## 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):

	36 🖬 🖬 10:53
🗄 🔁 Sales Orders	filters
Customer Id:	
0000001000	Sales Orders
Organization:	
UY01	
$\leftarrow$	

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



y eligiendo uno de ellos, ver su detalle:

	36 10:57			<sup>36</sup> ∥ 🗗 10:58
< 🛜 View Sales (	Drder	< 👩	View Sale	es Order
Order	Details		Order	Details
Sales Order 0000014386		Materia	<b>al:</b> M1000	Material de Venta
Customer Juan Perez	Organization UY01	Unit	ST	Currency UYU
Document Type TA	Document Date 3/5/15	Qty	2.000	<b>Price</b> 100.00
				<b>Total</b> 200.00
← 1			Ĵ	

(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

Page 2

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

🐻 WWSDSalesOrders* 🗙 🕼 ViewSDSalesOrder 🗙
Application Bar Insert
&ErrorWarning_m
GRID
Date: &SalesOrders.iten Type: &SalesOrders.iten
ID: &SalesOrders.iten
30 <sup>3</sup>
Any Platform, Default Orientations - 😋 Add Layout 😮 Delete Layout
😺 Layout * 🗟 Rules * 😰 Events * 🝸 Conditions 🗟 Variables 🖺 Documentation
ⓑ WWSDSalesOrders * ★ ⓑ ViewSDSalesOrder ★
- Application Bar
& ErrorWarning_m
Order Details
Sales Order &SalesOrderHead
Customer &SalesOrderHead Organization &SalesOrderHead
Document Type &SalesOrderHea Document Date &SalesOrderHeac
Style
Age
Any Platform, Default Orientations - C Add Layout C Delete Layout
😺 Layout 🕺 Rules 🕅 Events 🟋 Conditions 🛍 Variables 🕌 Documentation

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



ⓑ SAPAppMenu ★	-	🖥 Ask	SalesOrdersFilte	rs X			36 10:04
Load	•		Application Bar			🗄 🗾 Sales Orders	filters
10 6OptionIcon.FromImage(Sales_order_icon_sd) 11 6OptionName = 'Sales orders' 12 Load 13. Endevent	- III		Customer Id:	&CustomerId	Sales Orders	Customer Id: 0000001000	Sales Orders
14 15 Event 'Select' 16 Composite			Organization:	&OrganizationId		Organization:	
17 if &OptionName = 'Customers' 18 WWSDCustomers() 19 else							
20 if &OptionName = 'Materials' 21 WWSDMaterials() 22 else		7					
23 if &OptionName = 'Sales orders' 24 AstSalesOrdersFilters()							
26 - endif 27 - endif 28 - Endemonstre		Ale					
29 LEndevent	•		Any Platform	III			
Conditions & Variables Documentation		- 💷	/out 🗟 Rules	Events T Cond	litions 🕲 Variables 🖠		

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:

	oMenu 🗙 🔚 AskSalesOrdersFilters 🗙	-
		Start 👻
1	Event Start	T
2	&CustomerId = '0000001	000'
3	&OrganizationId = 'UYO	1'
4	Endevent	<u>^</u>
5		
6	Event 'View Sales Orders'	
7	WWSDSalesOrders( &Cust	omerId, &OrganizationId )
8	Endevent	
9		
10		
		_
	i 	
😡 Layout	t 🗟 Rules 🛣 Events 🍸 Conditions 🕅	🕄 Variables 📕 Documentation

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:

ĺ	5	WWSDS	alesO	rders *	×	TiewSDSa	lesOrder 🗙					-
		1	jp8	arm(	in:	&Custome:	rIdSearch,	in:	&SalesOrganizationSearch	1);		> 1411 441
												Ŧ
	•										Þ	
	V	Layout		Rules	*	Events *	Conditions	🕄 Var	iables ይ Documentation			

en el procedimiento invocado:

Page4

6	W/SDSalesOrders* 🗙 🖪 ViewSDSalesOrder 🗙		-
	Refres	ı	•
	1 Event Refresh	(Maggarag)	E
	3 If &Messages.Count = 0	anessages)	3
	4 6ErrorWarning msg.Visible = False		<b>^</b>
	5 else		
	6 & ErrorWarning_msg.Visible = True		
	7 for &Messages_Item in &Messages		
	<pre>8 &amp;ErrorWarning_msg = &amp;ErrorWarning_msg + ' ' + &amp;Messages_Item.Description</pre>	L .	Ξ
	9 - endfor		
	10 - endif		
	L Endevent		
	15 Event image1.1ap		
	15 Endevent		
	16		
	17 Event Image2.Tap		
	<pre>18 msg( 'Pending implementation')</pre>		-
1	· · · · · · · · · · · · · · · · · · ·		•
	.ayout * 🗟 Rules * 🗑 Events * 🍸 Conditions 🕼 Variables 📓 Documentation		

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:

ucture	Туре	Is Collection	Description		
$\chi$ sapSalesOrder			Sales Order		
📄 🚰 Properties					
	Character(10)		Sales document		
📄 =💊 Methods					
😥 =💊 CREATEFROMDAT2	None		Sales Order: Create		
💼 🖘 GETLIST	None		List		
	Character(20)		CustomerNumber		
SALES_ORGANIZATION	Character(8)		SalesOrganization		
	Character(36)		Material		
	Date		DocumentDate		
	Date		DocumentDateTo		
	Character(40)		PurchaseOrder		
	Character(70)		PurchaseOrderNumber		
	Character(2)		TransactionGroup		
	BAPIRETURN, Enterprise		Return		
MATERIAL_EVG	BAPIMGVMATNR, Enterprise		materialEVG		
SALES_ORDERS	BAPIORDERS, Enterprise	$\checkmark$	SalesOrders		



Video filmado con GeneXus X Evolution 3 – upgrade 3

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:

WWSDSales0	Irders * 🗴 🕫 ViewSDSalesOrder 🗙 🔂 SAPSalesOrderGETLIST 🗙	-
		•
1 - Ev	ent Refresh	-
2	SAPSalesOrderGETDETAIL( &SalesOrderId, &CustomerId, &SalesOrganization, &SalesOrde	rHea
3 占	If &Messages.Count = 0	
4	ErrorWarning msg.Visible = False	<b>^</b>
5	else	
6	<pre>&amp;ErrorWarning msg.Visible = True</pre>	
7 占	for &Messages Item in &Messages	=
8	<pre>&amp;ErrorWarning msg = &amp;ErrorWarning msg + ' ' + &amp;Messages Item.Description</pre>	
9 -	endfor	
10 -	endif	
11 - End	devent	
12		-
•	m	•
🛃 Lavout 🗟 E	Rules 🗑 Events 🝸 Conditions 🚯 Variables 📳 Documentation	
🛃 cayout   🖂 i		
		,
(g) 11110000lic30		
		•
1 SA	PSessionManagerConnect ( &SAPSessionManager, &Messages )	1.5
2		4
3 🗆 If	<pre>&amp;SAPSessionManager.ErrorCode.IsEmpty()</pre>	
4	//success	
5	sapSalesOrder.GETLIST( &CustomerId, &SalesOrganization, &MATERIAL, &DOCUMENT DATE	. &DC =
6	if sRETURN.CODE.ISEmpty()	
7	//success	
8	&SalesOrderHeader,SalesOrderId = &SalesOrderId	
9	&SalesOrderHeader.CustomerId = &CustomerId	
10	&SalesOrderHeader.Organization = &SalesOrganization	
11	Sheader ok = False	
12	for ASLES ORDERS ITEM IN ASLES ORDERS	-
		•
-		
🏖 Source 😡	Layout   🗟 Rules   🍸 Conditions   🔄 Variables   🖤 Help   🖆 Documentation	

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.

		🌯 🖬 2:55
< 🔂 Sales Ord	ers	<b></b> +
Date: 3/4/15	Type: TA	
ID: 0000014383		
Date: 3/4/15	Type: TA	
ID: 0000014384		
Date: 3/4/15	Type: TA	
ID: 0000014385		
Date: 3/5/15	Type: TA	
ID: 0000014386		
Date: 3/5/15	Type: TA	
ID: 0000014387		
$\hookrightarrow$	$\Box$	

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.

Page**O** 

Para implementar todo esto con GeneXus...

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

6	🖥 WWSDSalesOrders 🗙							
	•	Application Bar						
		&ErrorWarning_m						
		CRID						
		Date: &SalesOrders.iten Type: &SalesOrders.iten	=					
		ID: &SalesOrders.iten						
Style	Style							
ţ	₩ ₩							
Lay	🚊 📗 Any Platform, Default Orientations 🗸 😋 Add Layout 😮 Delete Layout							
V	🔯 Layout 🗟 Rules 📆 Events 🝸 Conditions 🚱 Variables 🖉 Documentation							

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

To WWSDSalesOrders ×	
+ Application Bar	
8:ErrorWarning_m     Select/Define User Event       CRID     Event Name: Inset]       Date:     8:SalesOrders.iten       ID:     8:SalesOrders.iten	

WWSDSales	sOrders * 🗙		
	ation Bar Insert		
&Error\	Warning_m		
GRID			
GRID Date:	&SalesOrders.iten	Type: &SalesOrders.iten	
CRID Date: ID:	&SalesOrders.iten &SalesOrders.iten	Type: &SalesOrders.iten	

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:

Select a category.	Select a Type:
Common Workflow Reporting Web Win Documentation	Dashboard     Panel for Smart Devices     Theme for Smart Devices     Work With for Smart Devices
Smart Device applicatio	n form. Interact with end user using events, show or request data and more.
Smart Device applicatio	n form. Interact with end user using events, show or request data and more. CreateSDSalesOrder
Smart Device application Name: Description:	In form. Interact with end user using events, show or request data and more. Create SDSalesOrder Create SDSales Order

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:

🕫 WWSDSalesOrders * 🗙 🛱 CreateSDSalesOrder * 🗙 🗸

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):



ame	Туре	Description	Is Collection
🚱 SalesOrderHeader		Sales Order Header	
<ul> <li>SalesOrderId</li> </ul>	Character(10)	Sales Order Id	
··· • DocType	Character(4)	Doc Type	
• DocDate	Date	Doc Date	
···· • CustomerId	Character(10)	Customer Id	
··· • CustomerName	Character(40)	Customer Name	
CustomerRole	Character(2)	Customer Role	
····   Organization	Character(4)	Organization	
<ul> <li>DistributionChannel</li> </ul>	Character(2)	Distribution Channel	
	Character(2)	Division	
Division     Structure     Documentation     SalesOrderItem	Character(2)	Division Description	Is Collection
	Character(2)	Division Description Sales Order Item	Is Collection
SalesOrderItem SalesOrderItem SalesOrderItem Teme	Character(2)	Division Description Sales Order Item Item Number	Is Collection
	Character(2) Type Numeric(6.0) Character(18)	Division Description Sales Order Item Item Number Material	Is Collection
	Character(2)	Division Description Sales Order Item Item Number Material Material Dsc	Is Collection
	Character(2)	Division  Description  Sales Order Item Item Number Material Material Dsc Pricing Unit	Is Collection
	Character(2)	Division Description Sales Order Item Item Number Material Material Dsc Pricing Unit Unit Of Measure	Is Collection
	Character (2) Type Numeric(6.0) Character (18) Character (40) Numeric(5.0) Character (3) Numeric(13.3)	Division Description Sales Order Item Item Number Material Material Dsc Pricing Unit Unit Of Measure Qty	Is Collection
SalesOrderItem Sales	Character(2)	Division Description Sales Order Item Item Number Material Material Dsc Pricing Unit Unit Of Measure Qty Net Value	Is Collection
	Character(2)	Division Description Sales Order Item Item Number Material Material Dsc Pricing Unit Unit Of Measure Qty Net Value Net Price	Is Collection

Definiremos entonces, una variable del tipo de datos SalesOrderHeader:

🕞 WWSDSalesOrders * 🗙 🖪 Cr	eateSDSalesOrder *	🗙 🐉 Sales Order Heade	er X
Name	Туре	↑ Is Collection	Description
🖃 🚷 Variables			
🕀 🚷 Standard Variables			
SalesOrderHeader	SalesOrderHeader		Sales Order Header
😺 Layout * 🗟 Rules 🗹 Events	Conditions *	🛍 Variables * 🙆 Doc	umentation

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

			ReadOnly
Name		Туре	Description I
🖃 🎲 Sal	esOrderHeader		Sales Ord
	SalesOrderId	Character(10)	Sales Ord
	DocType	Character(4)	Doc Type
	DocDate	Date	Doc Date
	CustomerId	Character(10)	Customer Id
	CustomerName	Character(40)	Customer
	CustomerRole	Character(2)	Customer
	Organization	Character(4)	Organization
	DistributionChannel	Character(2)	Distributio
	Division	Character(2)	Division
	Division	Character(2)	Division



li w	VSDSalesOrders * 🗴 🖪 CreateSDSalesOrder * 🗙	-
-	r Application Bar	
	Order	Â
	DocType &SalesOrderHead	
	DocDate &SalesOrderHead	
	CustomerId &SalesOrderHead	
	CustomerRole &SalesOrderHead	E
	Organization &SalesOrderHead	
	DistributionChannel &SalesOrder	
	Division &SalesOrderHead	
Style		
♦ IT		
Lay 🔋	Any Platform, Default Orientations 👻 😋 Add Layout 🔞 Delete Layout	

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

C	Yariable ⊉↓   Filter <b>Variable: &amp;Materi.</b> Name Description Column title Class	al Material Material Material	
C C C C C C C C C C C C C C C C C C C	r Variable 2↓   Filter Variable: &Materia Name Description Column title Class	al Material Material	
		al Material Material	
	<sup>2</sup> ↓         Filter           Variable:         &Materia           Name            Description            Column title            Class	al Material Material Material	
	Variable: &Materi Name Description Column title Class	al Material Material	
	Name Description Column title Class	Material Material Material	-
	Description Column title Class	Material Material	
	Column title Class	Material	
	Class		
		Attribute	
F		Help	
T	<ul> <li>Type Definition</li> </ul>		
Ť	Based on	(none)	-
	Data Type	Character	_
	Length	18	-
	Collection	False	- 11
	Dimensions	Scalar	- 11
	Initial value		- 11
	+ Validation		_
	+ Picture		- 11
	+ Control Info		- 11
	+ Behavior		- 11
	<ul> <li>Appearance</li> </ul>		- 11
	Automatic width se	cale Use Environment property value	- 11
	Auto Resize	True	-
	Width	20chr	-
		OK Cancel	

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

New Variable				
2↓   Filter				
Variable: &Qty		~		
Name	Qty			
Description	Qty			
Column title	Qty			
Class	Attribute			
	Help			
<ul> <li>Type Definition</li> </ul>				
Based on	(none)	Ε		
Data Type	Numeric			
Length	13			
Decimals	3			
Signed	False			
Collection	False			
Dimensions	Scalar			
Initial value				
+ Validation				
+ Picture				
+ Control Info				
+ Behavior				
<ul> <li>Appearance</li> </ul>				
Auto Resize	True	Ŧ		
	OK Cancel			

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:

		ReadOnly
Name	Туре	Description I
🖃 🎲 SalesOrderItem		Sales Ord
<ul> <li>ItemNumber</li> </ul>	Numeric(6.0)	Item Number
- • Material	Character(18)	Material
····  MaterialDsc	Character(40)	Material Dsc
····  PricingUnit	Numeric(5.0)	Pricing Unit
UnitOfMeasure	Character(3)	Unit Of Me
• Qty	Numeric(13.3)	Qty
• NetValue	Numeric(15.2)	Net Value
···· • NetPrice	Numeric(11.2)	Net Price
<ul> <li>ISOCodeCurrency</li> </ul>	Character(3)	ISOCode

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

Page 🗕

Image: Image	Ŧ
→ Application Bar	
	 •
Order Lines	
Material &Material Qty &Qty	
GRID	
&SalesOrderItems &SalesOrderItems Add	
	Ε
AS	
5	 _
Any Platform Default Orientations - C Add Layout C Delete Layout	 -
😺 Layout * 🗟 Rules 🕱 Events 🔻 Conditions * 🗟 Variables * 🙆 Documentation	

Para ello insertamos un botón asociado a un evento Add:

🕫 WWSDSalesOrders * 🗙 🕞 CreateSDSalesOrder *	x	*
→ Application Bar		
Order Lines		1
Material &Material	Qty &Qty Add	=
CRID BiSalarOrderitama	7	
Conescidenteins Conescidenteins		
Style		
ort		Ŧ
🚊 📗 Any Platform, Default Orientations 🝷 🛟 Add Lag	yout 😢 Delete Layout	
Layout * 🗟 Rules 😰 Events 🍸 Conditions * 🕴	Variables *	

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:

to www	alesOrders * 🗴 🖥 CreateSDSalesOrder 🗙 🗟 AddLine 🗙 🖛						
- /							
0	der Lines						
	Add a material to the Sales Order						
	Vlaterial &dMaterial Qty &Qty Add						
0	serted lines						
	RID ·						
	CsalesUrderitems						
ut Style							
😤 👖 Any Platform, Default Orientations 🗸 😯 Add Layout 😮 Delete Layout							
😡 Layo	🗟 Rules 🕅 Events 🝸 Conditions 💽 Variables 📓 Documentation						

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.

🖥 WWSDSalesOrders 🗙 🛅 CreateSDSalesOrder 🗙 🐼 AddLine 🗴 Select/Define User Event × Order Lines Event Name: Save OK Cancel Add a material to the Sa Qty &Qty Material &Material Add Inserted lines GRID &SalesOrderItems &SalesOrderItems Style Layout 📗 Any Platform, Default Orientations 🝷 🛟 Add Layout 🙁 Delete Layout 😺 Layout 🗟 Rules 🕱 Events 🝸 Conditions 🗟 Variables 🖉 Documentation

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:

 ${}^{\text{Page}}13$ 

icture	Туре	Is Collection	Description	
x sapSalesOrder			Sales Order	
🔄 🚰 Properties				
- SALESDOCUMENT	Character(10)		Sales document	
🗐 🐨 Methods				
CREATEFROMDAT2	None		Sales Order: Create	
	Character(20)		SalesDocumentIn	
	BAPISDHD1, Enterprise		OrderHeaderIn	
	BAPISDHD1X, Enterprise		OrderHeaderInx	
	BAPI_SENDER, Enterprise		Sender	
	Character(8)		BinaryRelationshiptype	
	Character(2)		IntNumberAssignment	
	Character(2)		BehaveWhenError	
Core Switch	BAPISDLS, Enterprise		LogicSwitch	
	Character(2)		Testrun	
	Character(2)		Convert	
	BAPIRET2, Enterprise	$\checkmark$	Return	
	BAPISDITM, Enterprise	$\checkmark$	OrderItemsIn	
	BAPISDITMX, Enterprise	$\checkmark$	OrderItemsInx	
	BAPIPARNR, Enterprise	$\checkmark$	OrderPartners	
	BAPISCHDL, Enterprise	$\checkmark$	OrderSchedulesIn	
	BAPISCHDLX, Enterprise	$\checkmark$	OrderSchedulesInx	
	BAPICOND, Enterprise	$\checkmark$	OrderConditionsIn	
	BAPICONDX, Enterprise	$\checkmark$	OrderConditionsInx	
	BAPICUCFG, Enterprise	$\checkmark$	OrderCfgsRef	
	BAPICUINS, Enterprise	$\checkmark$	OrderCfgsInst	
	BAPICUPRT, Enterprise	$\checkmark$	OrderCfgsPartOf	
	BAPICUVAL, Enterprise	$\checkmark$	OrderCfgsValue	
	BAPICUBLB, Enterprise	$\checkmark$	OrderCfgsBlob	
	BAPICUVK, Enterprise	$\checkmark$	OrderCfgsVk	

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:

MW/SDSalesOr	ders X		-
		Refresh	-
		nerrestr	
1 🖂 Eve	nt Refresh		10
2	SAPSalesOrderGETLIST( &CustomerIdSearch, &SalesOrganizationSea	rch, &SalesOrders	3, 3
3 户	If &Messages.Count = 0		<u> </u>
4	&ErrorWarning_msg.Visible = False		
5	else		=
6	<pre>&amp;ErrorWarning_msg.Visible = True</pre>		
7 白	for &Messages_Item in &Messages		
8	<pre>&amp;ErrorWarning_msg = &amp;ErrorWarning_msg + ' ' + &amp;Message</pre>	s_Item.Descriptio	on
9 -	endfor		
10 -	endif		
11 L En	event		-
			F.
🔯 Layout 🗟 R	ules 🕱 Events 🍸 Conditions 🚯 Variables 🖉 Documentation		
CreateSDSales	Urder X		
			•
2	AddLine( &Material, &Qty, &SalesOrderItems )		12
3 - End	event		-
4			=
5 🖂 Eve	nt 'Save'		
6	SAPSalesOrderCREATE( &SalesOrderHeader, &SalesOrderItems,	&SALESDOCUMENT, &	6Me
7 - End	event		=
8			
		1	
			,
R Lavout	ules 🗑 Events 🝸 Conditions 🚱 Variables 🚯 Documentation		

Para poder grabar, debemos crear el procedimiento:

 $_{\text{Page}}14$ 

Create New Obje	ct		
Select a Category:	Select a Type:		
Common Reporting Workflow Smart Devices Web Win Documentation	Business Process Diagram Dashboard Data Provider Data Selector Data Selector Data View Diagram Document Document Ex External Object	Image         Image	Structure Subtype Theme Theme Transact Web Co Web Pa Work Pa Work W
	< <u> </u>		ł
Solve procedural probler Name:	ns, process, update and report data.		
Description:	SAPSales Order CREATE	1	
Module/Folder:	Root Module	<b>~</b>	
		Create	Cancel

Definimos las variables que declararemos como parámetros:

🕞 WWSDSalesOrders 🗙 🕞 Cr	eateSDSalesOrder *	🗙 🔀 sapSalesOrder 🗴	SAPSalesOrderCREATE* ×	-
Name	Туре	Is Collection	Description	
E 🛃 Variables				
Comparison of the standard Variables     SalesOrderHeader     SalesOrderItems     SALESDOCUMENT     Messages	SalesOrderHeader SalesOrderItem Character(10) Messages		Sales Order Header Sales Order Items SALESDOCUMENT Messages	
🐱 Source 😺 Layout 🗟 Rules	* 🝸 Conditions 🗜	🗎 Variables * 🤣 Help	Documentation	

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:

ture	Туре	Is Collection	Description	
sapSalesOrder			Sales Order	
Properties				
SALESDOCUMENT	Character(10)		Sales document	
e 🍬 Methods				
CREATEFROMDAT2	None		Sales Order: Create	
SALESDOCUMENTIN	Character(20)		SalesDocumentIn	
	BAPISDHD1, Enterprise		OrderHeaderIn	
	BAPISDHD 1X, Enterprise		OrderHeaderInx	
SENDER.	BAPI_SENDER, Enterprise		Sender	
BINARY_RELATIONSHIPTYPE	Character(8)		BinaryRelationshiptype	
	Character(2)		IntNumberAssignment	
	Character(2)		BehaveWhenError	
	BAPISDLS, Enterprise		LogicSwitch	
" TESTRUN	Character(2)		Testrun	
	Character(2)		Convert	
	BAPIRET2, Enterprise	$\checkmark$	Return	
	BAPISDITM, Enterprise	$\checkmark$	OrderItemsIn	
	BAPISDITMX, Enterprise	$\checkmark$	OrderItemsInx	
	BAPIPARNR, Enterprise	$\checkmark$	OrderPartners	
	BAPISCHDL, Enterprise	$\checkmark$	OrderSchedulesIn	
ORDER_SCHEDULES_INX	BAPISCHDLX, Enterprise	$\checkmark$	OrderSchedulesInx	
ORDER_CONDITIONS_IN	BAPICOND, Enterprise	$\checkmark$	OrderConditionsIn	
ORDER_CONDITIONS_INX	BAPICONDX, Enterprise	$\checkmark$	OrderConditionsInx	
	BAPICUCFG, Enterprise	$\checkmark$	OrderCfgsRef	
ORDER_CFGS_INST	BAPICUINS, Enterprise	$\checkmark$	OrderCfgsInst	
ORDER_CFGS_PART_OF	BAPICUPRT, Enterprise	$\checkmark$	OrderCfgsPartOf	
ORDER_CFGS_VALUE	BAPICUVAL, Enterprise	$\checkmark$	OrderCfgsValue	
ORDER_CFGS_BLOB	BAPICUBLB, Enterprise		OrderCfgsBlob	
ORDER_CFGS_VK	BAPICUVK, Enterprise	$\checkmark$	OrderCfgsVk	
	PADICI DEE Enterprise		Order Ofgethe	

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:

🕼 CreateSDSalesOrder 🗙	∑ SAPSalesOrderCREATE* ×	🔀 sapSalesOrder 🗙 🍪 BAPISDHD1 🗙	₹ 4 ▷
Name	Type	Description	Is Collection
E BAPISDHD 1		BAPISDHD 1	
<ul> <li>REFOBJTYPE</li> </ul>	Character(10)	Object Type	
<ul> <li>REFOBJKEY</li> </ul>	Character(70)	Object key	
<ul> <li>REFDOCTYPE</li> </ul>	Character(10)	Document type of referenc	
DOC_TYPE	Character(4)	Sales Document Type	
- COLLECT_NO	Character(10)	Collective Number	
SALES_ORG	Character(4)	Sales Organization	
DISTR_CHAN	Character(2)	Distribution Channel	
- DIVISION	Character(2)	Division	
- SALES_GRP	Character(3)	Sales Group	
- SALES_OFF	Character(4)	Sales Office	
- REQ_DATE_H	Date	Requested delivery date	
DATE_TYPE	Character(1)	Proposed date type	
- PURCH_DATE	Date	Customer purchase order d	
- PO_METHOD	Character(4)	Customer purchase order t	
- PO_SUPPLEM	Character(4)	Purchase order number sup	
• REF_1	Character(12)	Your Reference	
• NAME	Character(35)	Name of orderer	
··· • TELEPHONE	Character(16)	Telephone Number	
- PRICE_GRP	Character(2)	Price group (customer)	
• CUST_GROUP	Character(2)	Customer group	
• SALES_DIST	Character(6)	Sales district	
PRICE_LIST	Character(2)	Price list type	
INCOTERMS1	Character(3)	Incoterms (Part 1)	
INCOTERMS2	Character(28)	Incoterms (Part 2)	
PMNTTRMS	Character(4)	Terms of Payment Key	
DLV_BLOCK	Character(2)	Delivery block (document h	
BILL_BLOCK	Character(2)	Billing block in SD document	
ORD_REASON	Character(3)	Order reason (reason for t	
	Character/1)	Complete delivery defined f	
💕 Structure 🔮 Docume	ntation		

ORDER\_ITEMS\_IN, colección, se utiliza para ingresar los datos de cada línea (de cada línea solamente registraremos MATERIAL Y TARGET\_QTY):

CreatesusalesUrder X 25 SA	SalesUrderUKEATE X X sapsa	alesurder × workAPISUITM ×	To Collection
ne	Туре	Description	Is Collection
BAPISDITM		BAPISDITM	
··· • ITM_NUMBER	Numeric(6.0)	Sales Document Item	
HG_LV_ITEM	Numeric(6.0)	Higher-level item in bill of m	
···· • PO_ITM_NO	Character(6)	Item Number of the Underl	
··· • MATERIAL	Character(18)	Material Number	
• ALT_TO_ITM	Numeric(6.0)	Item for which this item is a	
···· • CUST_MAT22	Character(22)	Customer's material number	
BATCH	Character(10)	Batch Number	
DLV_GROUP	Numeric(3.0)	Delivery group (items are d	
··· • PART_DLV	Character(1)	Partial delivery at item level	
REASON_REJ	Character(2)	Reason for rejection of quo	
BILL_BLOCK	Character(2)	Block	
BILL_DATE	Date	Billing date for billing index	
··· • PLANT	Character(4)	Plant	
STORE_LOC	Character(4)	Storage Location	
TARGET_QTY	Numeric(13.3)	Target quantity in sales units	
• TARGET_QU	Character(3)	Target quantity UoM	
···· • T_UNIT_ISO	Character(3)	Target quantity unit of mea	
ITEM_CATEG	Character(4)	Sales document item category	
- SHORT_TEXT	Character(40)	Short text for sales order item	
PRC_GROUP1	Character(3)	Material group 1	
PRC_GROUP2	Character(3)	Material group 2	
PRC_GROUP3	Character(3)	Material group 3	
PRC_GROUP4	Character(3)	Material group 4	
PRC_GROUP5	Character(3)	Material group 5	
PROD_HIERA	Character(18)	Product hierarchy	
MATL_GROUP	Character(9)	Material Group	
PURCH_NO_C	Character(35)	Customer purchase order n	
PURCH_DATE	Date	Customer purchase order d	
	Character(A)	Customer purchase order t	

ORDER\_SCHEDULES\_IN, se utiliza para agendar las fechas de entrega para cada material (registraremos únicamente la cantidad pedida):

🖻 CreateSDSalesOrder 🗙 🚡 SAPSalesOrder	CREATE* 🗙 🔀 sapSalesOr	der 🗙 🚱 BAPISCHDL* 🗙	₹ 4 ▷
Name	Туре	Description	Is Collection
BAPISCHDL		BAPISCHDL	
ITM_NUMBER	Numeric(6.0)	Sales Document Item	
<ul> <li>SCHED_LINE</li> </ul>	Numeric(4.0)	Delivery Schedule Line Number	
• REQ_DATE	Date	Schedule line date	
<ul> <li>DATE_TYPE</li> </ul>	Character(1)	Date type (day, week, mont	
<ul> <li>REQ_TIME</li> </ul>	DateTime	Arrival time	
• REQ_QTY	Numeric(13.3)	Order quantity in sales units	
REQ_DLV_BL	Character(2)	Schedule line blocked for deli	
<ul> <li>SCHED_TYPE</li> </ul>	Character(2)	Schedule line category	
MS_DATE	Date	Material Staging_Availability	
<ul> <li>TP_DATE</li> </ul>	Date	Transportation Planning Date	
<ul> <li>LOAD_DATE</li> </ul>	Date	Loading Date	
<ul> <li>GI_DATE</li> </ul>	Date	Goods Issue Date	
<ul> <li>TP_TIME</li> </ul>	DateTime	Transp. Planning Time (Local	
<ul> <li>MS_TIME</li> </ul>	DateTime	Material Staging Time (Local,	
LOAD_TIME	DateTime	Loading Time (Local Time Rel	
<ul> <li>GI_TIME</li> </ul>	DateTime	Time of Goods Issue (Local,	
<ul> <li>REFOBJTYPE</li> </ul>	Character(10)	Object Type	
<ul> <li>REFOBJKEY</li> </ul>	Character(70)	Object key	
<ul> <li>REFLOGSYS</li> </ul>	Character(10)	Logical system	
<ul> <li>DLV_DATE</li> </ul>	Date	Schedule line date	
DLV_TIME	DateTime	Arrival time	
REL_TYPE	Character(1)	Release type	
PLAN_SCHED_TYPE	Character(1)	Schedule line type EDI	

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

 $_{\mathsf{Page}}17$ 

e	Туре	Description	Is Collection	
BAPIPARNR		BAPIPARNR		
PARTN_ROLE	Character(2)	Partner Function		
- PARTN_NUMB	Character(10)	Customer Number		
ITM_NUMBER	Numeric(6.0)	Item number of the SD doc		
···· • TITLE	Character(15)	Form of address		
- NAME	Character(35)	Name 1		
NAME_2	Character(35)	Name 2		
• NAME_3	Character(35)	Name 3		
• NAME_4	Character(35)	Name 4		
- STREET	Character(35)	House number and street		
COUNTRY	Character(3)	Country Key		
COUNTR_ISO	Character(2)	Country key in ISO code		
POSTL_CODE	Character(10)	Postal Code		
POBX_PCD	Character(10)	P.O. Box Postal Code		
POBX_CTY	Character(35)	PO Box city		
• CITY	Character(35)	City		
IDISTRICT	Character(35)	District		
- REGION	Character(3)	Region (State, Province, C		
• PO_BOX	Character(10)	PO Box		
- TELEPHONE	Character(16)	First telephone number		
···· TELEPHONE2	Character(16)	Second telephone number		
• TELEBOX	Character(15)	Telebox number		
FAX_NUMBER	Character(31)	Fax Number		
• TELETEX_NO	Character(30)	Teletex number		
TELEX_NO	Character(30)	Telex number		
IANGU	Character(1)	Language Key		
IANGU_ISO	Character(2)	Language according to ISO		
UNLOAD_PT	Character(25)	Unloading Point		
- TRANSPZONE	Character(10)	Transportation zone to or f		
	Character(1E)	Tax Incidiction		

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).

🕼 WWSDSalesOrders 🗙 盾 Create	ⓑ WWSDSalesOrder x ⓑ CreateSDSalesOrder x ₺ SAPSalesOrderCREATE x 🗧 4 ▷ 🔀 sapSalesOrder x 🗸												
Name 🛧	Туре	Is Collection	Description		^	Struct	ture			Туре	Is	De	^
🖃 🚷 Variables							¢.	=Q CR	EATEFROMDAT2	None		Sale	
🔚 Autodefined Variables								- <b>°</b> e	SALESDOCUMENTIN	Character(20)		Sale	
··· BEHAVE_WHEN_ERROR	Character(2)		BEHAVE_WHEN_E	RROR				®	ORDER_HEADER_IN	BAPISDHD1, Enter		Ord	
BINARY_RELATIONSHIPTYPE	Character(8)		BINARY_RELATIO	NSHIPTYPE					ORDER_HEADER_INX	BAPISDHD1X, Ent		Ord	
• CONVERT	Character(2)		CONVERT					<u>_</u>	SENDER	BAPI_SENDER, En		Sen	
	BAPIPAREX, Enterp	$\checkmark$	EXTENSIONIN					<u></u>	BINARY_RELATIONSHIPTYPE	Character(8)		Bina	
INT_NUMBER_ASSIGNMENT	Character(2)		INT_NUMBER_ASS	SIGNMENT				<u></u>	INT_NUMBER_ASSIGNMENT	Character(2)		Int	
LOGIC_SWITCH	BAPISDLS, Enterprise		LOGIC_SWITCH					- <b>®</b> -	BEHAVE_WHEN_ERROR	Character(2)		Beh	
··· • Messages	Messages		Messages					- <b>®</b> -	LOGIC_SWITCH	BAPISDLS, Enterp		Logi	
······································	BAPICCARD, Enter	$\checkmark$	ORDER_CCARD					- <b>®</b>	TESTRUN	Character(2)		Tes	
······································	BAPICUBLB, Enterp	$\checkmark$	ORDER_CFGS_BL	OB				- 6	CONVERT	Character(2)		Con	
······································	BAPICUINS, Enterp	$\checkmark$	ORDER_CFGS_INS	ST				- 6	RETURN	BAPIRET2, Enterp	$\checkmark$	Ret	
	BAPICUPRT, Enterp	$\checkmark$	ORDER_CFGS_PA	RT_OF				- 6	ORDER_ITEMS_IN	BAPISDITM, Enter	$\checkmark$	Ord	
	BAPICUCFG, Enter	$\checkmark$	ORDER_CFGS_RE	F				<u>_</u>	ORDER_ITEMS_INX	BAPISDITMX, Ente	$\checkmark$	Ord	
	BAPICUREF, Enterp	$\checkmark$	ORDER_CFGS_RE	FINST				<u>_</u>	ORDER_PARTNERS	BAPIPARNR, Ente	$\checkmark$	Ord	
	BAPICUVAL, Enterp	$\checkmark$	ORDER_CFGS_VA	LUE				- 6	ORDER_SCHEDULES_IN	BAPISCHDL, Enter	$\checkmark$	Ord	
	BAPICUVK, Enterprise	$\checkmark$	ORDER_CFGS_VK					-6	ORDER_SCHEDULES_INX	BAPISCHDLX, Ent	$\checkmark$	Ord	
	BAPICOND, Enterpr	$\checkmark$	ORDER_CONDITIO	ONS_IN				- <b>®</b>	ORDER_CONDITIONS_IN	BAPICOND, Enter	$\checkmark$	Ord	
	BAPICONDX, Enter	$\checkmark$	ORDER_CONDITIO	ONS_INX				-	ORDER_CONDITIONS_INX	BAPICONDX, Ente	$\checkmark$	Ord	
··· • ORDER_HEADER_IN	BAPISDHD1, Enterp		ORDER_HEADER_	IN				- 6	ORDER_CFGS_REF	BAPICUCFG, Ente	$\checkmark$	Ord	
	BAPISDHD1X, Enter	$\checkmark$	ORDER_HEADER_	INX				- 6	ORDER_CFGS_INST	BAPICUINS, Enter	$\checkmark$	Ord	
	BAPISDITM, Enterp	$\checkmark$	ORDER_ITEMS_IN	l i				<u>_</u>	ORDER_CFGS_PART_OF	BAPICUPRT, Enter	$\checkmark$	Ord	
	BAPISDITMX, Enter	$\checkmark$	ORDER_ITEMS_IN	IX				<u>_</u>	ORDER_CFGS_VALUE	BAPICUVAL, Enter	$\checkmark$	Ord	
	BAPISDKEY, Enterp	$\checkmark$	ORDER_KEYS					<u>_</u>	ORDER_CFGS_BLOB	BAPICUBLB, Enter	$\checkmark$	Ord	
	BAPIPARNR, Enter	$\checkmark$	ORDER_PARTNER	S				- 6	ORDER_CFGS_VK	BAPICUVK, Enterp	$\checkmark$	Ord	
	BAPISCHDL, Enterp	$\checkmark$	ORDER_SCHEDUL	ES_IN				-6	ORDER_CFGS_REFINST	BAPICUREF, Enter	$\checkmark$	Ord	
	BAPISCHDLX, Enter	$\checkmark$	ORDER_SCHEDUL	ES_INX					ORDER_CCARD	BAPICCARD, Ente	$\checkmark$	Ord	
ORDER_TEXT	BAPISDTEXT, Enter	$\checkmark$	ORDER_TEXT					-	ORDER_TEXT	BAPISDTEXT, Ente	$\checkmark$	Ord	
PARTNERADDRESSES	BAPIADDR 1, Enter	$\checkmark$	PARTNERADDRES	SES				-@	ORDER_KEYS	BAPISDKEY, Enter	$\checkmark$	Ord	_
	DADIDETO Enternino		DETLIDNI					0	EVTENICIONITM	DADTDADEV Entor		=+ l	<u> </u>
🐱 Source 😡 Layout 🗟 Rules 🎙	🍸 Conditions 😢 V	ariables 🥏 H	Help 👫 Docume	entation		🔊 S	truct	ture	🤣 Help 🔮 Documentation	I			

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	$\checkmark$
··· • ORDER_ITEMS_IN_Item	BAPISDITM, Enterprise	
	BAPISDITMX, Enterprise	$\checkmark$
Lo mismo sucede con ORDE	R SCHEDULES IN:	

🖁 =	ORDER_SCHEDULES_IN	BAPISCHDL, Enterprise	$\checkmark$
•	ORDER_SCHEDULES_IN_Item	BAPISCHDL, Enterprise	
🔡 =	ORDER_SCHEDULES_INX	BAPISCHDLX, Enterprise	$\checkmark$

Y con ORDER\_PARTNERS:

	ORDER_PARTNERS	BAPIPARNR, Enterprise	$\checkmark$
•	ORDER_PARTNERS_Item	BAPIPARNR, Enterprise 🛛 🖂	
	ORDER_SCHEDULES_IN	BAPISCHDL, Enterprise	$\checkmark$

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):



GX Start	t Page 🗴 🗟 WWSDSalesOrders 🗙 🗟 CreateSDSalesOrder 🗙 🐱 SAPSalesOrderCREATE* 🗙	
- /	Application Bar Save	
	Order Lines	
	DocType &SalesOrderHead	
	DocDate &SalesOrderHead	
	CustomerId &SalesOrderHead	
	CustomerRole &SalesOrderHead	
	Organization &SalesOrderHead	
	DistributionChannel &SalesOrder	
	Division &SalesOrderHead	
yle		
rt S		

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:

Add a mate	rial to th	ne Sales Oro	der					
Material	&Mater	rial		Qty	&Qty	]	Add	
Inserted line	5							Ó
GRID		-			· .			
&SalesOrde	rItems	&SalesOr	derItems		4			
					J			 

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:

CreateSDSalesOrder X	2. SAPSalesUrderCREATE ×	BAPIRET2 ×	<b>₩</b> .4
lame	Туре	Description	Is Collection
BAPIRET2		BAPIRET2	
··· • TYPE	Character(1)	Message type: S Su	
• ID	Character(20)	Message Class	
··· • NUMBER	Numeric(3.0)	Message Number	
···  MESSAGE	Character(220)	Message Text	
IOG_NO	Character(20)	Application log: log	
IOG_MSG_NO	Numeric(6.0)	Application log: Inte	
···  MESSAGE_V1	Character(50)	Message Variable	
···  MESSAGE_V2	Character(50)	Message Variable	
··· • MESSAGE_V3	Character(50)	Message Variable	
··· • MESSAGE_V4	Character(50)	Message Variable	
- PARAMETER	Character(32)	Parameter Name	
• ROW	Numeric(10.0)	Lines in parameter	
- FIELD	Character(30)	Field in parameter	
<ul> <li>SYSTEM</li> </ul>	Character(10)	Logical system from	

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:

GX Start Page	🗙 🚡 WWSDSalesOrders 🗴 🚡 CreateSDSalesOrder * 🗙 😭 SalesOrderHeaderInit * 🗴	x 🔽
1 S	alesOrderHeader	15
2 🖂 {		들
3	DocType = "TA"	-
4	DocDate = Today()	
5	CustomerId = "0000001000"	
6	CustomerRole = "WE"	
7	Organization = "UY01"	
8	DistributionChannel = ""	
9	Division = "01"	
10 - }		
		-
1.		•
Source*	🗟 Rules 🚱 Variables 🤣 Help 🖺 Documentation	

## Aquí vemos la salida:

![](_page_23_Picture_4.jpeg)

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

Page 24

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:

![](_page_24_Picture_4.jpeg)

Ahora sí, F5

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

![](_page_24_Picture_7.jpeg)

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).

361 🖉 3:15	36 🖬 🖬 3:16	3°1 🖬 3:30
< 📝 Create SDSales Order 🛛 🗸	< 📝 Create SDSales Order	< 🔁 View Sales Order
Order Lines	Order Lines	Order Details
DocType TA	Material Qty M1000 3 Add	Sales Order 0000014493
DocDate	Inserted lines	CustomerOrganizationJuan PerezUY01
9/7/15		
CustomerId	M1000 3.000	Document Type Document Date
000001000		10 3/1/13
CustomerRole		
WE		
Organization		
UY01		

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:

🐼 Start Page 🗴 🗟 WWSDSalesOrders 🗴 🗟 CreateSDSalesOrder 🗙 🚋 SAPSalesOrderCREATE* 🗙 💱 BAPIRET2 🗴		-		
	'Save'	•		
1 Event 'Add'		12		
2 AddLine( &Material, &Qty, &SalesOrderItems )				
3 Endevent		=		
4		ĥ		
5 Event 'Save'				
6 Composite				
7 SAPSalesOrderCREATE ( &SalesOrderHeader, &SalesOrderItems, &SALESDOCUMENT, &Messag	(es)	Ξ		
8 ViewSDSalesOrder.CallOptions.Type = CallType.Replace				
9 ViewSDSalesOrder( & SALESDOCUMENT, & SalesOrderHeader.CustomerId, & SalesOrderHeader.Organization )				
10 - Endcomposite				
11 LEndevent		-		
	•			
🕼 Layout 🗟 Rules 📆 Events 🍸 Conditions 🔕 Variables 📓 Documentation				

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.

![](_page_26_Picture_9.jpeg)