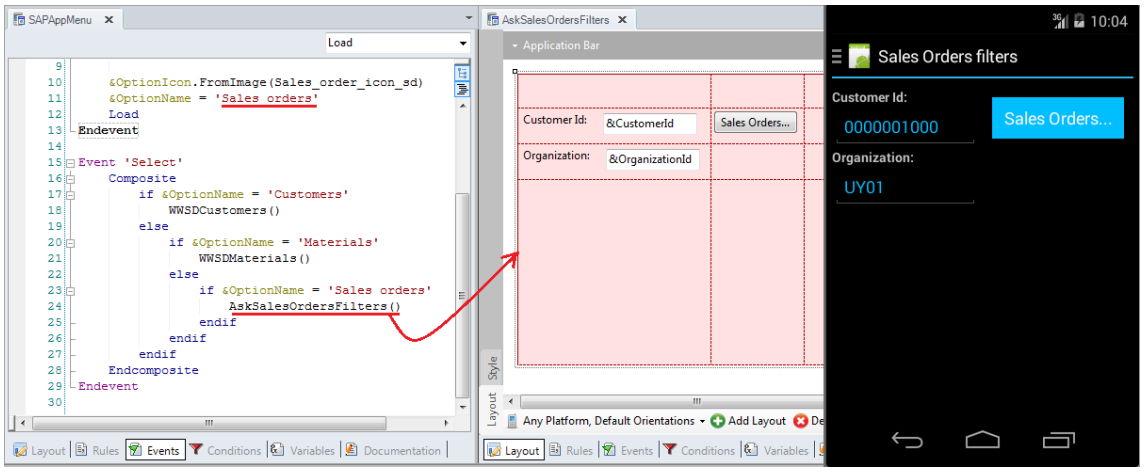
SAPSessionManagerConnect e ingresar los datos de conexión a su SAP ERP. De lo contrario fallará la ejecución pues están vacíos.)



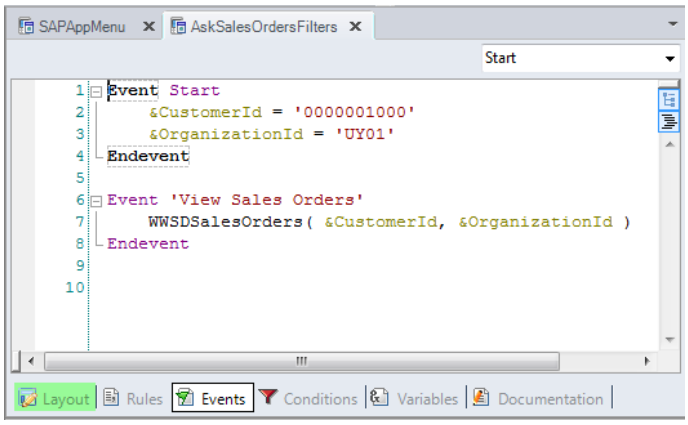


Observemos como novedad, que desde el menú de entrada no llamamos directamente al Work With, sino al panel intermedio AskSalesOrdersFilters:



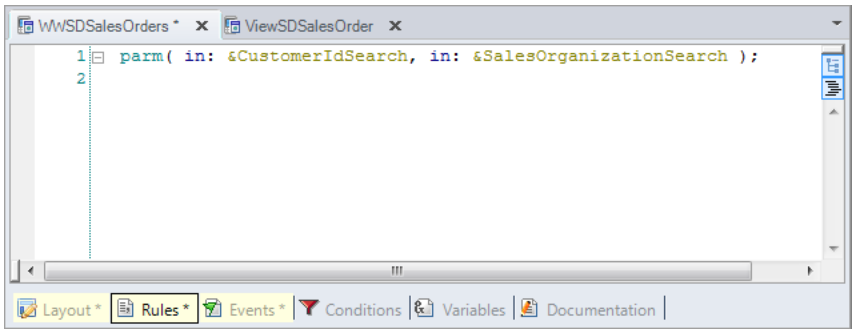


que tiene en el layout dos variables no Readonly, es decir, que el usuario puede ingresarles valor. Y vemos que se han inicializado en el evento Start:

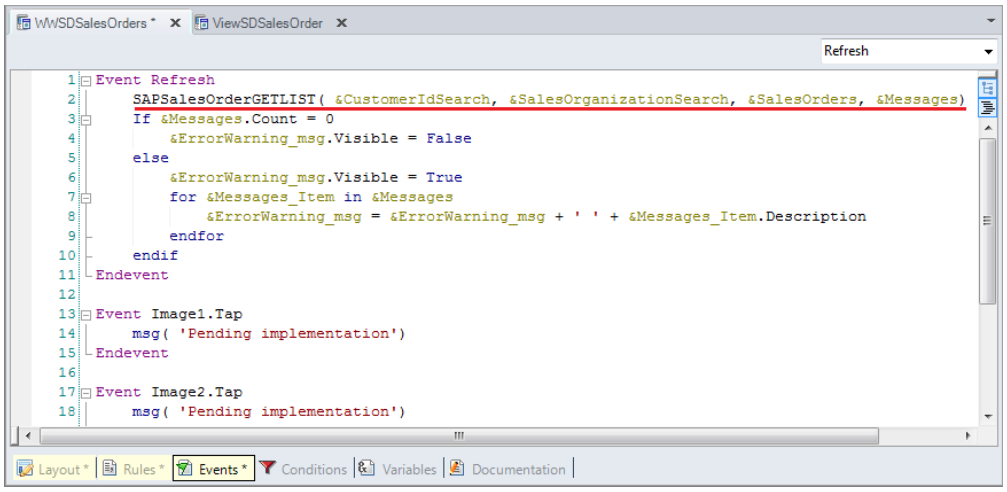


mientras que en el evento asociado al botón, se invoca, ahora sí, al trabajar con SalesOrders, pasando por parámetros estas dos variables, para que el work with pueda filtrar por sus valores.

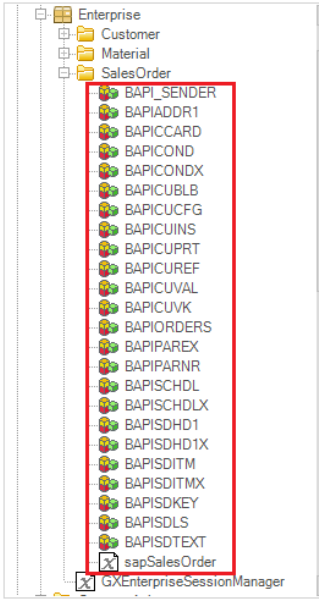
Evidentemente, para mostrar los pedidos de venta del cliente y organización recibidos por parámetro:



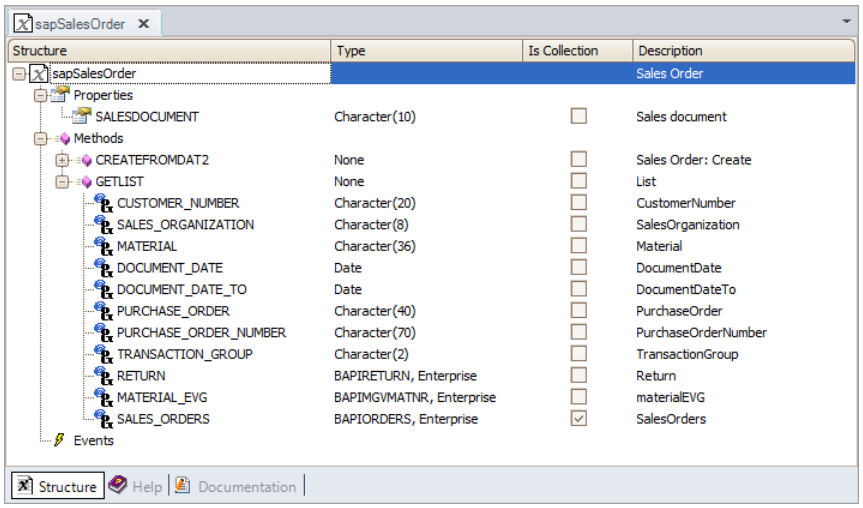
en el procedimiento invocado:



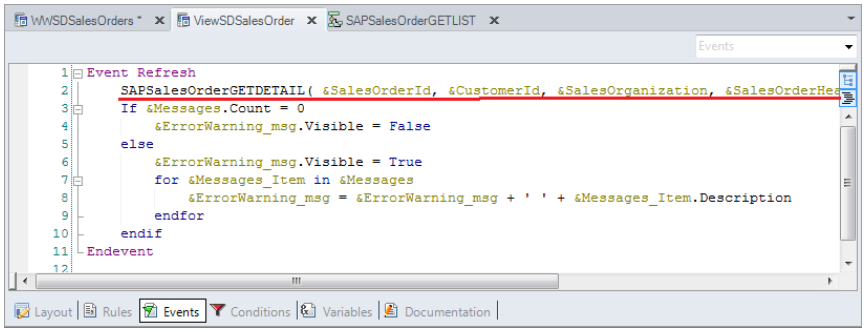
tuvimos que llamar al método GETLIST de la bapi SalesOrder importada previamente del SAP ERP a través del conector. Aquí vemos el objeto externo y los tipos de datos que se importaron en GeneXus:

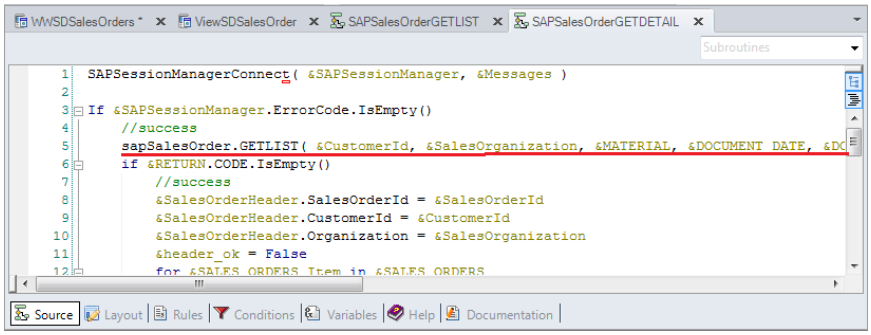


En el objeto externo sapSalesOrder vemos que hemos importado dos métodos:



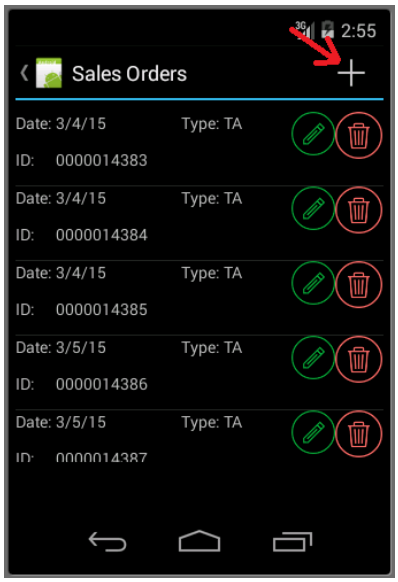
el primero lo utilizaremos para crear un nuevo pedido de venta, y el segundo, GETLIST, ha sido utilizado para obtener los pedidos de venta que se muestran en el Work With, así como para obtener el pedido que se muestra en Detalle cuando se lo selecciona del listado:





Hasta aquí lo que tenemos.

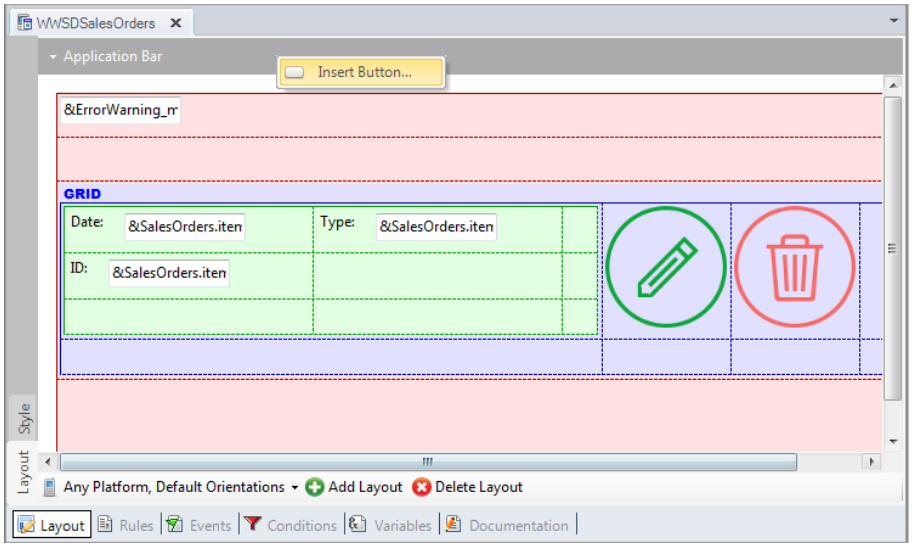
Querremos agregar un botón en la barra de la aplicación del Work With, para permitir desde allí ingresar un nuevo pedido de ventas.



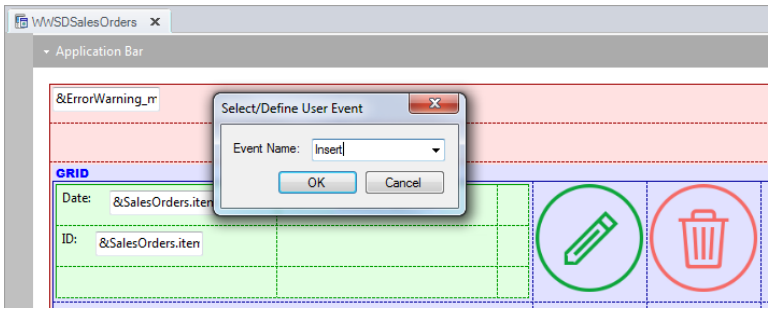
Llenando los datos del cabezal, que aparecen inicializados por defecto, pero que pueden cambiarse, y especificando qué cantidad de qué materiales se están pidiendo. Aquí solamente uno. Al grabar, queda ingresado el pedido de venta en el sistema.

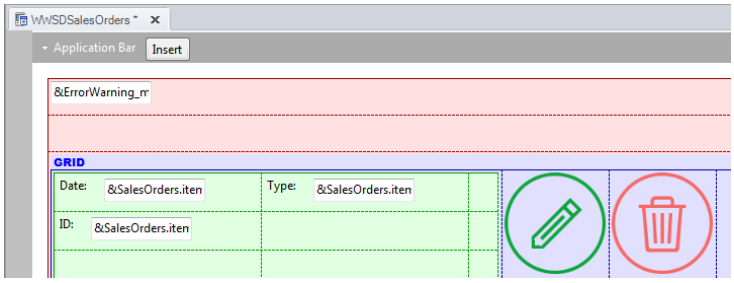
Para implementar todo esto con GeneXus…

En el Work With hacemos botón derecho sobre la Application Bar:



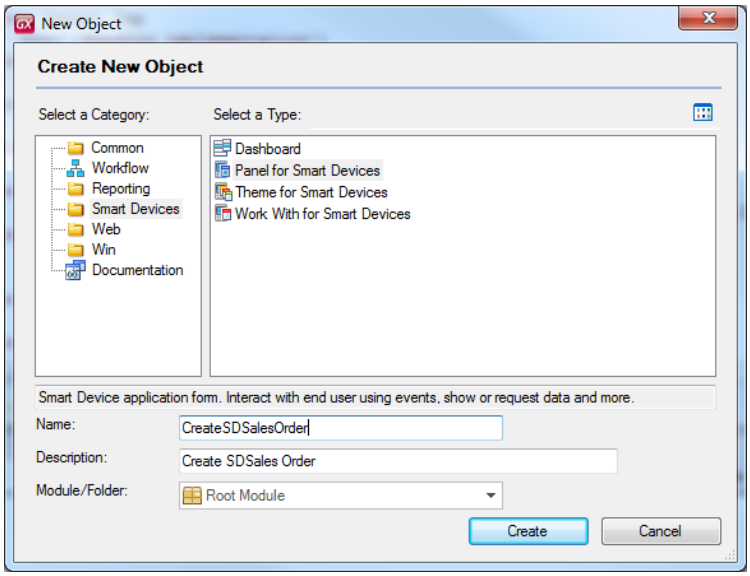
Insert Button (que nos pide el nombre del evento que tendrá asociado, al que llamaremos Insert):



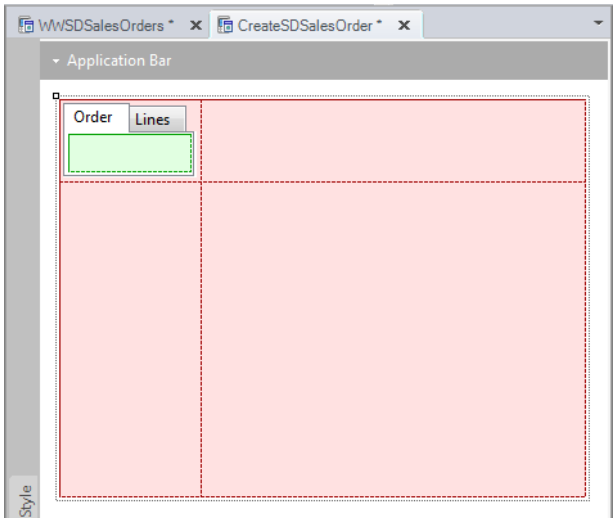


En este evento programaremos la invocación al panel que vimos en ejecución, en el que el usuario ingresará los datos del pedido para, al grabar, insertarlo en la base de datos del ERP.

Debemos crear ese panel:

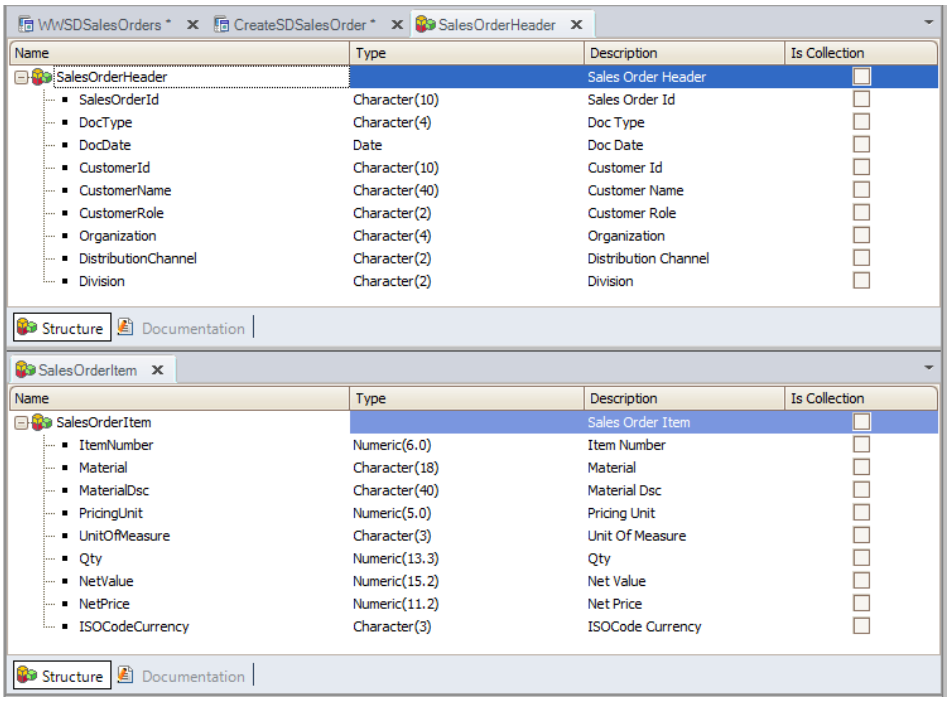


El usuario deberá poder ingresar desde esta pantalla el cabezal de la orden, y las líneas con los pedidos de cada material. Insertamos, por tanto, un tab control para separar una cosa de la otra:

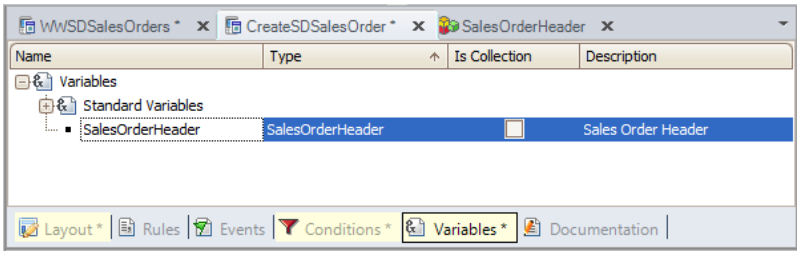


A los datos del cabezal los llamaremos Order, y a los de las líneas, Lines. Y eliminaremos la tercera solapa, que no será necesaria.

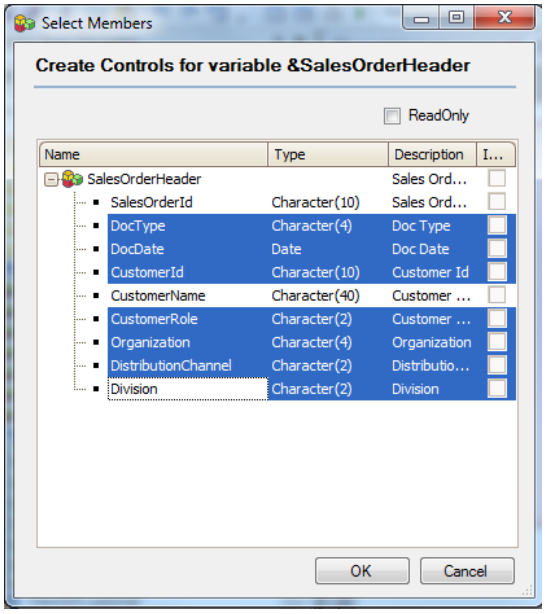
Observemos que para implementar las pantallas de WorkWith y de View ya habíamos definido dos tipos de datos estructurados: SalesOrderHeader para representar el cabezal (lo utilizábamos en el Work with… y en la solapa Orders del View) y SalesOrderItem, para almacenar cada línea de pedido (lo utilizábamos como colección en la solapa Details del View):

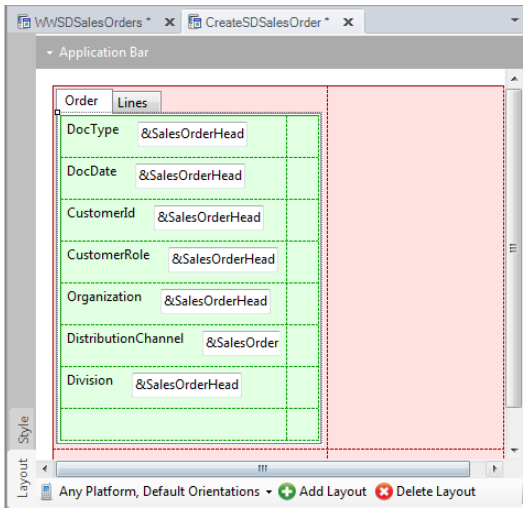


Definiremos entonces, una variable del tipo de datos SalesOrderHeader:

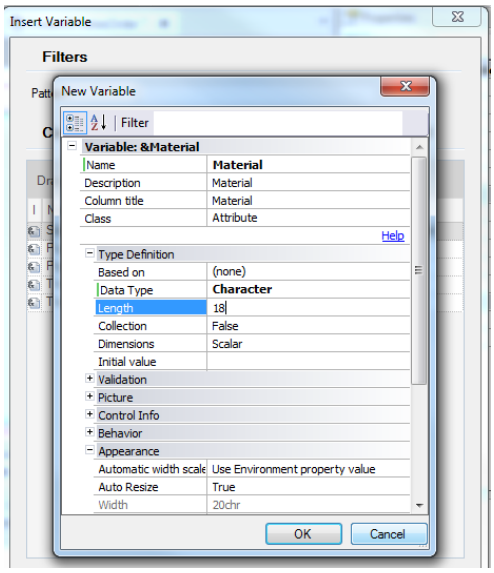


Y la insertamos en la pantalla para que el usuario pueda ingresar los campos del cabezal:

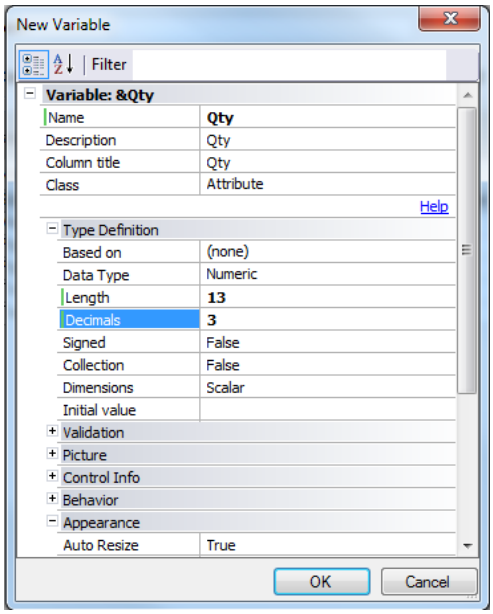




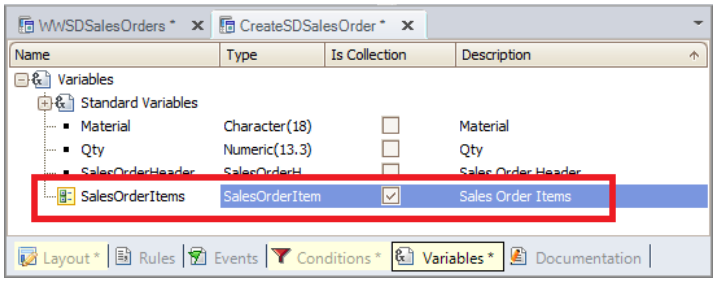
Y para las líneas, le pediremos que de cada una ingrese únicamente el material (character de 18):



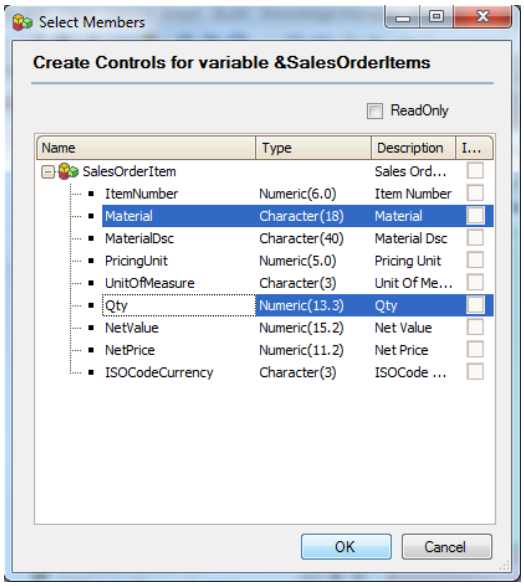
Y la cantidad llevada, quantity, un numérico de 13, de los cuales 3 son decimales:



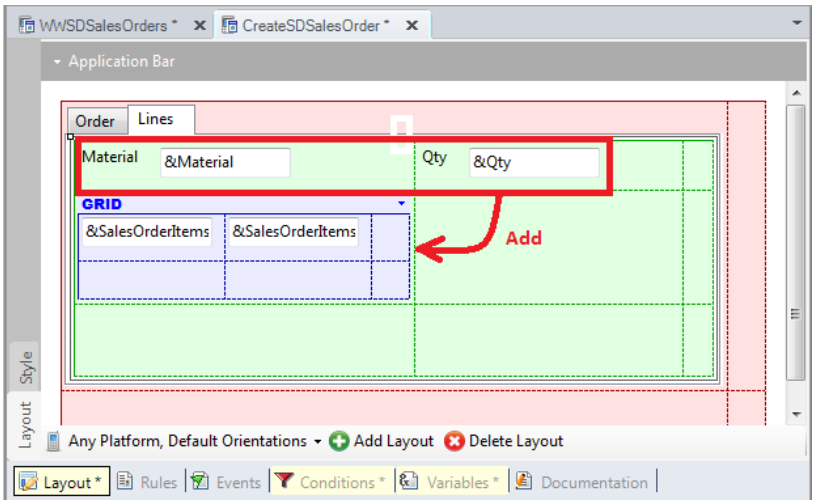
Necesitamos una variable que vaya almacenando todas las líneas que el usuario vaya ingresando en la pantalla. Para ello, definimos una variable SalesOrderItem**s** de tipo SalesOrderItem, pero colección:



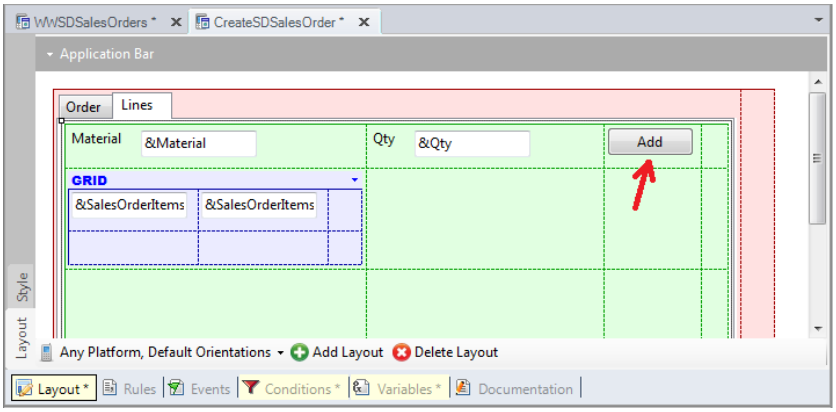
La insertamos en el layout, eligiendo mostrar únicamente el material y la cantidad:



Luego, sólo nos está faltando que cuando el usuario ingrese material y cantidad en estas variables, se ingrese un nuevo ítem en esta colección:



Para ello insertamos un botón asociado a un evento Add:



en cuyo código deberemos invocar a un procedimiento, al que llamaremos AddLine, al que le pasaremos como parámetros de entrada las dos variables cargadas por el usuario: material y quantity, y como parámetro de entrada/salida la colección de líneas que tenemos hasta el momento (al principio, vacía), para que nos devuelva esta misma colección, pero con una nueva línea, con los valores de las variables.

Aquí ya lo hemos creado. Vemos su declaración de parámetros a través de su regla parm:

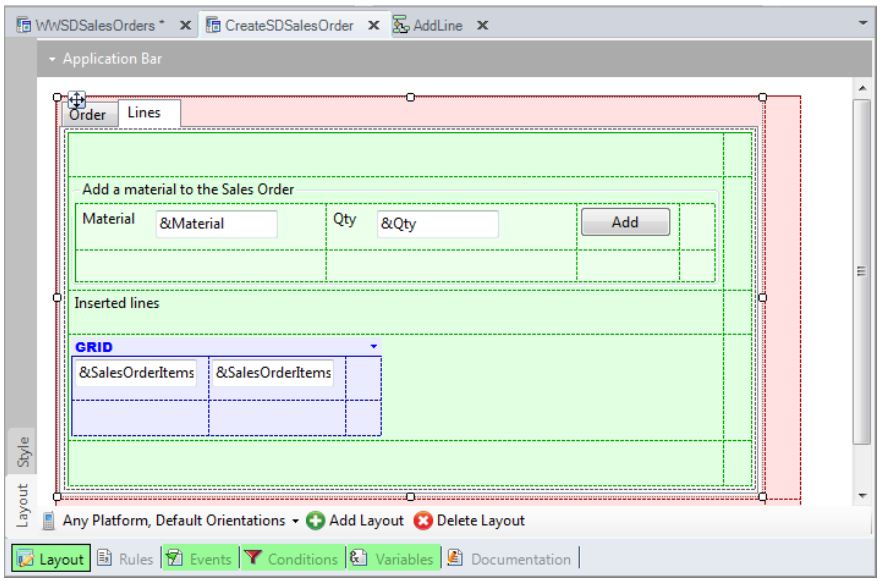
```
parm( in: &Material, in: &Qty, inout: &SalesOrderItems);
```

y si vamos al Source, todo lo que hace es agregar un item con los valores de las variables recibidas a la colección también recibida:

```
&SalesOrderItem.Material = &Material
&SalesOrderItem.Qty = &Qty
&SalesOrderItems.Add(&SalesOrderItem)
```

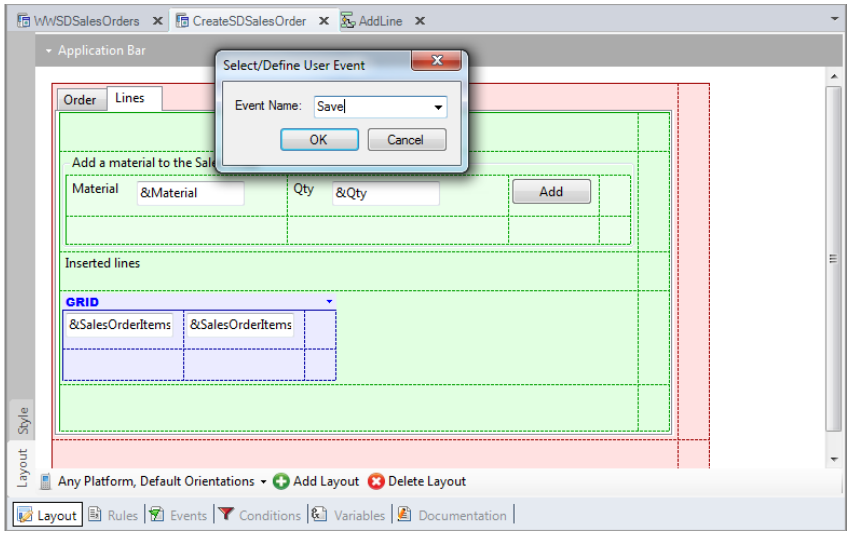
(Por ahora el método Add de una colección no puede especificarse dentro de un evento que se ejecuta en el dispositivo, y es por eso que tuvimos que hacerlo a través de este procedimiento y no directamente aquí. Distinto hubiera sido el caso si el evento que hacía el Add era uno del Server, como el Start, el Refresh o el Load).

Observemos que hemos mejorado un poco la estética de la solapa Lines:



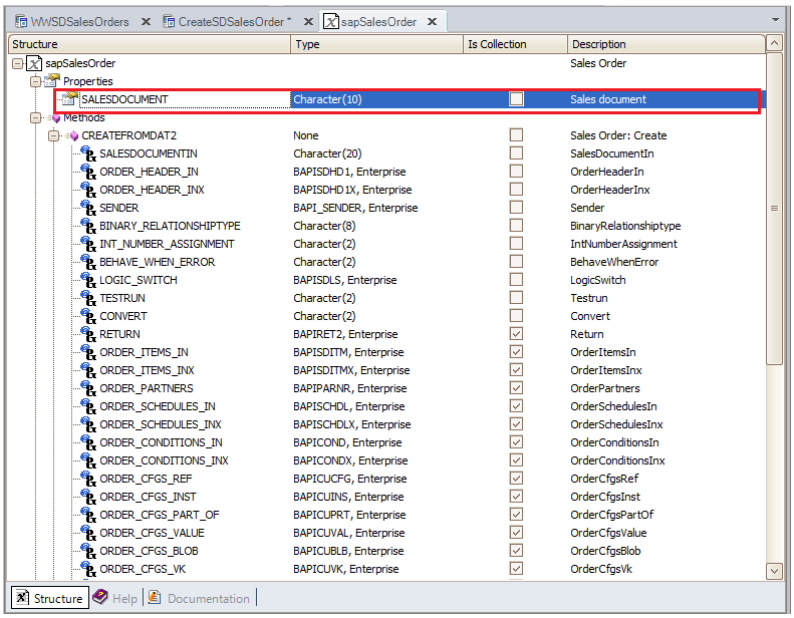
Una vez que el usuario haya dado valor a las variables del cabezal y haya ingresado las líneas, tenemos que ofrecerle un botón para que se cree el pedido de venta con esos datos.

Para ello insertamos un botón en la application bar, a cuyo evento asociado llamaremos Save:

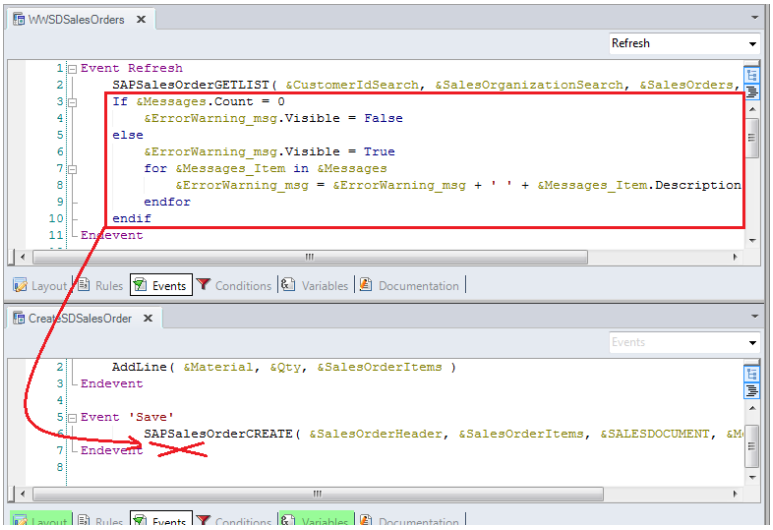


Donde invocaremos a un procedimiento que se conectará con el SAP ERP para intentar crear un nuevo pedido de venta con la información ingresada por el usuario en la pantalla.

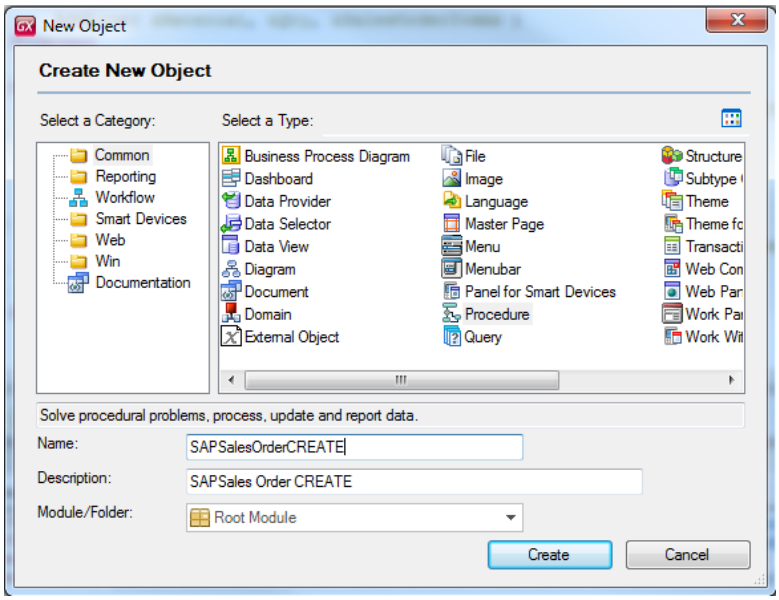
Lo llamaremos SAPSalesOrderCREATE, y debemos pasarle por parámetros de entrada las dos variables colección que contienen los datos de cabezal y líneas, esto es: &SalesOrderHeader y &SalesOrderItems, y nos devolverá una variable con el identificador del pedido ingresado, &SALESDOCUMENT, que si nos fijamos en la bapi, debe ser del tipo Character de 10:



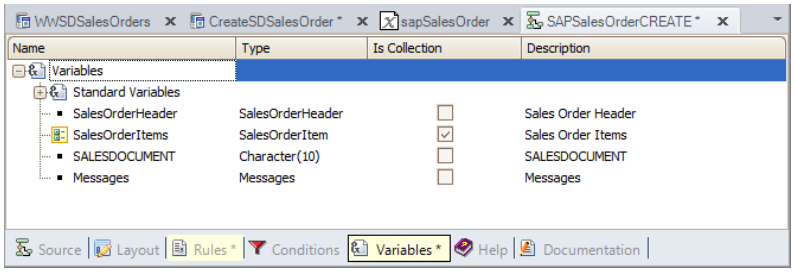
y otra, como siempre, &messages, con los posibles mensajes de advertencia o error, que al tratarse de un evento ejecutado en el cliente, es decir, en el dispositivo y no en el servidor (como era el caso del refresh) se manejarán automáticamente, no teniendo que recorrer esa colección para mostrarlos en pantalla explícitamente:



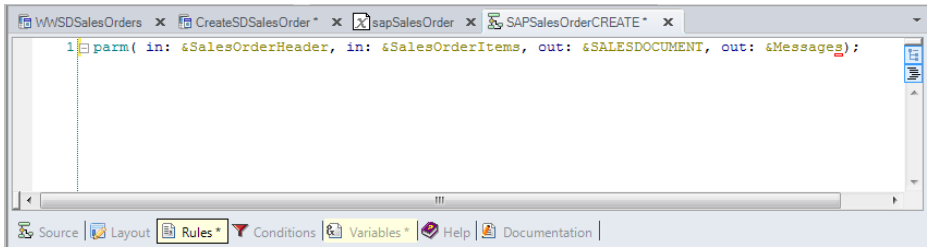
Para poder grabar, debemos crear el procedimiento:



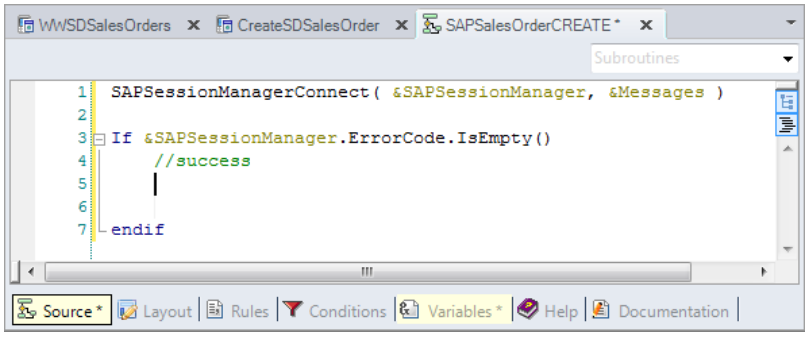
Definimos las variables que declararemos como parámetros:



Y las colocamos en la regla parm:

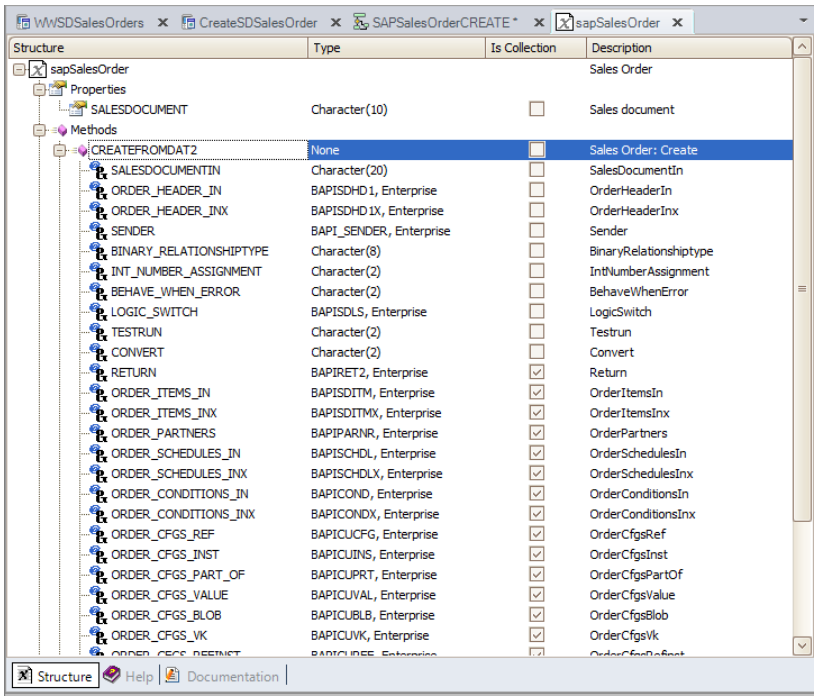


Lo primero que debemos hacer es conectarnos, como siempre, al ERP. Definimos, entonces, una variable SAPSessionManager, del tipo del objeto externo GXEnterpriseSessionManager e invocamos al procedimiento que realiza la conexión:

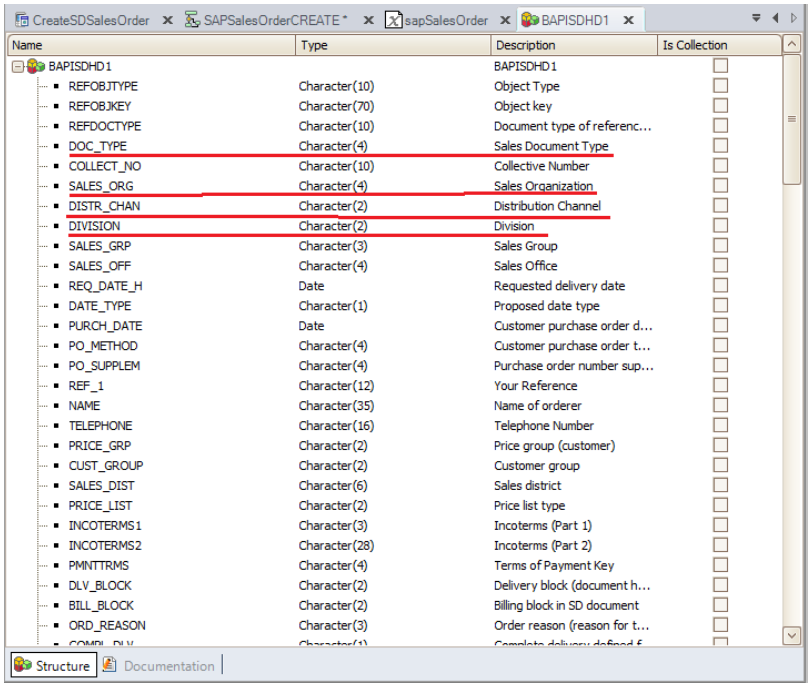


Si no hubo error de conexión, entonces debemos intentar crear la Sale Order.

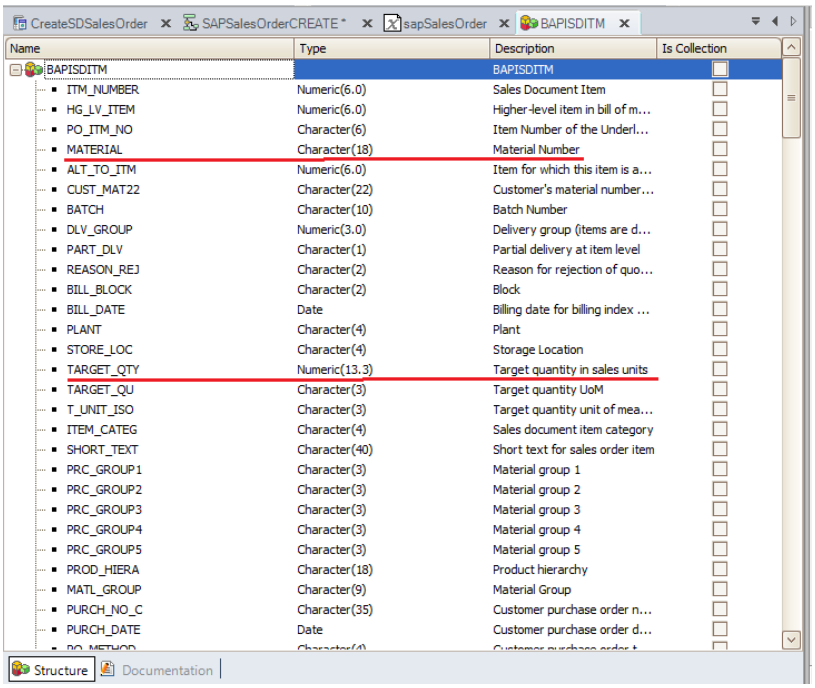
Para ello, utilizaremos el método CREATEFROMDAT2 del objeto externo que implementa la comunicación RFC con la bapi. Vemos que es necesario pasarle todos estos parámetros:



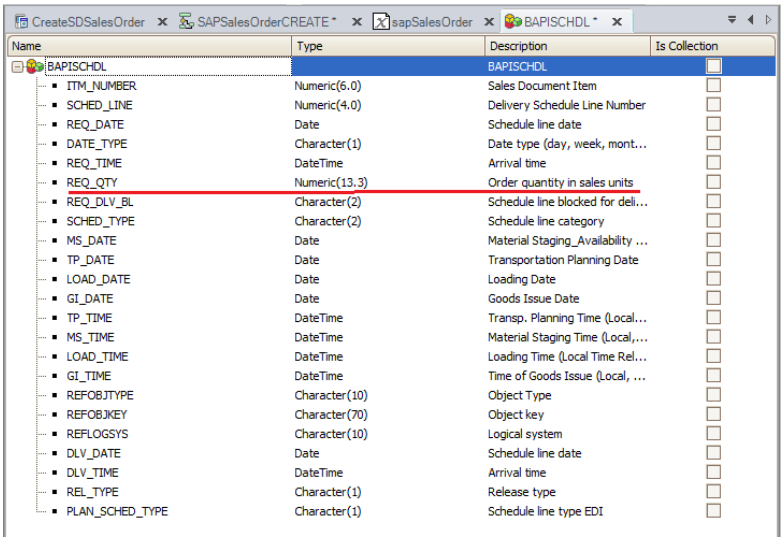
De los cuales ORDER_HEADER_IN se utiliza para ingresar los datos del cabezal. Sólo ingresaremos aquellos que le ofrecimos al usuario en la pantalla:



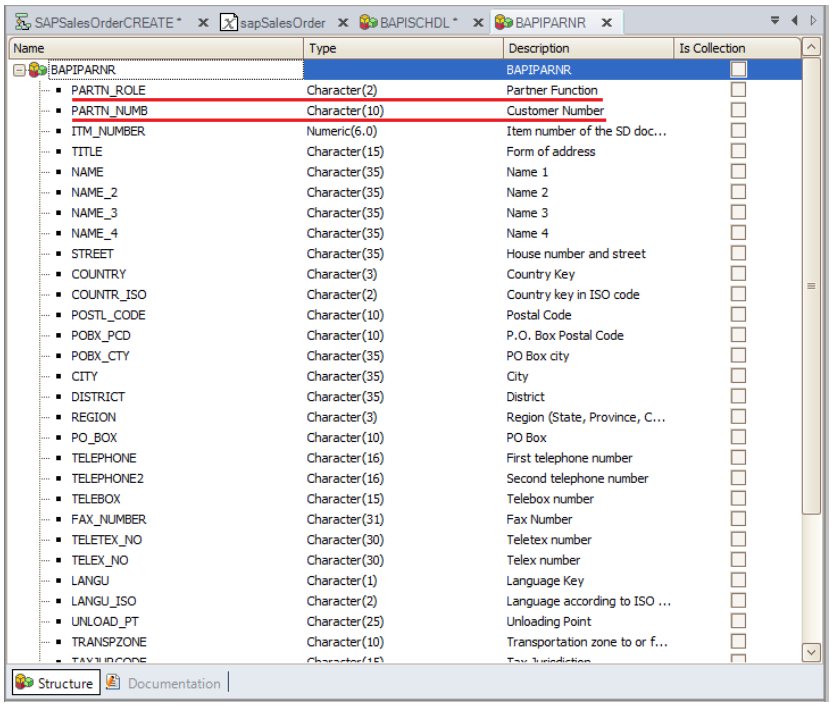
ORDER_ITEMS_IN, colección, se utiliza para ingresar los datos de cada línea (de cada línea solamente registraremos MATERIAL Y TARGET_QTY):



ORDER_SCHEDULES_IN, se utiliza para agendar las fechas de entrega para cada material (registraremos únicamente la cantidad pedida):

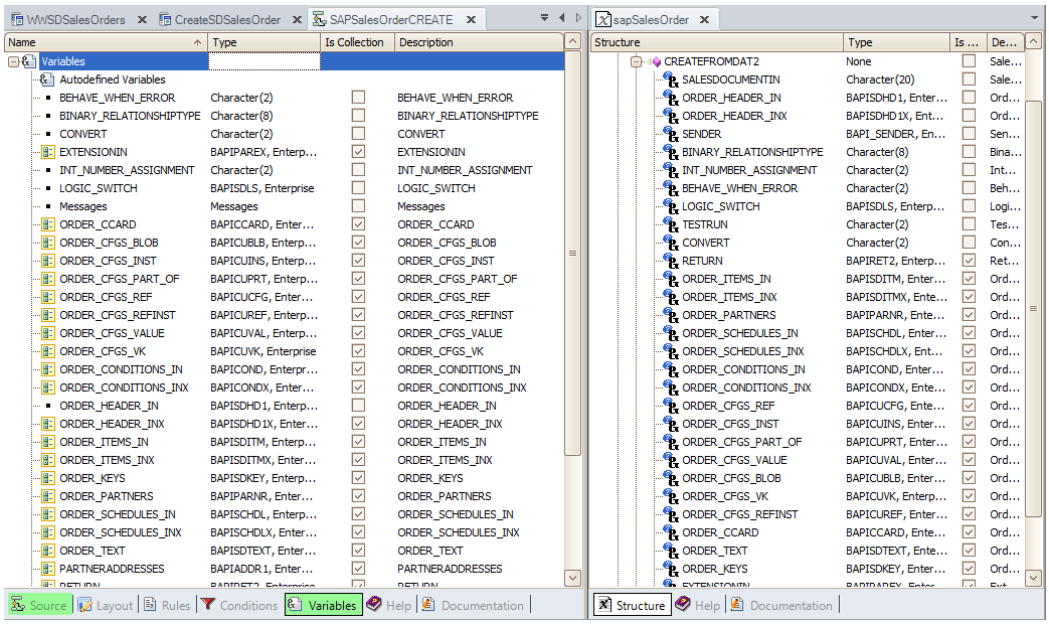


Y ORDER_PARTNERS, también colección, será utilizado en nuestro caso con un único item, para registrar al cliente:

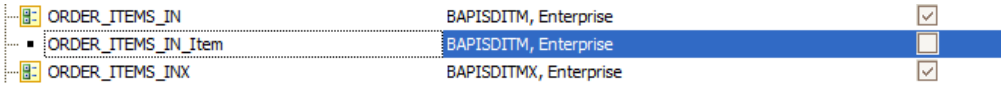


Empezaremos por arrastrar los tipos de datos utilizados por el método, para crear variables –el el tipo BAPIRET2 ya era utilizado por Materials– y les cambiamos su nombre para que coincida con el de cada parámetro, por simplicidad (marcando como colección los que lo son).

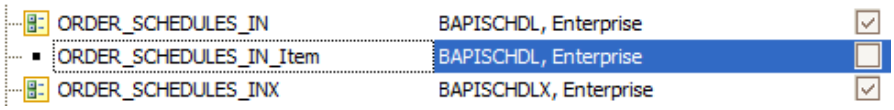
Aquí ya tenemos todas las variables definidas, incluyendo las de tipos simples:



Tenemos que cargar las cuatro que habíamos mencionado. Como &ORDER_ITEMS_IN es una colección, necesitamos una variable del tipo de cada item:

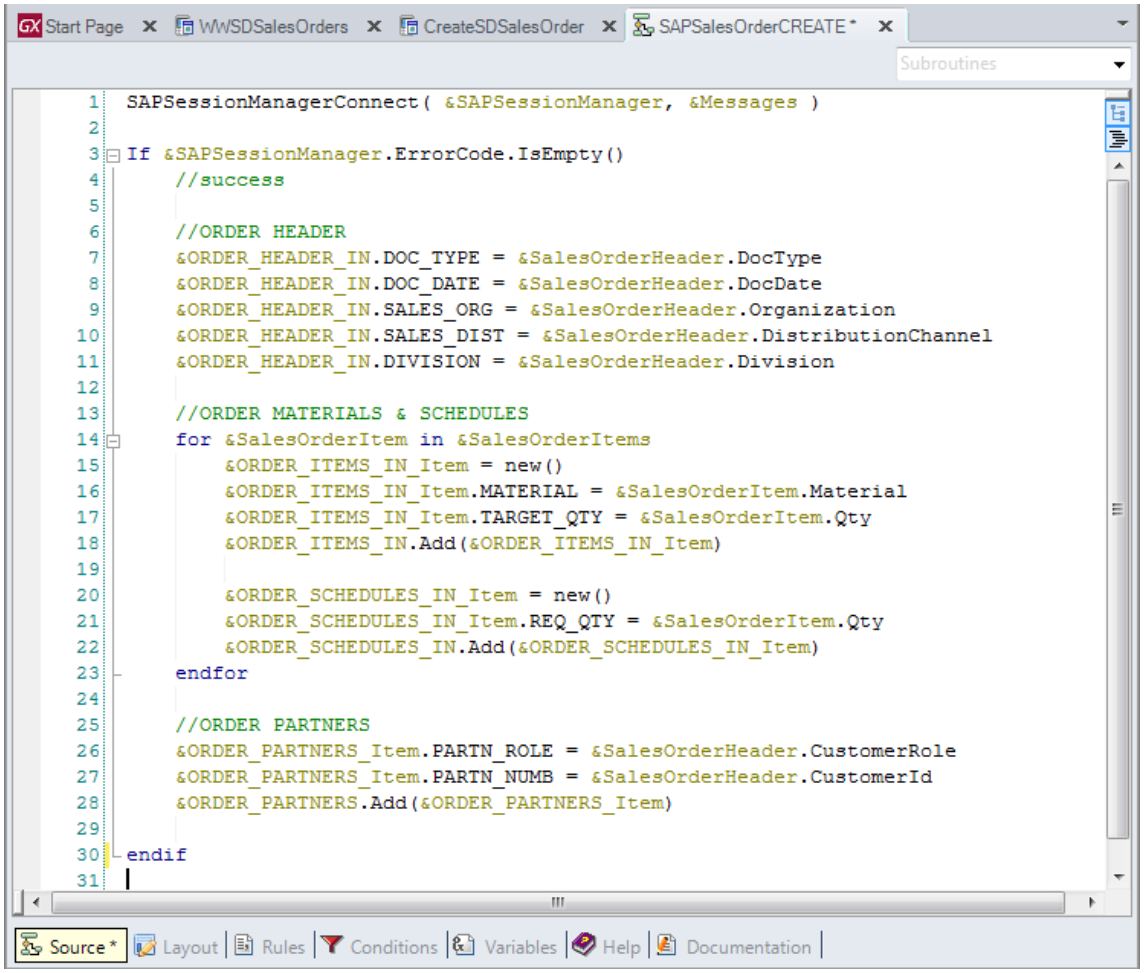


Lo mismo sucede con ORDER_SCHEDULES_IN:

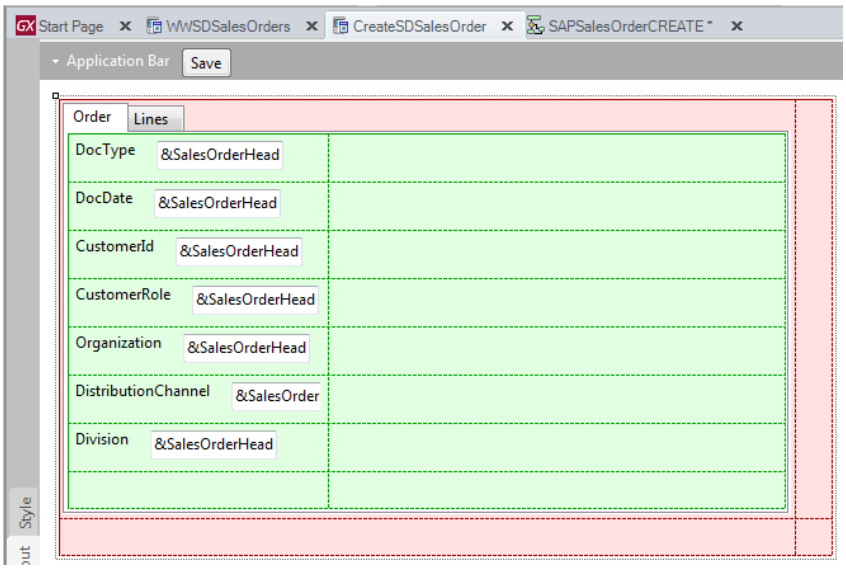


Y con ORDER_PARTNERS:

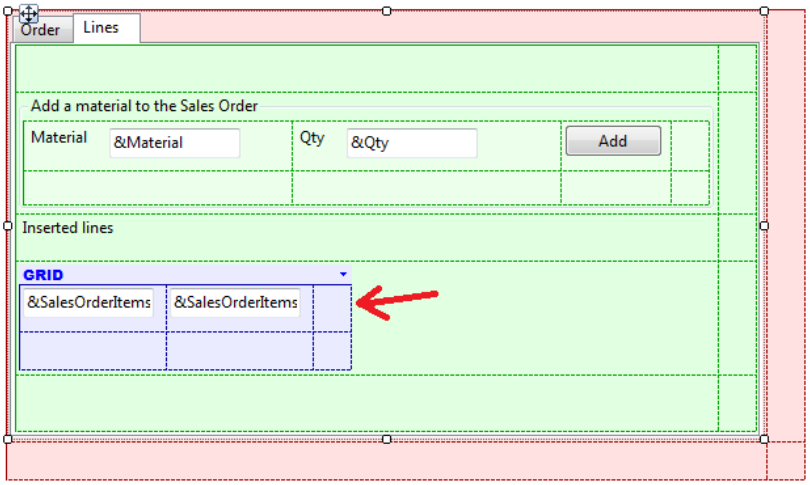
Aquí ya hemos incluído el código que carga esas variables:



La ORDER_HEADER_IN, con la información extraída de algunos elementos de la variable estructurada &SalesOrderHeader recibida por parámetro (y que el usuario llenó en esta pantalla):



Los items de la colección ORDER_ITEMS_IN, se van tomando de cada item de la colección &SalesOrderItems también recibida por parámetro y que el usuario cargó en la pantalla:

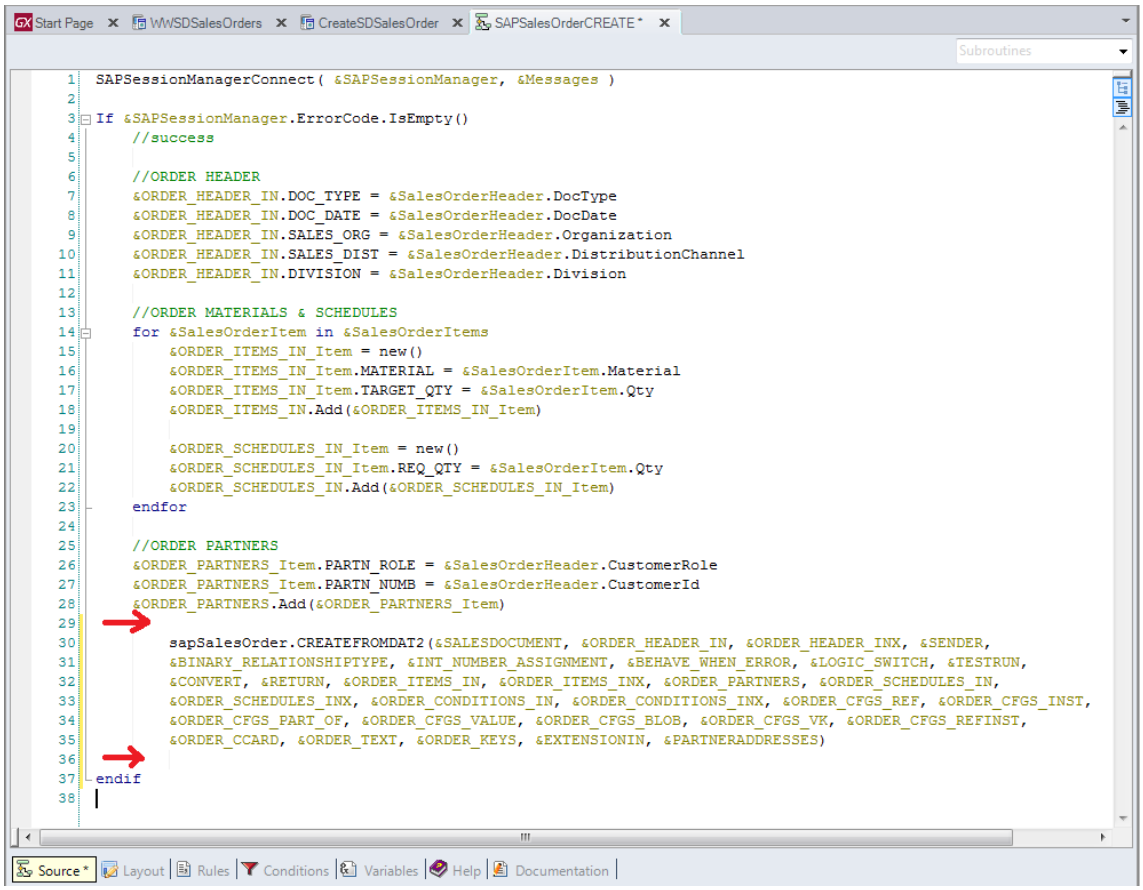


Así como la colección de ORDER-SCHEDULES.

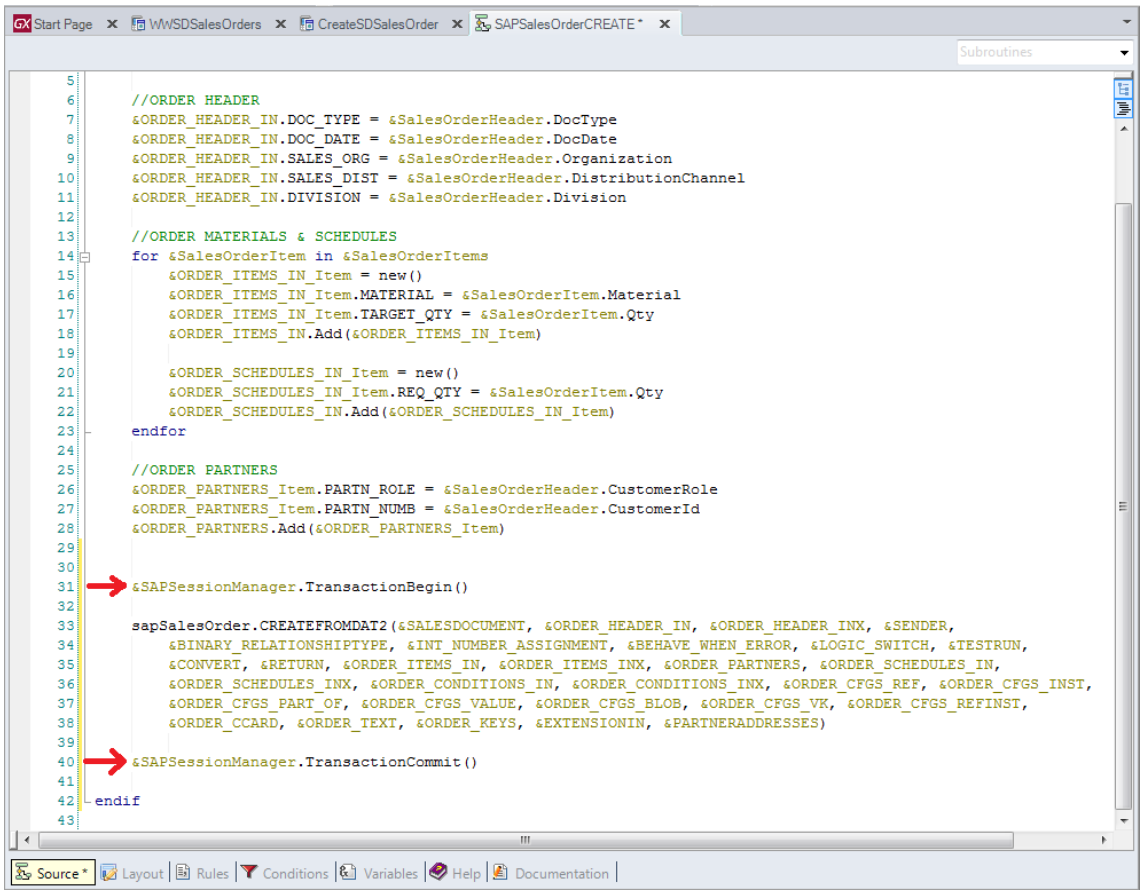
Y Por último, se agrega un único partner a la colección ORDER_PARTNERS, con los datos del cliente tomados de &SalesOrderHeader.

Ya tenemos, entonces, todas las variables de entrada cargadas. Estamos listos para invocar al método CREATEFROMDAT2 de la bapi… ingresando en orden todos los parámetros.

A diferencia de los métodos que utilizamos hasta el momento, como este método graba en la base de datos del ERP, debemos indicar que con él se conforma una Unidad de trabajo lógica, es decir que todas las operaciones realizadas desde antes de que comience, hasta su finalización, deberán realizarse todas o ninguna. Lo que en la jerga de base de datos se denomina transacción.

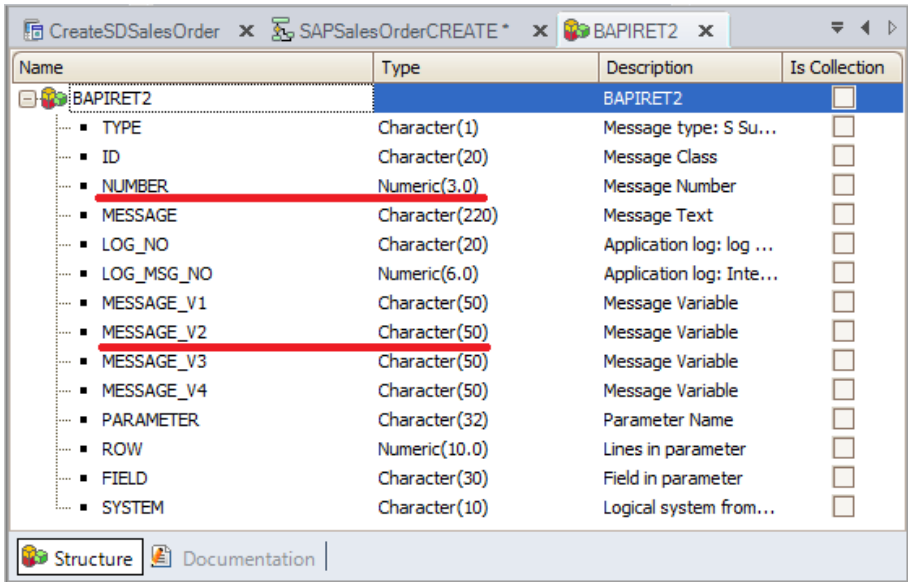


Debemos entonces, definir el comienzo y el fin de esa transacción de base de datos, o unidad de trabajo lógica. Para ello, el objeto GXEnterpriseSessionManager tiene los métodos TransactionBegin y TransactionCommit, por lo que los utilizamos antes y después de invocar al método CREATEFROMDAT2 a través de la variable de sesión:

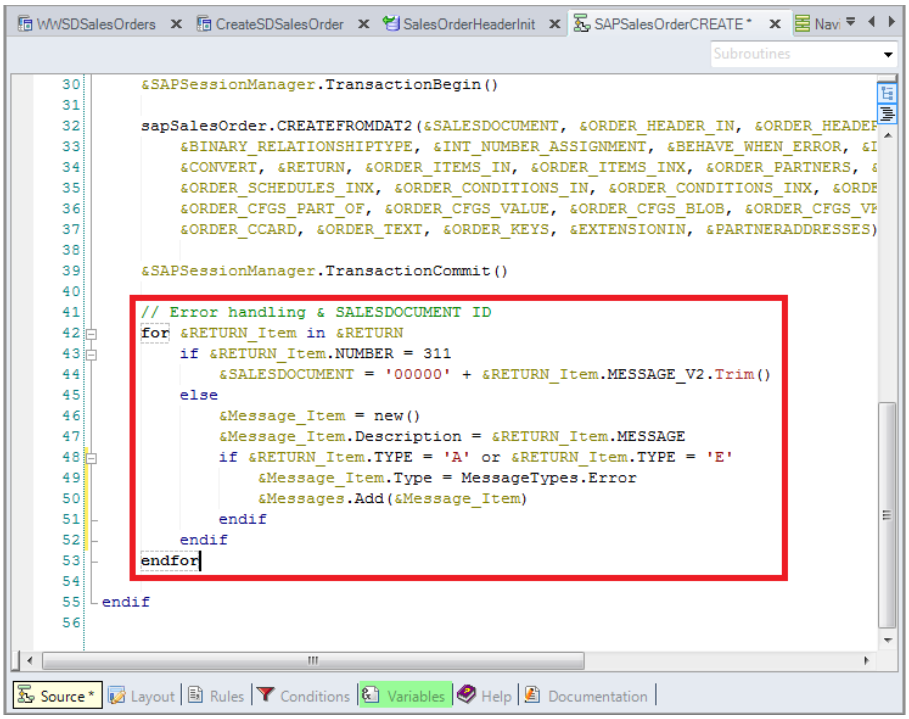


Sólo nos resta cargar la colección &Messages con los mensajes y devolver el identificador del pedido de venta que se generó en el ERP.

Los mensajes se encuentran en el parámetro &RETURN devuelto por la bapi. Observemos que se trata de una colección de items del tipo BAPIRET2. Será el item de número 311 el que contendrá el identificador de pedido generado, en el campo MESSAGE_V2:



Por lo que escribimos el siguiente código:



Sólo nos interesa registrar los mensajes de Abort o Error, además del 311. Definimos las variables &Message_Item y &RETURN_Item y grabamos.

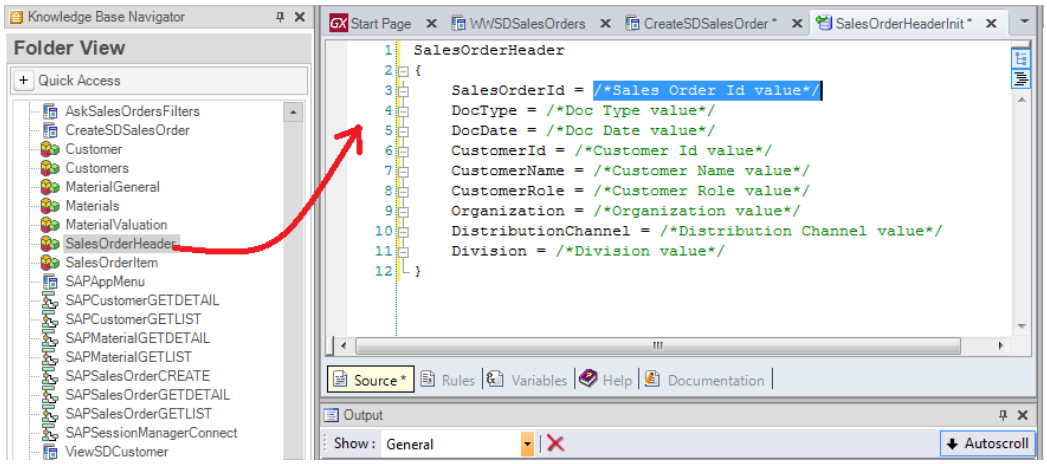
Volvamos al panel que pide los datos de la Sale Order al usuario.

Agreguemos una inicialización para las variables de la solapa Order (las del cabezal), para que el usuario no tenga que ingresarlas de cero. Para ello, en el evento Start que se ejecuta en el servidor al abrirse el panel, la primera vez y sólo esa vez, a la variable &SalesOrderHeader le asignaremos el resultado de ejecutar un objeto DataProvider:

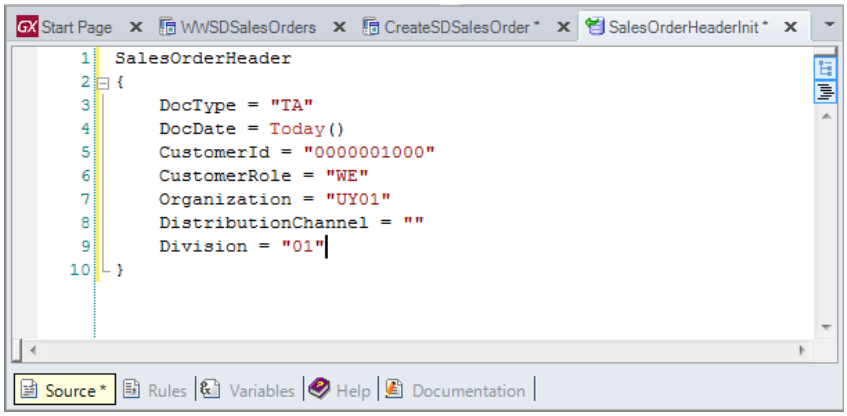
```
Event Start
    &SalesOrderHeader = SalesOrderHeaderInit()
Endevent
```

que lo que hace es devolver cargado el tipo de datos estructurado. Lo llamaremos SalesOrderHeaderInit.

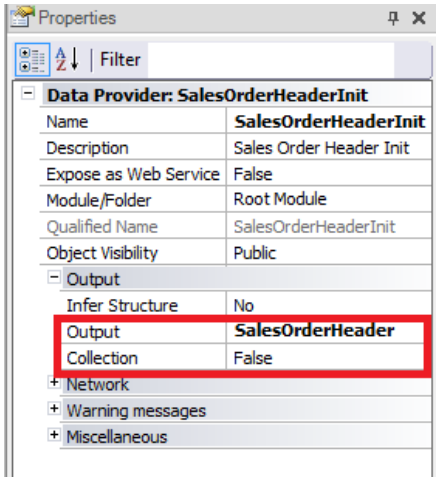
Arrastramos el tipo de datos estructurado SalesOrderHeader, que será el que el DataProvider devolverá cargado:



y simplemente le damos valor a los elementos que queramos inicializar:



Aquí vemos la salida:



Una vez que se invocó al procedimiento para crear el pedido de venta, querremos llamar al panel que permite visualizar la orden. Por lo que hacemos:

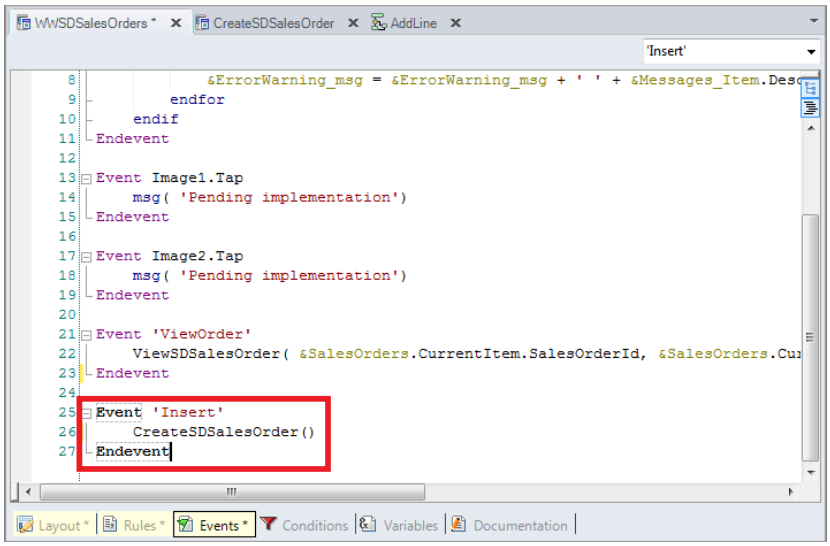
```
Event 'Save'
    SAPSalesOrderCREATE( &SalesOrderHeader, &SalesOrderItems, &SALESDOCUMENT, &Messages )
    ViewSDSalesOrder( &SALESDOCUMENT, &SalesOrderHeader.CustomerId, &SalesOrderHeader.Organization)
Endevent
```

Si intentamos grabar vemos que nos pide que especifiquemos el comando Composite, necesario en los eventos que se ejecutan en el cliente, y que están compuestos de dos o más invocaciones, como es nuestro caso.

Lo especificamos. Y grabamos.

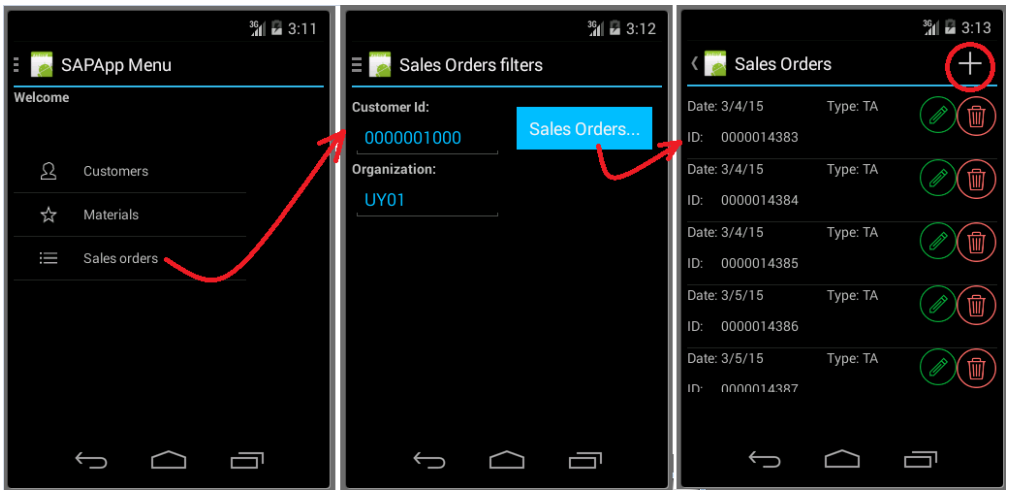
```
Event 'Save'
    Composite
        SAPSalesOrderCREATE( &SalesOrderHeader, &SalesOrderItems, &SALESDOCUMENT, &Messages )
        ViewSDSalesOrder( &SALESDOCUMENT, &SalesOrderHeader.CustomerId, &SalesOrderHeader.Organization)
    endcomposite
Endevent
```

Ahora estamos en condiciones de probar lo realizado. Sólo nos está faltando invocar desde el WorkWith de Sales Orders al panel creado:



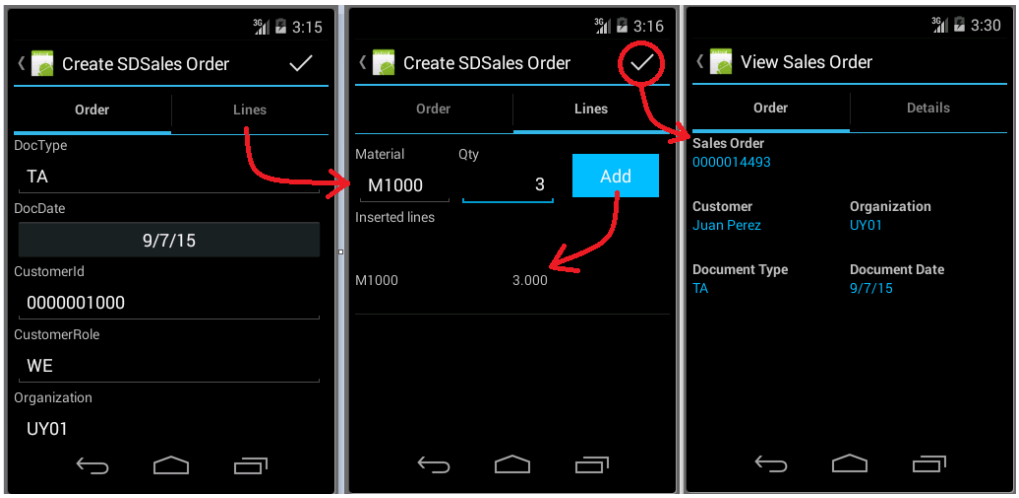
Ahora sí, F5

Elegimos customer y organization, y vemos todos sus pedidos de venta. Elegimos agregar uno nuevo:

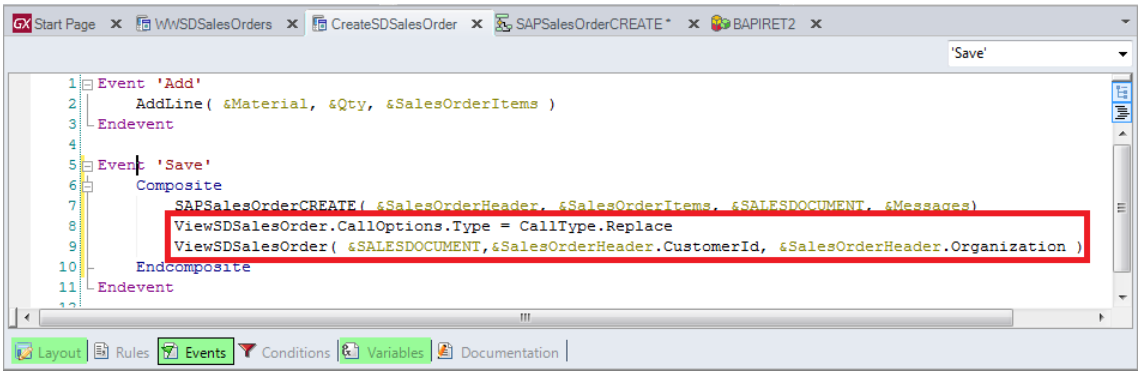


Y dejamos los valores que nos ofrece por defecto, cambiando solamente la fecha. Elegimos el material M1000 (tenemos habilitado el autocomplete), cantidad 5, y lo agregamos como línea

del pedido. Y grabamos. Nos está abriendo el panel View, que nos muestra todos los datos del pedido ingresado (al que le dio este número).



Si queremos volver haciendo back, va a la pantalla de Create, lo que no nos interesa. Quisiéramos que vuelva al Work With que la invocó en primera instancia. Para ello, alcanza con escribir:



Así, logramos que el View llamado se coloque en el stack de invocaciones donde estaba el objeto actual, reemplazándolo, el panel de Create. De esta manera, cuando se haga back del view, se volverá no al create, sino al panel que lo llamó. Como podemos ver en ejecución.

Con eso terminaremos nuestra demo. Si desea ver esta aplicación con más elementos de diseño y con una pantalla de login donde se permite conectarse a diferentes ERPs cambiando los datos de conexión, e ingresando luego un usuario y password válidos para esa instancia,

acceda a … http://samples.genexusserver.com/xev3/

con su usuario de gxtechnical. Si no lo tiene, créese uno y allí elija la Knowledge base que se muestra:

DemoXev3SAP

Puede visualizar aquí algunos aspectos de la KB. Si quiere estudiarla en detalle y probarla, vaya a GeneXus, y créese una KB a partir de la del servidor de KBs que veíamos.

Aquí ingresa la url del servidor, ingresa su usuario y password, conexión que puede testear si lo desea… tras lo cual se le muestran todas las KBs de ese servidor… elige la que desea y pide que se le cree una KB local conectada con esa del servidor, e inicializada con todos sus objetos. Esta KB tiene además, un desarrollo web.

Por más información sobre el desarrollo de aplicaciones conectadas al SAP ERP con GeneXus (incluyendo aplicaciones Web), le recomendamos acceder a nuestro wiki:

http://wiki.genexus.com/commwiki/servlet/hwikibypageid?26013

Gracias por acompañarnos.