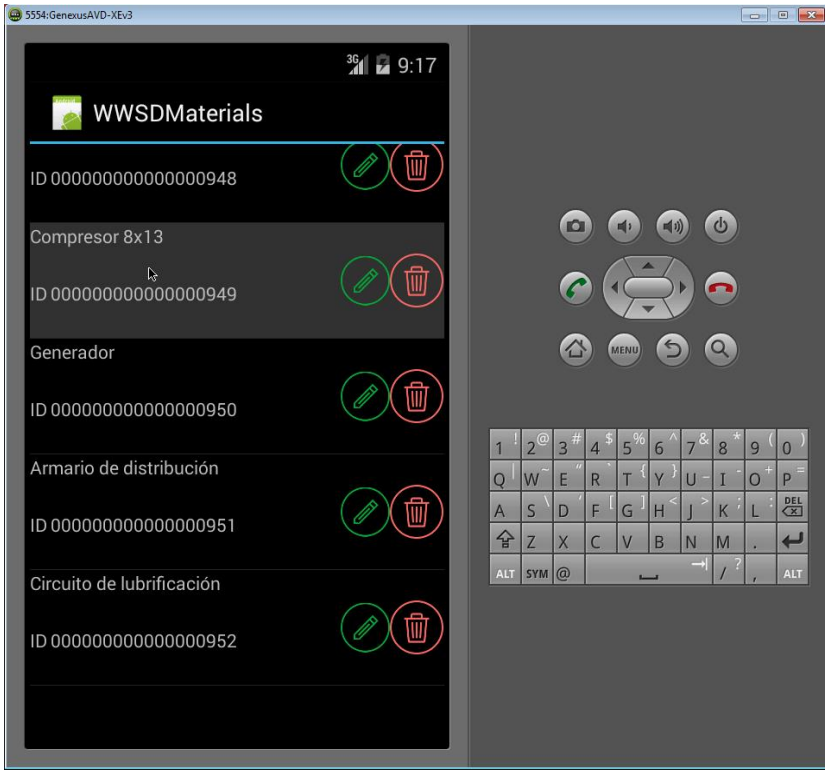
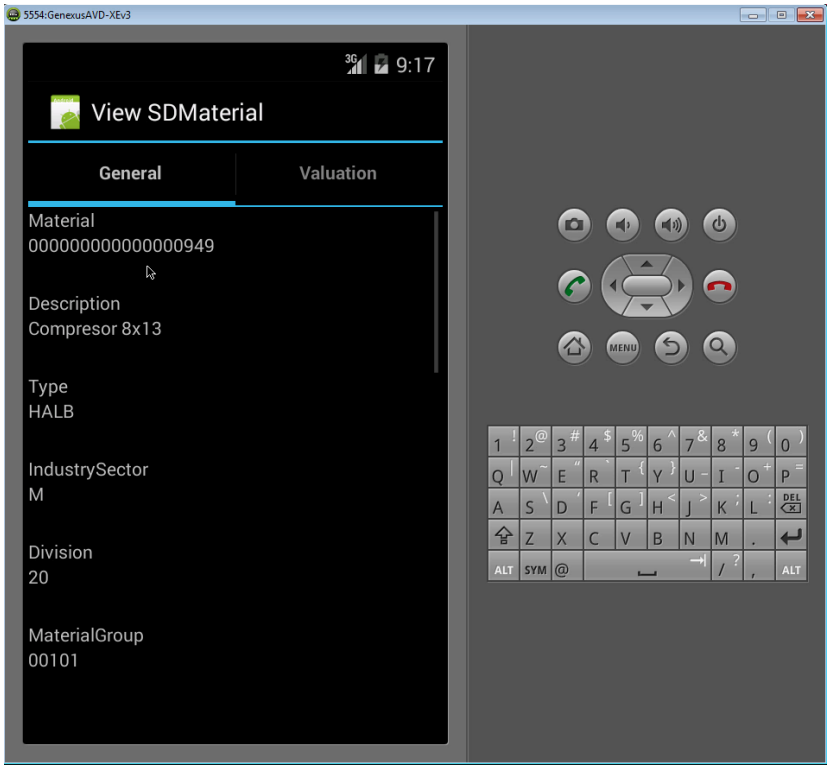
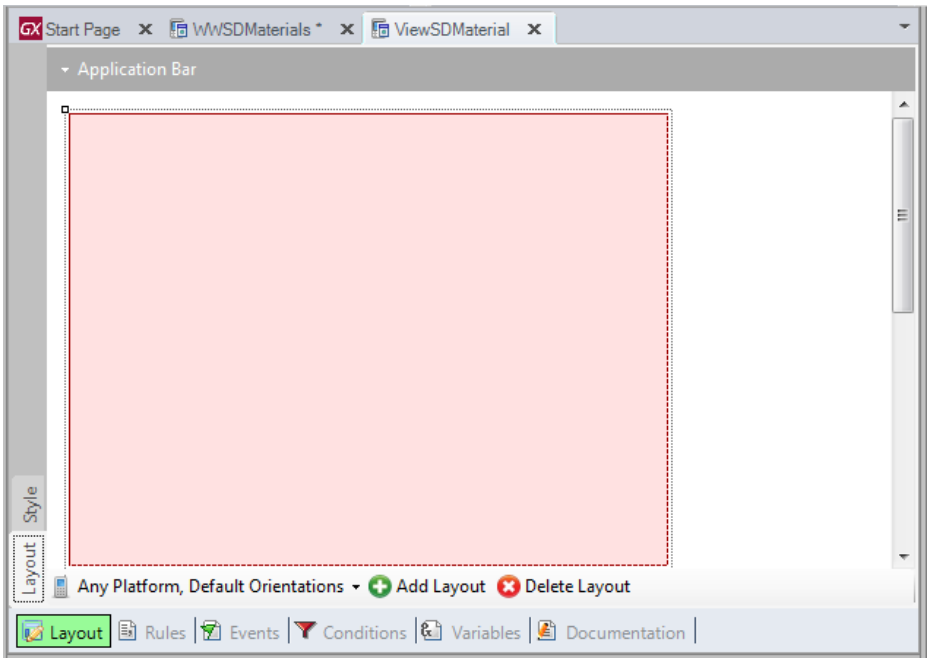
Cómo visualizar la información detallada de un material (desde el trabajar con materiales del SAP ERP)

Para que cuando el usuario haga tap sobre un material:

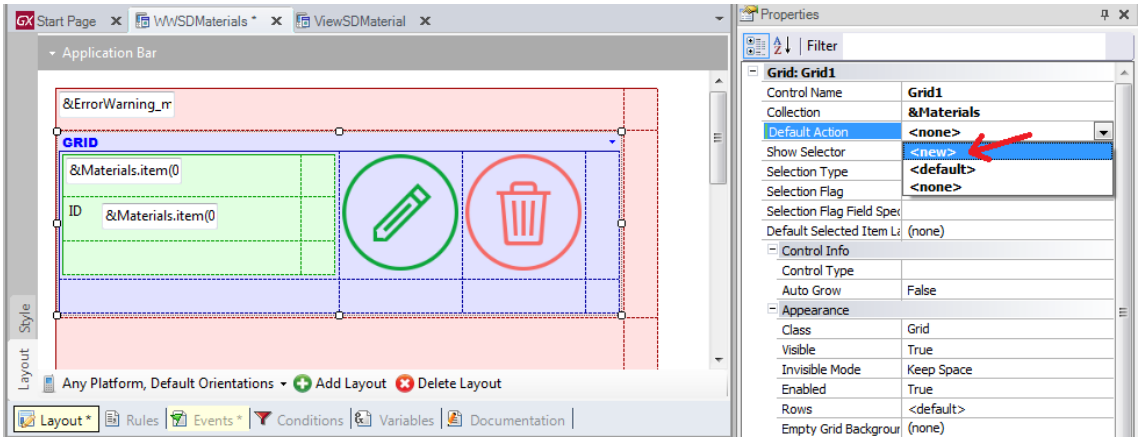




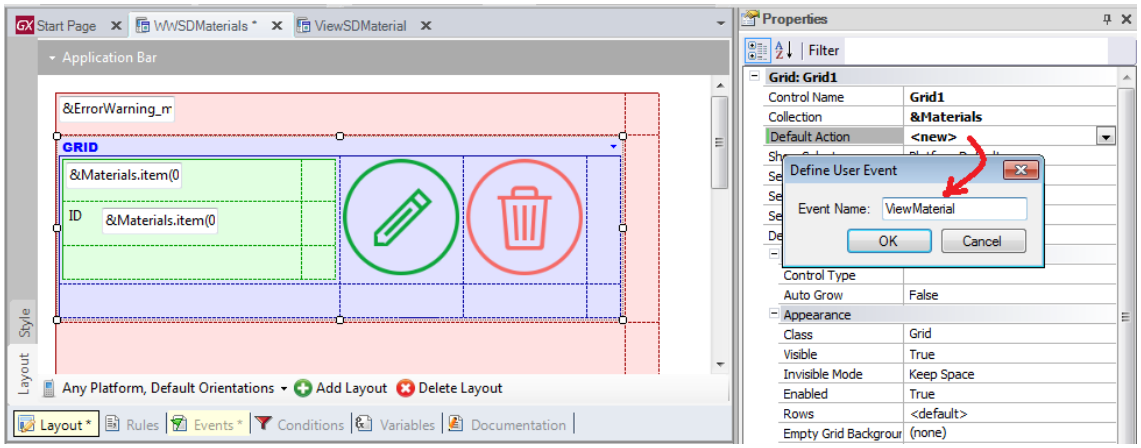
se invoque a un panel que muestre la información completa de ese material, tendremos que, por un lado, crear el panel (aquí lo mostramos ya creado, aunque vacío):



y por otro, tendremos que indicar a nivel del grid del Work With que la **Default Action** cuando se elige una línea (es decir, se hace tap sobre ella), es invocar a este panel. Para eso creamos un nuevo evento ya asociado a la Default Action:



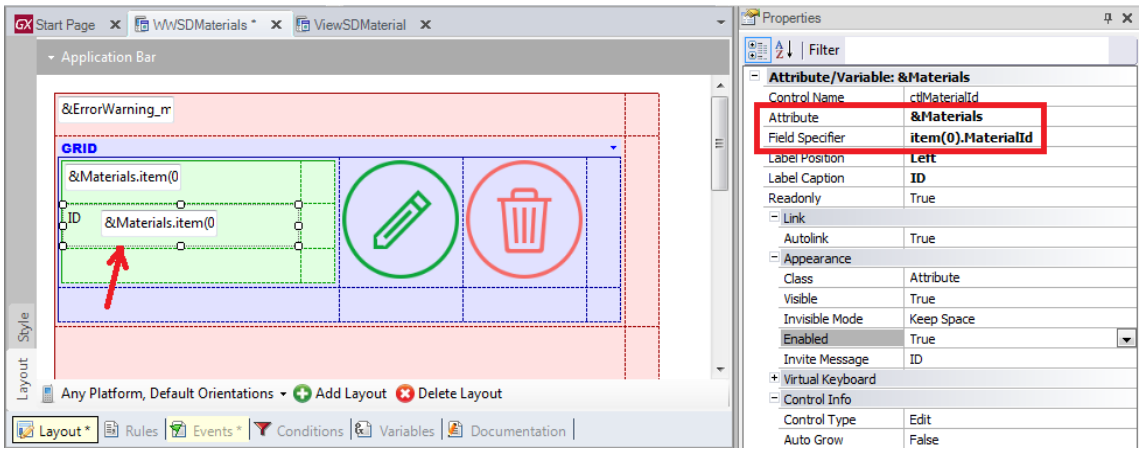
Al que llamaremos **ViewMaterial**:



Vamos a la solapa de eventos… Aquí programaremos la invocación al nuevo panel:

```
Event 'ViewMaterial'
    ViewSDMaterial()
endevent
```

que deberá recibir por parámetro el material, ya que su objetivo es mostrar su información. ¿Qué material? El contenido de la variable colección, correspondiente al item que se ha elegido en ese momento, elemento MaterialId:

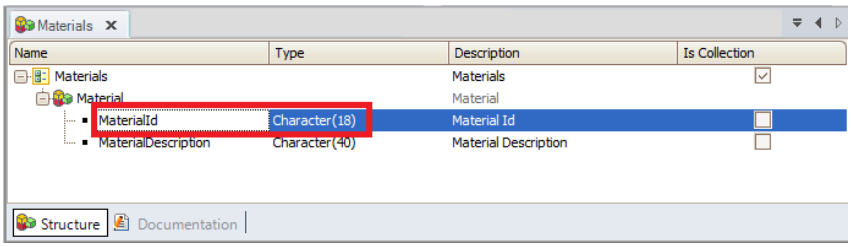



```
Event 'ViewMaterial'
    ViewSDMaterial( &Materials.CurrentItem.MaterialId)
endevent
```

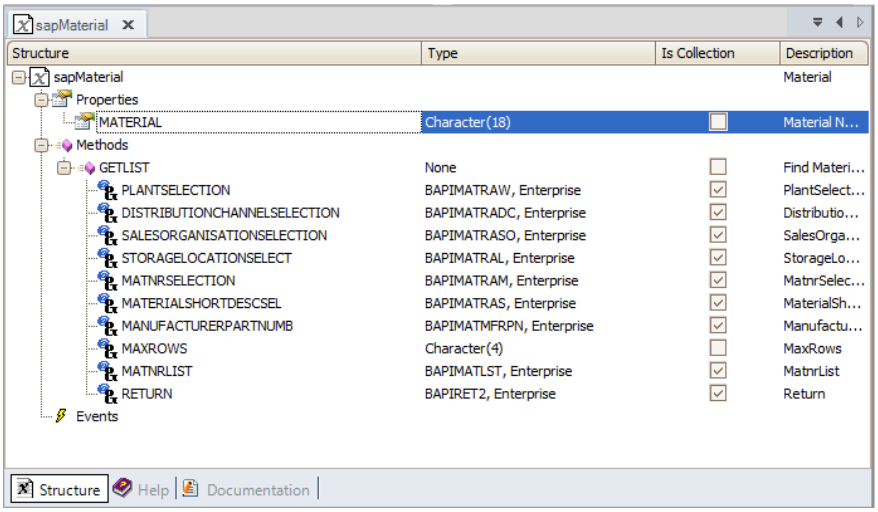
En el panel ViewSDMaterial, tenemos que declarar el parámetro de entrada:

```
parm( in: &MaterialId );
```

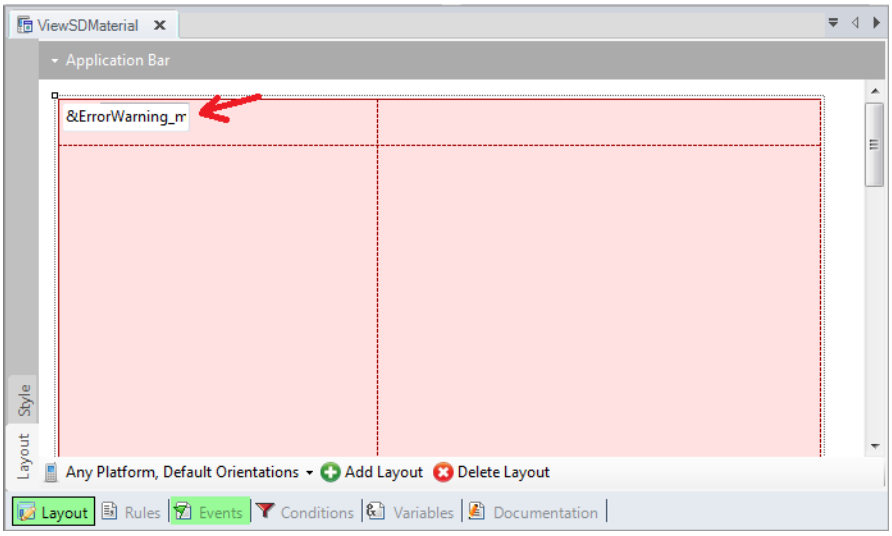
siendo MaterialId un Character de 18:



(correspondía a la clave del Business object Material, del SAP):



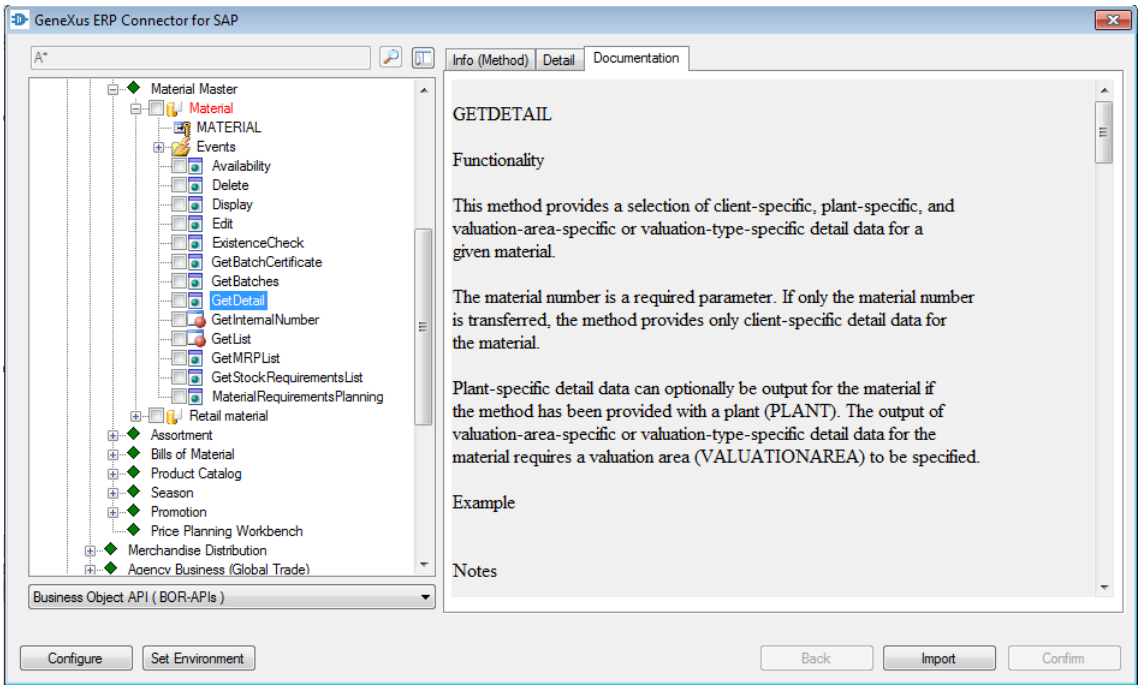
Aquí hemos colocado la misma variable para mostrar los posibles mensajes de warning o error que manejábamos en el panel que listaba los materiales.



En este caso, deberemos llamar a otro método del mismo Business Object Material del ERP. Tenemos que importarlo: Tools/Application Integration/SAP bapi Import….

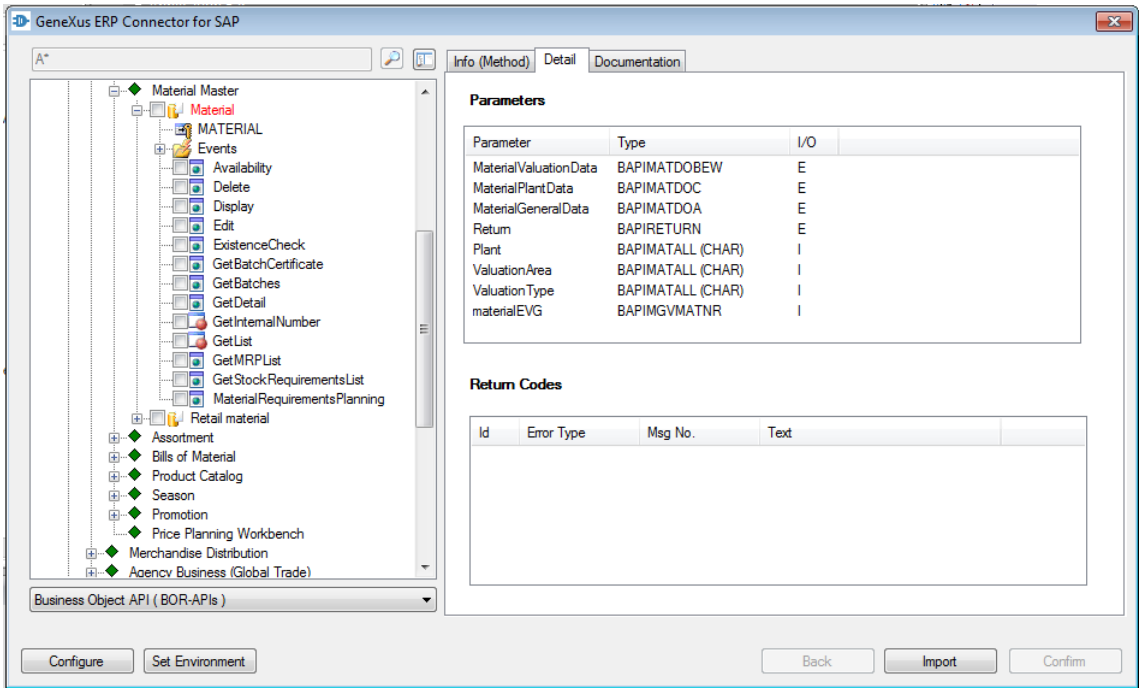
Nos conectamos…

Volvemos a buscar el Business Object Material y esta vez elegimos el método GetDetail.

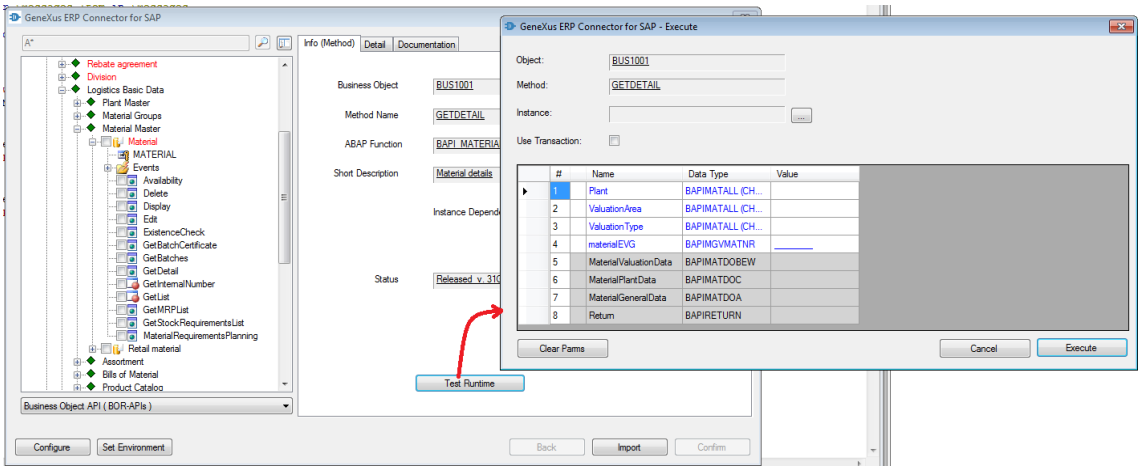


A diferencia del GetList, este es un método de instancia y no de clase. Lo que hará que para utilizarlo en GeneXus, debamos necesariamente invocarlo a través de una variable de tipo de datos el objeto sapMaterial y no directamente. Ya lo veremos.

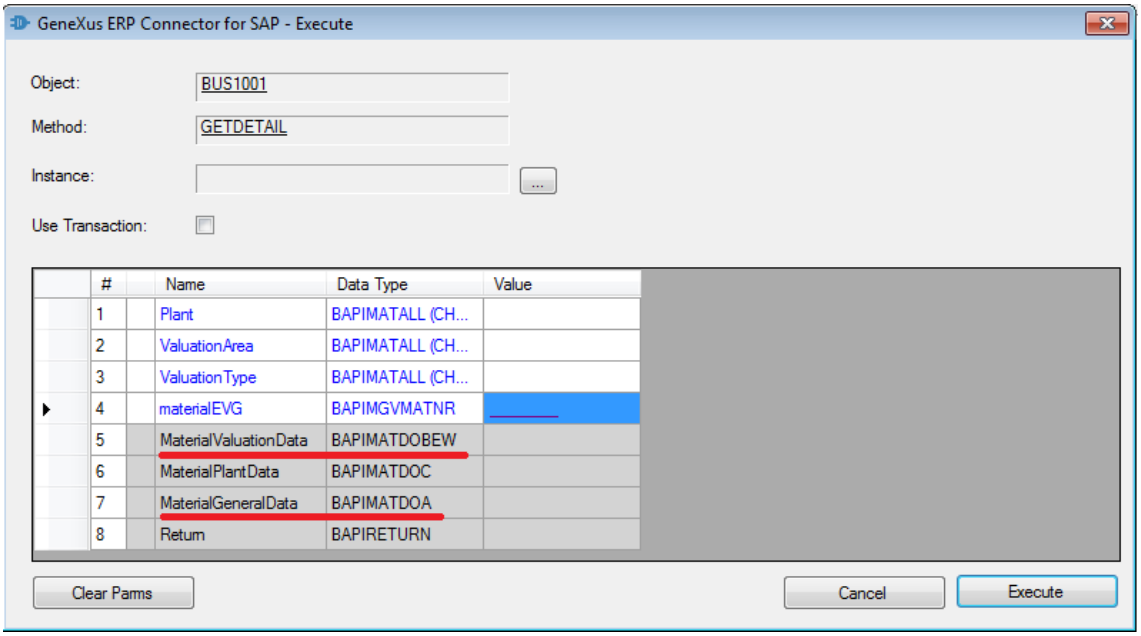
El método tiene estos 8 parámetros, de los cuales los primeros cuatro son de salida:



Podríamos probar el método, si lo deseáramos, como hicimos antes con el GetList.

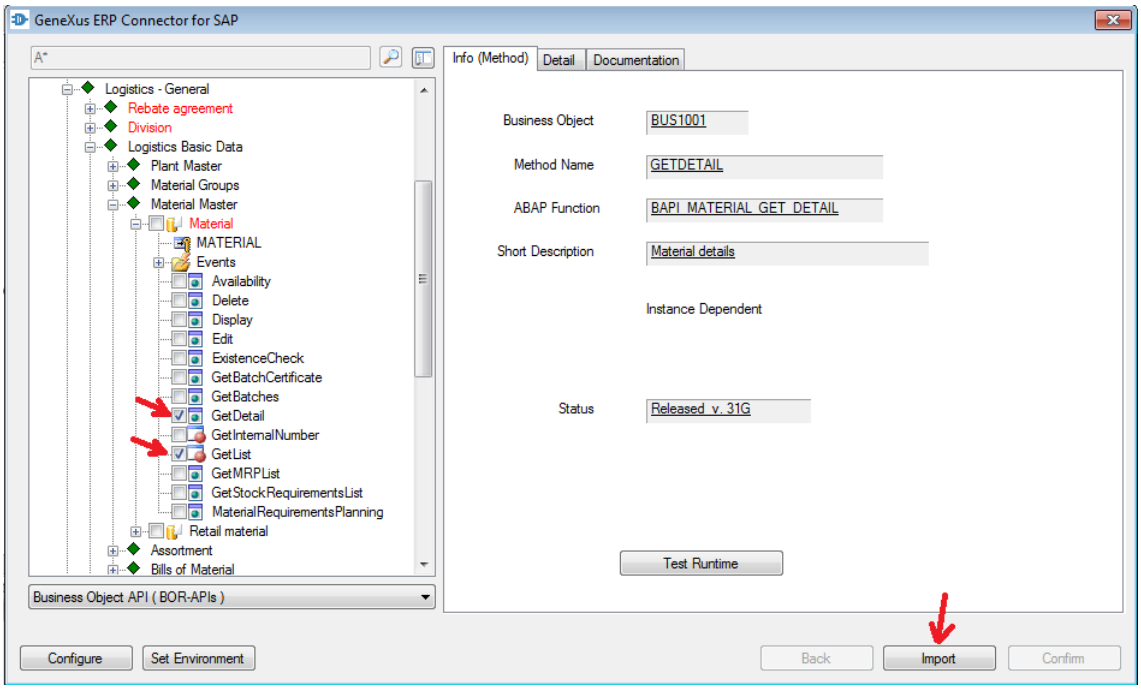


En dos parámetros nos devolverá la información del material: uno, el 7, corresponderá a la información general del material y otro, el 5, a datos **de costo** del material.

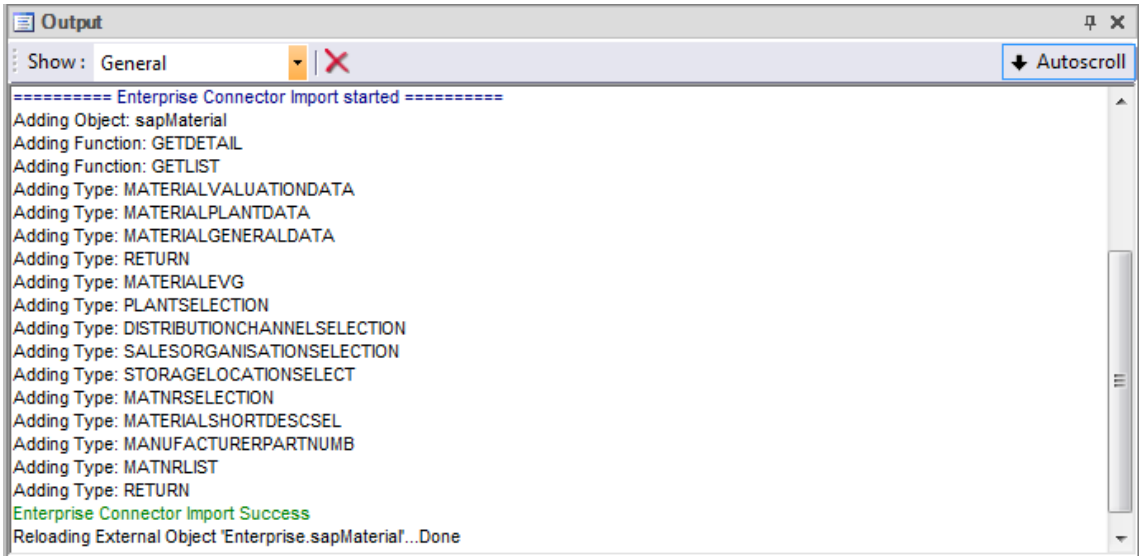


Importemos el método sin testearlo.

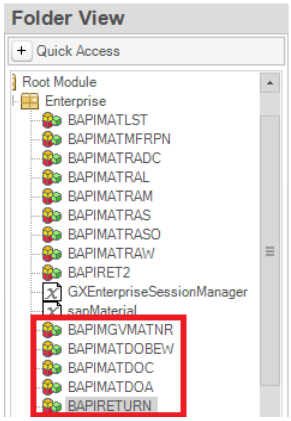
Como ya hicimos antes, lo seleccionamos (y dejamos seleccionado el GetList también) e importamos:



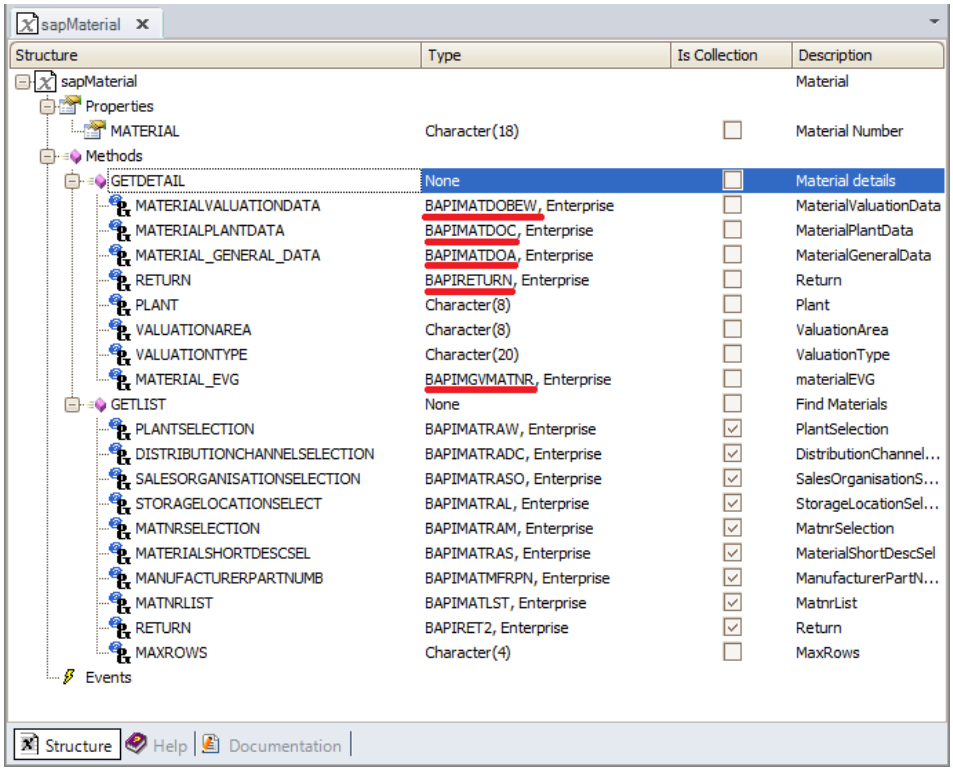
Vemos en la ventana de Output que se vuelve a importar el objeto sapMaterial, la función GETLIST y la nueva, GETDETAIL, junto con los tipos de datos estructurados necesarios para los parámetros:



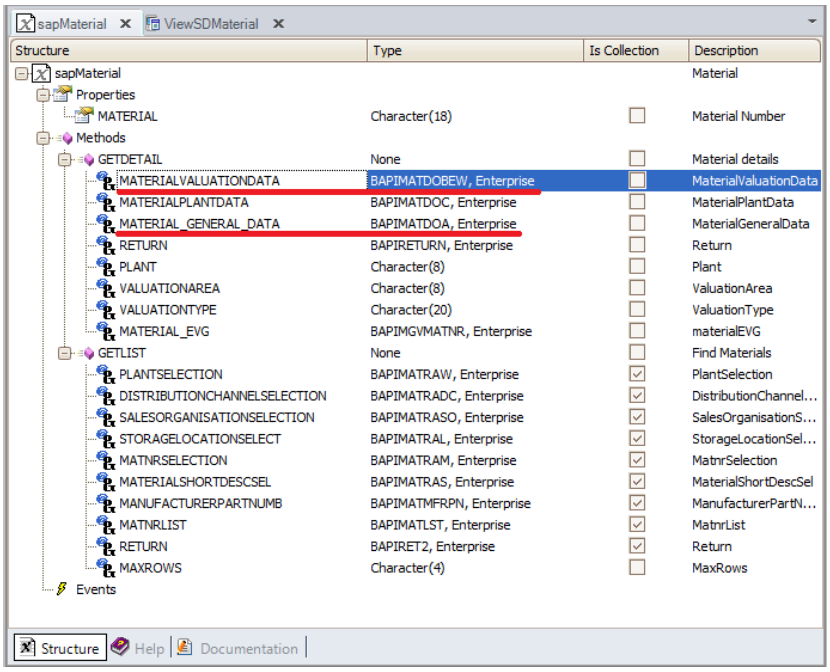
que vemos en el Folder View. Estos son los nuevos:



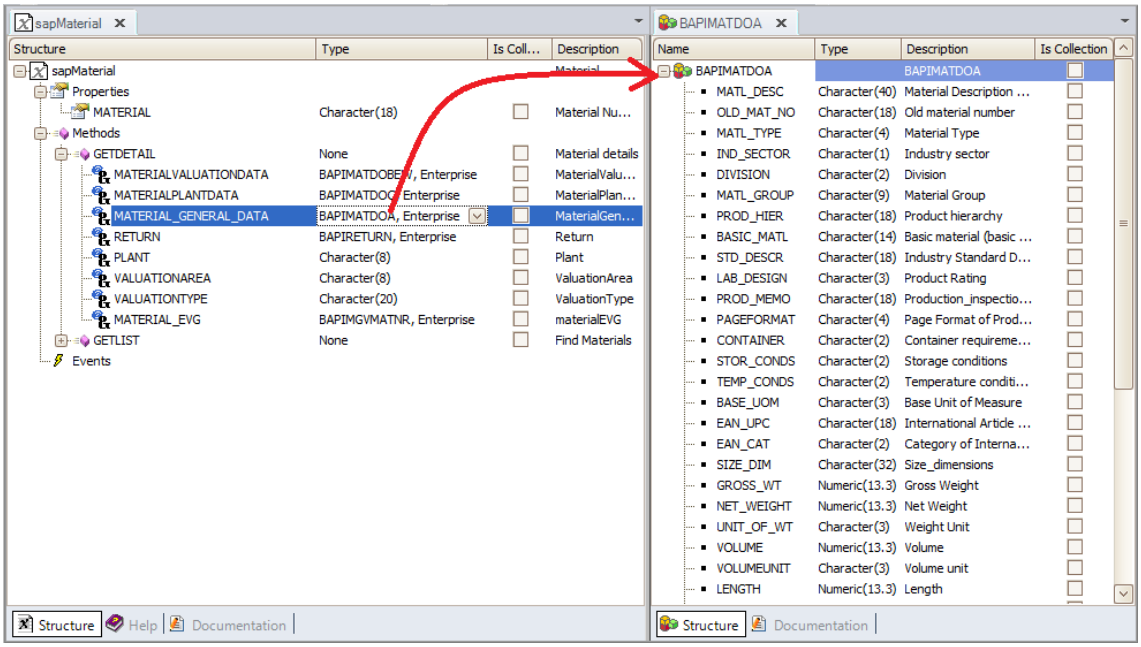
Si abrimos el objeto sapMaterial, los vemos como tipos de algunos de los parámetros del nuevo método:



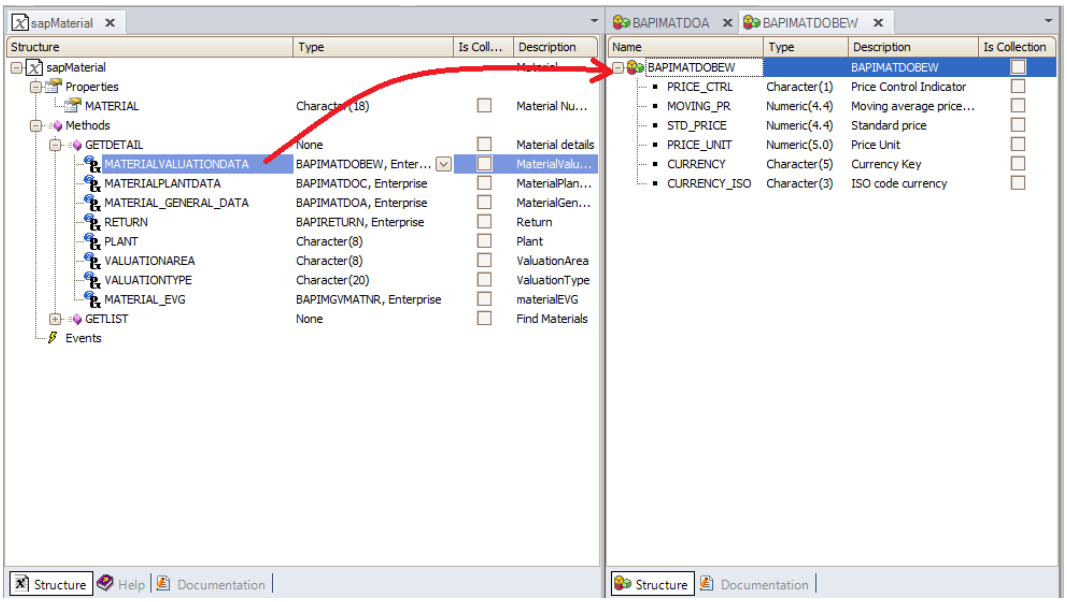
Estos son los dos parámetros de salida que contendrán la información relevante del material:



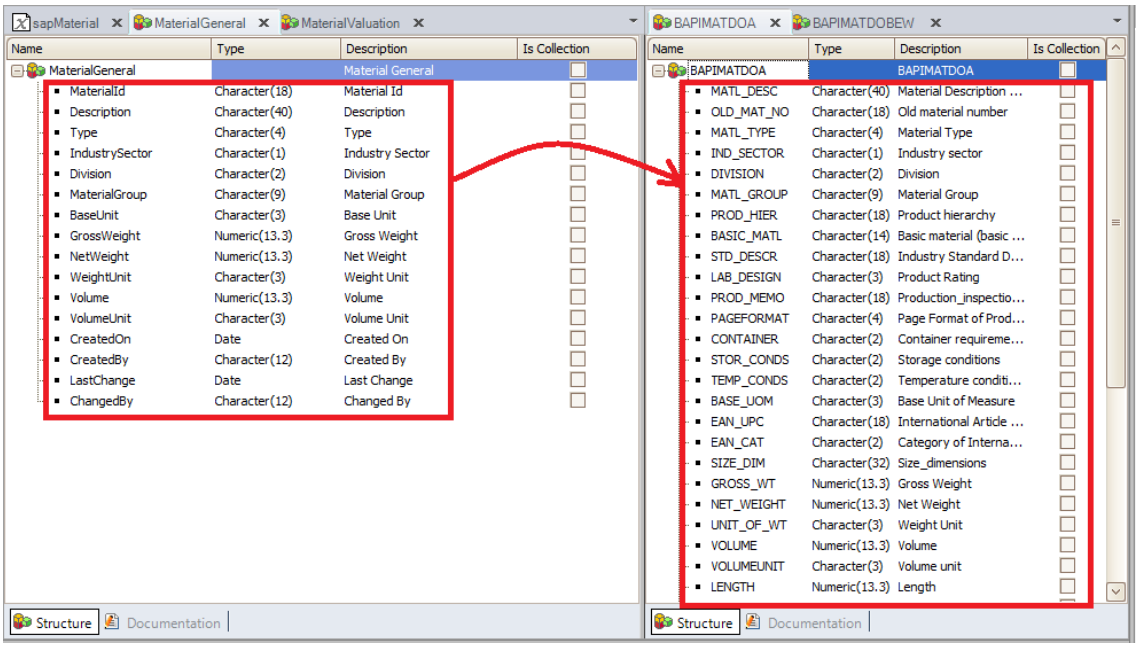
Los datos generales del material, como descripción, tipo, sector de la industria, división, etcétera:



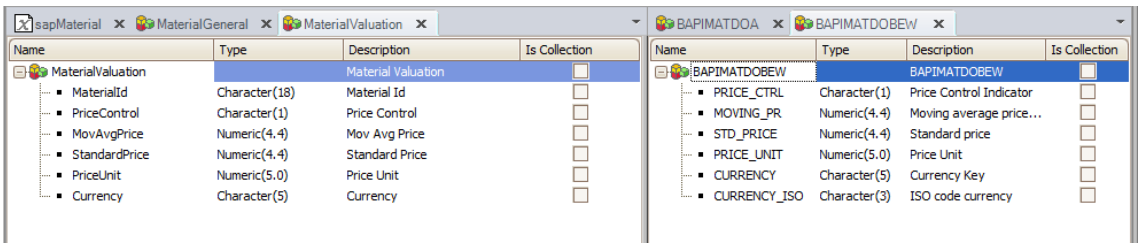
Y datos **del costo** del material, como el precio:



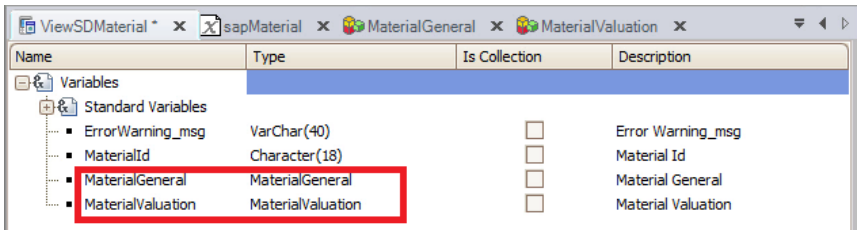
No querremos mostrar en nuestro panel toda esta información, sino sólo una parte, por lo que hemos creado dos tipos de datos estructurados: MaterialGeneral que representa un subconjunto de la información del primer SDT:



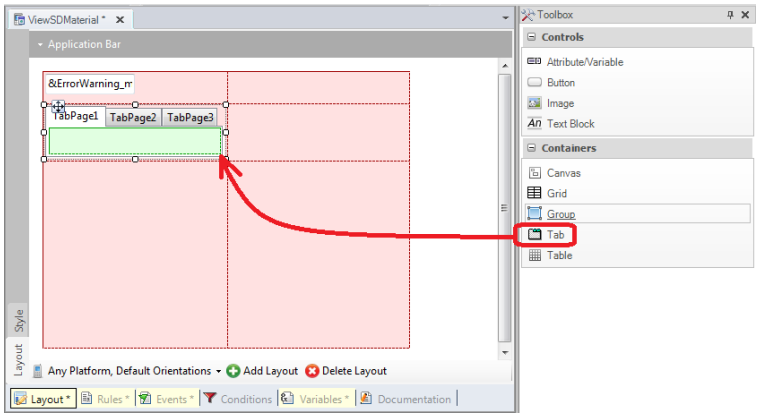
Y MaterialValuation, que muestra la información del segundo. Le hemos agregado a ambos MaterialId por si queremos indicar la información de qué material se trata.



Lo que haremos, entonces, será definir dos variables con esos tipos de datos en el panel ViewSDMaterial (que mostrará la información del material recibido por parámetro):

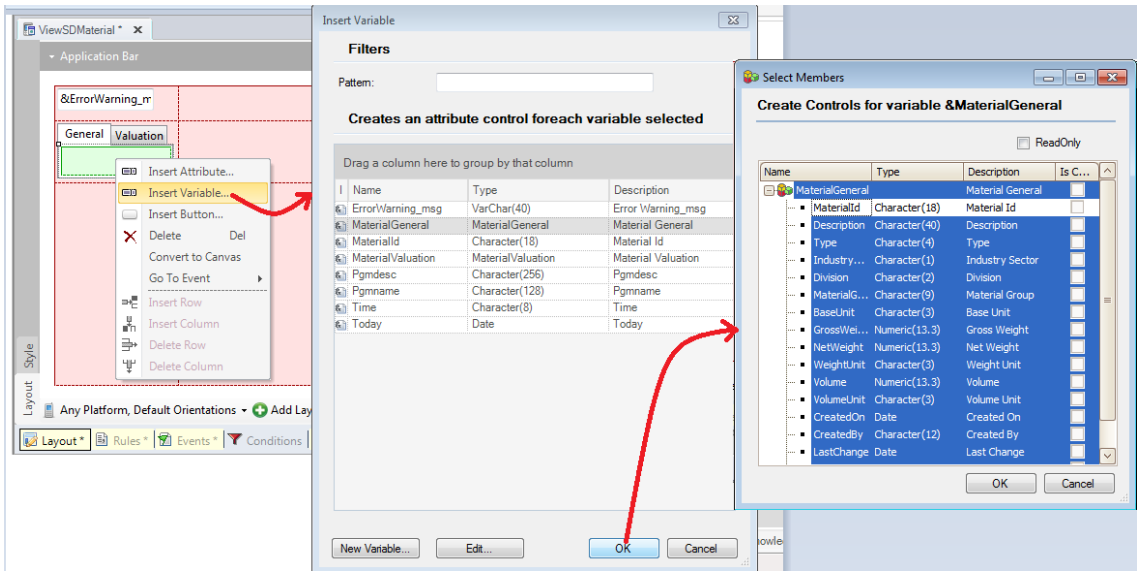


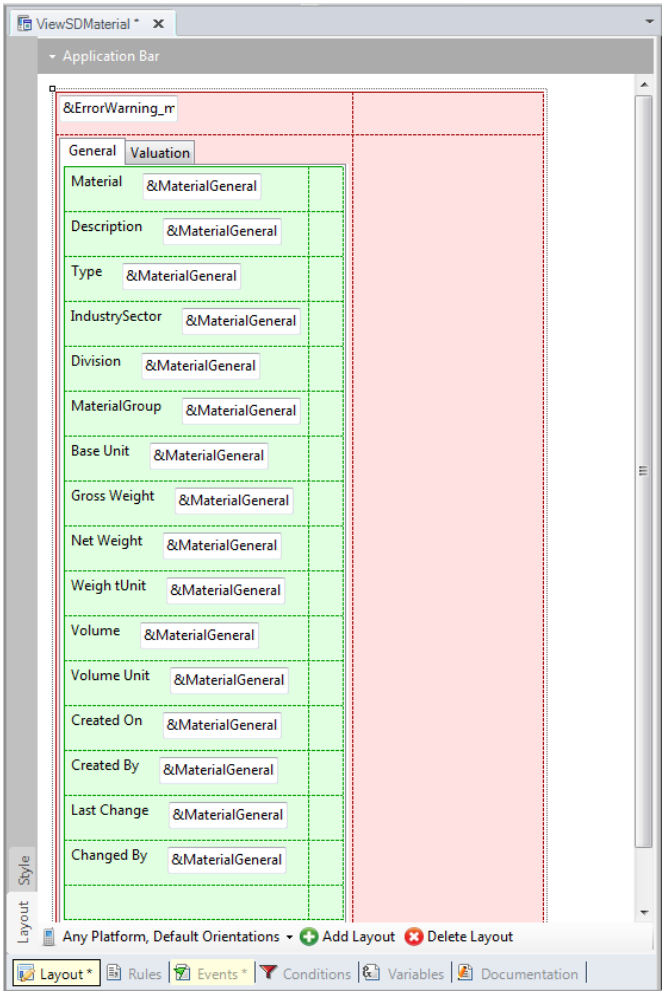
Insertamos en el layout un control tab:



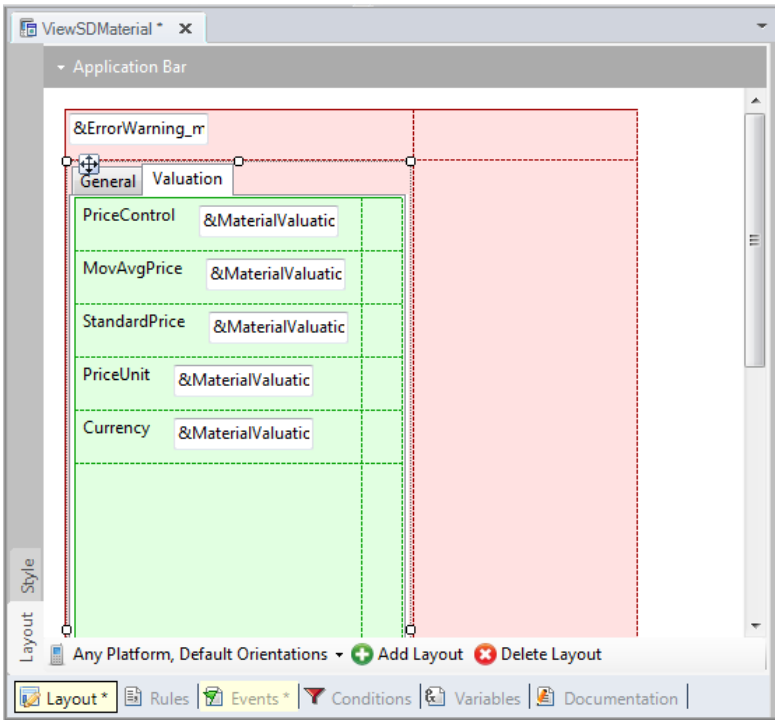
para poder tener dos solapas: una para la General information y otra para la Valuation information. La tercera pestaña la eliminamos.

En la primera insertamos la variable estructurada MaterialGeneral (con todos sus elementos):





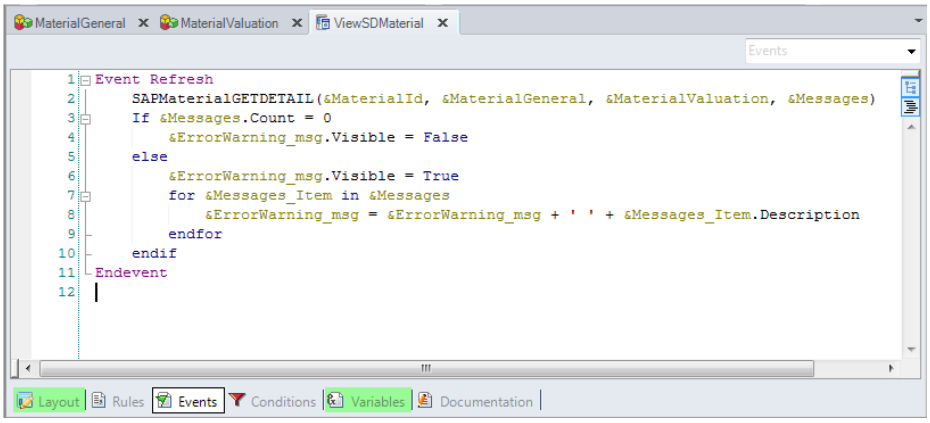
y en la segunda la variable estructurada MaterialValuation (sin el Material Id).



Cambiamos todas las variables a readonly.

Como hicimos en el caso del Work With, en el evento Refresh deberemos llamar a un procedimiento (que tendremos que crear, al que llamaremos `SAPMaterialGETDETAIL`), que será quien invoque a la bapi correspondiente vía RFC, y devuelva cargadas estas dos variables.

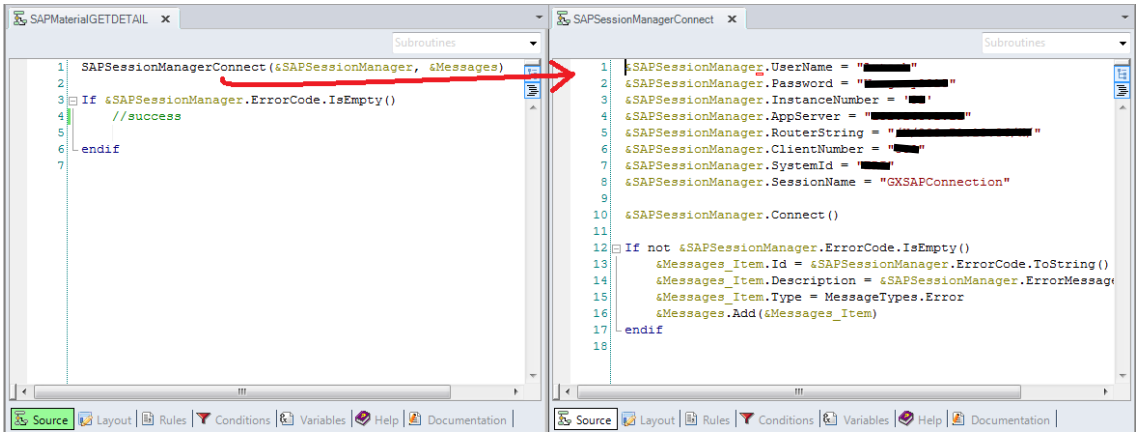
A ese procedimiento tendremos que pasarle por parámetro de entrada, el identificador de material, y las variables que acabamos de insertar en el layout, que devolverá cargadas: &MaterialGeneral y &MaterialValuation. Además, devolverá la colección de mensajes de advertencia y error, que serán manejados como lo hicimos antes:



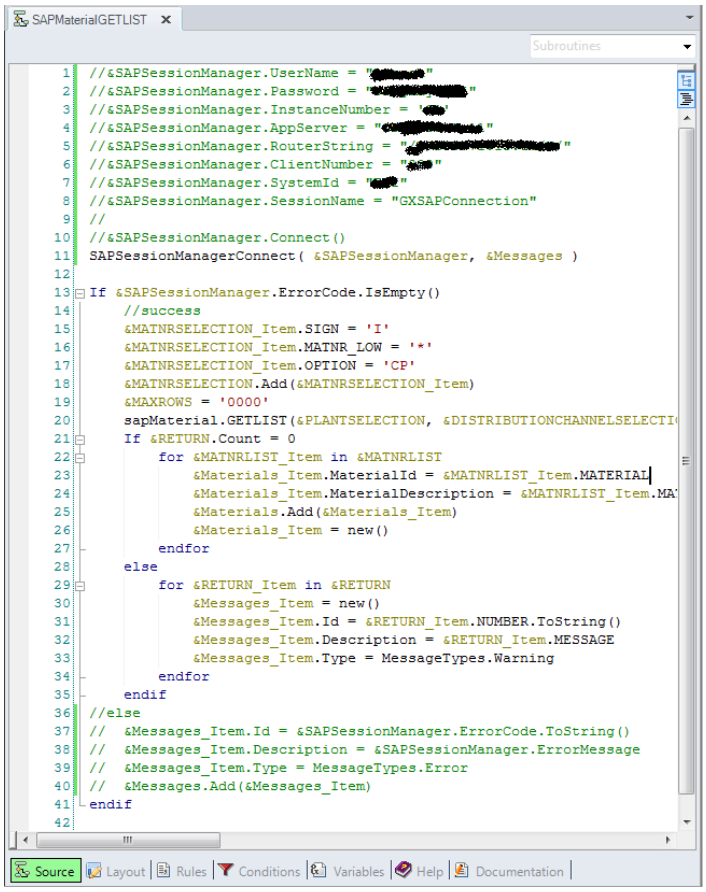
Aquí mostramos el procedimiento ya creado. Vemos la declaración de parámetros:

```
parm( in: &MaterialId, out: &MaterialGeneral, out: &MaterialValuation, out: &Messages);
```

y en el Source vemos que para no tener que repetir cada vez el mismo código de conexión al SAP ERP, lo hemos encapsulado en otro procedimiento con dos parámetros de salida: una variable &SAPSessionManager del tipo del objeto GXEnterpriseSessionManager, y una variable &Messages que tendrá cargado el error de conexión, en caso de ocurrir:



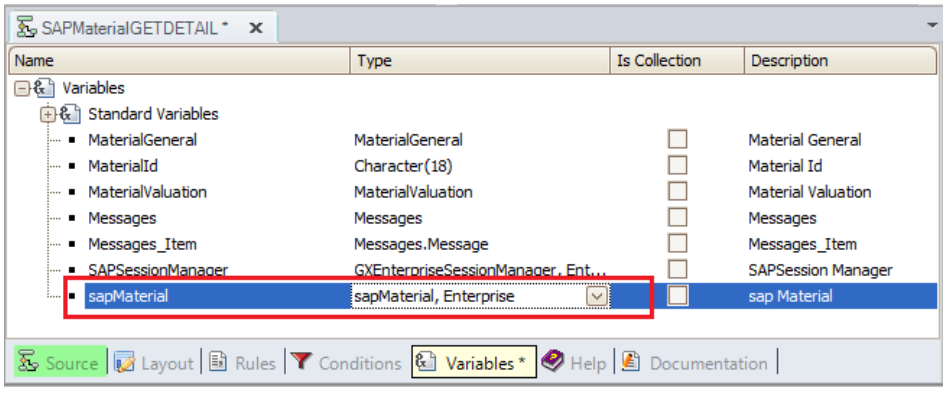
Debido a esto hemos cambiado también el código del procedimiento que ya teníamos, para ahora llamar al procedimiento de conexión:



Luego de conectarnos al ERP, entonces, tenemos que invocar al método GETDETAIL. Como ese método es de instancia y no de clase, como mencionamos, no podremos escribir directamente:

sapMaterial.GETDETAIL

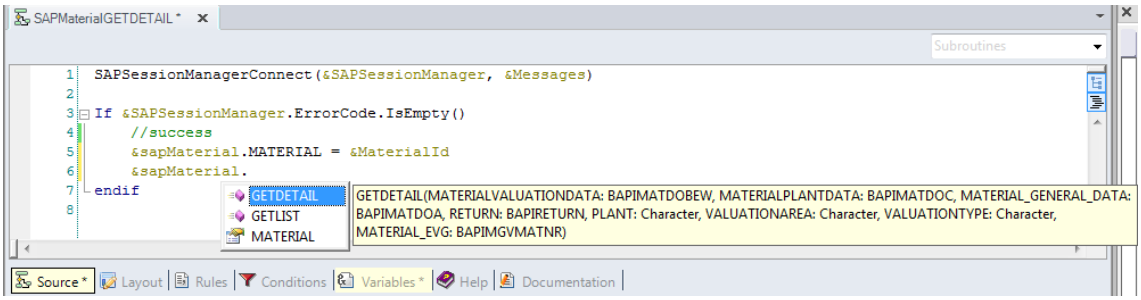
sino que deberemos crearnos una variable del tipo de datos sapMaterial:



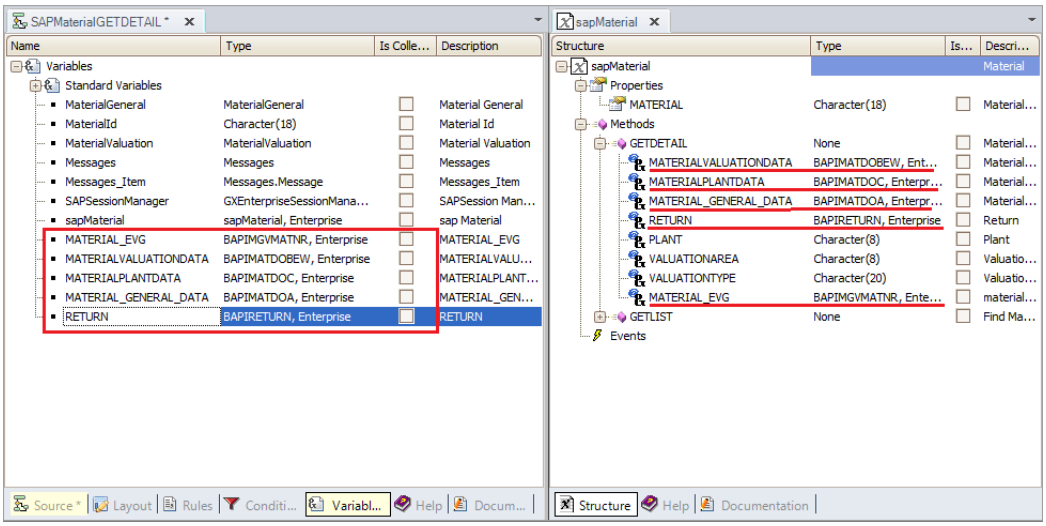
E indicar cuál será el identificador de material que se cargará en su instancia (será el recibido por parámetro, MaterialId):

```
If &SAPSessionManager.ErrorCode.IsEmpty()
        //success
        &sapMaterial.MATERIAL = &MaterialId
endif
```

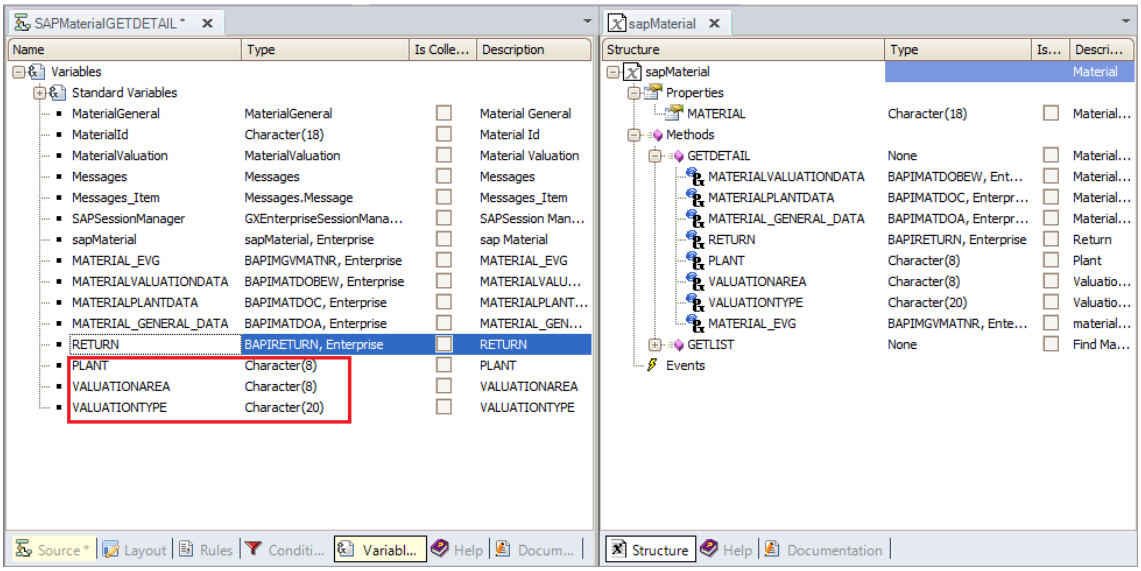
y luego sí, invocar al método GETDETAIL:



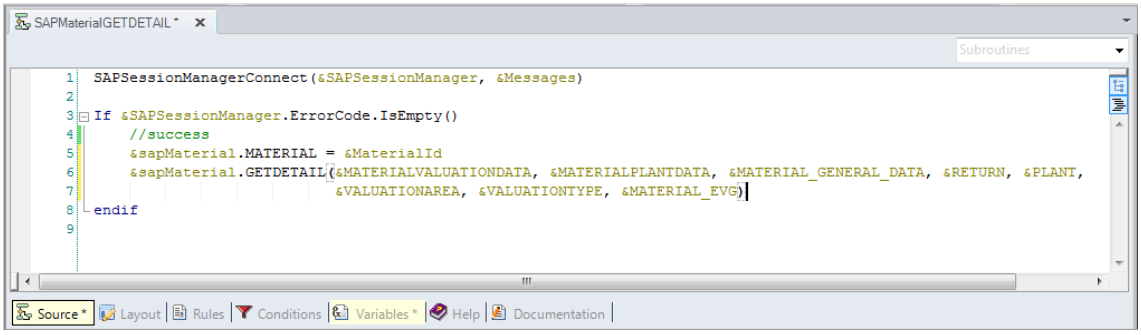
Otra vez, crearemos variables para todos los parámetros, arrastrando los SDTs desde el folder view, y colocándoles el mismo nombre por claridad:



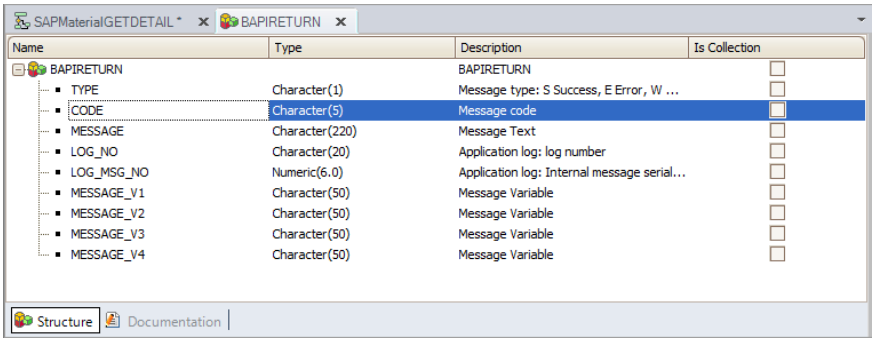
Y definiendo variables para los parámetros de tipos simples:



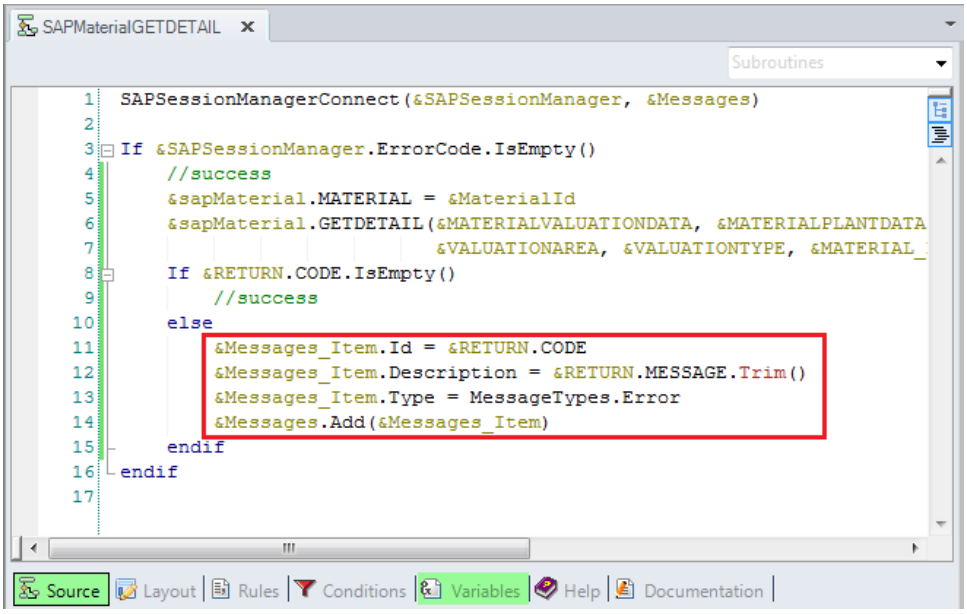
Ahora sí, invocamos al método, pasándole todos sus parámetros (vacíos):



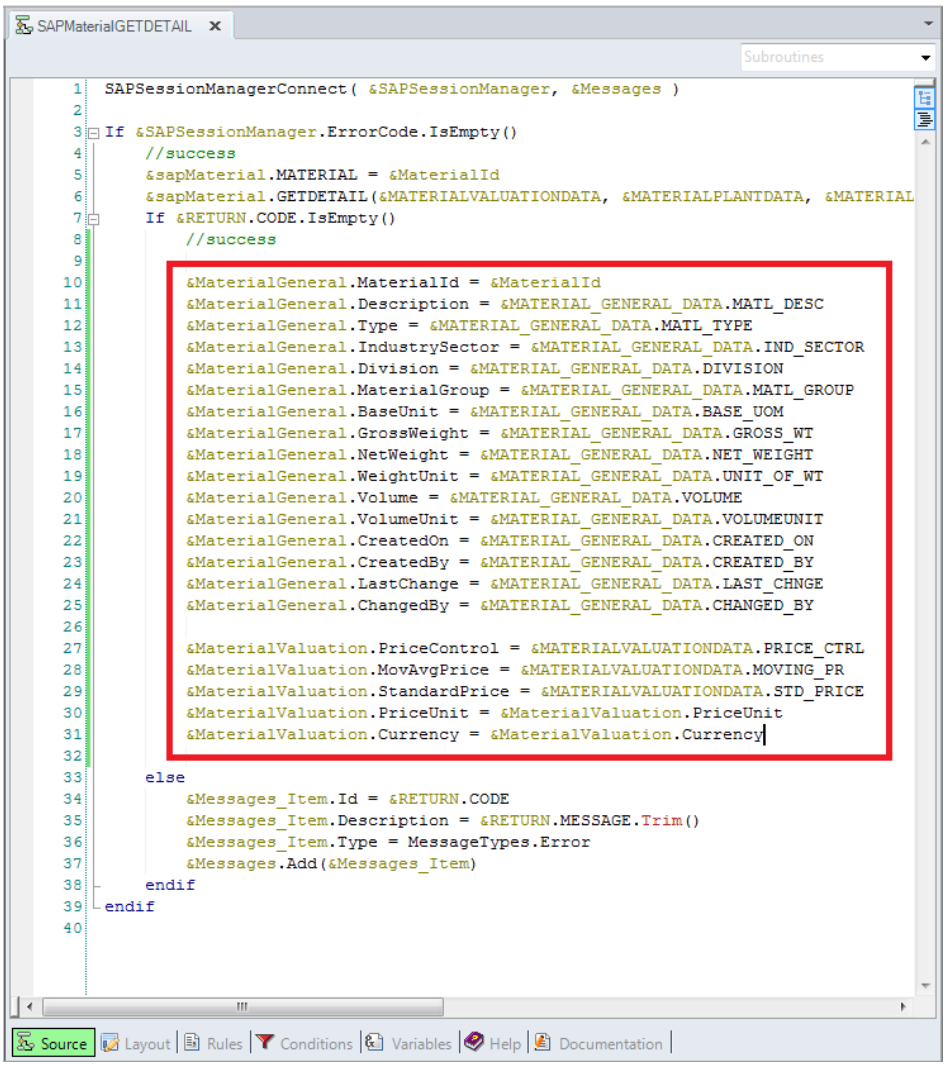
La ejecución fue exitosa si el CODE del Return está vacío:



Si no lo está: agregamos un item a la colección de mensajes, en forma similar a como lo habíamos hecho antes con la lista de materiales.

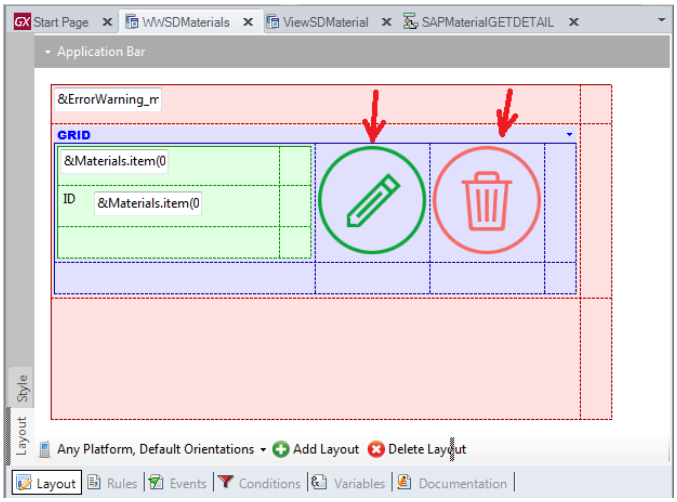


En caso de éxito, sólo nos está restando cargar las variables de salida, de acuerdo a los dos parámetros de salida del método:

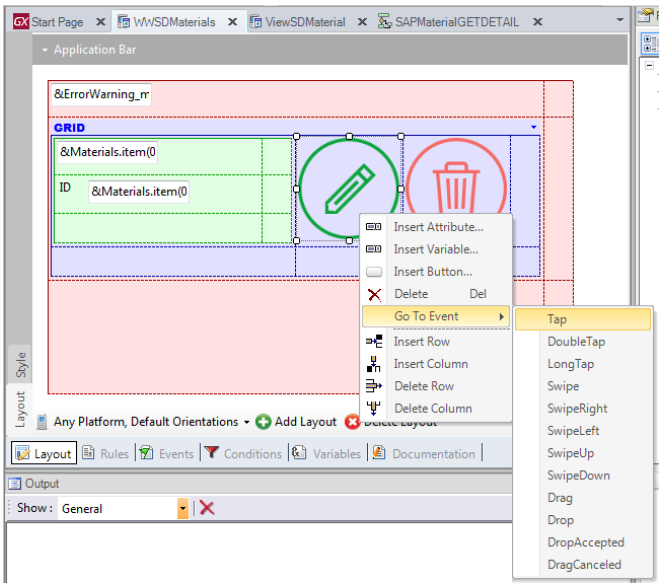


Ya estamos listos para probar. Botón derecho, Run, sobre el objeto main.

Para completar la implementación del Work With de Materiales, nos estaría haciendo falta programar la posibilidad de modificar un material o de eliminarlo…



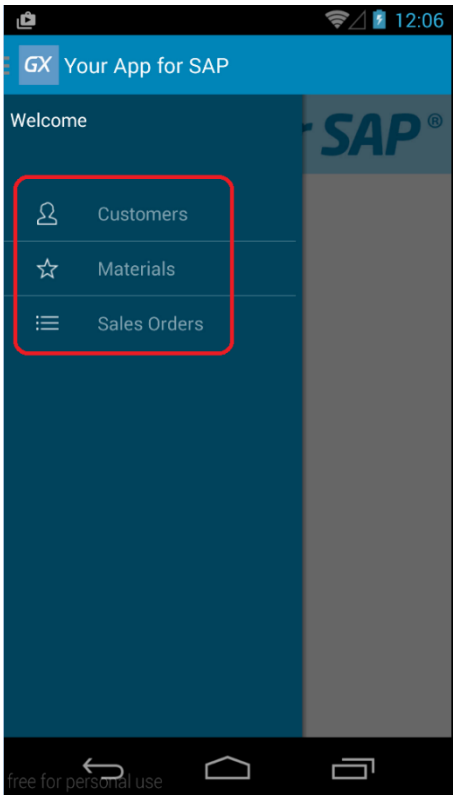
Para ello alcanzará con definir los eventos tap de cada control:



Y allí invocar al método apropiado del external object sapMaterial (previamente importándolo del ERP, como hemos hecho hasta ahora con los métodos GetList y GetDetail).

Dejamos esta implementación pendiente, para que usted investigue y vea cómo hacerlo.

En el próximo video, una vez agregado el Work With de clientes, que será en todo análogo al de materiales, veremos cómo implementar un menú de entrada, para que el usuario elija qué Work With desea utilizar en cada momento. Ese menú luego incluirá otra opción, que será trabajar con las Sales Orders.


12:06
GX Your App for SAP
Welcome
SAP®
Customers
Materials
Sales Orders
free for personal use


GeneXus™
TRAINING PROGRAM