Laboratorio: primeros pasos en el desarrollo con GeneXus para Smart Devices

Septiembre 2019

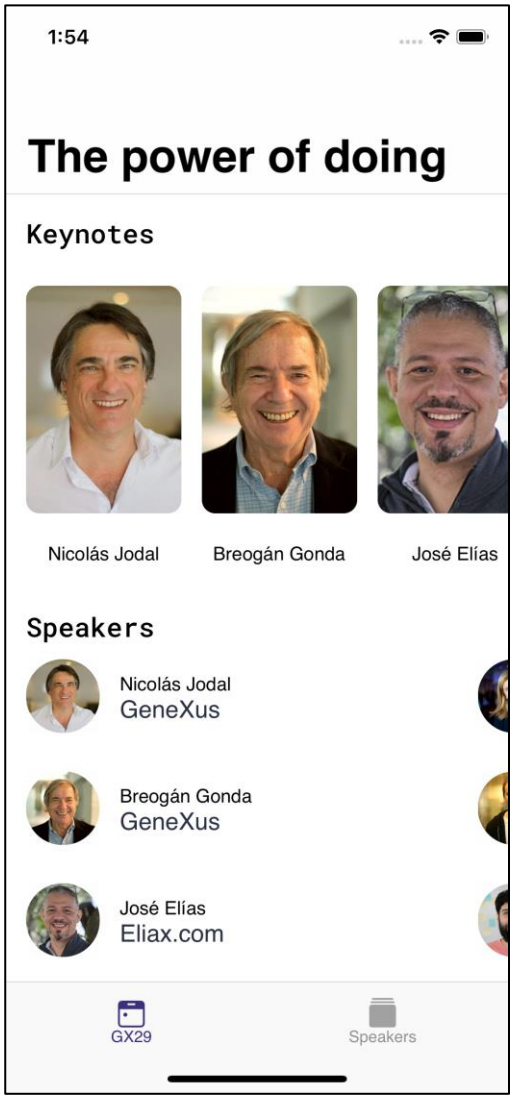
CONTENIDO

OBJETIVO

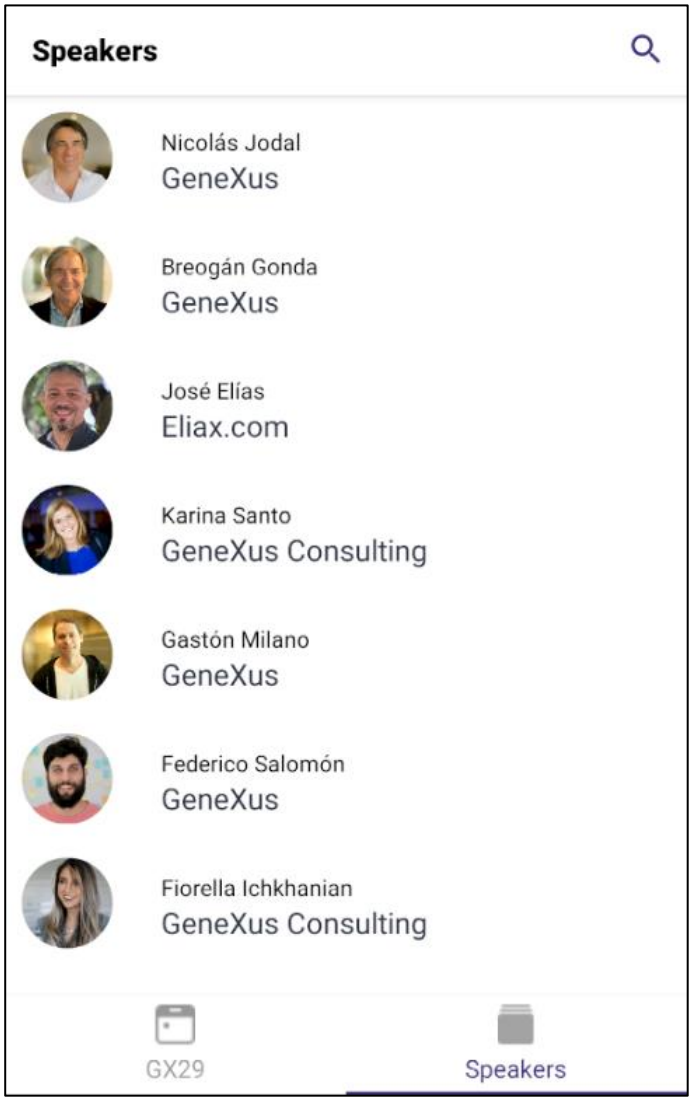
En este Laboratorio usted dispondrá de una guía paso a paso, para la construcción de una aplicación simple para Smart Devices utilizando GeneXus 16. El objetivo de este laboratorio sin embargo, no es la capacitación, sino la familiarización con la simplicidad del desarrollo de aplicaciones para dispositivos inteligentes con GeneXus.

Se trabajará sobre un fragmento de la aplicación del evento GX29. Se partirá de una KB ya inicializada con sus entidades, y se pretende terminar el laboratorio con una aplicación Android o iOS, aplicando diseño y algunas funcionalidades.

Las siguientes, son dos pantallas de ejemplo que crearemos en este laboratorio, entre otras.



iOS

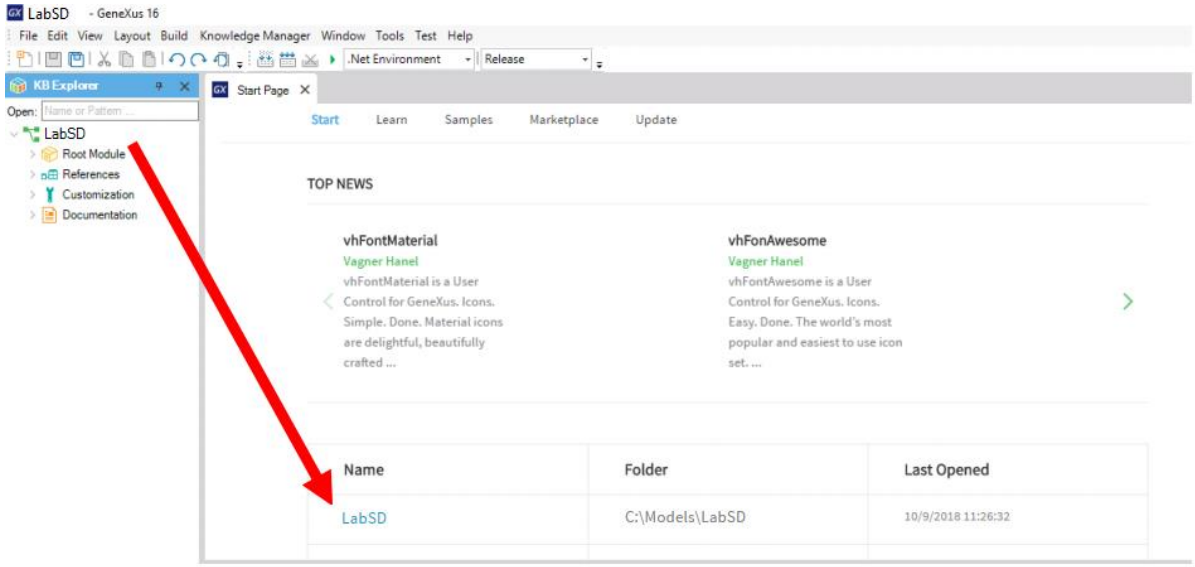


Android

COMENCEMOS

Para este laboratorio utilizaremos la versión GeneXus 16.

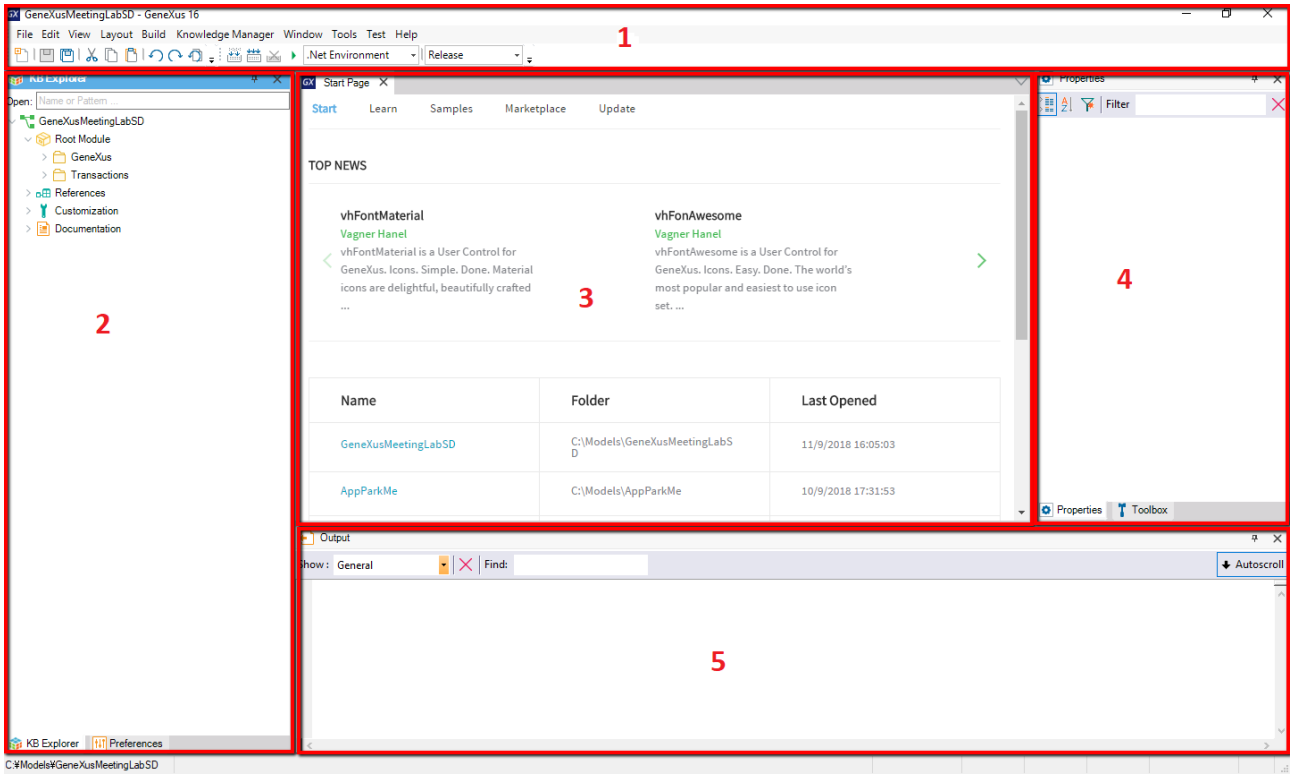
Una vez abierto GeneXus, abrir la base de conocimiento (Knowledge Base) LabSD, desde las Recent Knowledge Bases de la Start Page, como se muestra en la imagen:



Ésta KB, está acotada para los fines específicos del laboratorio. Contiene una aplicación Web y sus entidades, las cuales tomaremos como punto de partida para crear nuestra versión Smart Devices de la misma.

INTRODUCCIÓN – AMBIENTE GENEXUS

La siguiente será la pantalla de bienvenida con la cual nos encontraremos al abrir GeneXus. Repasaremos un poco sus partes antes de seguir adelante (de estar familiarizado con el IDE de GeneXus puede saltarse al primer paso):



1. Toolbar: Proporciona acceso a todas las opciones de la Knowledge Base GeneXus.

2. Knowledge Base Navigator: Es un conjunto de menús contextuales para navegar sobre objetos, ya sea en su vista por directorios (Folder View), categorías (Category View), o también para visualizar la lista de últimos cambios (Latest Changes View) y Propiedades del modelo (Preferences).

3. Principal: Aquí se despliega la Start Page y una pestaña por cada uno de los objetos abiertos.

4. Propiedades y toolbox: de controles, objetos y variables que estén seleccionadas.

5. Salida (Output): de las distintas operaciones (especificación, generación, compilación, etc.).

PASO 1 – EJECUCIÓN Y CARGA DE DATOS INICIALES

Como podemos ver en el Folder View del Knowledge Base Navigator, contamos con una aplicación precargada, la cual, como ya mencionamos, está únicamente implementada para web con sus entidades y será nuestro trabajo crear su versión Android / iOS.

En este laboratorio ejecutaremos nuestra aplicación de forma local, como se puede ver en las siguientes propiedades del generador .NET (en Preferences de la Knowledge Base Navigator), bajo el nodo Execution:

- Deploy to cloud = No
- Web Root = http://localhost/LabSD.NetEnvironment/

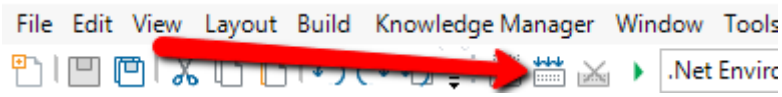
Vamos a modificar la propiedad Web Root, cambiando "localhost" por el nombre de la máquina virtual a la que estamos conectados, la cual podemos ver en la parte superior de la pantalla:

Ejemplo:

Pasemos ahora a configurar las propiedades del Data Store:

- Database name = LABSD**TuNombre**
- Server Name = localhost\SQLEXPRESS

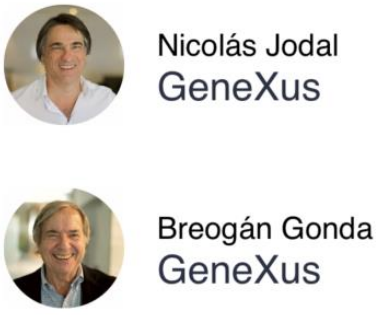
Luego de hecho esto, hacemos Rebuild All



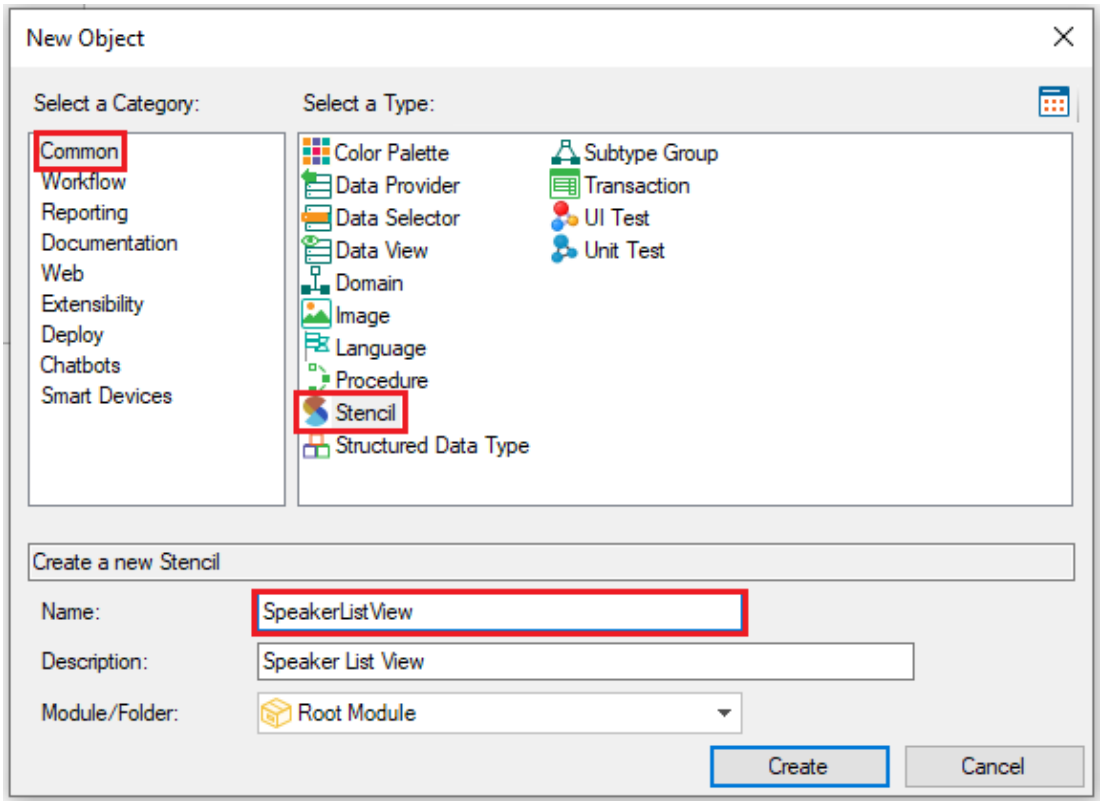
Al aparecer la ventana de Impact Analysis, hacemos clic en Create para que cree las tablas en nuestra base de datos. De esta forma, al dar Successful en el Output, ya tenemos las entidades creadas con sus datos precargados.

PASO 2 – TRABAJANDO CON STENCILS

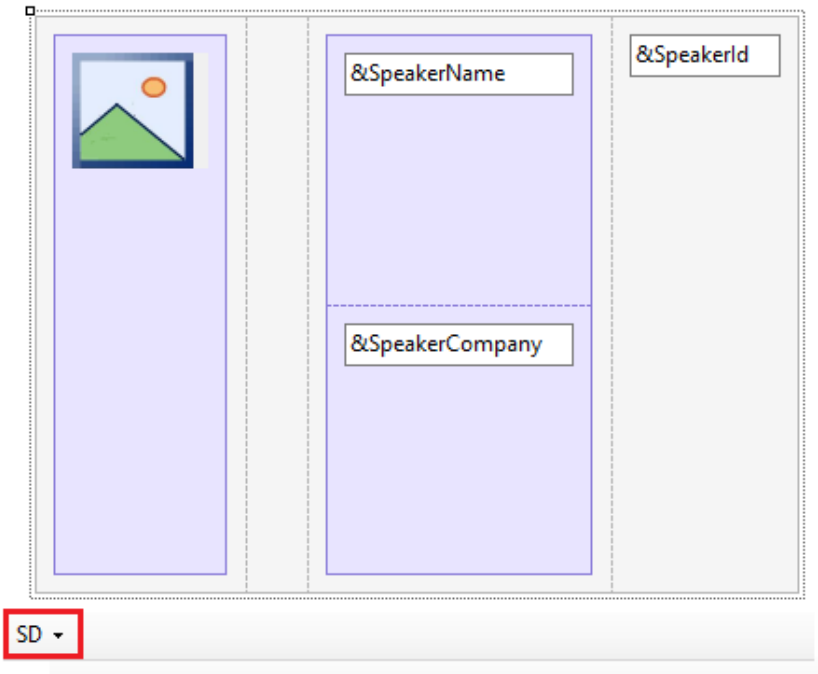
Usualmente en el desarrollo de aplicaciones, reutilizamos un mismo diseño en diferentes pantallas. Por ejemplo, en nuestra aplicación del evento GX29, es muy común ver en distintas partes, información de oradores desplegada de la siguiente manera:



En el caso del presente laboratorio, vamos a mostrar la información de oradores de esa forma en dos SDPanels distintos, por lo cual procederemos a crear un objeto Stencil llamado "SpeakerListView" que modele dicho diseño.



Luego de creado el objeto, procedemos a crear 4 variables: “SpeakerId”, “SpeakerName”, “SpeakerImage” y “SpeakerCompany” y arrastrarlas al layout, diagramándolas de la siguiente manera:



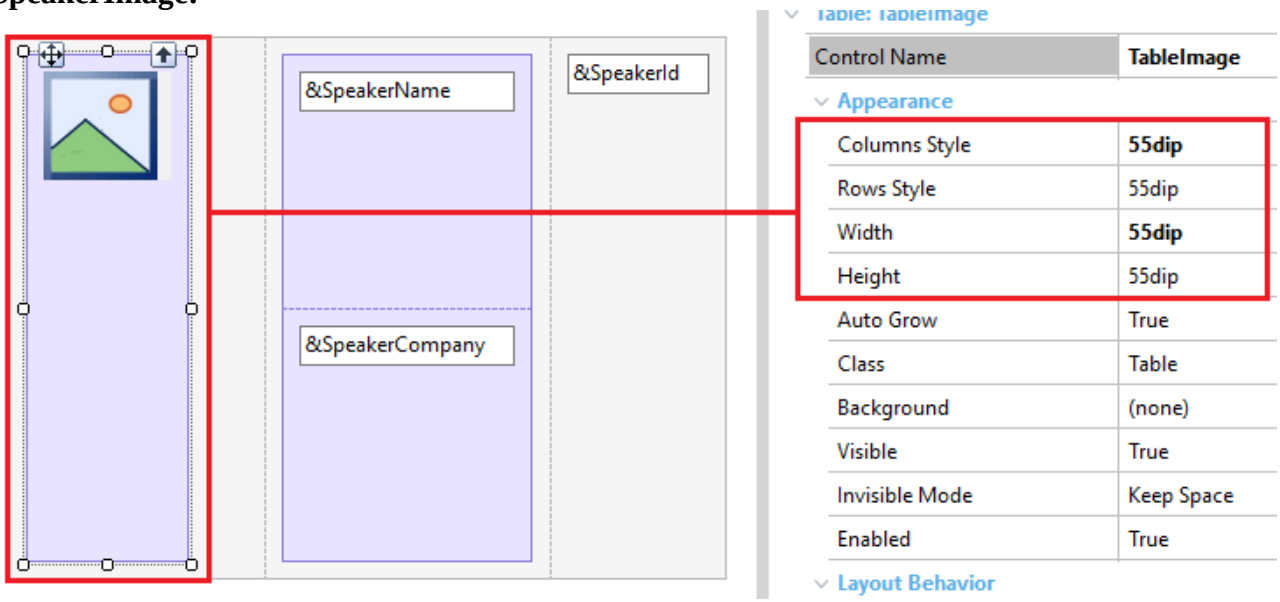
Notar que tanto SpeakerImage como SpeakerName y SpeakerCompany están contenidos en sus respectivas tablas (no olvidar agregarlas). También podremos ver que los atributos al agregarlos, se nos muestran con un texto del lado izquierdo (el “Label Caption”), esto podemos quitarlo cambiando la propiedad “Label Position = none” para cada atributo. Además, le pondremos a **“SpeakerId”** la propiedad **“Visible = false”**.

Ahora procederemos a asignar las propiedades de estos objetos.
Las propiedades de la “Main Table” deben ingresarse de la siguiente forma:

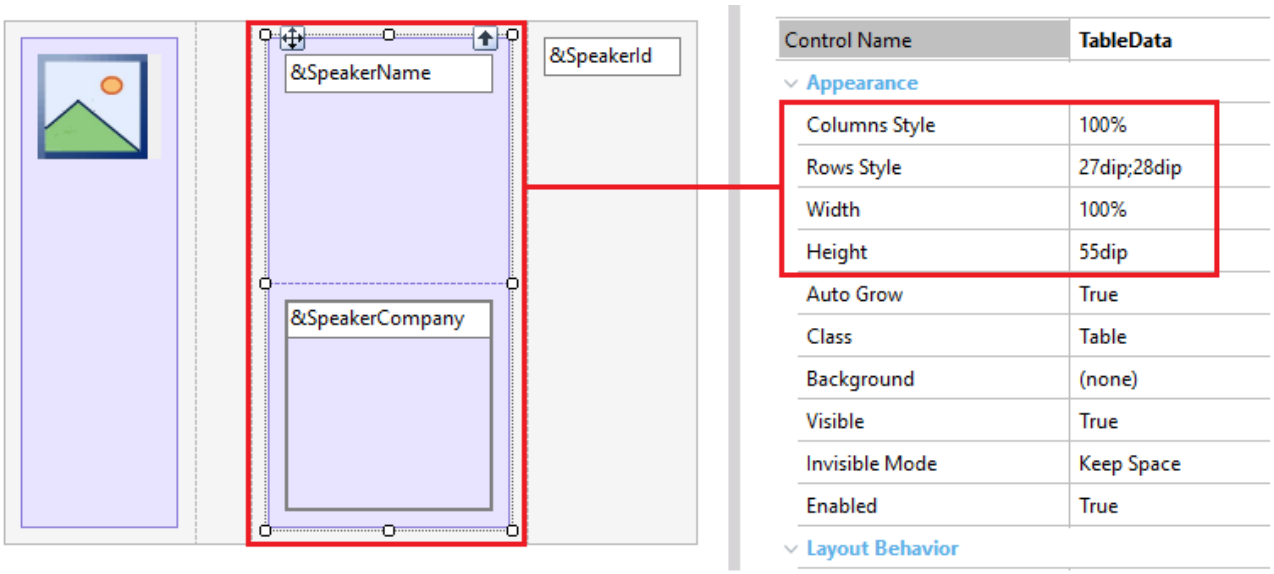
Table: MainTable	
Control Name	**MainTable**
Appearance	
Columns Style	**55dip;15dip;100%;0dip**
Rows Style	55dip
Width	100%
Height	100%
Auto Grow	True
Class	**TableSpeakerListItem**
Background	(none)
Visible	True
Invisible Mode	Keep Space
Enabled	True
Layout Behavior	
Expand Bounds	Background Only
Expand Bounds Limit	Behind System Bars
Expand Bounds Directions	Top, Left, Bottom, Right

A continuación, pasaremos a asignar las propiedades de la tabla contenedora de SpeakerImage y de la tabla contenedora de SpeakerName y SpeakerCompany.

SpeakerImage:



SpeakerName y **SpeakerCompany:**

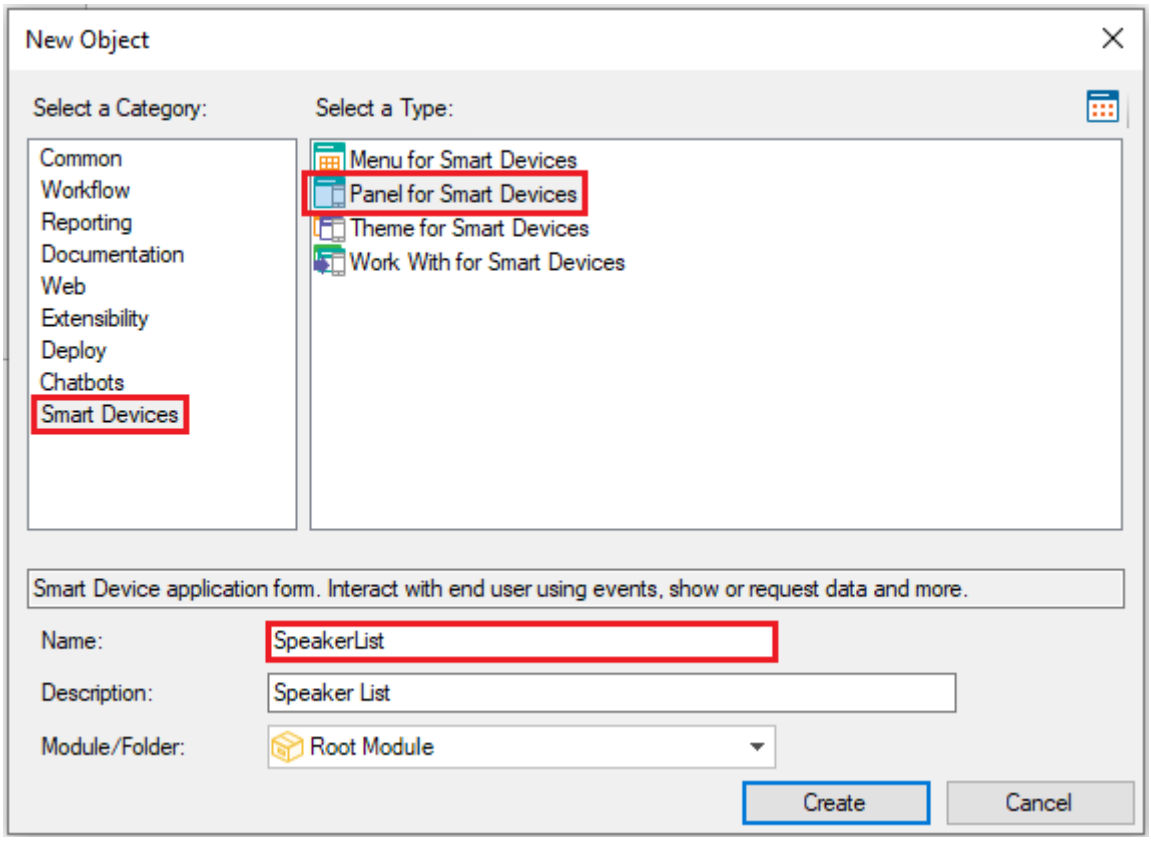


Por último, vamos a setear en la propiedad "Class" de SpeakerImage, el valor "ImageSpeakerList", "SpeakerName" el valor "AttributeSpeakerName" y "SpeakerCompany" el valor "AttributeSpeakerCompany". Guardamos los cambios del objeto.

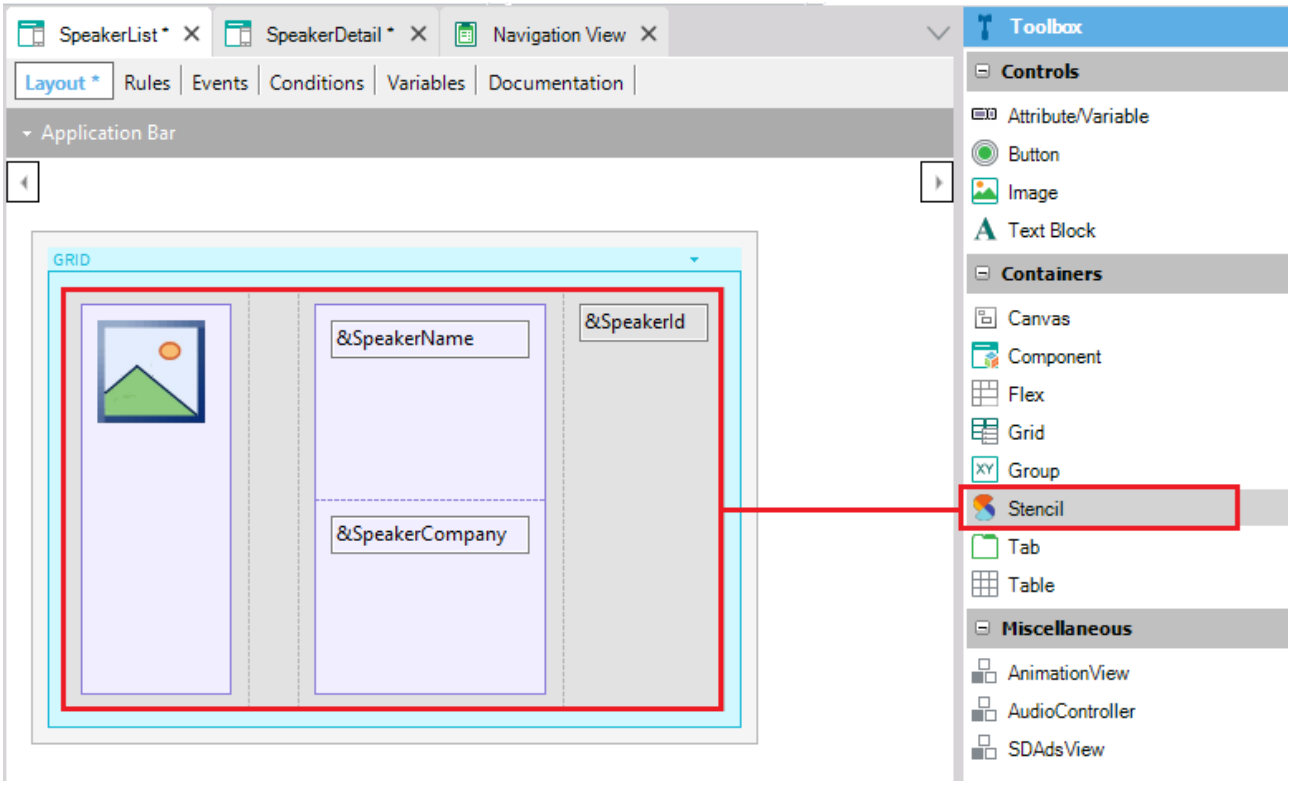
Nuestro diseño esta pronto y listo para usarse en todas las pantallas en las que lo precisemos.

PASO 3 – CREANDO SD PANELS Y MENU

Procedemos entonces a crear los objetos SD para nuestra aplicación. Para ello, hacemos **clic derecho** sobre la carpeta **AppSD**, **New > Object**, seleccionamos Category: **Smart Device**, Type: **Panel for Smart Devices** y de nombre le ponemos **SpeakerList**.

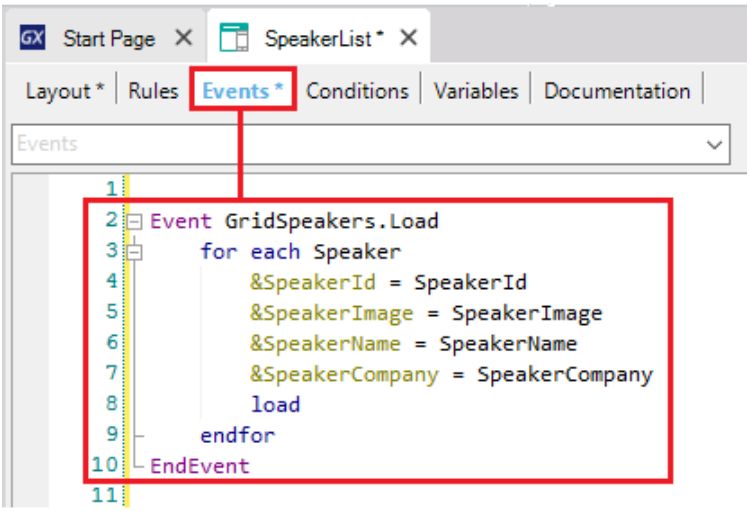


Luego, arrastramos desde la *Toolbox* el control grid hacia nuestro *layout*. Se nos desplegara un *popup* en el cual simplemente pondremos **"OK"**. Se creará entonces un grid (con una variable &Today, la cual borraremos) que será el encargado de mostrar todos los oradores del evento en el formato que ya tenemos creado en el objeto stencil. Entonces, arrastramos dicho stencil desde el *Toolbox* hacia dentro del grid, como muestra la siguiente imagen:

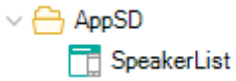


Una vez tengamos ese formato creado, lo único que nos falta es crear la función de carga para dicho grid. Nos posicionamos entonces sobre el grid, y cambiaremos su nombre en la propiedad "Control Name", asignándole el valor "GridSpeakers".

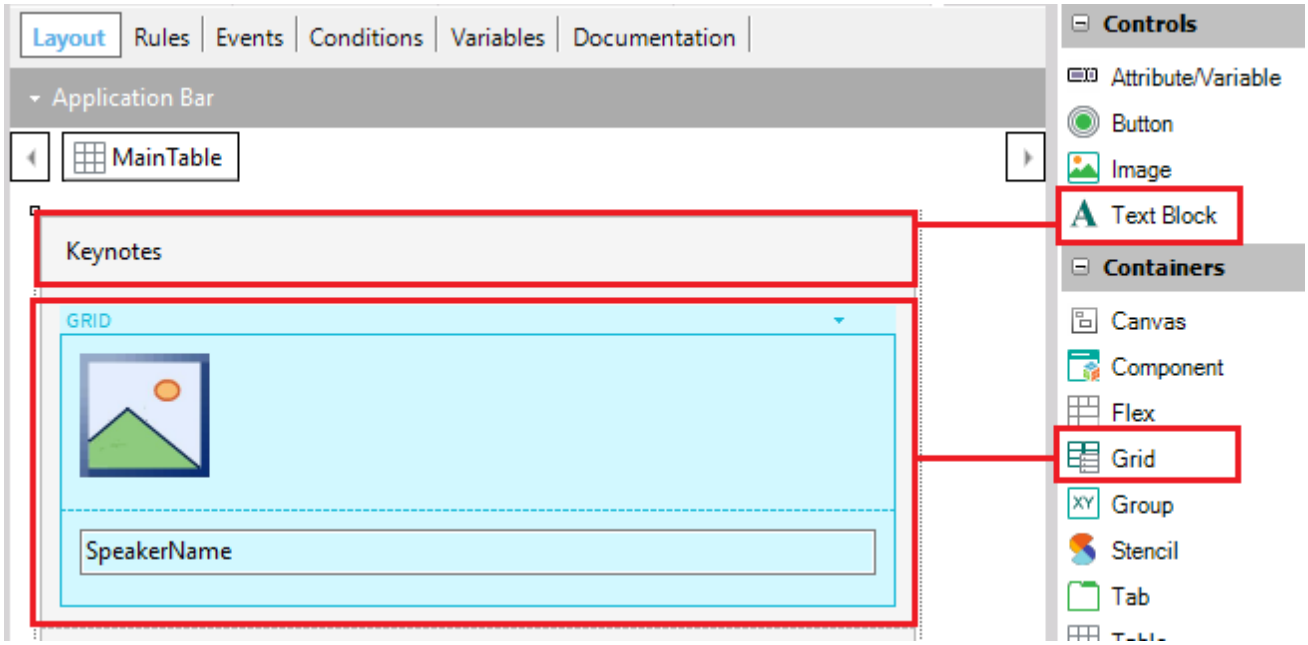
Luego vamos al tab "Events" en la parte superior del layout y vamos a crear el siguiente evento de carga de datos:



Para finalizar, hacemos clic en el botón de guardar y ya tenemos nuestro panel creado con éxito.



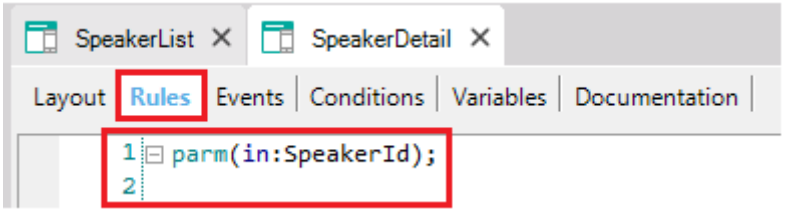
Repetimos el proceso y creamos otro **Panel for Smart Devices** de nombre **GXHome** y nuevamente arrastramos un **Grid** seleccionando esta vez, los atributos **SpeakerImage** y **SpeakerName**, y un Text Block para el título, de esta forma:



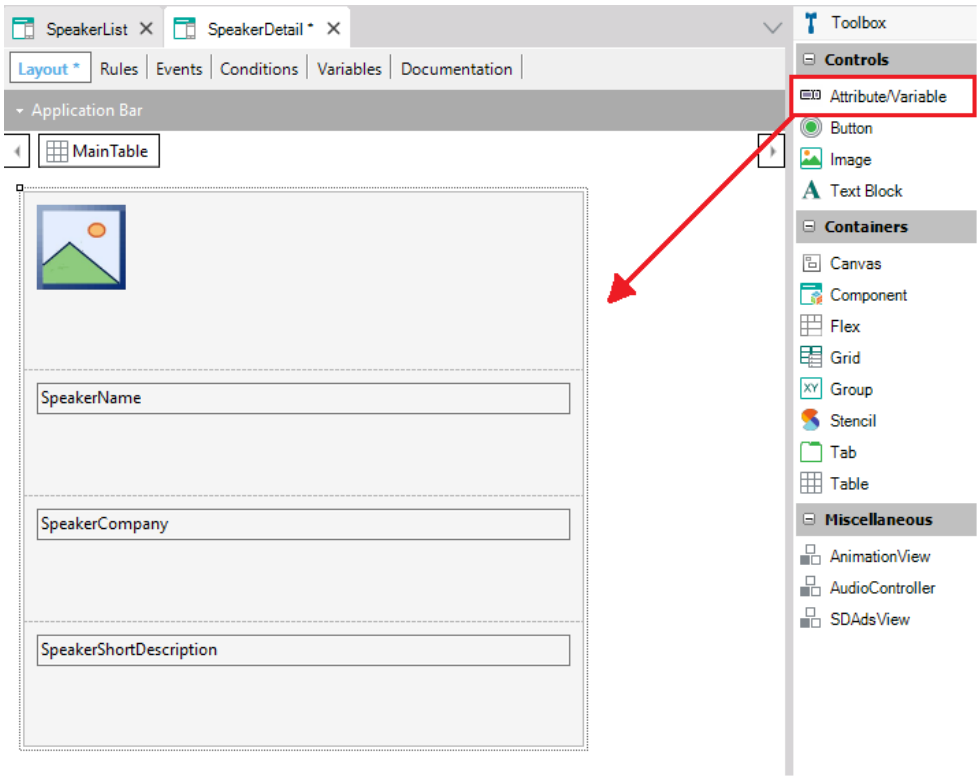
Dado que en este grid solamente queremos mostrar los oradores con Keynote, procedemos a poner en la propiedad "**Conditions**" del grid, la siguiente línea: "**SpeakerIsKeynote = true;**"

Guardamos (**CTRL + S**) y ya tenemos creado nuestro segundo Panel, el cual va a mostrar una grilla con las Keynotes destacadas, entre otros datos que veremos más adelante.

Queremos que al hacer **tap** sobre uno de los oradores (Keynotes), nos muestre en detalle la información del mismo. Para ello vamos crear otro Panel de nombre **SpeakerDetail**, agregándole en las **rules** que reciba como parámetro el ID del orador en el cual hicimos tap en la pantalla anterior, de la siguiente manera:



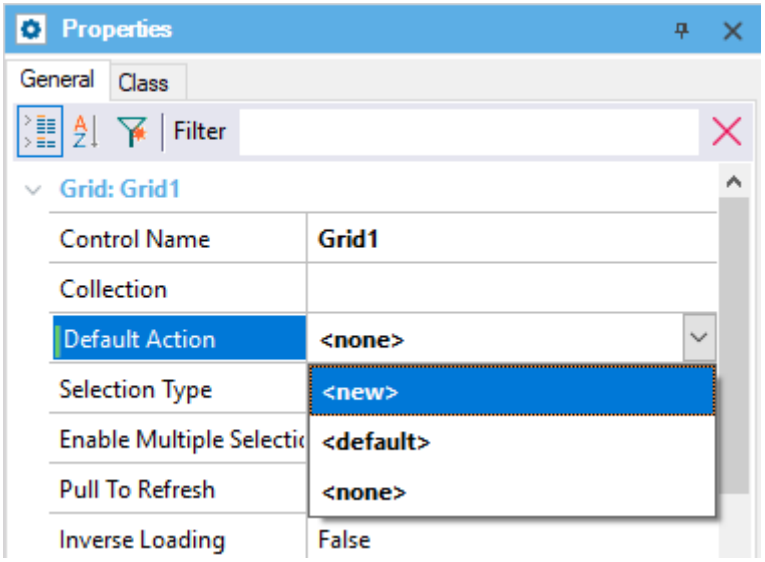
Luego volviendo al **Layout**, arrastramos los atributos **SpeakerImage, SpeakerName, SpeakerCompany y SpeakerShortDescription**:



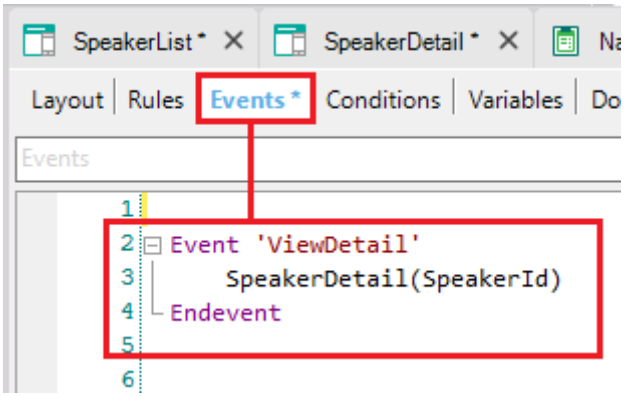
No olvidar quitar los labels de los atributos y a "**SpeakerShortDescription**" asignarle la propiedad "**Auto Grow = true**".

Guardamos y volvemos al Panel **GXHome**, y en las propiedades del **Grid** vamos a crear una nueva acción para que cuando se haga tap en algún Keynote, nos abra el panel **SpeakerDetail** con la información del orador.

Para ello, hacemos clic en la propiedad **Default Action > new** del Grid:



Con nombre "**ViewDetail**" y el siguiente código:

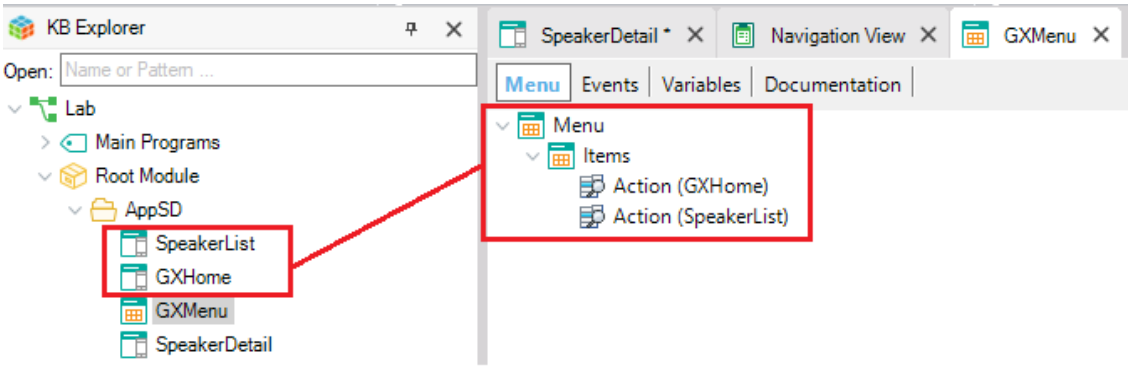


CREACIÓN DEL MENU DE LA APLICACIÓN

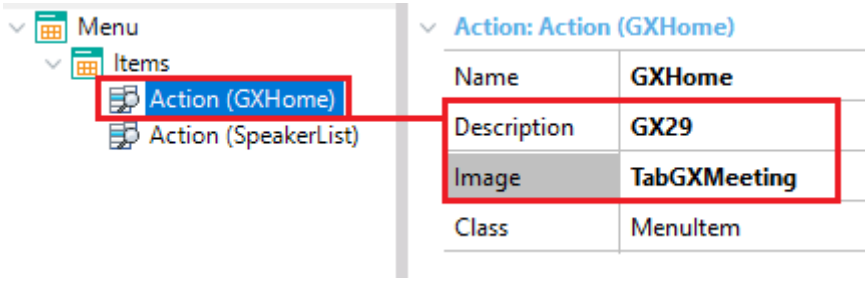
Ahora vamos a crear el **Menu for SD** para poder navegar entre los paneles recientemente creados y que además sea nuestro punto de entrada en la aplicación.

Para ello, nuevamente botón derecho sobre la carpeta **AppSD**, **New > Object**, y esta vez elegimos el Type: **Menu for Smart Devices** colocándole el nombre **GXMenu**.

Resta simplemente agregarle los paneles que queremos poder acceder desde el menú, para ello arrastramos **GXHome** y **SpeakerList** desde KB Explorer hacia **Items**, obteniendo la siguiente imagen:



Una vez hecho esto, procederemos a asignar imágenes de iconos descriptivas a las dos opciones de menú recién creadas. Para hacerlo, primero nos posicionamos sobre **"Action (GXHome)"** y en la propiedad **"Image"** seleccionamos el siguiente valor: **"TabGXMeeting".** En la propiedad **"Description"** escribimos el texto **"GX29".**

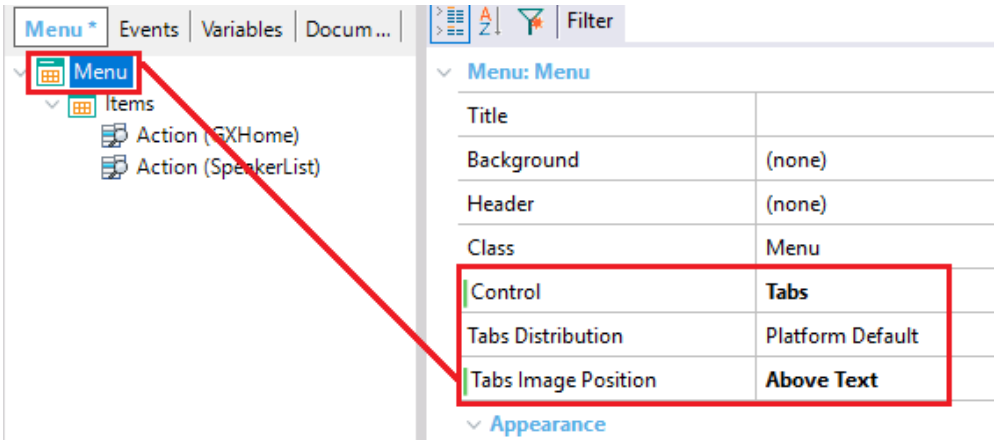


Análogamente para el ítem **"Action (SpeakerList)"** seleccionaremos la imagen **"TabSchedule"** y en la descripción pondremos **"Speakers".**

Ahora, vamos a seleccionar un **nombre e icono** para nuestra aplicación.
Para hacer esto, vamos a posicionarnos sobre nuestro **GXMenu** y en sus propiedades buscaremos **"Application Title"**. A dicha propiedad le pondremos el valor **"GX29"**.

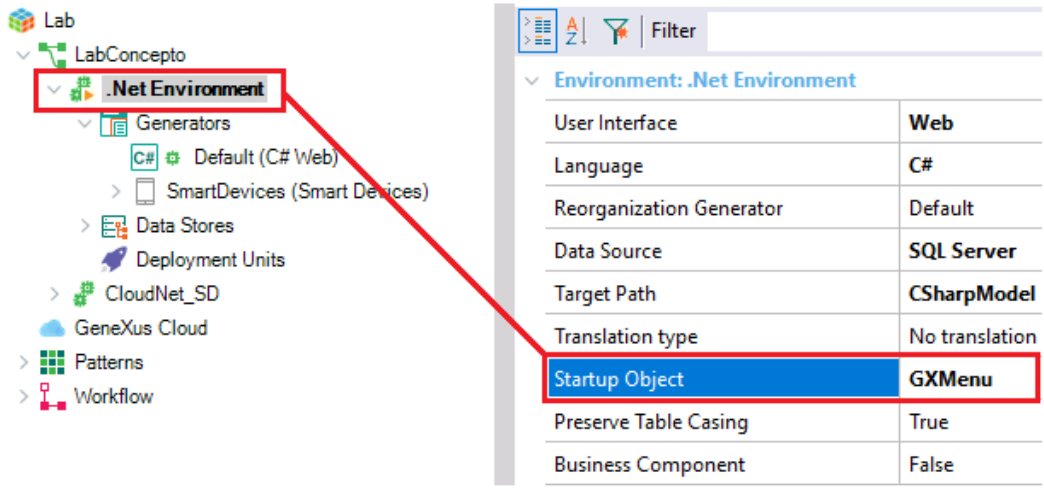
Luego, para setear un icono, buscamos la propiedad **“Android Application Icon”** o **“Apple Application Icon”** (dependiendo con qué dispositivo vayamos a realizar las pruebas) y le asignamos el valor **“GXMeetingIconAndroid”** o **“GXMeetingIcon”** respectivamente.

Finalmente, vamos a setear la propiedad “Tabs” (como muestra la imagen) para que nuestro menú en la aplicación se muestre como pestañas en la parte inferior.



Hacemos clic en **Save All** para que se guarden todos los cambios realizados hasta el momento.

Ya estamos listos para **ejecutar** la primera versión de nuestra aplicación. Para esto, sólo nos resta especificar que el Menu que recién creamos será el punto de entrada de la aplicación. Eso se hace mediante la propiedad **Startup Object** del .Net Environment:

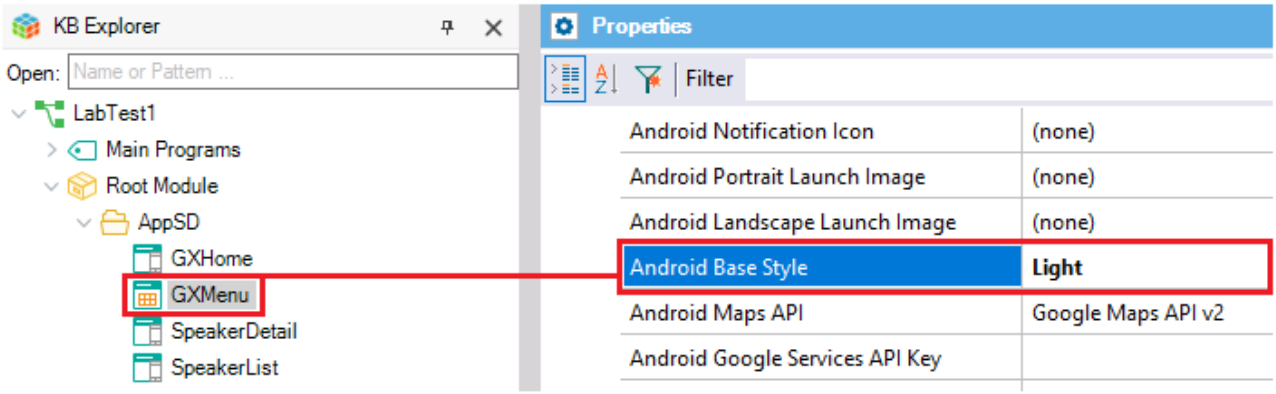


Nota para usuarios Android:

Para los participantes que vayan a ejecutar la aplicación en un dispositivo Android, deben hacer una configuración más que refiere a el “base style” que vamos a utilizar. El base style es el color

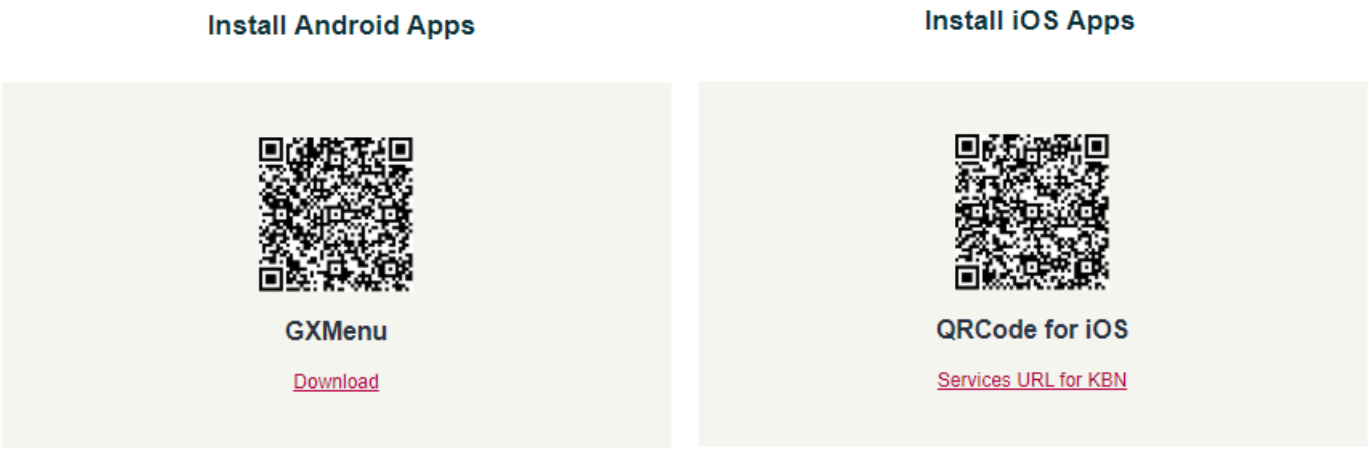
base que se va a utilizar en los layouts Android, en este caso pondremos el valor “light”, o sea **fondos blancos, fuentes negras y application bars blancas.**

Para ello, vamos a posicionarnos sobre el objeto GXMenu y cambiar su propiedad como muestra la siguiente imagen:



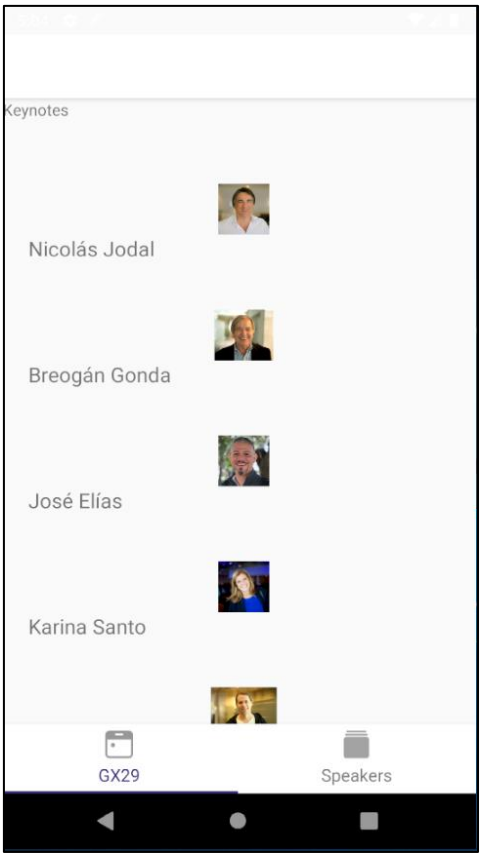
PASO 4 - EJECUCIÓN DE LA APLICACIÓN EN ANDROID E IOS

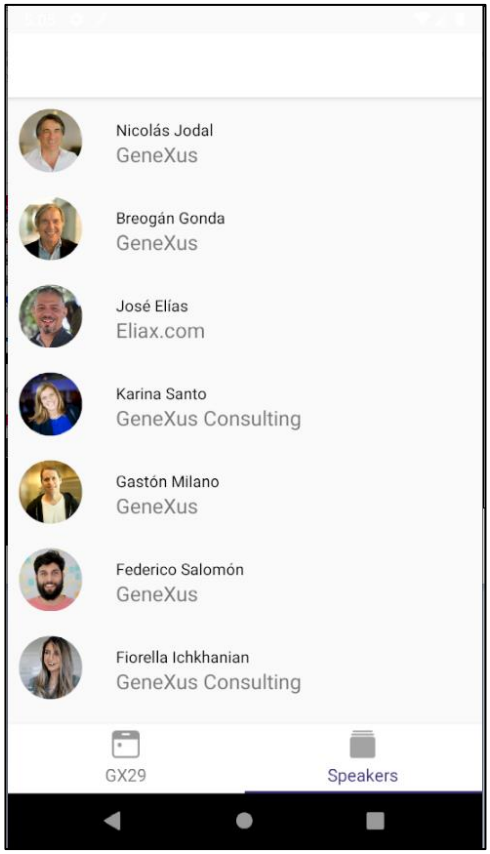
Hecho esto, ya podemos correr nuestra aplicación. Para ello, hacemos "**build all**" y luego "**View > Show QR codes**". Se abrirá una página web con los códigos QR que tendremos que leer desde nuestro dispositivo Android o iOS respectivamente.



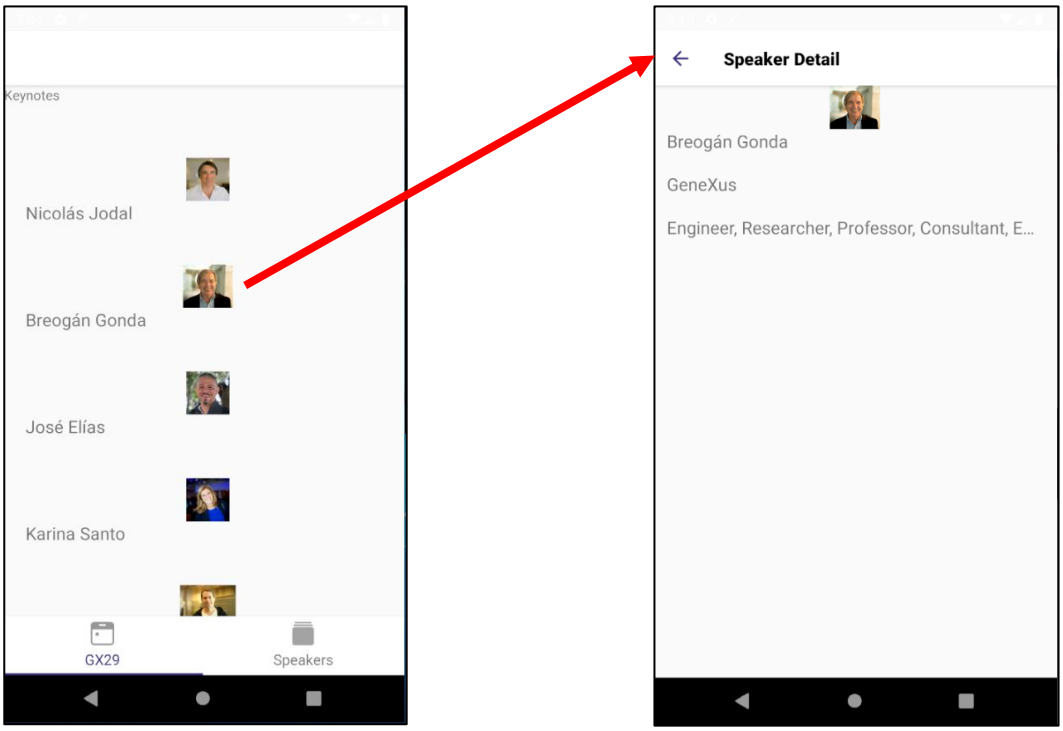
Para el caso de **iOS** se deberá tener instalado previamente la app "**GeneXus 16 KB Navigator**" que se puede encontrar en Apple Store.

Al ejecutar, podemos ver como se carga el Startup Object que habíamos configurado previamente: el Menu for SD **GXMenu**, que nos va a mostrar en la parte inferior, las opciones de menú que tenemos hasta el momento: "**GX29**" donde podremos ver los keynotes y "**Speakers**" donde podremos ver la lista de oradores.





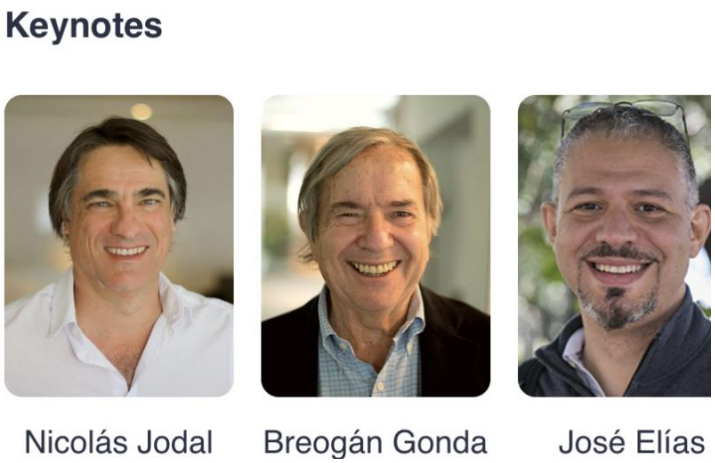
También, si hacemos tap sobre un orador, nos va a llevar a su detalle, tal como lo programamos.



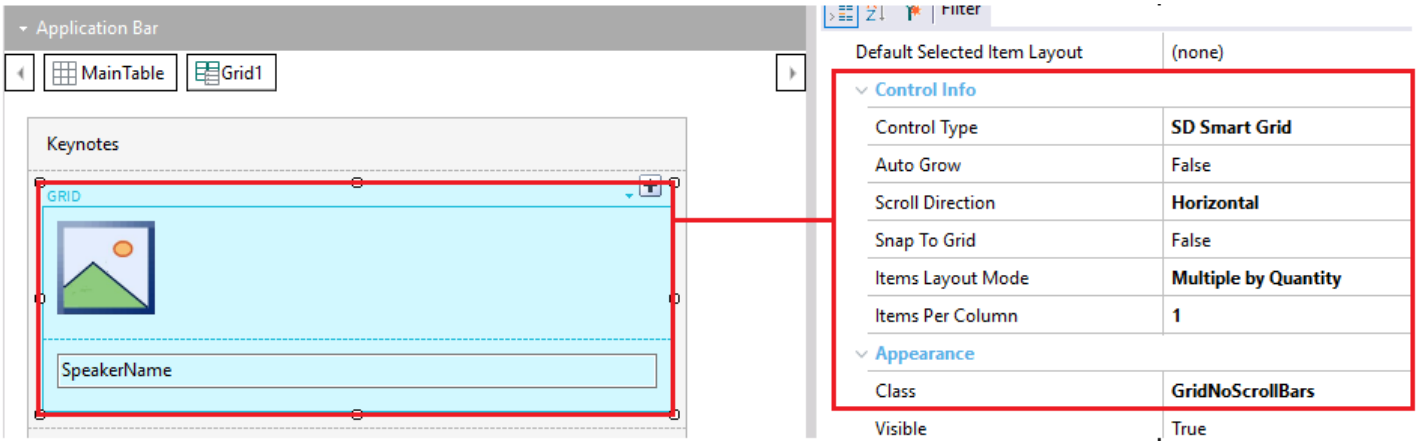
PASO 5 – AGREGANDO DISEÑO

Agregaremos ahora un poco de diseño para que nuestra aplicación se vea mejor.

Para mejorar el diseño del panel GXHome, vamos a modificar algunas de las propiedades del grid. La idea es que el resultado final se vea como la siguiente imagen, con scroll horizontal hacia la derecha:



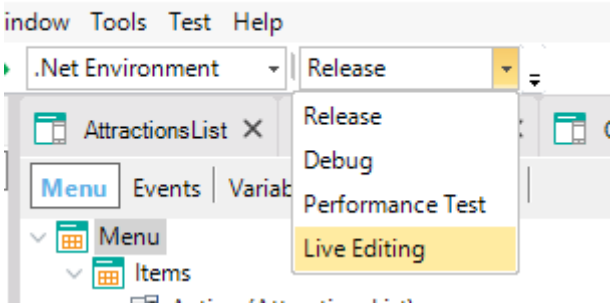
Para este fin, utilizaremos el control “SD Smart Grid”, configurándolo de la siguiente manera:



Luego, al atributo SpeakerImage que se encuentra dentro del grid, le seleccionaremos la clase “ImageKeynoteSpeaker”, para cambiar su aspecto y hacerlo mas adecuado al objetivo que nos propusimos. Haremos algo similar con el atributo SpeakerName, al cual le asignaremos la clase “AttributeSpeakerName”.

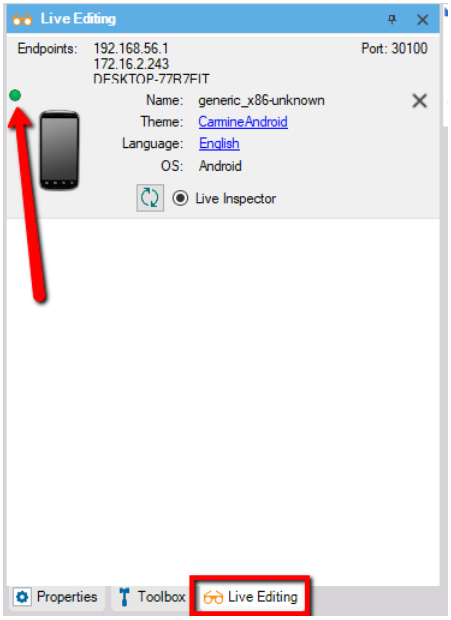
Antes de ejecutar para ver como va quedando nuestra aplicación con estos cambios, activaremos la funcionalidad de **Live Editing** que nos va a permitir ver en tiempo real, los cambios de diseños que realicemos en GeneXus, directamente en el dispositivo.

Para esto, debemos cambiar el valor del combo localizado en la segunda fila de la barra de herramientas, configurando el valor "**Live Editing**" en lugar del valor "**Release**" que se encuentra seleccionado.

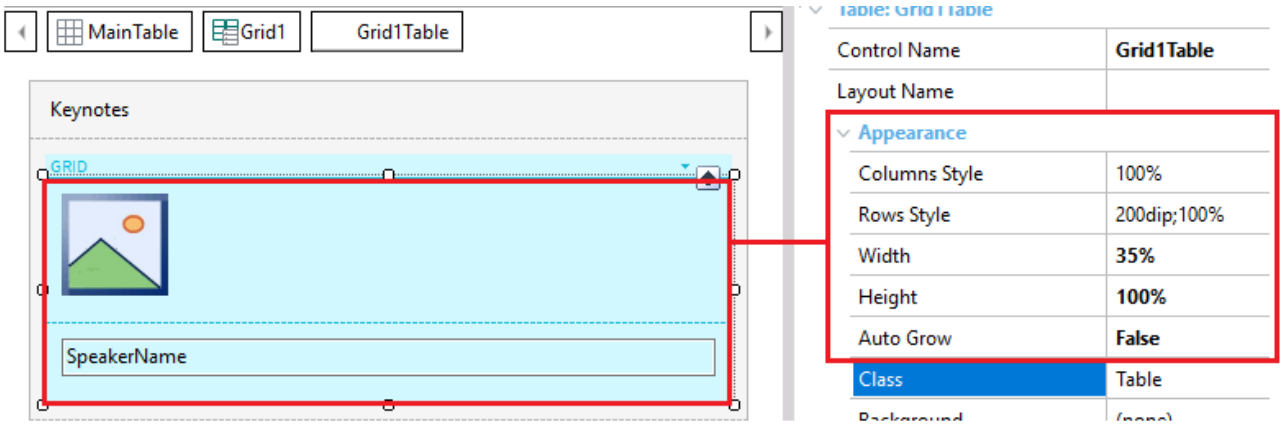


Hecho esto, volvemos a hacer "**Build All"** y leemos nuevamente los códigos QR en "**View > Show QR codes**". De esta forma va a quedar conectado **GeneXus con nuestra aplicación**. Es recomendable tener el celular con la aplicación abierta a la vista para cuando estemos modificando el diseño en GeneXus.

Verifique que el Live Editing quedó correctamente activado haciendo clic en la opción "Live Editing" en el toolbox y que aparezca el siguiente punto verde:



Con live editing prendido y funcionando en nuestro dispositivo, vamos a cambiarle las medidas a la tabla dentro del grid para que se vea mejor. De la siguiente forma:

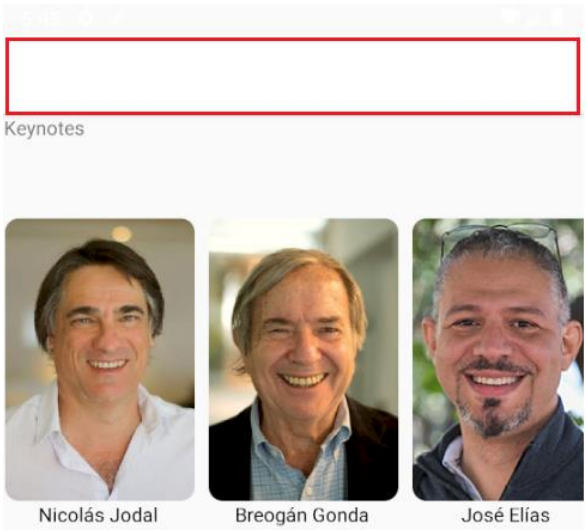


Tanto la imagen como el SpeakerName deben estar alineados horizontalmente al centro, asignando la propiedad correspondiente:

Horizontal Alignment	**Center**

Nota para usuarios Android:

Al momento de ejecutar la aplicación notarán que el application bar queda vacío sin texto.



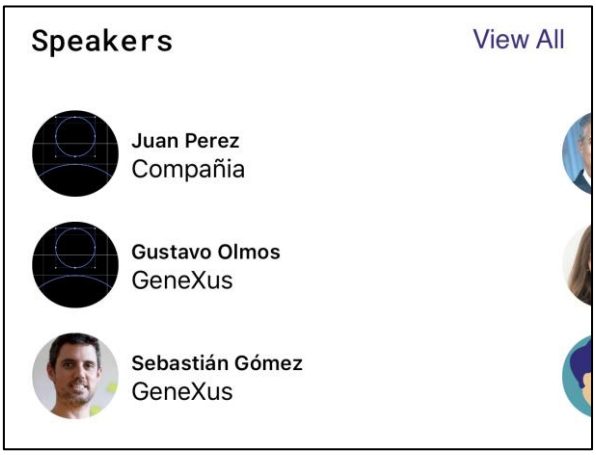
Procederemos a agregarle texto para mejorar el aspecto del panel. Para ello vamos al objeto GXHome y en la pestaña "Events", creamos el siguiente código:

```
Event Refresh
    Form.Caption = "The power of doing"
EndEvent
```

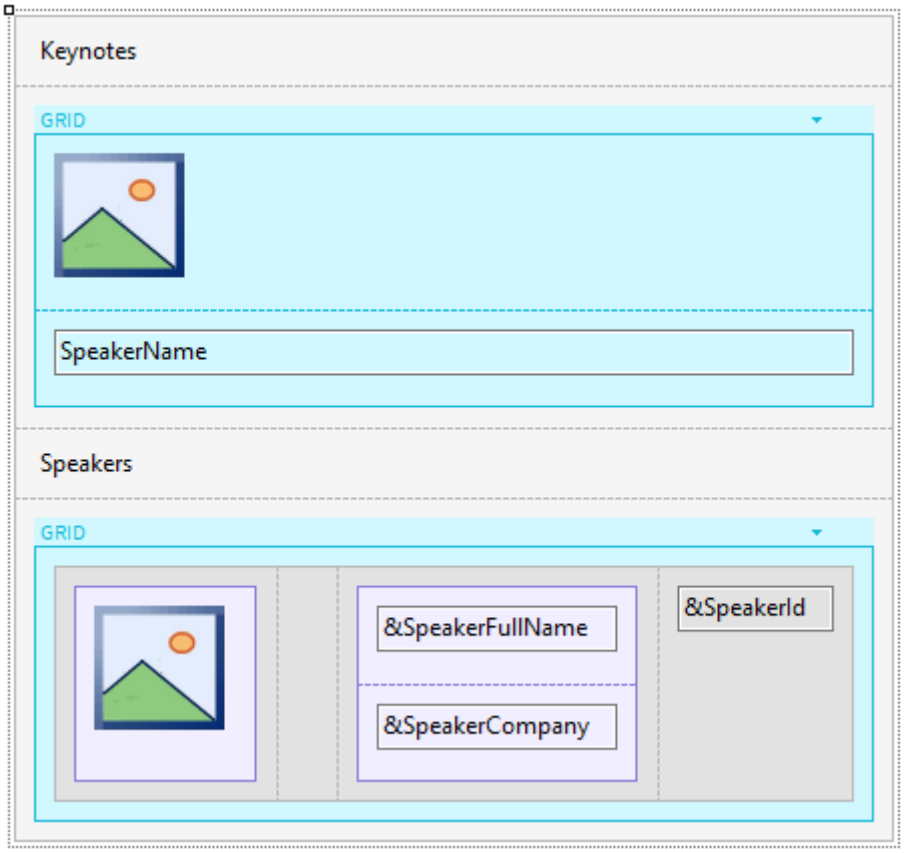
Haremos lo mismo para el panel SpeakerList, pero con el texto “Speakers”.

NUEVAS FORMAS DE MOSTRAR LISTAS

Agregaremos otro estilo de lista a nuestra pantalla de GXHome, en la que mostraremos oradores. El objetivo es llegar a un resultado como este:



Para lograrlo procederemos a agregar un nuevo Text Block con el texto “Speakers” y otro grid que contenga nuestro Stencil ya creado. Notar que como vamos a mostrar oradores con el mismo estilo estético que en nuestro otro panel “SpeakerList”, podemos reutilizar el Stencil que ya tenemos creado. El resultado final debería verse algo asi:



Para darle el diseño deseado al grid de Speakers vamos a setearlo de la siguiente forma:

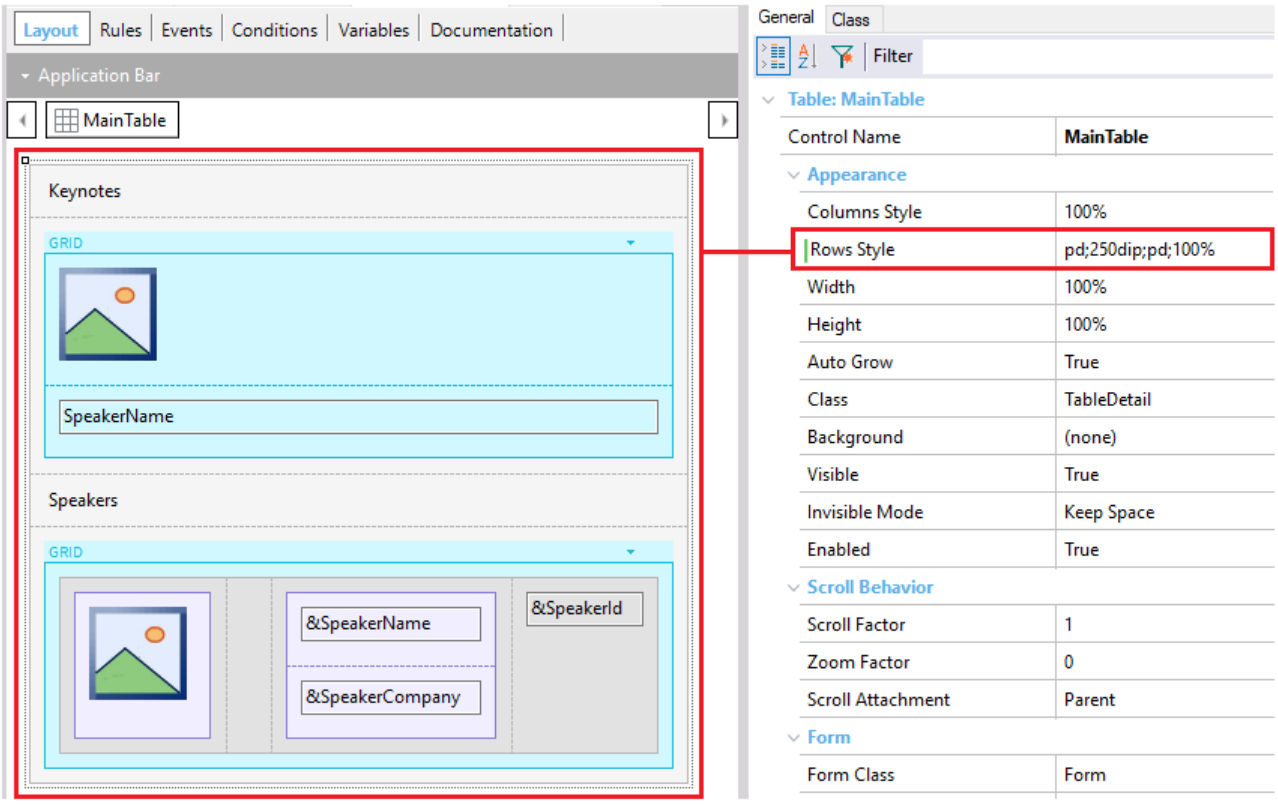
Control Info	
Control Type	**SD Smart Grid**
Auto Grow	**True**
Scroll Direction	**Horizontal**
Snap To Grid	**True**
Items Layout Mode	**Staggered by Quantity**
Items Per Column	**3**
Appearance	
Class	**GridNoScrollBars**

Ahora necesitamos cargar el grid, ya que, al ser formado con variables, no tiene tabla base y debemos indicarle a GeneXus como se va a nutrir de datos. Para esto, en la pestaña "Events" de GXHome, escribimos el siguiente código:

```
Event GridSpeakers.Load
    for each Speaker
        &SpeakerId = SpeakerId
        &SpeakerImage = SpeakerImage
        &SpeakerFullName = SpeakerName
        &SpeakerCompany = SpeakerCompany
        load
    endfor
EndEvent
```

Notar que el nombre del evento **"GridSpeakers.load"** refiere al nombre del grid en el layout, si no lo modificaste, por defecto tendrá el nombre **"Grid2"**.

Por último, antes de volver a ejecutar para ver todos los cambios realizados, vamos a asignar un tamaño a la MainTable del layout para que muestre correctamente los grids que realizamos.

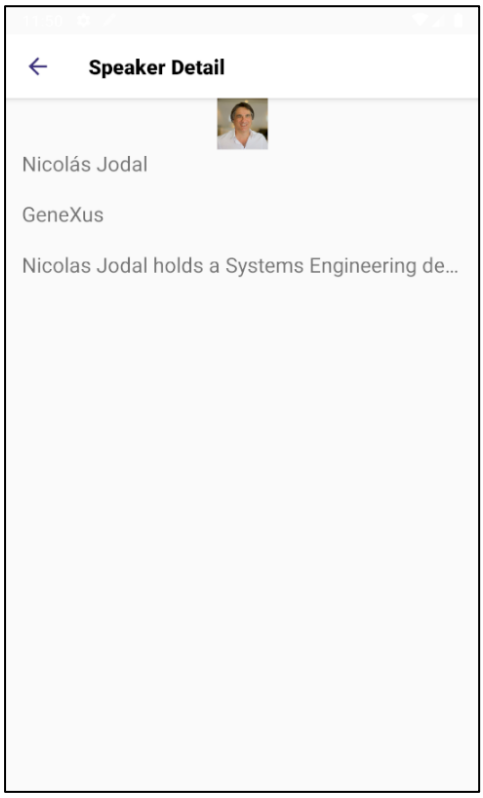


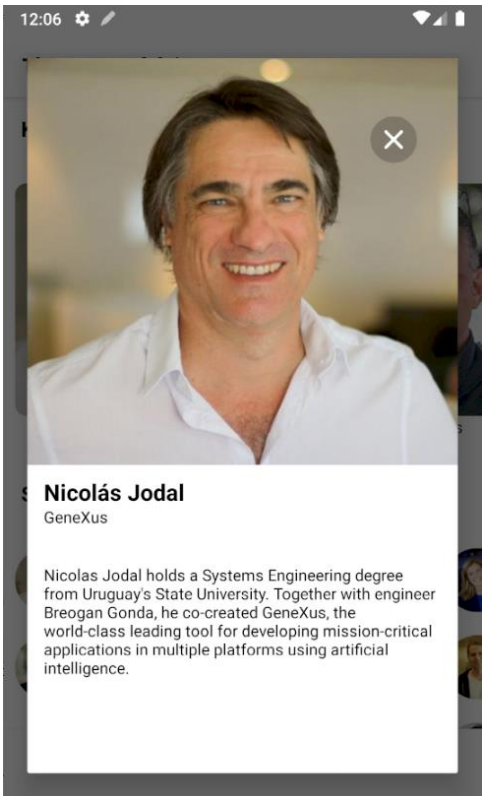
Como ultimo detalle, apliquemos la clase "TxtKeynotesTitle" a los dos text block que tenemos en el layout: Keynotes y Speakers.

Ahora ya podemos volver a ejecutar nuestra aplicación, haciendo "build all" y luego "View > Show QR codes" para leer nuevamente el código QR.

PASO 6 – SPEAKER DETAIL UTILIZANDO CANVAS.

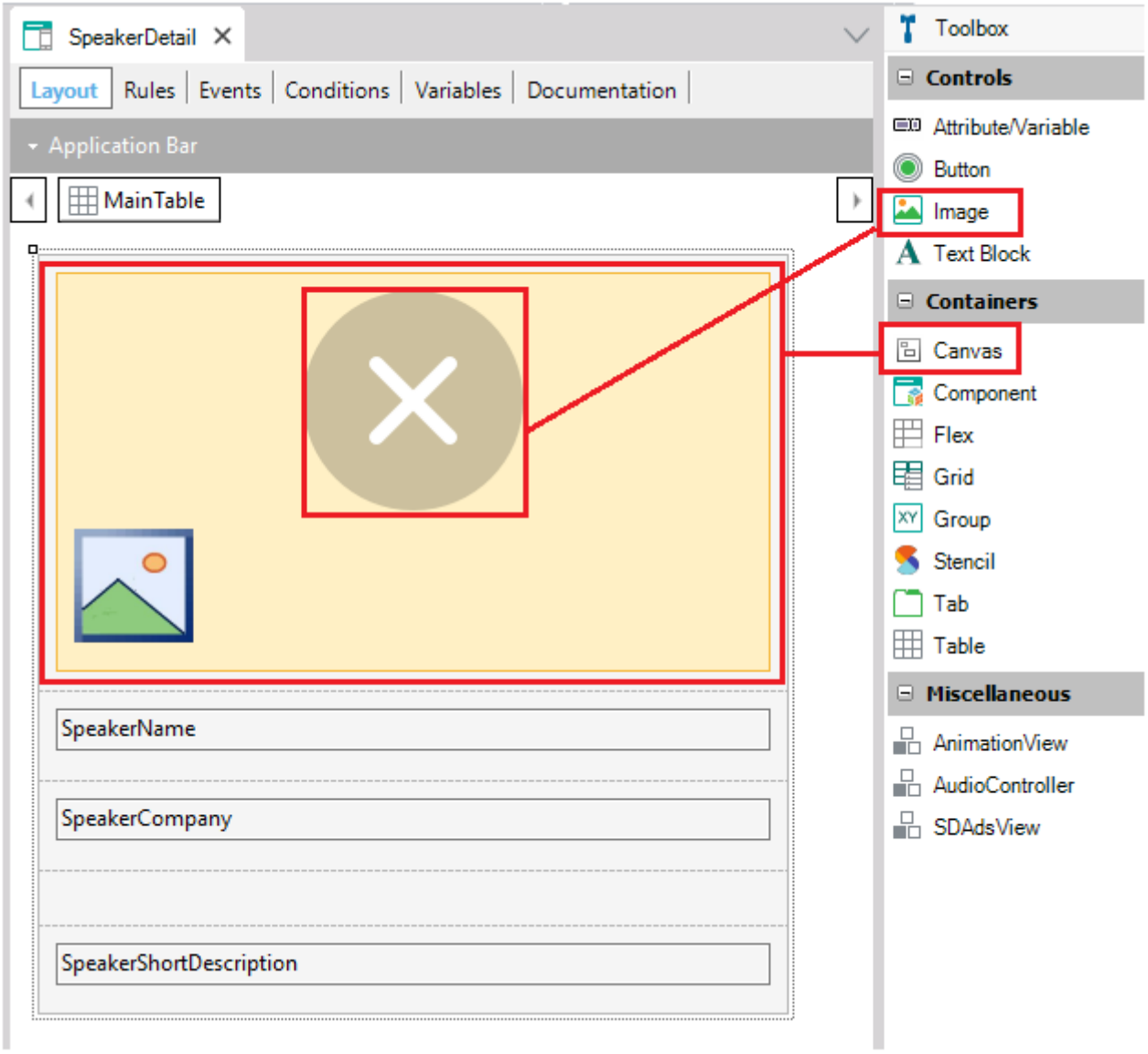
Procederemos ahora a darle diseño al panel de detalle de un orador. Veremos una imagen de como lo tenemos hasta el momento y otra con el resultado al que queremos llegar.





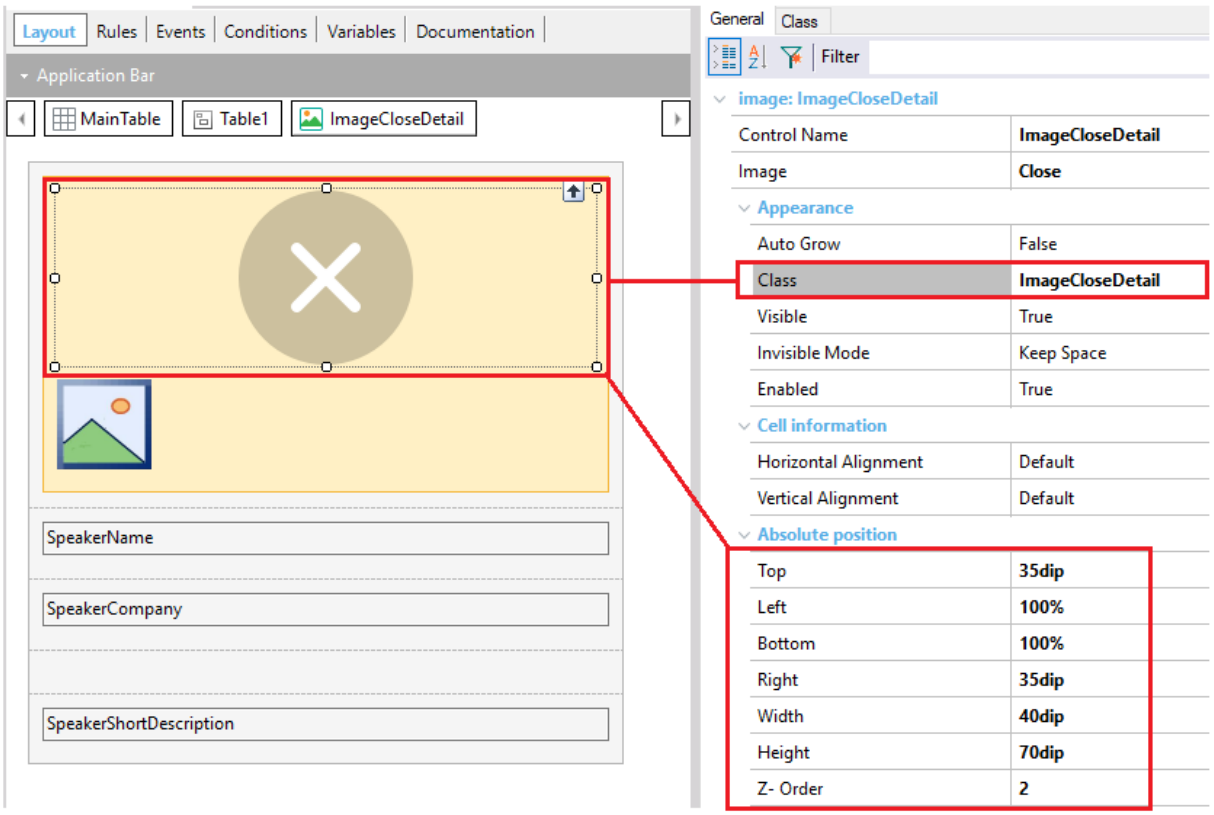
Para lograr este diseño, lo primero que debemos hacer es, en el objeto SpeakerDetail, crear un canvas en nuestro layout, que nos va a permitir tener una foto (en este caso la de Nicolás) con una imagen superpuesta (la cruz para cerrar el panel).

Procedemos entonces a hacer un drag & drop de un canvas a nuestro layout, dentro del cual también incluiremos una imagen, en este caso buscaremos la imagen con nombre "Close". Finalmente arrastramos dentro del canvas el atributo SpeakerImage que ya teníamos en la MainTable y el resultado debería verse de la siguiente manera:

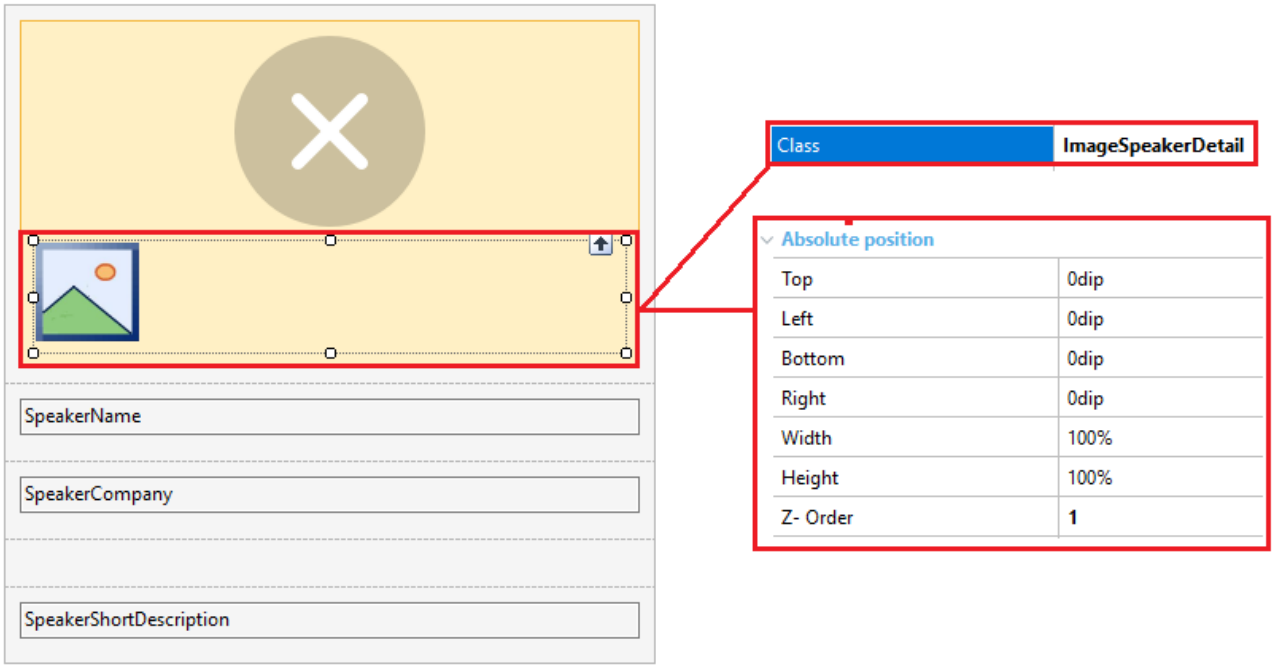


Acto seguido, debemos especificarle al canvas qué objeto va posicionado encima del otro, es decir, debemos configurar la propiedad **"Z-Order"** de cada elemento dentro del canvas.
A mayor Z-Order, más "arriba" quedará posicionado el elemento.

Comencemos modificando las propiedades de la imagen "close".
Deberían quedar como la siguiente imagen:



Y las propiedades de SpeakerImage, de la siguiente manera:



Finalmente, vamos a modificar el tamaño de las filas en la “MainTable” para darle el espacio deseado a cada elemento.

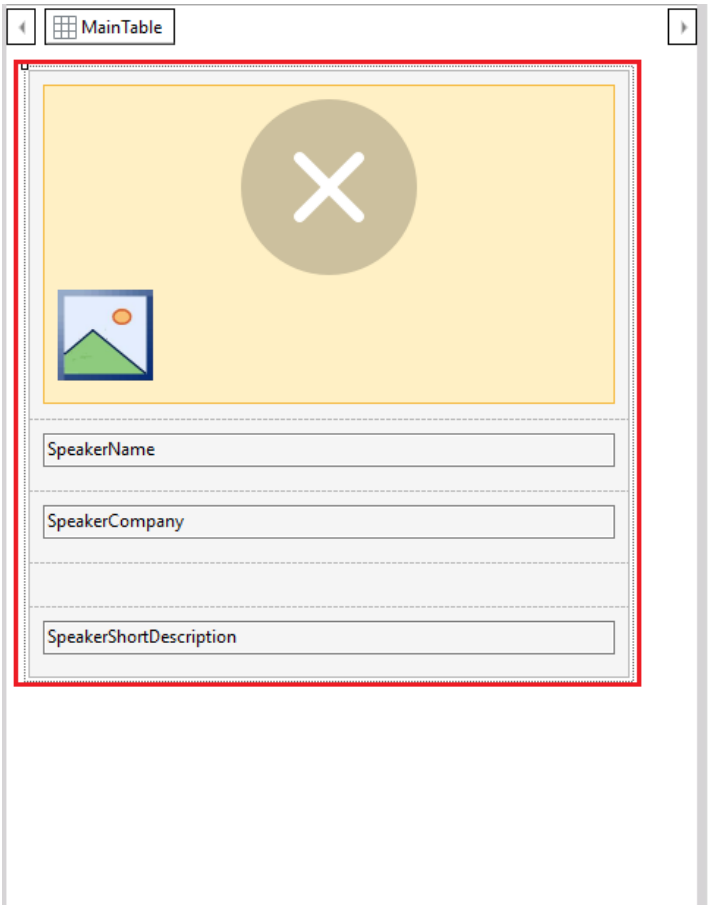


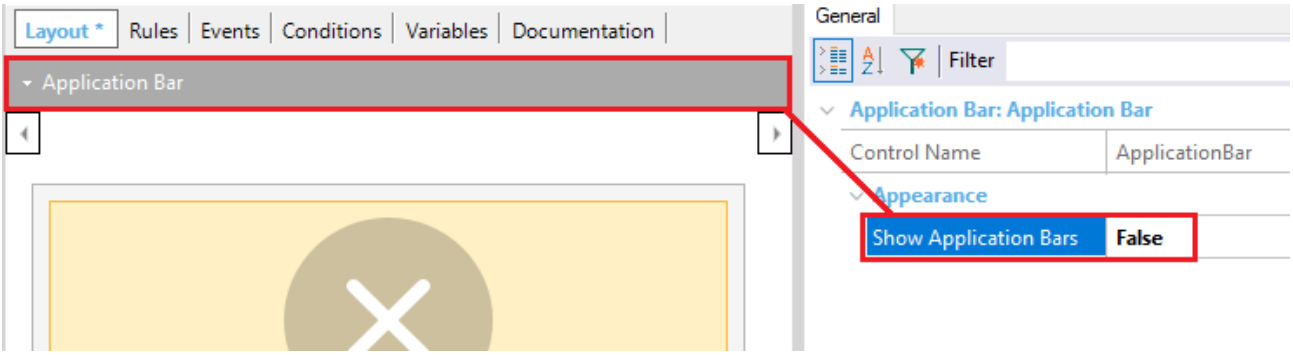
Table: MainTable

Control Name	MainTable
Appearance	
Columns Style	100%
Rows Style	350dip;35dip;25dip;25dip;pd
Width	100%
Height	100%
Auto Grow	True
Class	TableDetail
Background	(none)
Visible	True
Invisible Mode	Keep Space
Enabled	True
Scroll Behavior	
Scroll Factor	1
Zoom Factor	0
Scroll Attachment	Parent
Form	
Form Class	**FormPopup**
Enable Header Row Pattern	**True**
Header Row Application Bars Cla	ApplicationBarsHeaderRow
Layout Behavior	
Expand Bounds	**Content & Background**
Expand Bounds Limit	Behind System Bars
Expand Bounds Directions	Top

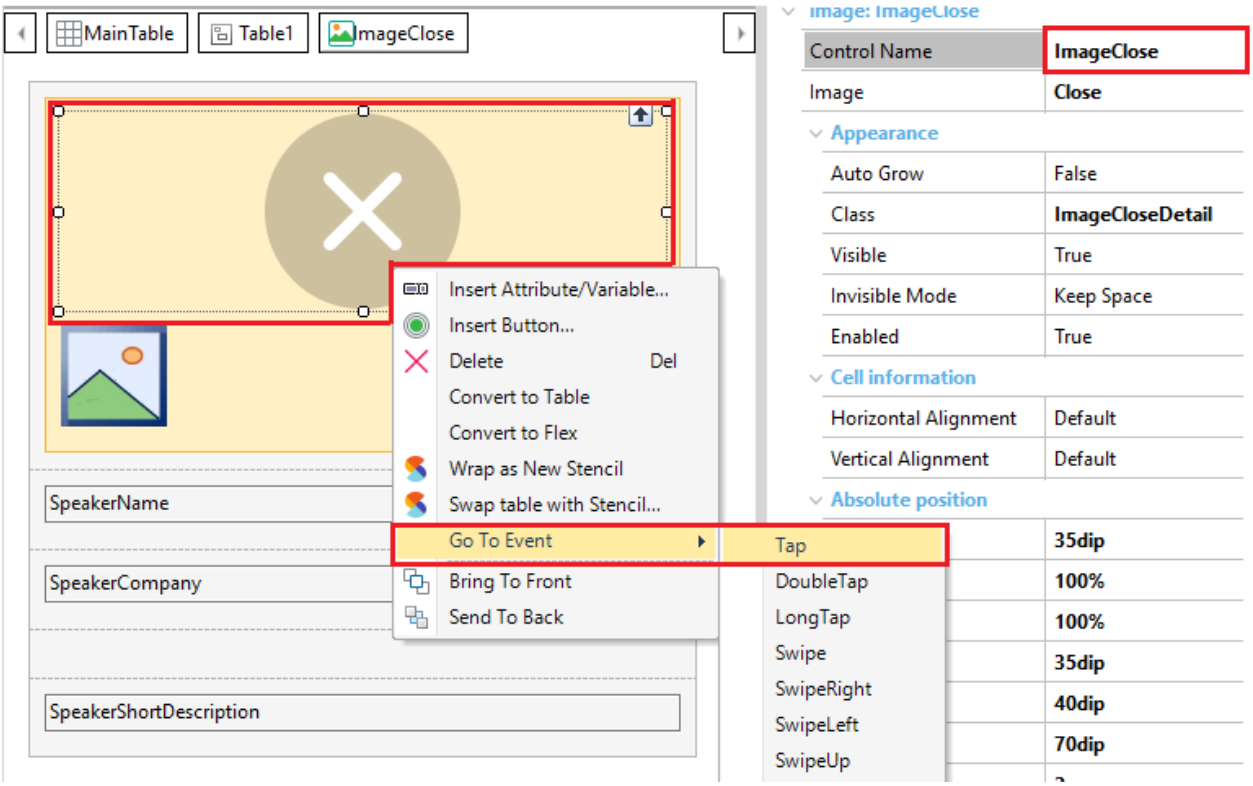
Agregamos los tamaños deseados para las filas y también asignamos la clase “FormPopup” para que el panel al ser llamado se despliegue como popUp sin application bar.

Acto seguido, asignamos en las propiedades de nuestro canvas, a la propiedad “Height” el valor “350dip”, para que tenga el espacio requerido para desplegar la imagen.

Pasemos ahora a posicionamos sobre el application bar del panel, y le asignamos la propiedad “Show Application Bars = false”, como muestra la siguiente imagen:



Tenemos que asignarle comportamiento al botón “Close” que acabamos de colocar en el layout, por lo que vamos a cambiar su nombre a “ImageClose” y crear un evento que haga el retorno al panel llamador, de la siguiente manera:



En el evento programamos el comportamiento deseado:

```
Event ImageClose.Tap
    return
Endevent
```

Una vez terminado, procedemos a setear las ultimas propiedades “class” a nuestros atributos del “SpeakerDetail”.

El atributo “SpeakerName” deberá tener asignada la propiedad:

“Class = AttributeSpeakerDetailName”

El atributo “SpeakerCompany” la propiedad:

“Class = AttributeSpeakerDetailCompany”

Y el atributo “SpeakerShortDescription” la propiedad:

“Class = AttributeSpeakerDetailDescription”.

Finalmente, podemos ejecutar nuevamente la aplicación para ver todos los cambios en funcionamiento, haciendo “Build” y leyendo nuevamente el código QR.

¡GRACIAS POR PARTICIPAR!

Glosario

Live Editing

https://wiki.genexus.com/commwiki/servlet/wiki?27806,Live+Editing+in+Smart+Devices+applications,

Menu for Smart Devices

https://wiki.genexus.com/commwiki/servlet/wiki?16321,Category%3AMenu+for+Smart+Devices+object

Canvas control

https://wiki.genexus.com/commwiki/servlet/wiki?22452,Canvas%20control

Stencils

https://wiki.genexus.com/commwiki/servlet/wiki?38418,Category%3AStencil+object,