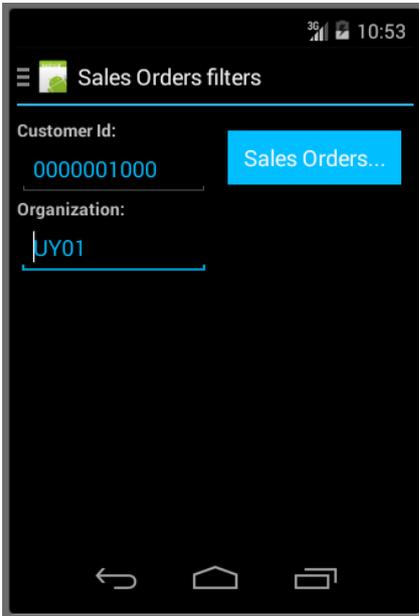
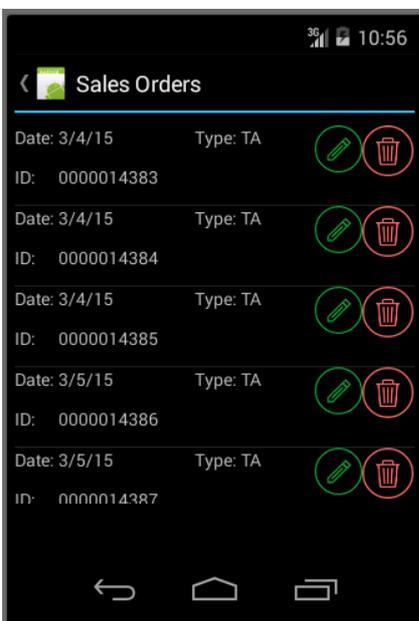


Cómo ingresar un pedido de venta en el SAP ERP desde la aplicación móvil creada con GeneXus

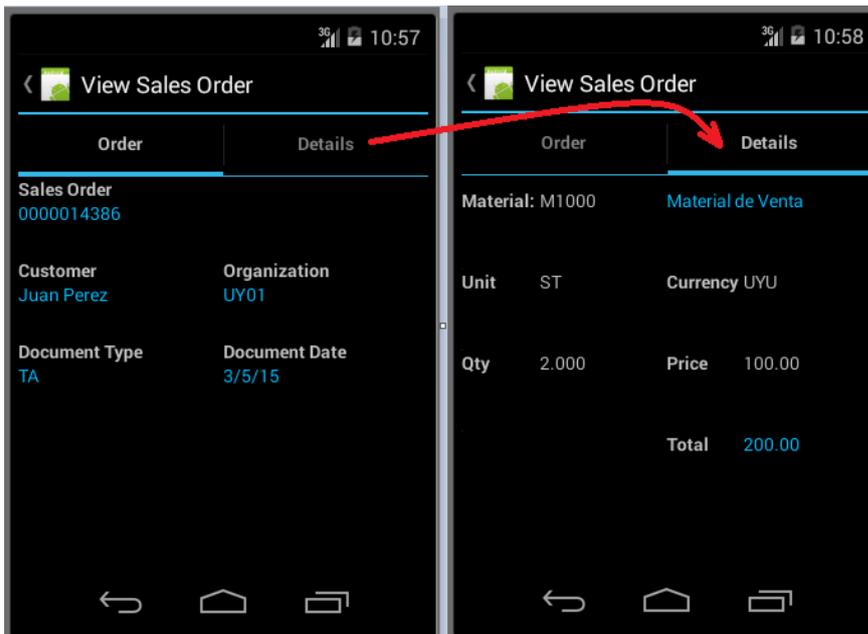
Partiremos de una aplicación en la que, además de poder trabajar con la información de los clientes y materiales, agregamos la posibilidad de, ingresando un cliente y organización (con estos valores por defecto, pero que pueden ser modificados por el usuario):



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

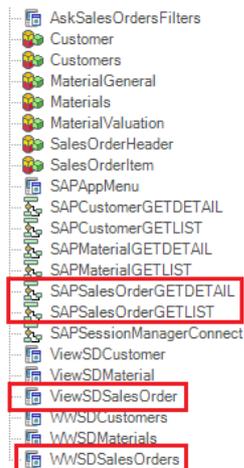


y eligiendo uno de ellos, ver su detalle:



(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:



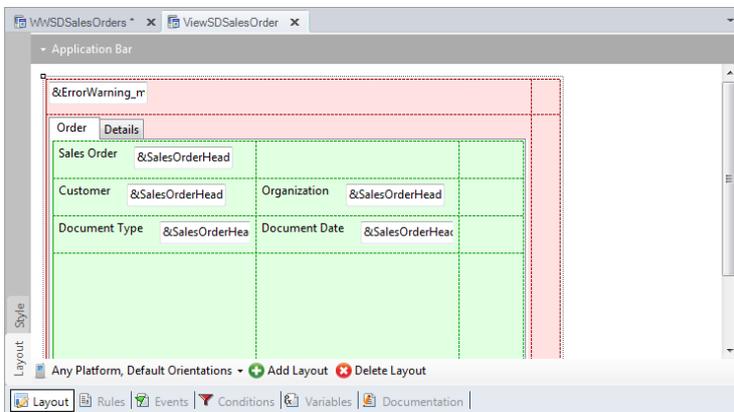
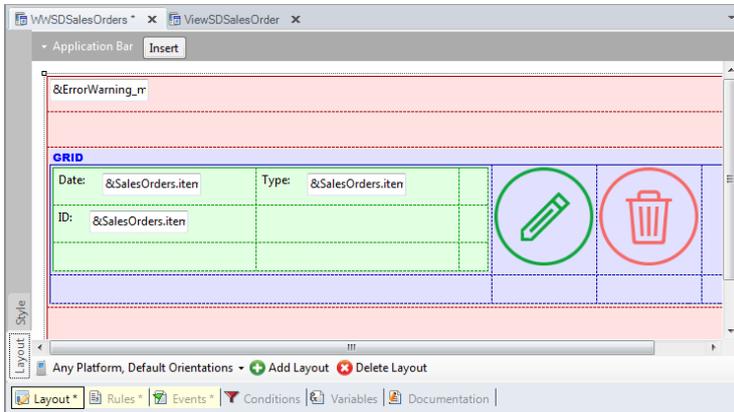
por lo que no entraremos en detalles.

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.

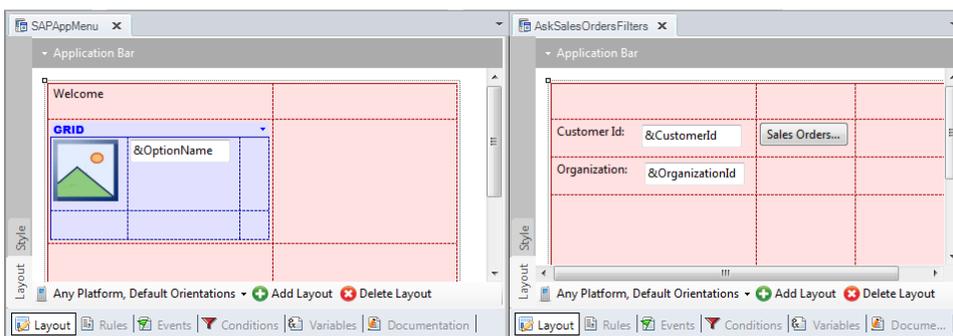
(Puede importarse el xpz adjunto, SalesOrders.xpz, para incorporar estos objetos a su Knowledge base sin tener que desarrollarlos de cero. Deberá existir en su KB el procedimiento SAPSessionManagerConnect que realiza la conexión al SAP ERP. De lo contrario fallará la importación. El xpz SAPSessionManagerConnect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

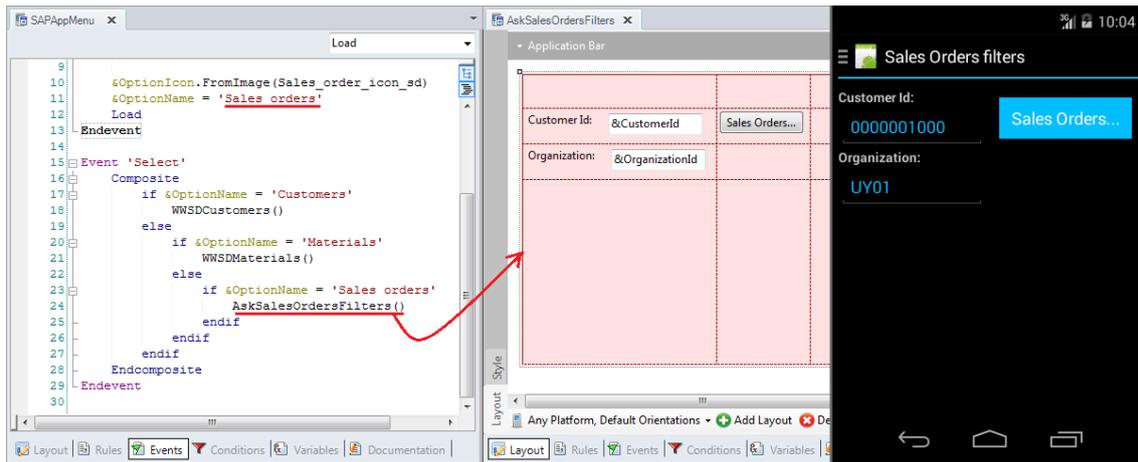
Si quiere importar en una KB vacía todos los objetos desarrollados hasta el momento, acceda al archivo KBStatePreCreateSalesOrder.xpz. Deberá abrirl el procedimiento

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

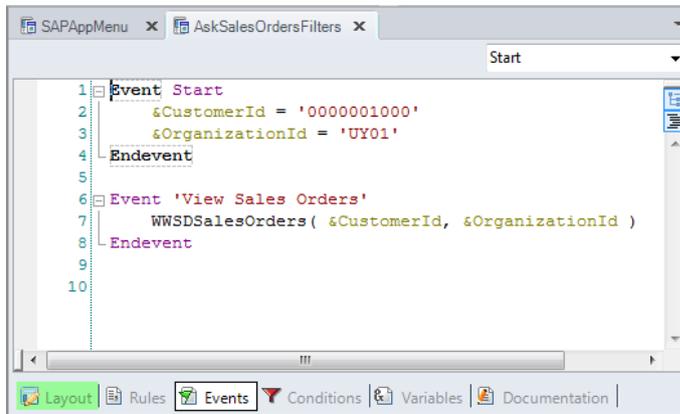


Observemos como novedad, que desde el menú de entrada no llamamos directamente al Work With, sino al panel intermedio AskSalesOrdersFilters:



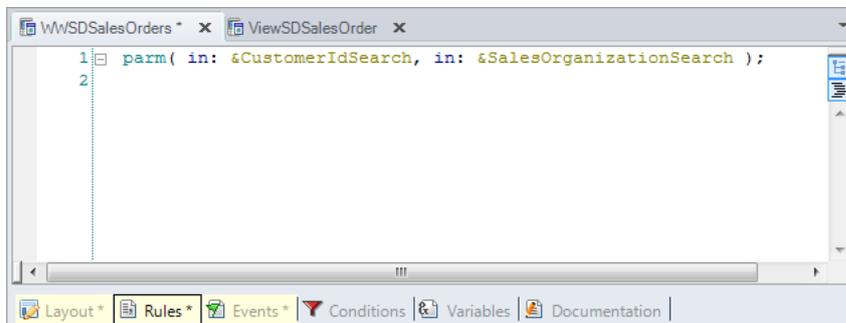


que tiene en el layout dos variables no Readonly, es decir, que el usuario puede ingresarles valor. Y vemos que se han inicializado en el evento Start:



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:



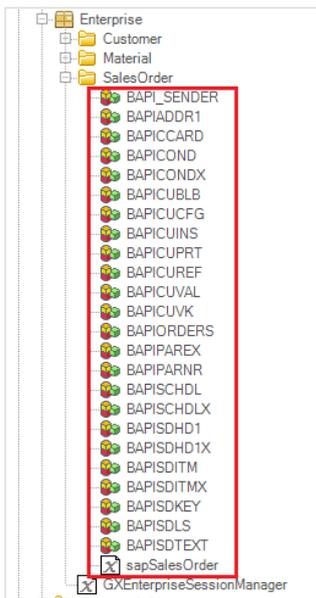
en el procedimiento invocado:

```

1  Event Refresh
2  SAPSalesOrderGETLIST( &CustomerIdSearch, &SalesOrganizationSearch, &SalesOrders, &Messages)
3  If &Messages.Count = 0
4      &ErrorWarning_msg.Visible = False
5  else
6      &ErrorWarning_msg.Visible = True
7      for &Messages_Item in &Messages
8          &ErrorWarning_msg = &ErrorWarning_msg + ' ' + &Messages_Item.Description
9      endfor
10     endif
11 Endevent
12
13 Event Image1.Tap
14     msg( 'Pending implementation')
15 Endevent
16
17 Event Image2.Tap
18     msg( 'Pending implementation')

```

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:

Structure	Type	Is Collection	Description
Properties			
SALESDOCUMENT	Character(10)	<input type="checkbox"/>	Sales document
Methods			
CREATEFROMDAT2	None	<input type="checkbox"/>	Sales Order: Create
GETLIST	None	<input type="checkbox"/>	List
CUSTOMER_NUMBER	Character(20)	<input type="checkbox"/>	CustomerNumber
SALES_ORGANIZATION	Character(8)	<input type="checkbox"/>	SalesOrganization
MATERIAL	Character(36)	<input type="checkbox"/>	Material
DOCUMENT_DATE	Date	<input type="checkbox"/>	DocumentDate
DOCUMENT_DATE_TO	Date	<input type="checkbox"/>	DocumentDateTo
PURCHASE_ORDER	Character(40)	<input type="checkbox"/>	PurchaseOrder
PURCHASE_ORDER_NUMBER	Character(70)	<input type="checkbox"/>	PurchaseOrderNumber
TRANSACTION_GROUP	Character(2)	<input type="checkbox"/>	TransactionGroup
RETURN	BAPIRETURN, Enterprise	<input type="checkbox"/>	Return
MATERIAL_EVG	BAPIMGVATNR, Enterprise	<input type="checkbox"/>	materialEVG
SALES_ORDERS	BAPIORDERS, Enterprise	<input checked="" type="checkbox"/>	SalesOrders
Events			

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:

```

1 Event Refresh
2 SAPSalesOrderGETDETAIL( &SalesOrderId, &CustomerId, &SalesOrganization, &SalesOrderHeader)
3 If &Messages.Count = 0
4   &ErrorWarning_msg.Visible = False
5 else
6   &ErrorWarning_msg.Visible = True
7   for &Messages_Item in &Messages
8     &ErrorWarning_msg = &ErrorWarning_msg + ' ' + &Messages_Item.Description
9   endfor
10 endif
11 Endevent
12

```

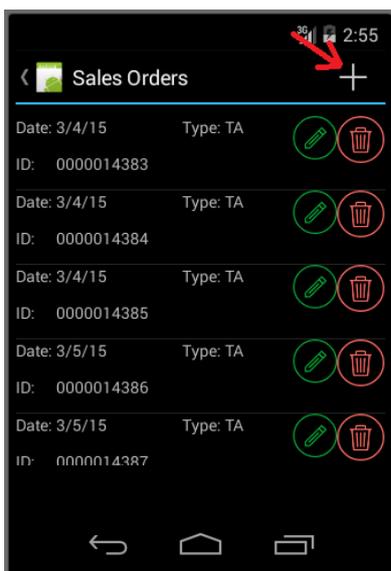
```

1 SAPSessionManagerConnect( &SAPSessionManager, &Messages )
2
3 If &SAPSessionManager.ErrorCode.IsEmpty()
4   //success
5   sapSalesOrder.GETLIST( &CustomerId, &SalesOrganization, &MATERIAL, &DOCUMENT DATE, &D
6   if &RETURN.CODE.IsEmpty()
7     //success
8     &SalesOrderHeader.SalesOrderId = &SalesOrderId
9     &SalesOrderHeader.CustomerId = &CustomerId
10    &SalesOrderHeader.Organization = &SalesOrganization
11    &header_ok = False
12    for &SALES_ORDERS_Item in &SALES_ORDERS

```

Hasta aquí lo que tenemos.

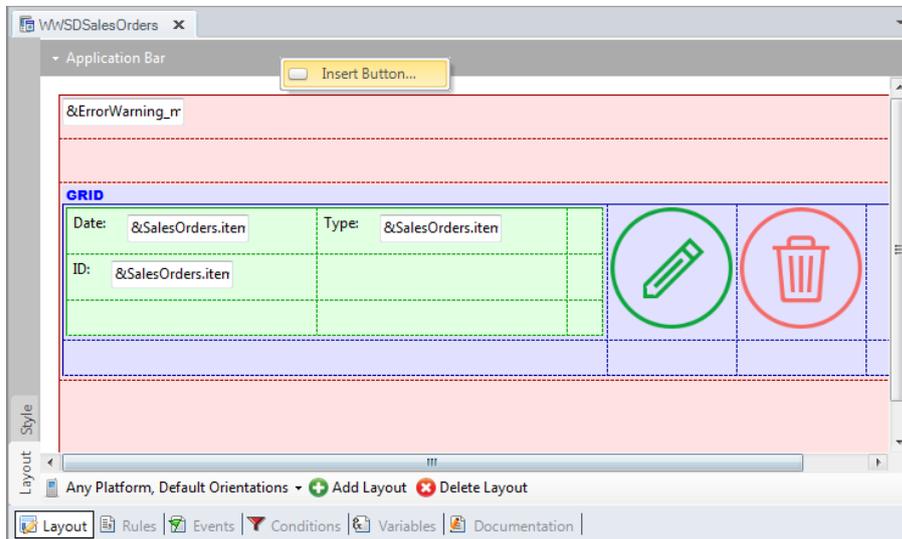
Querremos agregar un botón en la barra de la aplicación del Work With, para permitir desde allí ingresar un nuevo pedido de ventas.



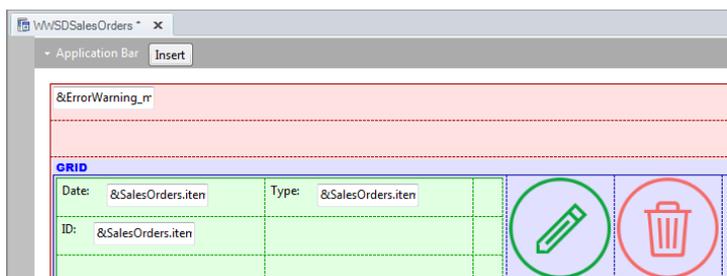
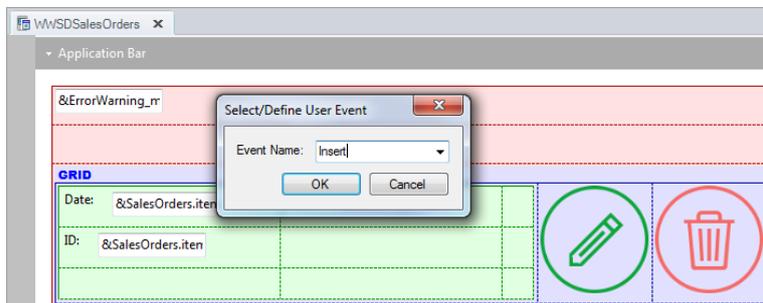
Llenando los datos del cabezal, que aparecen inicializados por defecto, pero que pueden cambiarse, y especificando qué cantidad de qué materiales se están pidiendo. Aquí solamente uno. Al grabar, queda ingresado el pedido de venta en el sistema.

Para implementar todo esto con GeneXus...

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

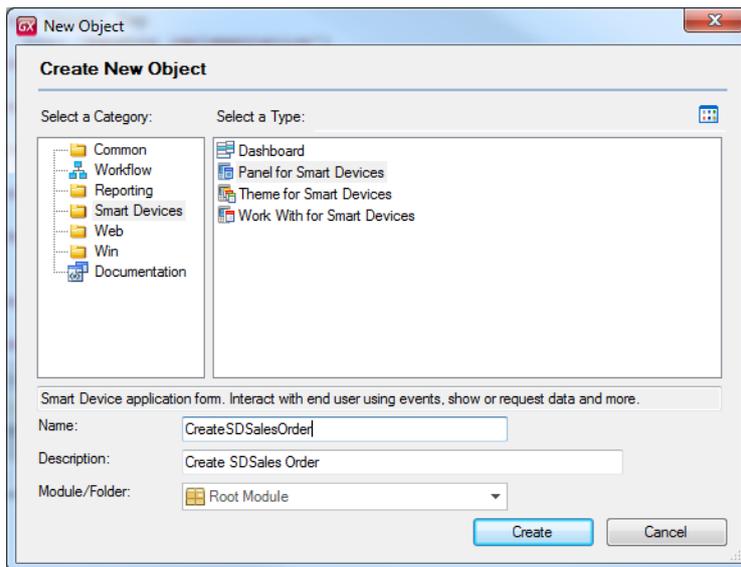


Insert Button (que nos pide el nombre del evento que tendrá asociado, al que llamaremos Insert):

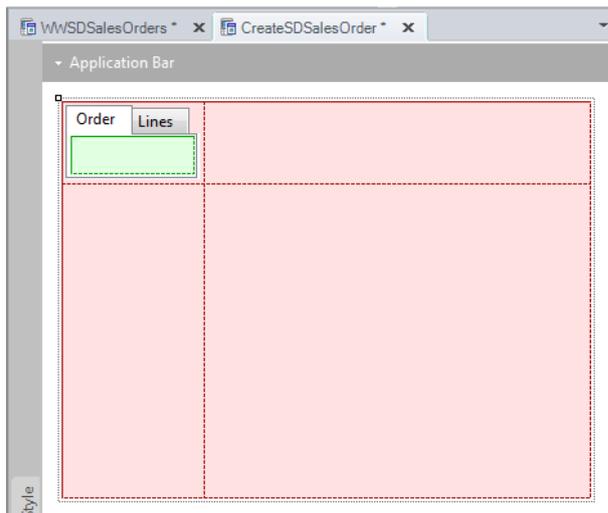


En este evento programaremos la invocación al panel que vimos en ejecución, en el que el usuario ingresará los datos del pedido para, al grabar, insertarlo en la base de datos del ERP.

Debemos crear ese panel:

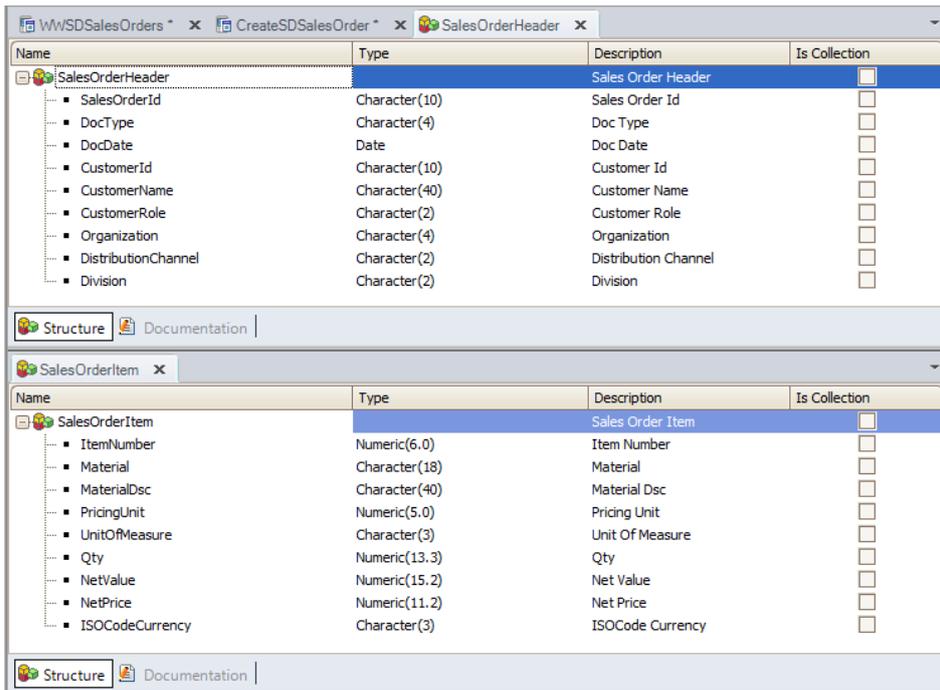


El usuario deberá poder ingresar desde esta pantalla el cabezal de la orden, y las líneas con los pedidos de cada material. Insertamos, por tanto, un tab control para separar una cosa de la otra:

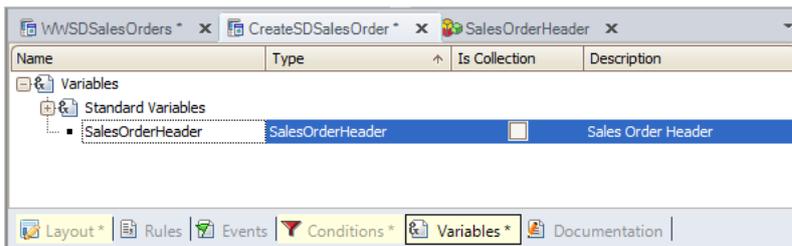


A los datos del cabezal los llamaremos Order, y a los de las líneas, Lines. Y eliminaremos la tercera solapa, que no será necesaria.

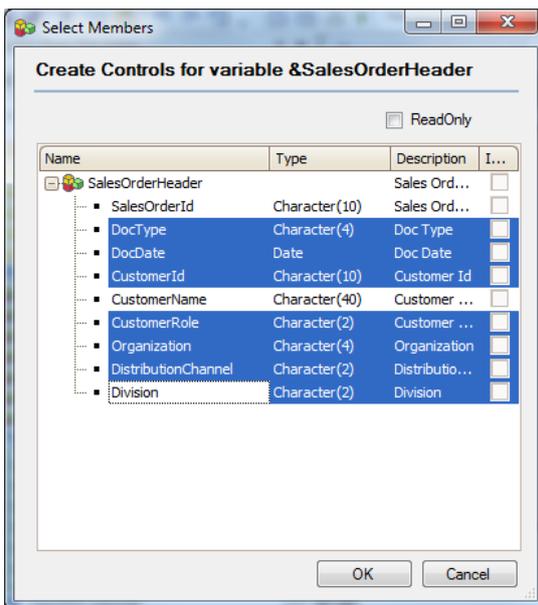
Observemos que para implementar las pantallas de WorkWith y de View ya habíamos definido dos tipos de datos estructurados: SalesOrderHeader para representar el cabezal (lo utilizábamos en el Work with... y en la solapa Orders del View) y SalesOrderItem, para almacenar cada línea de pedido (lo utilizábamos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

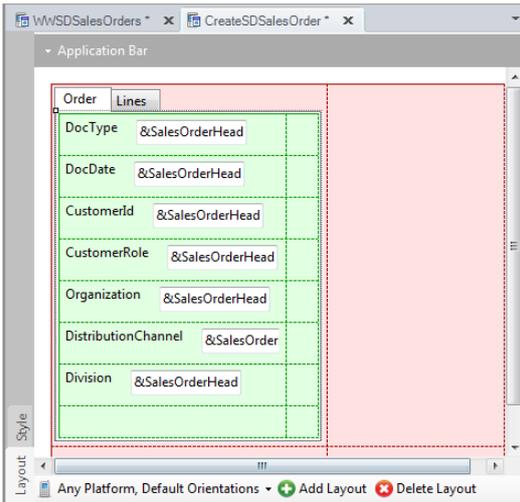


Definiremos entonces, una variable del tipo de datos SalesOrderHeader:

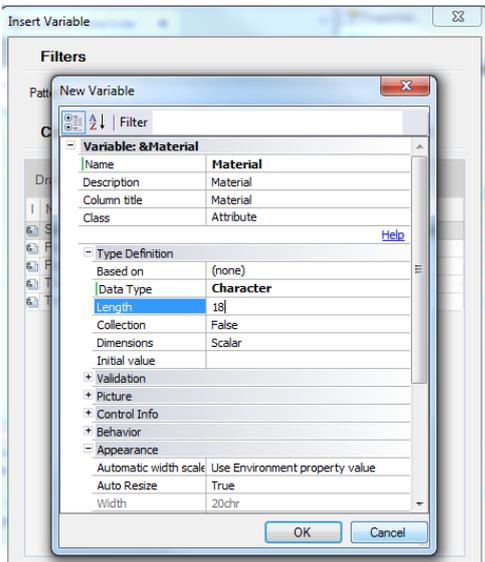


Y la insertamos en la pantalla para que el usuario pueda ingresar los campos del cabezal:

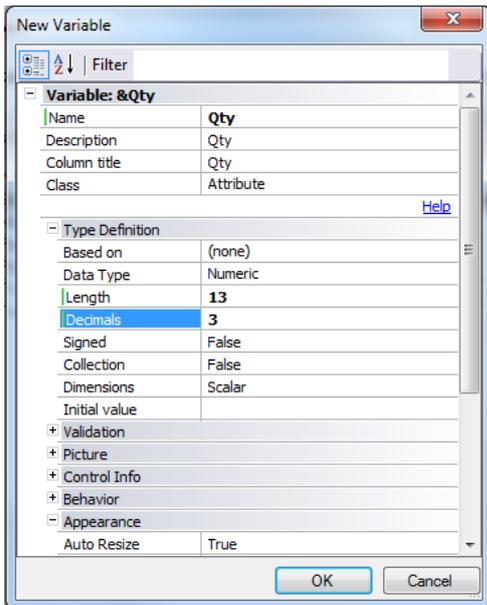




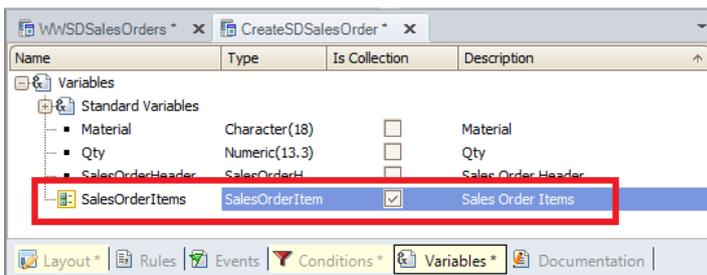
Y para las líneas, le pediremos que de cada una ingrese únicamente el material (caracter de 18):



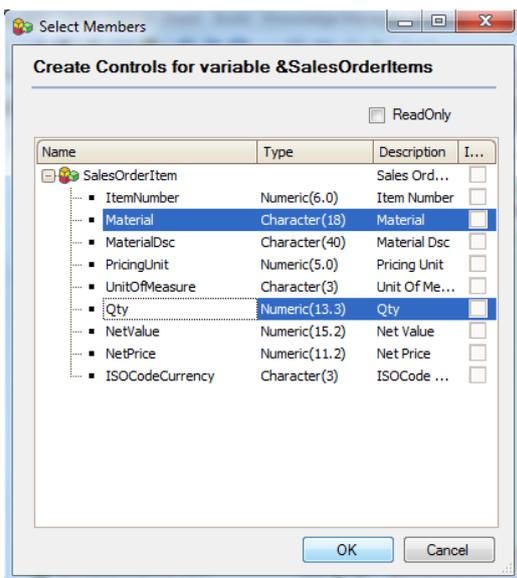
Y la cantidad llevada, quantity, un numérico de 13, de los cuales 3 son decimales:



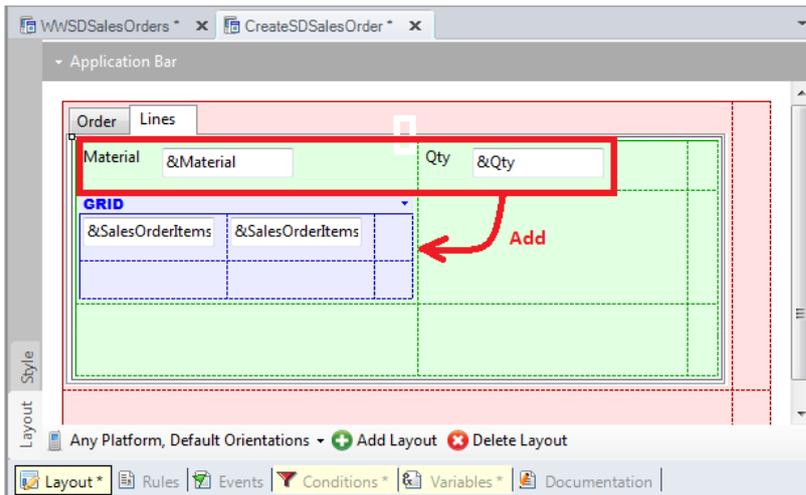
Necesitamos una variable que vaya almacenando todas las líneas que el usuario vaya ingresando en la pantalla. Para ello, definimos una variable SalesOrderItems de tipo SalesOrderItem, pero colección:



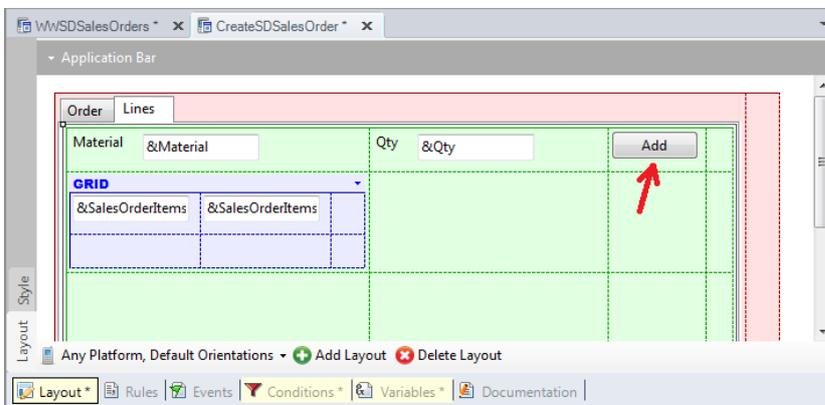
La insertamos en el layout, eligiendo mostrar únicamente el material y la cantidad:



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:

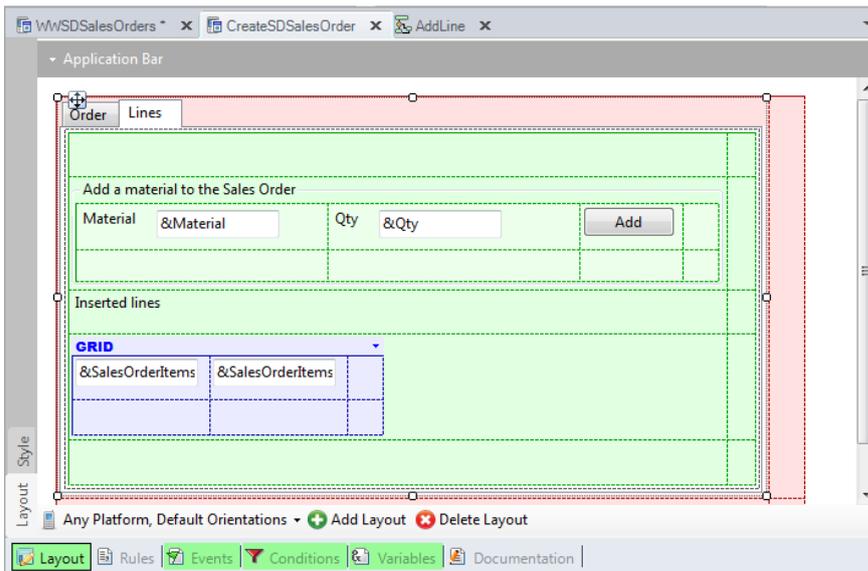
```
parm( in: &Material, in: &Qty, inout: &SalesOrderItems );
```

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)
```

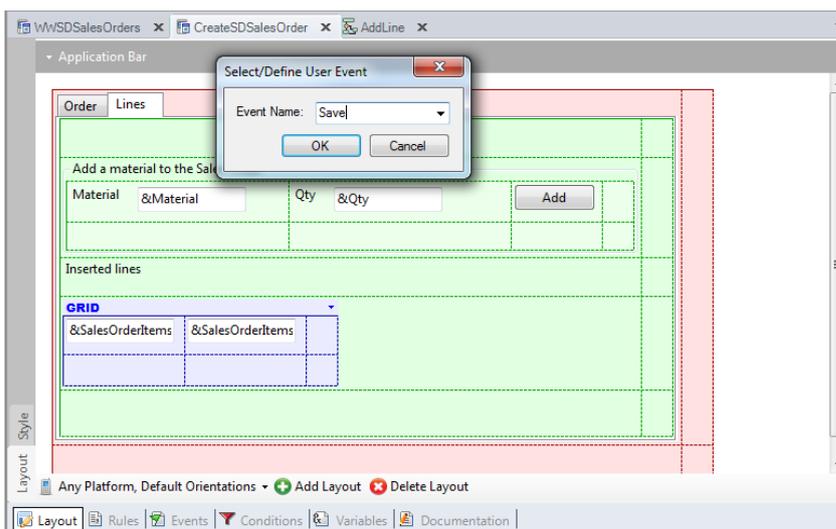
(Por ahora el método Add de una colección no puede especificarse dentro de un evento que se ejecuta en el dispositivo, y es por eso que tuvimos que hacerlo a través de este procedimiento y no directamente aquí. Distinto hubiera sido el caso si el evento que hacía el Add era uno del Server, como el Start,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

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



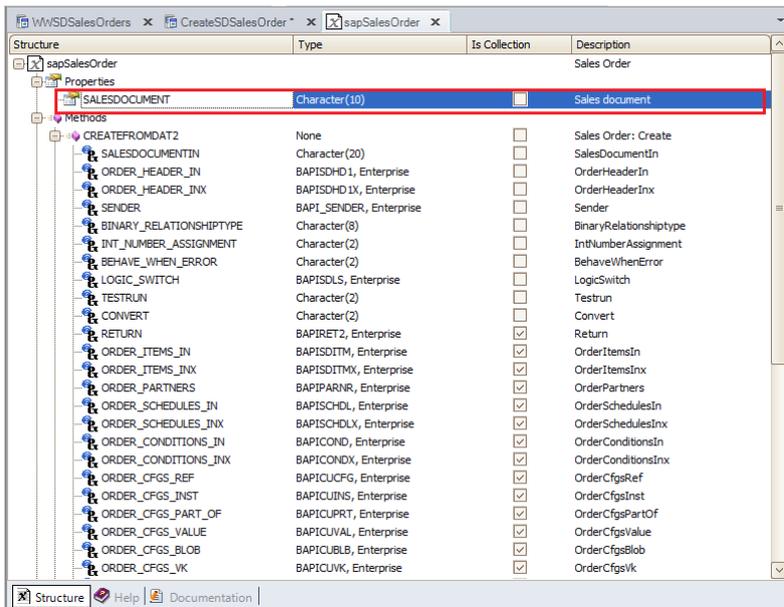
Una vez que el usuario haya dado valor a las variables del cabezal y haya ingresado las líneas, tenemos que ofrecerle un botón para que se cree el pedido de venta con esos datos.

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

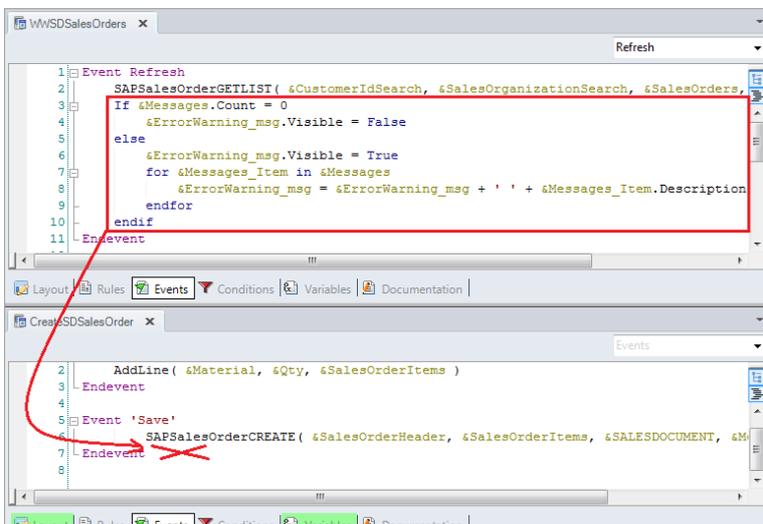


Donde invocaremos a un procedimiento que se conectará con el SAP ERP para intentar crear un nuevo pedido de venta con la información ingresada por el usuario en la pantalla.

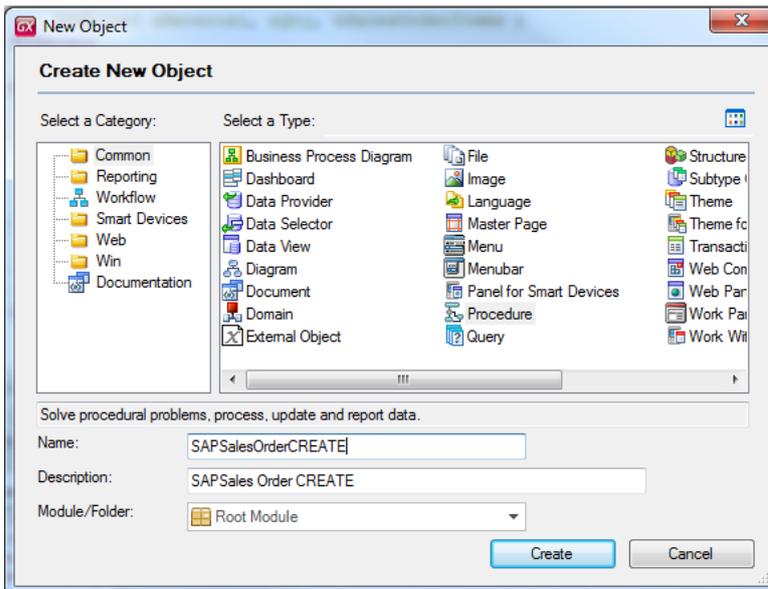
Lo llamaremos SAPSalesOrderCREATE, y debemos pasarle por parámetros de entrada las dos variables colección que contienen los datos de cabezal y líneas, esto es: &SalesOrderHeader y &SalesOrderItems, y nos devolverá una variable con el identificador del pedido ingresado, &SALESDOCUMENT, que si nos fijamos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



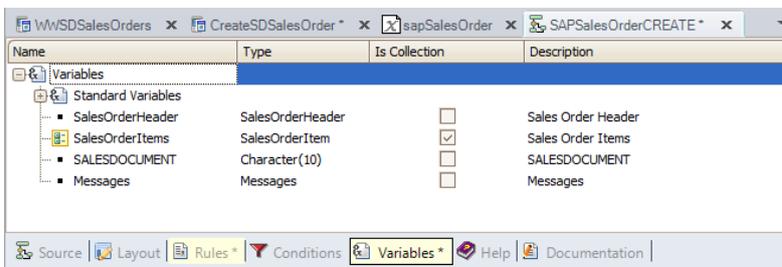
y otra, como siempre, &messages, con los posibles mensajes de advertencia o error, que al tratarse de un evento ejecutado en el cliente, es decir, en el dispositivo y no en el servidor (como era el caso del refresh) se manejarán automáticamente, no teniendo que recorrer esa colección para mostrarlos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



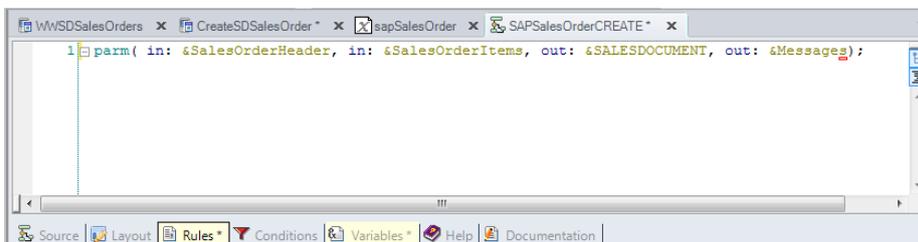
Para poder grabar, debemos crear el procedimiento:



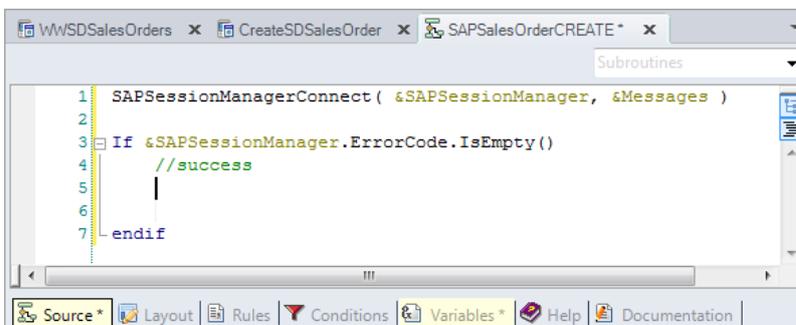
Definimos las variables que declaramos como parámetros:



Y las colocamos en la regla parm:

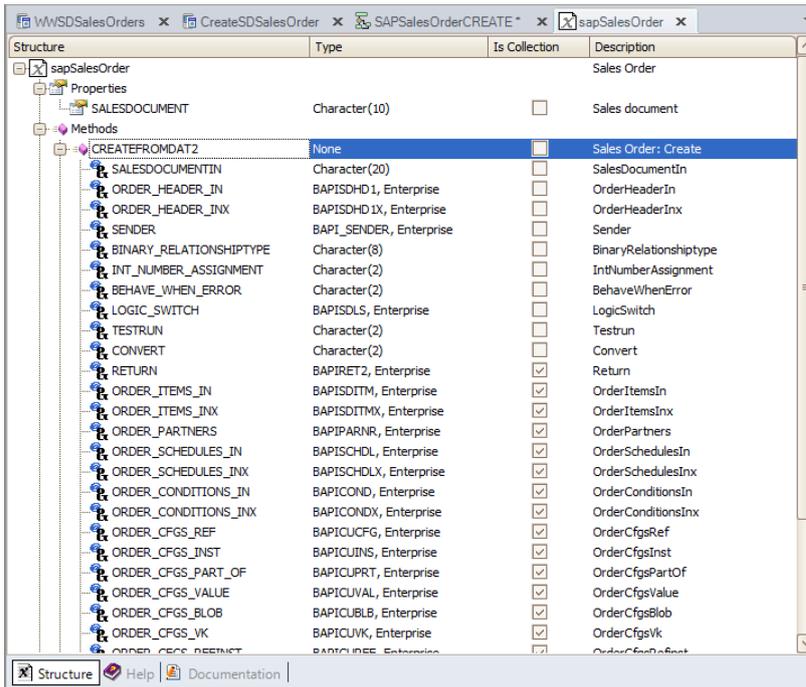


Lo primero que debemos hacer es conectarnos, como siempre, al ERP. Definimos, entonces, una variable SAPSessionManager, del tipo del objeto externo GXEnterpriseSessionManager e invocamos al procedimiento que realiza la conexión:

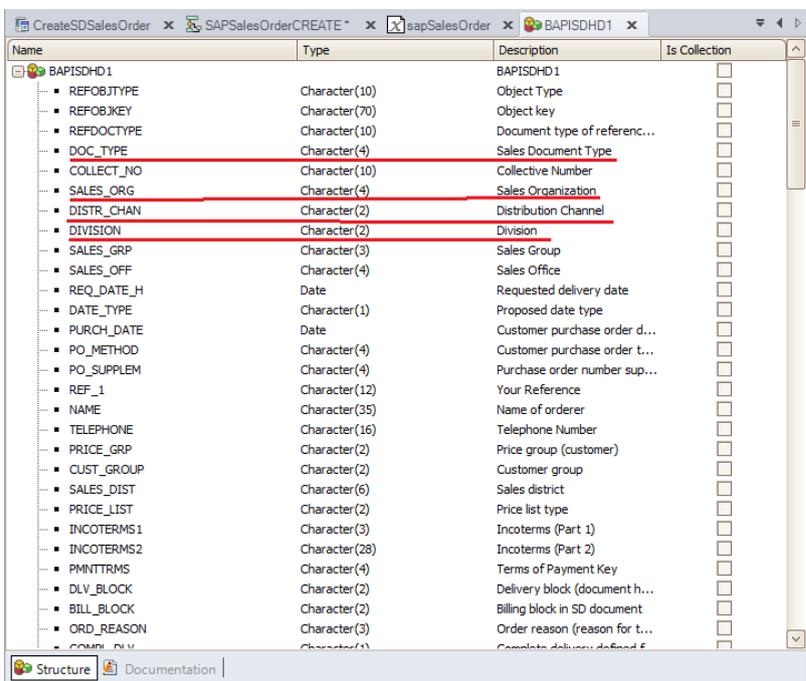


Si no hubo error de conexión, entonces debemos intentar crear la Sale Order.

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:



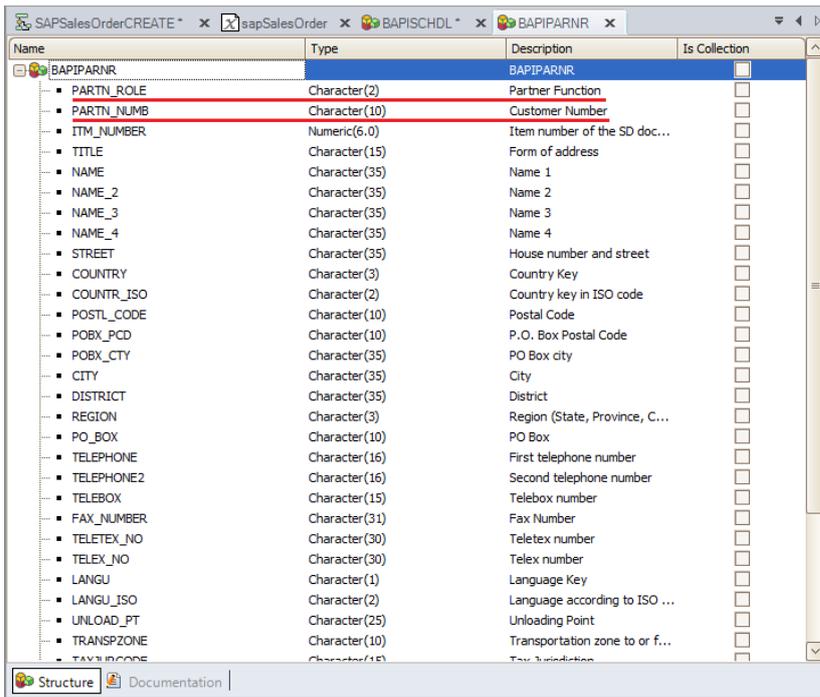
ORDER_ITEMS_IN, colección, se utiliza para ingresar los datos de cada línea (de cada línea solamente registraremos MATERIAL Y TARGET_QTY):

Name	Type	Description	Is Collection
BAPISDITM		BAPISDITM	<input type="checkbox"/>
ITM_NUMBER	Numeric(6.0)	Sales Document Item	<input type="checkbox"/>
HG_IV_ITEM	Numeric(6.0)	Higher-level item in bill of m...	<input type="checkbox"/>
PO_ITM_NO	Character(6)	Item Number of the Underl...	<input type="checkbox"/>
MATERIAL	Character(18)	Material Number	<input type="checkbox"/>
ALT_TO_ITM	Numeric(6.0)	Item for which this item is a...	<input type="checkbox"/>
CUST_MAT22	Character(22)	Customer's material number...	<input type="checkbox"/>
BATCH	Character(10)	Batch Number	<input type="checkbox"/>
DLV_GROUP	Numeric(3.0)	Delivery group (Items are d...	<input type="checkbox"/>
PART_DLV	Character(1)	Partial delivery at item level	<input type="checkbox"/>
REASON_REJ	Character(2)	Reason for rejection of quo...	<input type="checkbox"/>
BILL_BLOCK	Character(2)	Block	<input type="checkbox"/>
BILL_DATE	Date	Billing date for billing index ...	<input type="checkbox"/>
PLANT	Character(4)	Plant	<input type="checkbox"/>
STORE_LOC	Character(4)	Storage Location	<input type="checkbox"/>
TARGET_QTY	Numeric(13.3)	Target quantity in sales units	<input type="checkbox"/>
TARGET_QU	Character(3)	Target quantity UoM	<input type="checkbox"/>
T_UNIT_ISO	Character(3)	Target quantity unit of mea...	<input type="checkbox"/>
ITEM_CATEG	Character(4)	Sales document item category	<input type="checkbox"/>
SHORT_TEXT	Character(40)	Short text for sales order item	<input type="checkbox"/>
PRC_GROUP1	Character(3)	Material group 1	<input type="checkbox"/>
PRC_GROUP2	Character(3)	Material group 2	<input type="checkbox"/>
PRC_GROUP3	Character(3)	Material group 3	<input type="checkbox"/>
PRC_GROUP4	Character(3)	Material group 4	<input type="checkbox"/>
PRC_GROUP5	Character(3)	Material group 5	<input type="checkbox"/>
PROD_HIERA	Character(18)	Product hierarchy	<input type="checkbox"/>
MATL_GROUP	Character(9)	Material Group	<input type="checkbox"/>
PURCH_NO_C	Character(35)	Customer purchase order n...	<input type="checkbox"/>
PURCH_DATE	Date	Customer purchase order d...	<input type="checkbox"/>
PO_METHOD	Character(6)	Customer purchase order t...	<input type="checkbox"/>

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

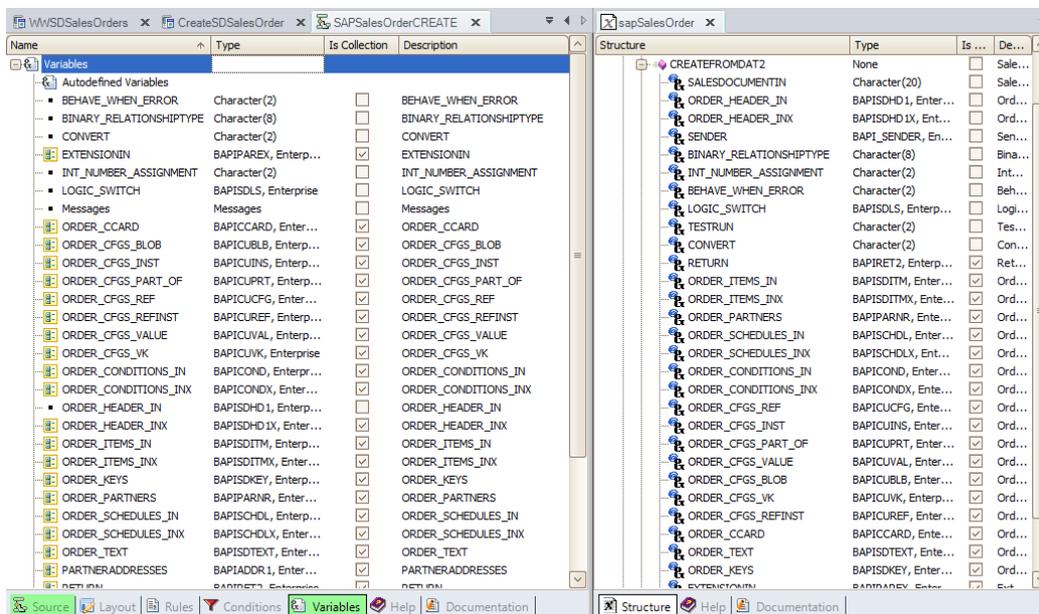
Name	Type	Description	Is Collection
BAPISCHDL		BAPISCHDL	<input type="checkbox"/>
ITM_NUMBER	Numeric(6.0)	Sales Document Item	<input type="checkbox"/>
SCHED_LINE	Numeric(4.0)	Delivery Schedule Line Number	<input type="checkbox"/>
REQ_DATE	Date	Schedule line date	<input type="checkbox"/>
DATE_TYPE	Character(1)	Date type (day, week, mont...	<input type="checkbox"/>
REQ_TIME	DateTime	Arrival time	<input type="checkbox"/>
REQ_QTY	Numeric(13.3)	Order quantity in sales units	<input type="checkbox"/>
REQ_DLV_BL	Character(2)	Schedule line blocked for deli...	<input type="checkbox"/>
SCHED_TYPE	Character(2)	Schedule line category	<input type="checkbox"/>
MS_DATE	Date	Material Staging_Availability ...	<input type="checkbox"/>
TP_DATE	Date	Transportation Planning Date	<input type="checkbox"/>
LOAD_DATE	Date	Loading Date	<input type="checkbox"/>
GI_DATE	Date	Goods Issue Date	<input type="checkbox"/>
TP_TIME	DateTime	Transp. Planning Time (Local...	<input type="checkbox"/>
MS_TIME	DateTime	Material Staging Time (Local...	<input type="checkbox"/>
LOAD_TIME	DateTime	Loading Time (Local Time Rel...	<input type="checkbox"/>
GI_TIME	DateTime	Time of Goods Issue (Local, ...	<input type="checkbox"/>
REFOBJTYPE	Character(10)	Object Type	<input type="checkbox"/>
REFOBJKEY	Character(70)	Object key	<input type="checkbox"/>
REFLOGSYS	Character(10)	Logical system	<input type="checkbox"/>
DLV_DATE	Date	Schedule line date	<input type="checkbox"/>
DLV_TIME	DateTime	Arrival time	<input type="checkbox"/>
REL_TYPE	Character(1)	Release type	<input type="checkbox"/>
PLAN_SCHED_TYPE	Character(1)	Schedule line type EDI	<input type="checkbox"/>

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 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:

ORDER_ITEMS_IN	BAPISDITM, Enterprise	<input checked="" type="checkbox"/>
ORDER_ITEMS_IN_Item	BAPISDITM, Enterprise	<input type="checkbox"/>
ORDER_ITEMS_INX	BAPISDITMX, Enterprise	<input checked="" type="checkbox"/>

Lo mismo sucede con ORDER_SCHEDULES_IN:

ORDER_SCHEDULES_IN	BAPISCHDL, Enterprise	<input checked="" type="checkbox"/>
ORDER_SCHEDULES_IN_Item	BAPISCHDL, Enterprise	<input type="checkbox"/>
ORDER_SCHEDULES_INX	BAPISCHDLX, Enterprise	<input checked="" type="checkbox"/>

Y con ORDER_PARTNERS:

ORDER_PARTNERS	BAPIPARNR, Enterprise	<input checked="" type="checkbox"/>
ORDER_PARTNERS_Item	BAPIPARNR, Enterprise	<input type="checkbox"/>
ORDER_SCHEDULES_IN	BAPISCHDL, Enterprise	<input checked="" type="checkbox"/>

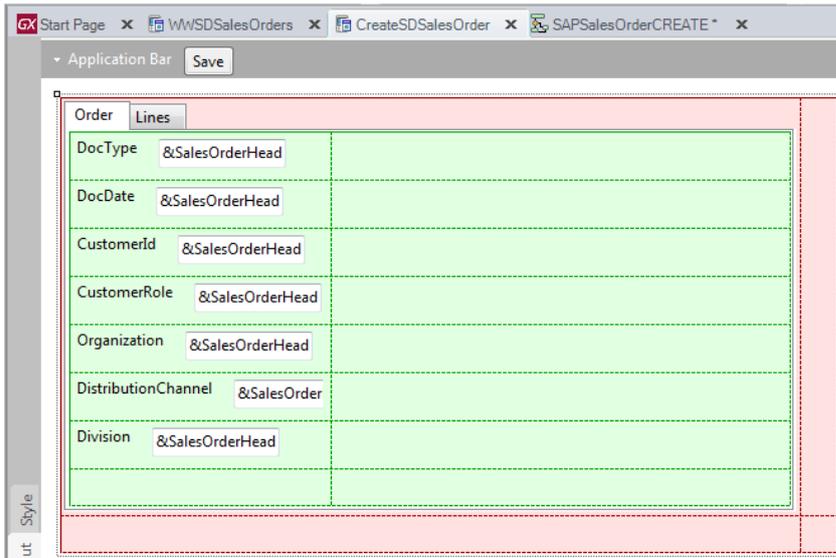
Aquí ya hemos incluido el código que carga esas variables:

```

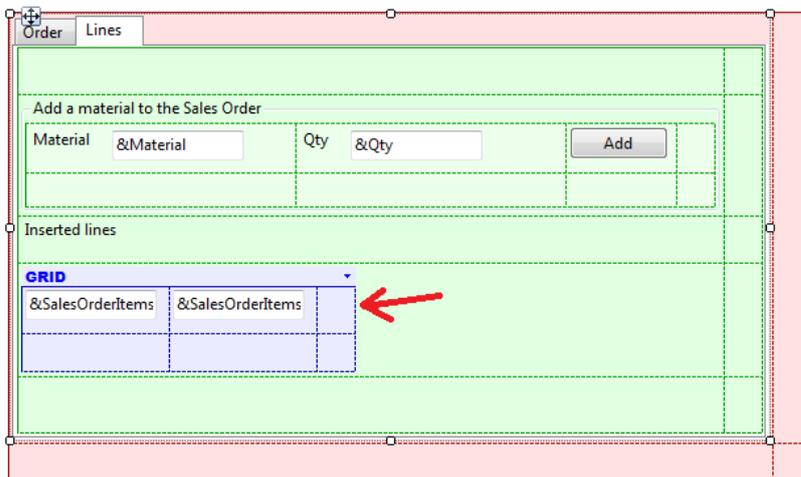
1  SAPSessionManagerConnect( &SAPSessionManager, &Messages )
2
3  If &SAPSessionManager.ErrorCode.IsEmpty()
4      //success
5
6      //ORDER HEADER
7      &ORDER_HEADER_IN.DOC_TYPE = &SalesOrderHeader.DocType
8      &ORDER_HEADER_IN.DOC_DATE = &SalesOrderHeader.DocDate
9      &ORDER_HEADER_IN.SALES_ORG = &SalesOrderHeader.Organization
10     &ORDER_HEADER_IN.SALES_DIST = &SalesOrderHeader.DistributionChannel
11     &ORDER_HEADER_IN.DIVISION = &SalesOrderHeader.Division
12
13     //ORDER MATERIALS & SCHEDULES
14     for &SalesOrderItem in &SalesOrderItems
15         &ORDER_ITEMS_IN_Item = new()
16         &ORDER_ITEMS_IN_Item.MATERIAL = &SalesOrderItem.Material
17         &ORDER_ITEMS_IN_Item.TARGET_QTY = &SalesOrderItem.Qty
18         &ORDER_ITEMS_IN.Add(&ORDER_ITEMS_IN_Item)
19
20         &ORDER_SCHEDULES_IN_Item = new()
21         &ORDER_SCHEDULES_IN_Item.REQ_QTY = &SalesOrderItem.Qty
22         &ORDER_SCHEDULES_IN.Add(&ORDER_SCHEDULES_IN_Item)
23     endfor
24
25     //ORDER PARTNERS
26     &ORDER_PARTNERS_Item.PARTN_ROLE = &SalesOrderHeader.CustomerRole
27     &ORDER_PARTNERS_Item.PARTN_NUMB = &SalesOrderHeader.CustomerId
28     &ORDER_PARTNERS.Add(&ORDER_PARTNERS_Item)
29
30 endif
31

```

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.

```
1 SAPSessionManagerConnect( &SAPSessionManager, &Messages )
2
3 If &SAPSessionManager.ErrorCode.IsEmpty()
4     //success
5
6     //ORDER HEADER
7     &ORDER_HEADER_IN.DOC_TYPE = &SalesOrderHeader.DocType
8     &ORDER_HEADER_IN.DOC_DATE = &SalesOrderHeader.DocDate
9     &ORDER_HEADER_IN.SALES_ORG = &SalesOrderHeader.Organization
10    &ORDER_HEADER_IN.SALES_DIST = &SalesOrderHeader.DistributionChannel
11    &ORDER_HEADER_IN.DIVISION = &SalesOrderHeader.Division
12
13    //ORDER MATERIALS & SCHEDULES
14    for &SalesOrderItem in &SalesOrderItems
15        &ORDER_ITEMS_IN_Item = new()
16        &ORDER_ITEMS_IN_Item.MATERIAL = &SalesOrderItem.Material
17        &ORDER_ITEMS_IN_Item.TARGET_QTY = &SalesOrderItem.Qty
18        &ORDER_ITEMS_IN.Add(&ORDER_ITEMS_IN_Item)
19
20        &ORDER_SCHEDULES_IN_Item = new()
21        &ORDER_SCHEDULES_IN_Item.REQ_QTY = &SalesOrderItem.Qty
22        &ORDER_SCHEDULES_IN.Add(&ORDER_SCHEDULES_IN_Item)
23    endfor
24
25    //ORDER PARTNERS
26    &ORDER_PARTNERS_Item.PARTN_ROLE = &SalesOrderHeader.CustomerRole
27    &ORDER_PARTNERS_Item.PARTN_NUMB = &SalesOrderHeader.CustomerId
28    &ORDER_PARTNERS.Add(&ORDER_PARTNERS_Item)
29
30    sapSalesOrder.CREATEFROMDAT2(&SALESDOCUMENT, &ORDER_HEADER_IN, &ORDER_HEADER_INX, &SENDER,
31    &BINARY_RELATIONSHIPTYPE, &INT_NUMBER_ASSIGNMENT, &BEHAVE_WHEN_ERROR, &LOGIC_SWITCH, &TESTRUN,
32    &CONVERT, &RETURN, &ORDER_ITEMS_IN, &ORDER_ITEMS_INX, &ORDER_PARTNERS, &ORDER_SCHEDULES_IN,
33    &ORDER_SCHEDULES_INX, &ORDER_CONDITIONS_IN, &ORDER_CONDITIONS_INX, &ORDER_CFGS_REF, &ORDER_CFGS_INST,
34    &ORDER_CFGS_PART_OF, &ORDER_CFGS_VALUE, &ORDER_CFGS_BLOB, &ORDER_CFGS_VK, &ORDER_CFGS_REFINST,
35    &ORDER_CCARD, &ORDER_TEXT, &ORDER_KEYS, &EXTENSIONIN, &PARTNERADDRESSES)
36
37 endif
38
```

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:

```

5
6 //ORDER HEADER
7 &ORDER_HEADER_IN.DOC_TYPE = &SalesOrderHeader.DocType
8 &ORDER_HEADER_IN.DOC_DATE = &SalesOrderHeader.DocDate
9 &ORDER_HEADER_IN.SALES_ORG = &SalesOrderHeader.Organization
10 &ORDER_HEADER_IN.SALES_DIST = &SalesOrderHeader.DistributionChannel
11 &ORDER_HEADER_IN.DIVISION = &SalesOrderHeader.Division
12
13 //ORDER MATERIALS & SCHEDULES
14 for &SalesOrderItem in &SalesOrderItems
15     &ORDER_ITEMS_IN_Item = new()
16     &ORDER_ITEMS_IN_Item.MATERIAL = &SalesOrderItem.Material
17     &ORDER_ITEMS_IN_Item.TARGET_QTY = &SalesOrderItem.Qty
18     &ORDER_ITEMS_IN.Add(&ORDER_ITEMS_IN_Item)
19
20     &ORDER_SCHEDULES_IN_Item = new()
21     &ORDER_SCHEDULES_IN_Item.REQ_QTY = &SalesOrderItem.Qty
22     &ORDER_SCHEDULES_IN.Add(&ORDER_SCHEDULES_IN_Item)
23
24 endfor
25
26 //ORDER PARTNERS
27 &ORDER_PARTNERS_Item.PARTN_ROLE = &SalesOrderHeader.CustomerRole
28 &ORDER_PARTNERS_Item.PARTN_NUMB = &SalesOrderHeader.CustomerId
29 &ORDER_PARTNERS.Add(&ORDER_PARTNERS_Item)
30
31 → &SAPSessionManager.TransactionBegin()
32
33 sapSalesOrder.CREATEFROMDAT2(&SALESDOCUMENT, &ORDER_HEADER_IN, &ORDER_HEADER_INX, &SENDER,
34     &BINARY_RELATIONSHIPTYPE, &INT_NUMBER_ASSIGNMENT, &BEHAVE_WHEN_ERROR, &LOGIC_SWITCH, &TESTRUN,
35     &CONVERT, &RETURN, &ORDER_ITEMS_IN, &ORDER_ITEMS_INX, &ORDER_PARTNERS, &ORDER_SCHEDULES_IN,
36     &ORDER_SCHEDULES_INX, &ORDER_CONDITIONS_IN, &ORDER_CONDITIONS_INX, &ORDER_CFGS_REF, &ORDER_CFGS_INST,
37     &ORDER_CFGS_PART_OF, &ORDER_CFGS_VALUE, &ORDER_CFGS_BLOB, &ORDER_CFGS_VK, &ORDER_CFGS_REFINST,
38     &ORDER_CCARD, &ORDER_TEXT, &ORDER_KEYS, &EXTENSIONIN, &PARTNERADDRESSES)
39
40 → &SAPSessionManager.TransactionCommit()
41
42 endif
43

```

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:

Name	Type	Description	Is Collection
BAPIRET2		BAPIRET2	<input type="checkbox"/>
TYPE	Character(1)	Message type: S Su...	<input type="checkbox"/>
ID	Character(20)	Message Class	<input type="checkbox"/>
NUMBER	Numeric(3.0)	Message Number	<input type="checkbox"/>
MESSAGE	Character(220)	Message Text	<input type="checkbox"/>
LOG_NO	Character(20)	Application log: log ...	<input type="checkbox"/>
LOG_MSG_NO	Numeric(6.0)	Application log: Inte...	<input type="checkbox"/>
MESSAGE_V1	Character(50)	Message Variable	<input type="checkbox"/>
MESSAGE_V2	Character(50)	Message Variable	<input type="checkbox"/>
MESSAGE_V3	Character(50)	Message Variable	<input type="checkbox"/>
MESSAGE_V4	Character(50)	Message Variable	<input type="checkbox"/>
PARAMETER	Character(32)	Parameter Name	<input type="checkbox"/>
ROW	Numeric(10.0)	Lines in parameter	<input type="checkbox"/>
FIELD	Character(30)	Field in parameter	<input type="checkbox"/>
SYSTEM	Character(10)	Logical system from...	<input type="checkbox"/>

Por lo que escribimos el siguiente código:

```

30  &SAPSessionManager.TransactionBegin()
31
32  sapSalesOrder.CREATEFROMDAT2(&SALESDOCUMENT, &ORDER_HEADER_IN, &ORDER_HEADER_IN,
33  &BINARY_RELATIONSHIPTYPE, &INT_NUMBER_ASSIGNMENT, &BEHAVE_WHEN_ERROR, &I
34  &CONVERT, &RETURN, &ORDER_ITEMS_IN, &ORDER_ITEMS_INX, &ORDER_PARTNERS, &
35  &ORDER_SCHEDULES_INX, &ORDER_CONDITIONS_IN, &ORDER_CONDITIONS_INX, &ORDE
36  &ORDER_CFGS_PART_OF, &ORDER_CFGS_VALUE, &ORDER_CFGS_BLOB, &ORDER_CFGS_VF
37  &ORDER_CCARD, &ORDER_TEXT, &ORDER_KEYS, &EXTENSIONIN, &PARTNERADDRESSES)
38
39  &SAPSessionManager.TransactionCommit()
40
41  // Error handling & SALESDOCUMENT ID
42  for &RETURN_Item in &RETURN
43  if &RETURN_Item.NUMBER = 311
44  &SALESDOCUMENT = '00000' + &RETURN_Item.MESSAGE_V2.Trim()
45  else
46  &Message_Item = new()
47  &Message_Item.Description = &RETURN_Item.MESSAGE
48  if &RETURN_Item.TYPE = 'A' or &RETURN_Item.TYPE = 'E'
49  &Message_Item.Type = MessageTypes.Error
50  &Messages.Add(&Message_Item)
51  endif
52  endif
53  endfor
54
55 -endif
56

```

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:

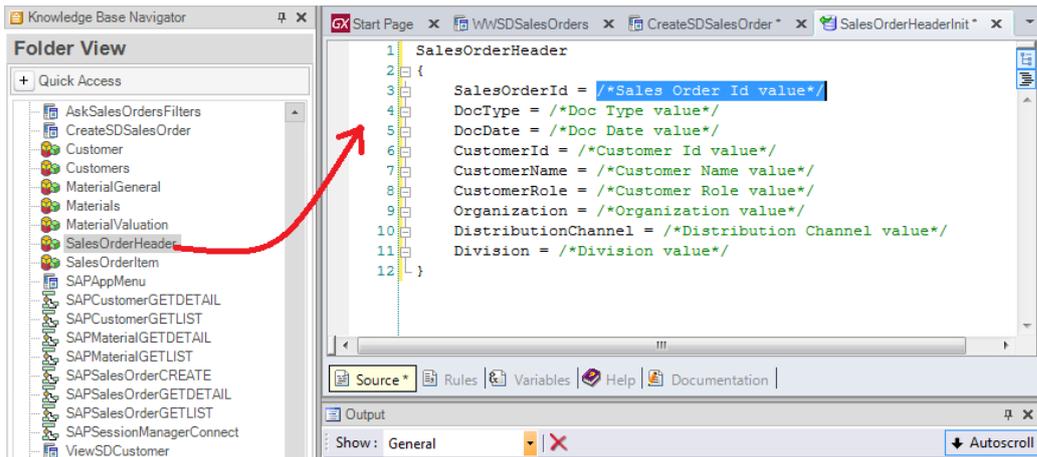
```

|Event Start
    &SalesOrderHeader = SalesOrderHeaderInit()
-Endevent

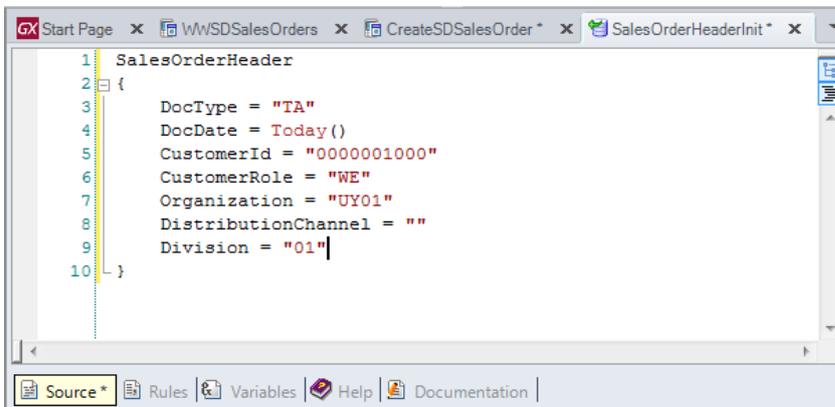
```

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

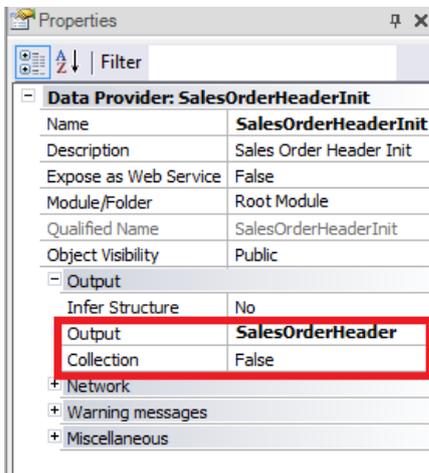
Arrastramos el tipo de datos estructurado SalesOrderHeader, que será el que el DataProvider devolverá cargado:



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



Aquí vemos la salida:



Una vez que se invocó al procedimiento para crear el pedido de venta, queremos 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

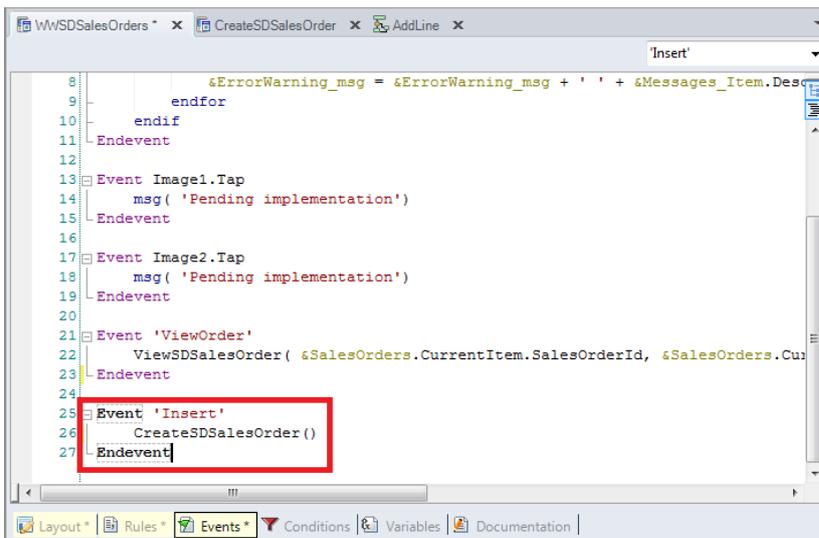
```

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.

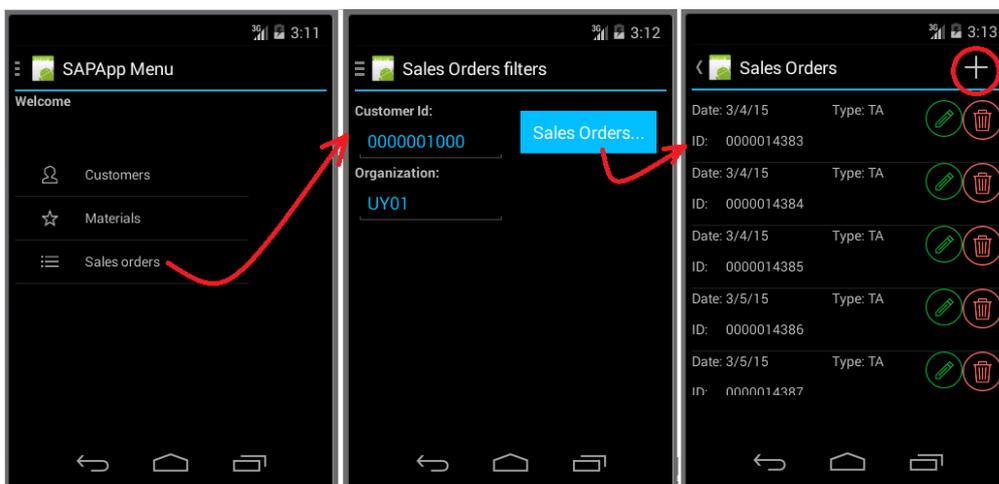
```
Event 'Save'  
  Composite  
    SAPSalesOrderCREATE( &SalesOrderHeader, &SalesOrderItems, &SALESDOCUMENT, &Messages )  
    ViewSDSalesOrder( &SALESDOCUMENT, &SalesOrderHeader.CustomerId, &SalesOrderHeader.Organization)  
  endcomposite  
Endevent
```

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



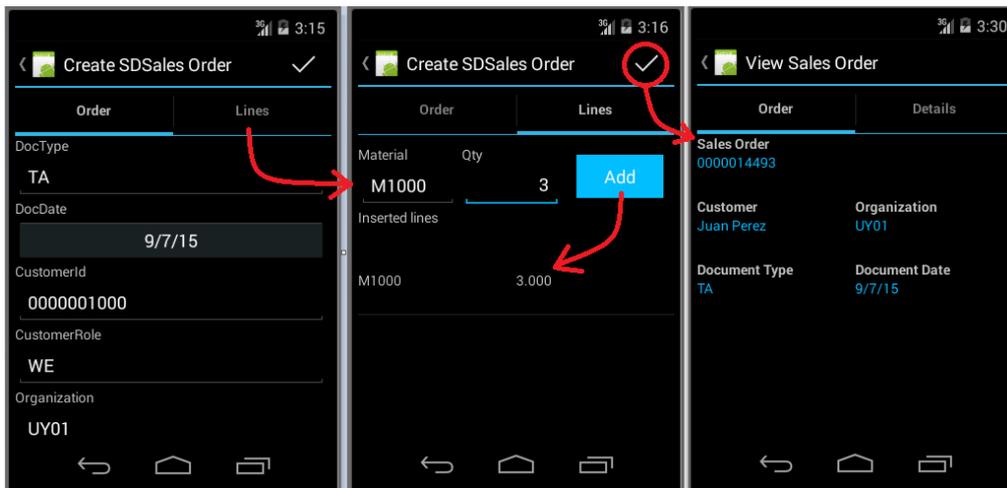
Ahora sí, F5

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:

```
1 Event 'Add'
2   AddLine( &Material, &Qty, &SalesOrderItems )
3 Endevent
4
5 Event 'Save'
6   Composite
7     SAPSalesOrderCREATE( &SalesOrderHeader, &SalesOrderItems, &SALESDOCUMENT, &Messages)
8     ViewSDSalesOrder.CallOptions.Type = CallType.Replace
9     ViewSDSalesOrder( &SALESDOCUMENT, &SalesOrderHeader.CustomerId, &SalesOrderHeader.Organization )
10  Endcomposite
11 Endevent
12
```

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.

