

Using Apis to add functionalities



22-Using-apis.to.add.functionalities_sp



Behavior: events Using apis to add functionalities

Cecilia Fernández | GeneXus Training

Device Resources



Semantic Domains

Smart Devices APIs

Las aplicaciones para dispositivos móviles requerirán integrarse con recursos del dispositivo, tanto físicos como lógicos.

Por ejemplo deberán poder realizar llamadas telefónicas, enviar mensajes a través de los programas de chat o de mail instalados, acceder y usar la libreta de direcciones, la cámara, el GPS junto con las aplicaciones de mapas, así como interactuar con Facebook y Twitter.

Algunas de estas funcionalidades se implementaban automáticamente a través de los dominios semánticos... como por ejemplo...

Name	Type
Speaker	Speaker
SpeakerId	Id
SpeakerName	Name
SpeakerSurname	Surname
SpeakerFullName	VarChar(60)
SpeakerImage	Image
SpeakerCVMini	VarChar(1K)
CountryId	Id
CountryName	Name
SpeakerPhone	Phone
SpeakerAddress	Address
SpeakerEmail	Email

Name	Country
Blengio, Alejandro	USA
Bonilla, Fabian	Uruguay
Cardozo, Armando	Uruguay
Cimas, Alejandro	Uruguay
Fernandez, Gonzalo	Uruguay
Gonda, Breogan	Uruguay

teniendo el dominio semántico Phone para el atributo SpeakerPhone

22-Using-apis.to.add.functionalities.sp
Device Resources

Semantic Domains

Name	Type
Speaker	Speaker
SpeakerId	Id
SpeakerName	Name
SpeakerSurname	Surname
SpeakerFullName	VarChar(60)
SpeakerImage	Image
SpeakerCVMini	VarChar(1K)
CountryId	Id
CountryName	Name
SpeakerPhone	Phone
SpeakerAddress	Address
SpeakerEmail	Email

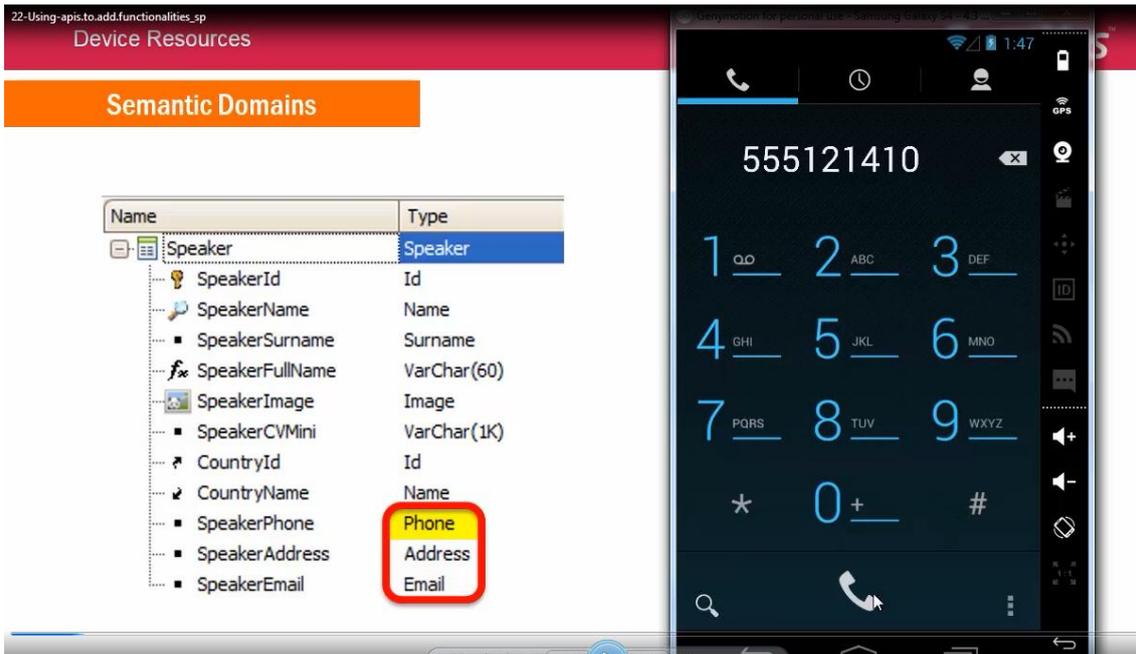
si vamos al detalle de un orador,

22-Using-apis.to.add.functionalities.sp
Device Resources

Semantic Domains

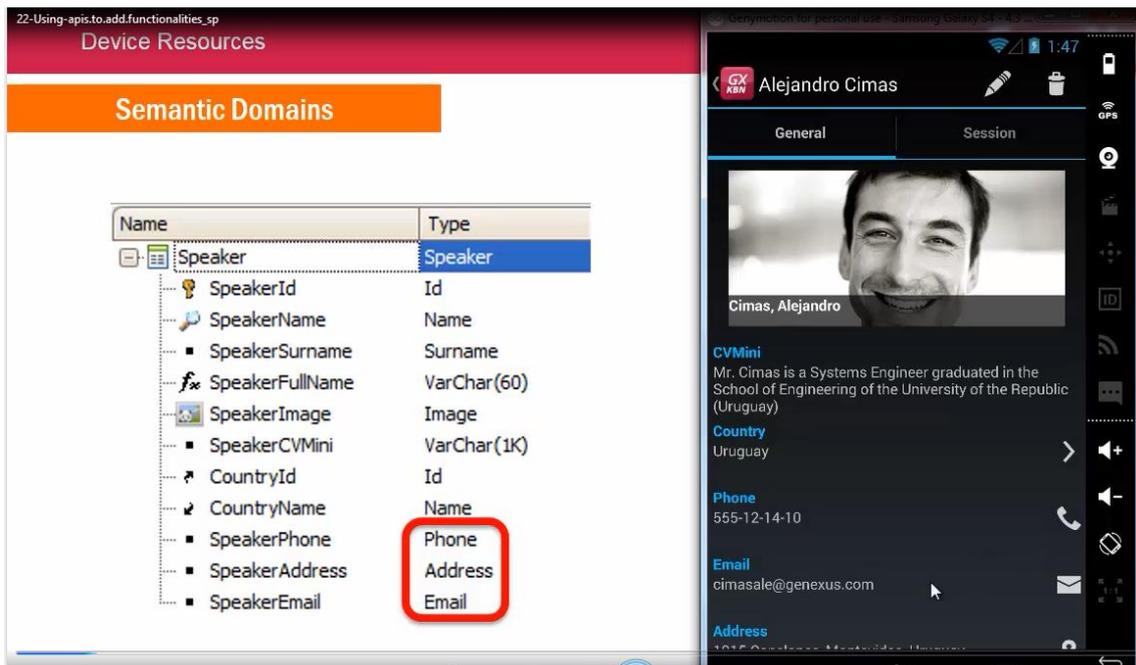
Name	Type
Speaker	Speaker
SpeakerId	Id
SpeakerName	Name
SpeakerSurname	Surname
SpeakerFullName	VarChar(60)
SpeakerImage	Image
SpeakerCVMini	VarChar(1K)
CountryId	Id
CountryName	Name
SpeakerPhone	Phone
SpeakerAddress	Address
SpeakerEmail	Email

y hacemos TAP sobre su campo Phone

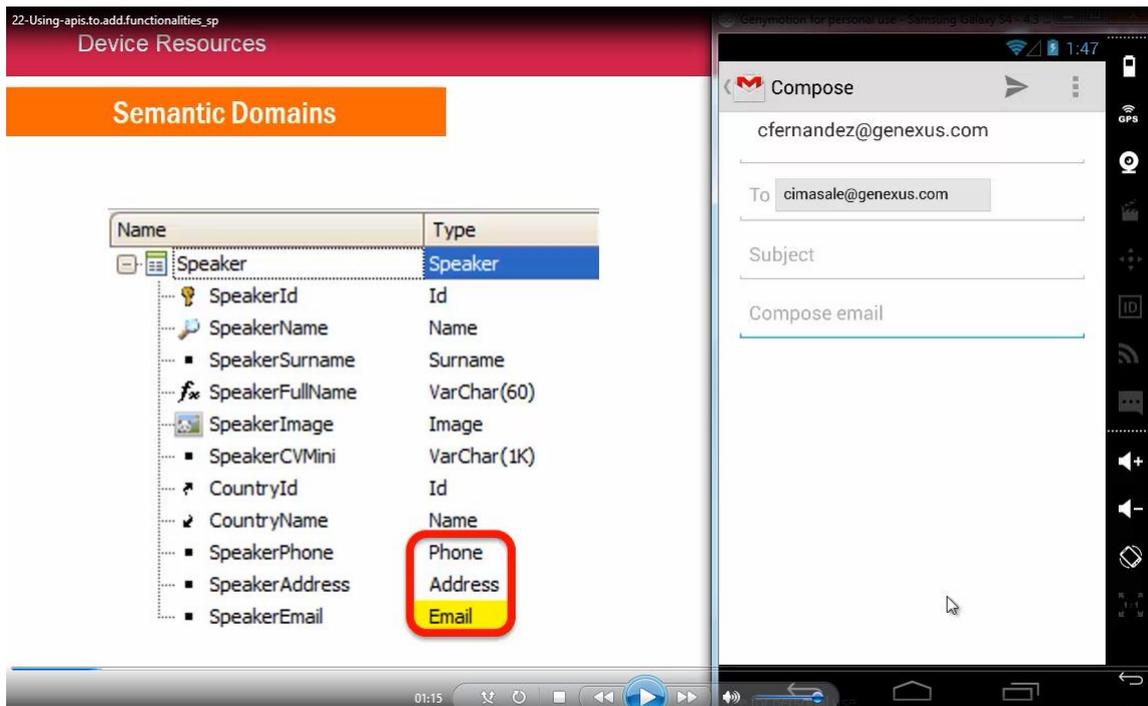


se nos abre la aplicación instalada en el dispositivo, para hacer la llamada telefónica.

Por otro lado, si hacemos TAP sobre el campo Email, de dominio semántico: Email

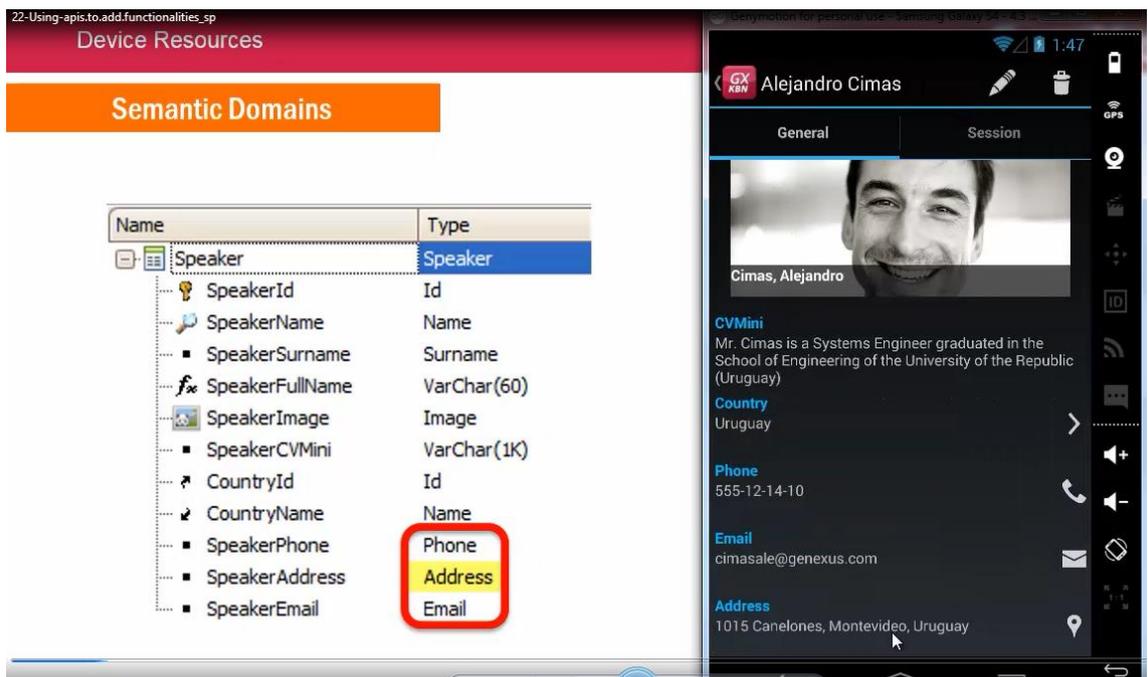


se nos abre la aplicación instalada en el dispositivo

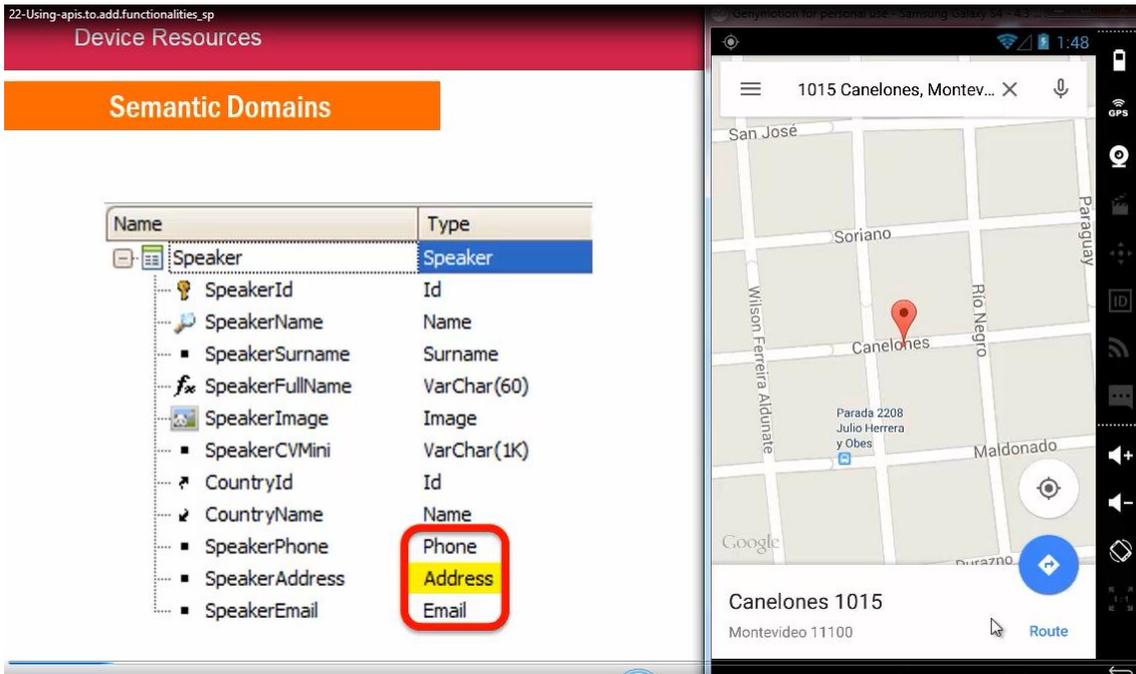


para escribir y enviar un email al orador correspondiente.

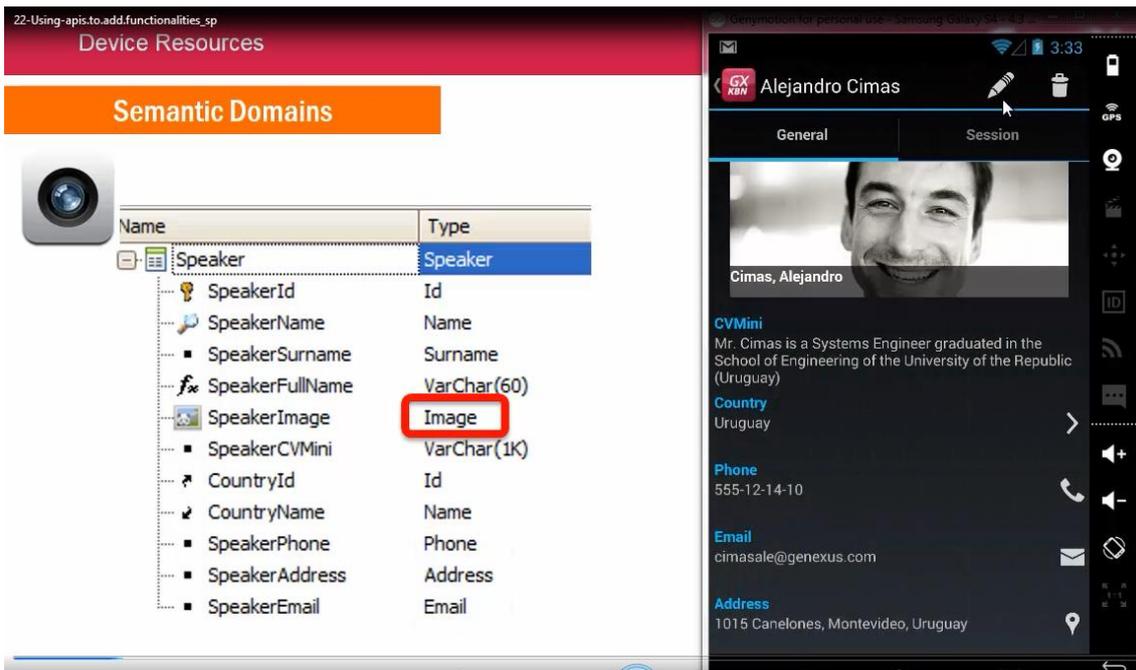
Y por último, si vamos al campo Address y hacemos TAP sobre la dirección



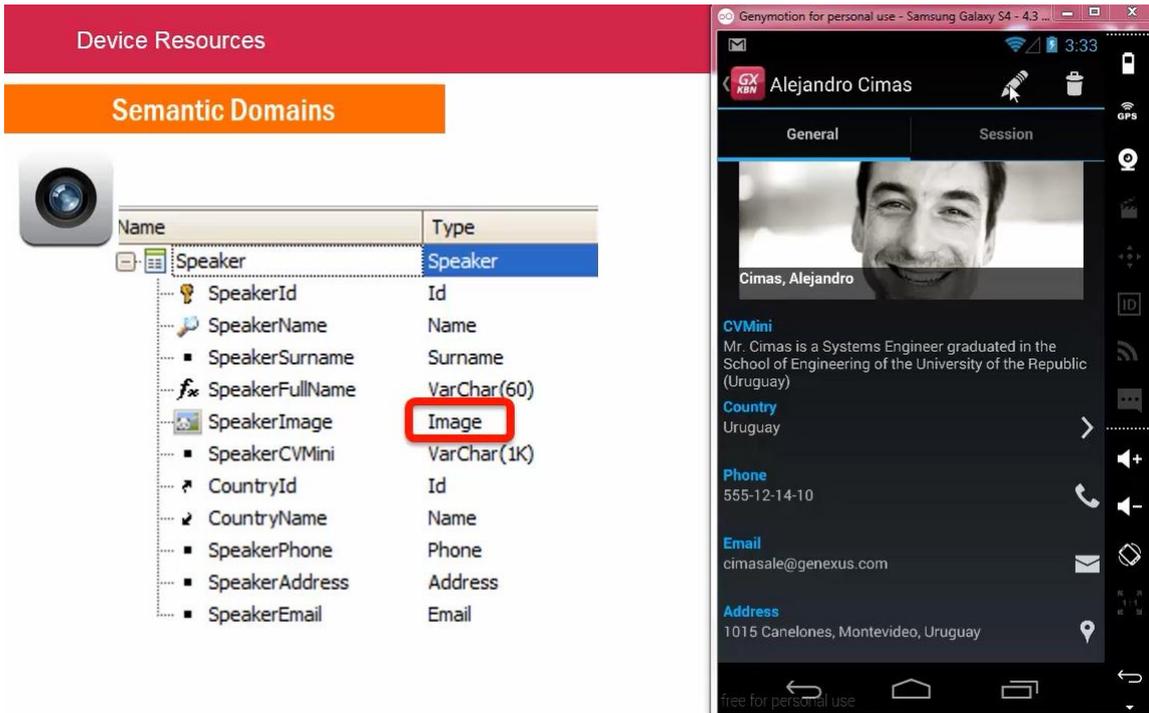
se nos abren los mapas instalados en el dispositivo



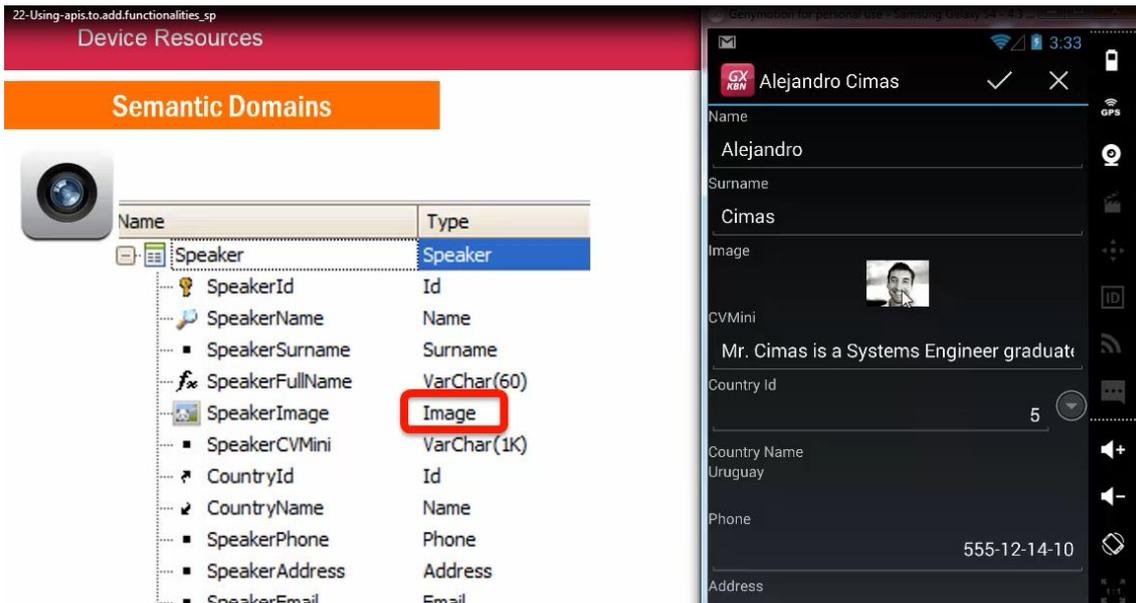
mostrándonos entonces la dirección en el mapa.



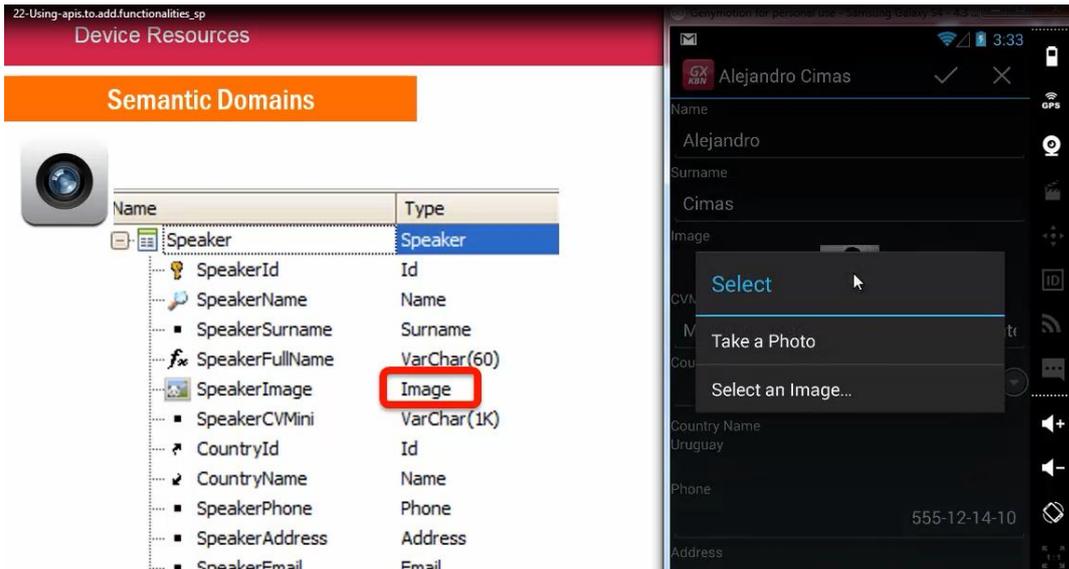
Por otro lado, sólo con tener el tipo de datos Image para un atributo, esto hará que en el Detail



cuando editemos la información



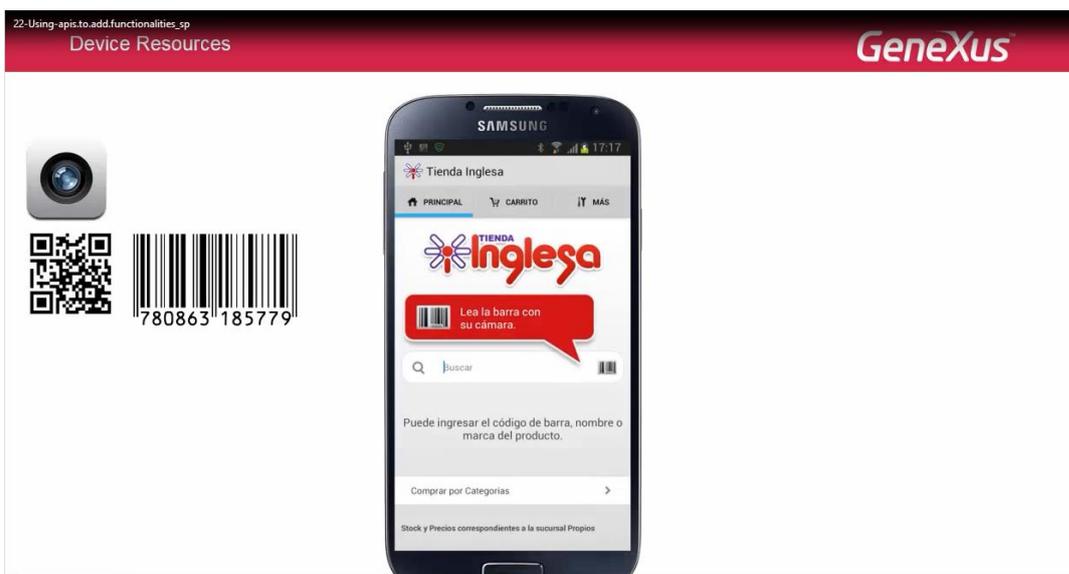
nos ofrezca el control correspondiente, tomar una foto haciendo uso de la cámara del dispositivo



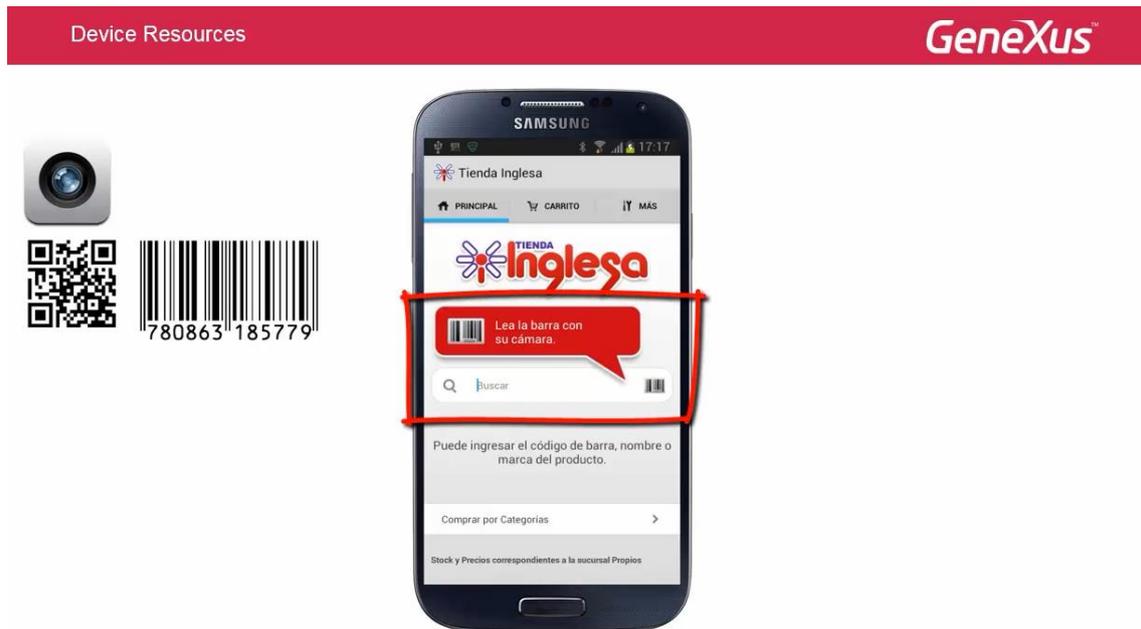
La cámara también puede ser usada para scanear QR codes o códigos de barra.



Por ejemplo para esta aplicación desarrollada para una cadena de supermercados

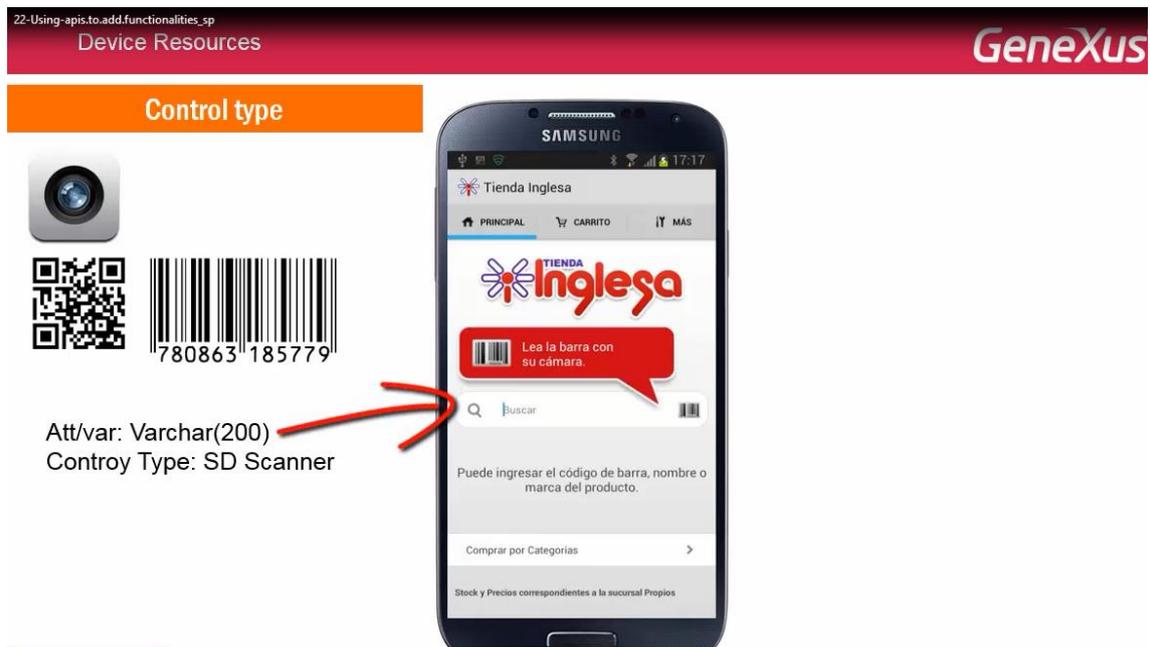


podemos escanear el código de barra del producto que tenemos en casa y que se nos acaba de terminar



para buscarlo en la aplicación y ver su precio actual o agregarlo al carrito de compras.

Para ello tenemos un par de opciones.



Teniendo un atributo o variable de tipo character o varchar, cambiándole su tipo de control a SD Scanner, cuando el control es de entrada, se agregará automáticamente un botón de Scan, que al ser presionado, abrirá la aplicación lectora de códigos de barras que el dispositivo tenga instalada.

22-Using-apis.to.add.functionalities_sp
Device Resources

GeneXus

Control type





Att/var: Varchar(200)
Controy Type: SD Scanner



Smart Devices APIs

```

Event &code.Tap
...
    &code = SacannerAPI.ScanBarcode()
    ShowProduct( &code )
...
endevent

```

La otra alternativa, será programar un evento, que podría ser el Tap sobre la variable

22-Using-apis.to.add.functionalities_sp
Device Resources

GeneXus

Control type





Att/var: Varchar(200)
Controy Type: SD Scanner



Smart Devices APIs

```

Event &code.Tap
...
    &code = SacannerAPI.ScanBarcode()
    ShowProduct( &code )
...
endevent

```

donde utilizemos el método ScanBarcode de una api

22-Using apis to add functionalities.sp
Device Resources

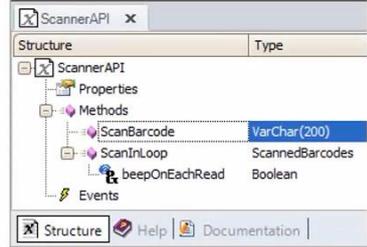
Control type



Att/var: Varchar(200)
Controy Type: SD Scanner



Smart Devices APIs



```

Event &code.Tap
...
  &code = SacannerAPI.ScanBarcode()
  ShowProduct( &code )
...
endevent

```

provista por GeneXus para tal fin.

La api es como un objeto externo que provee propiedades, métodos y eventos, para abstraer su implementación y proveer las funcionalidades.

Aquí estaríamos invocando un método que ejecutará el programa lector de códigos de barras que tenga instalado el dispositivo que utilizará, por su parte, la cámara:

22-Using apis to add functionalities.sp
Device Resources

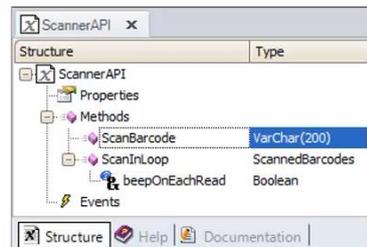
Control type



Att/var: Varchar(200)
Controy Type: SD Scanner



Smart Devices APIs



```

Event &code.Tap
...
  &code = SacannerAPI.ScanBarcode()
  ShowProduct( &code )
...
endevent

```

El valor leído, será devuelto por el método

22-Using-apis.to.add.functionalities.sp
Device Resources

GeneXus

Control type



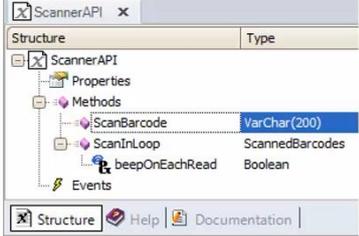


780863 185779

Att/var: Varchar(200)
Controy Type: SD Scanner



Smart Devices APIs



```

Event &code.Tap
...
  &code = SacannerAPI.ScanBarcode()
  ShowProduct( &code )
...
endevent

```

y podremos luego, invocar a un panel for Smart Devices que muestre toda la información de ese producto

22-Using-apis.to.add.functionalities.sp
Device Resources

GeneXus

Control type



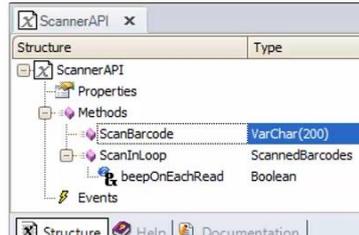


780863 185779

Att/var: Varchar(200)
Controy Type: SD Scanner



Smart Devices APIs



```

Event &code.Tap
...
  &code = SacannerAPI.ScanBarcode()
  ShowProduct( &code )
...
endevent

```

En algunos casos, las Apis necesitarán trabajar con tipos de datos estructurados, que vendrán definidos automáticamente en GeneXus, junto con ellas.

Por ejemplo, para poder leer una serie de códigos de barras, uno detrás del otro, para recién luego procesarlos, es que existe un sdt colección y un método ScanInLoop.

Device Resources



Semantic Domains

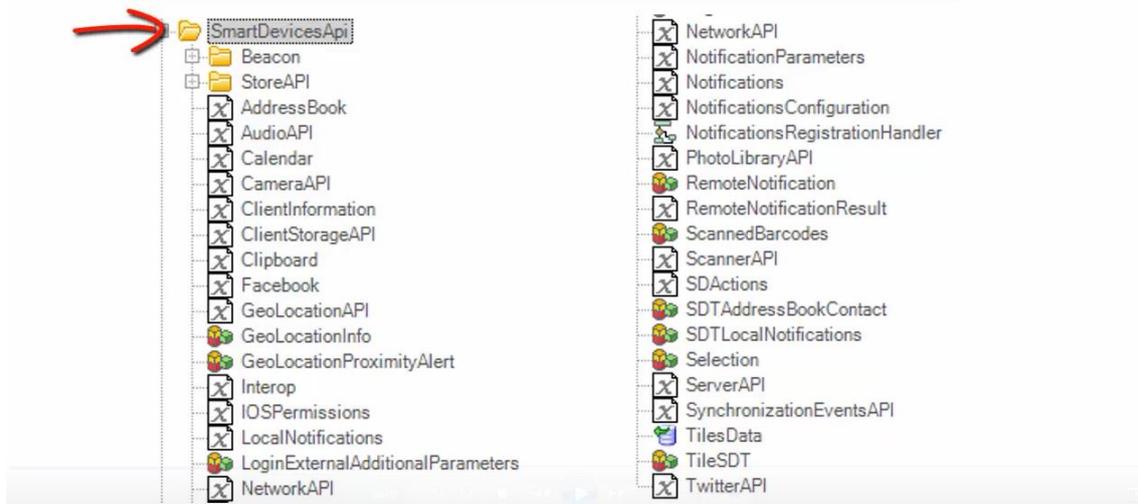
Smart Devices APIs

En definitiva, GeneXus nos brindará un conjunto de apis para poder agregar diferentes funcionalidades a la aplicación móvil; en particular para permitir la integración con otras aplicaciones y funcionalidades del dispositivo.

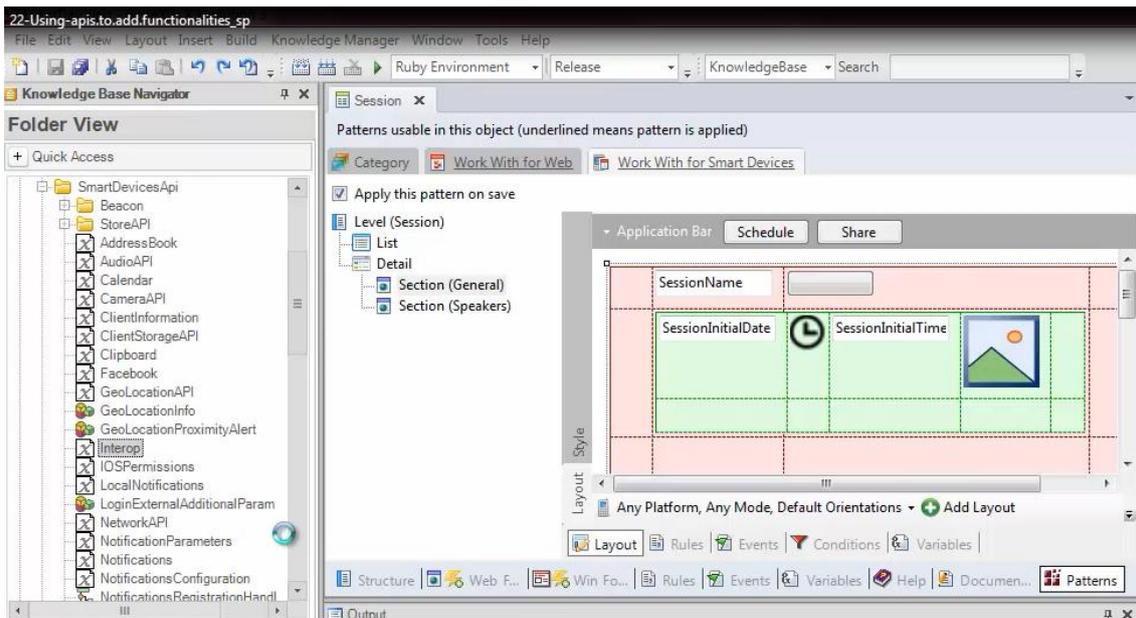
Device Resources

GeneXus™

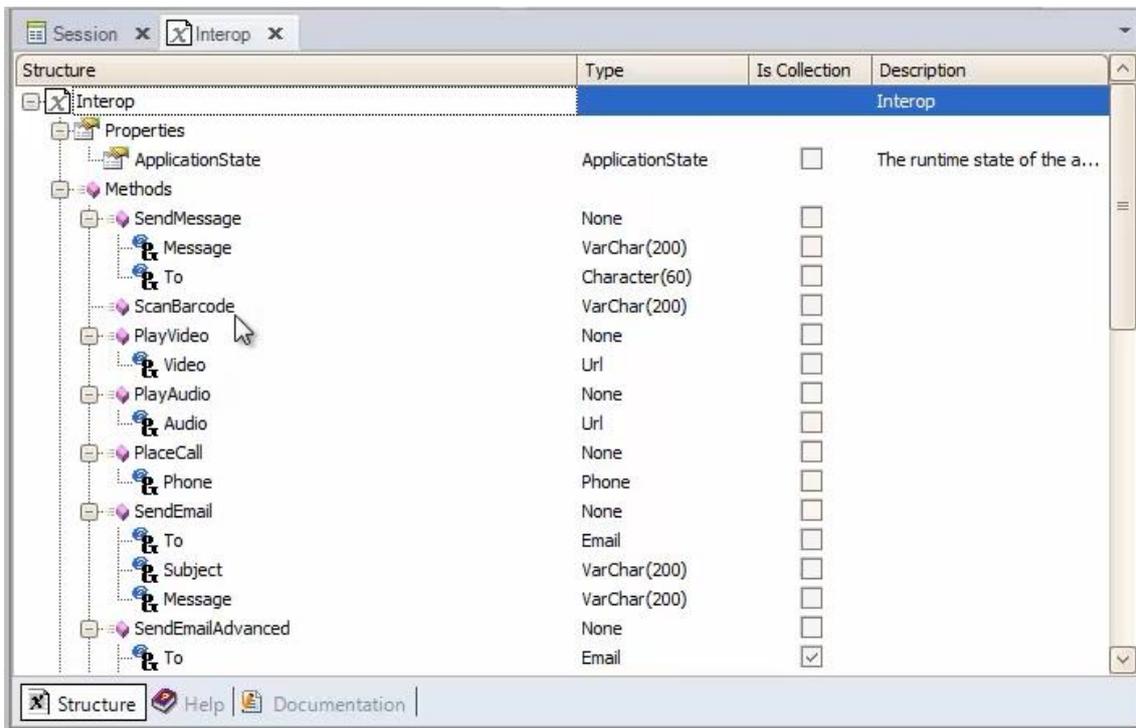
Smart Devices APIs



Es dentro del folder SmartDevicesApi que se encuentran todas las apis y los sdt's, procs y data providers, que estas requieran.

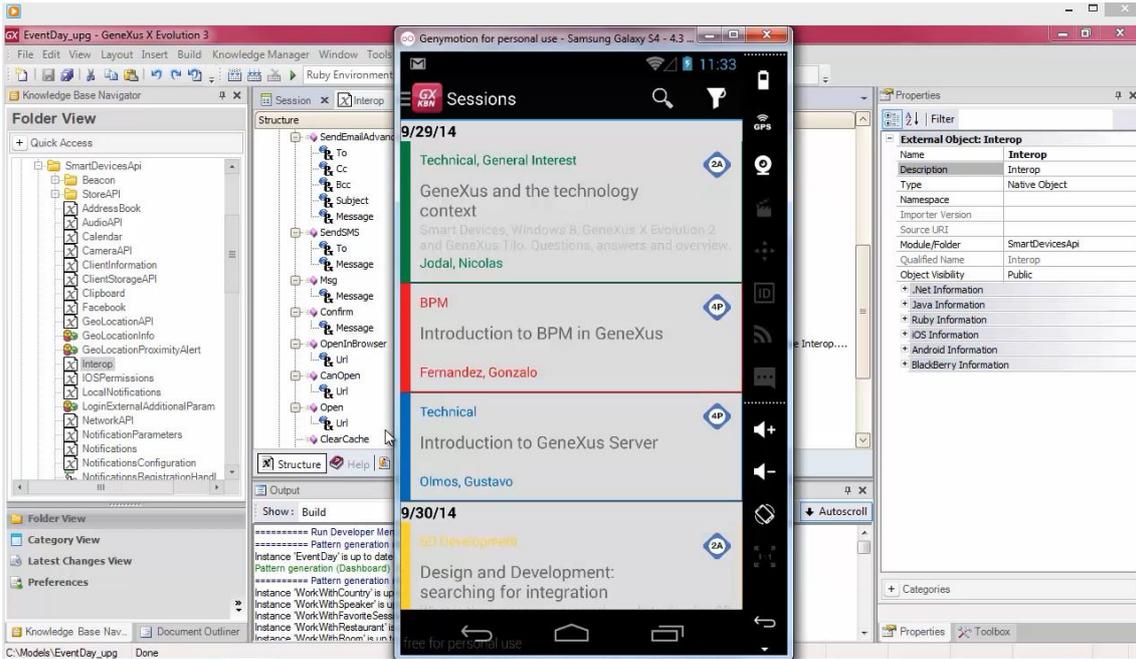


Así, algunas ofrecen la interoperabilidad con aplicaciones de mensajería que permiten

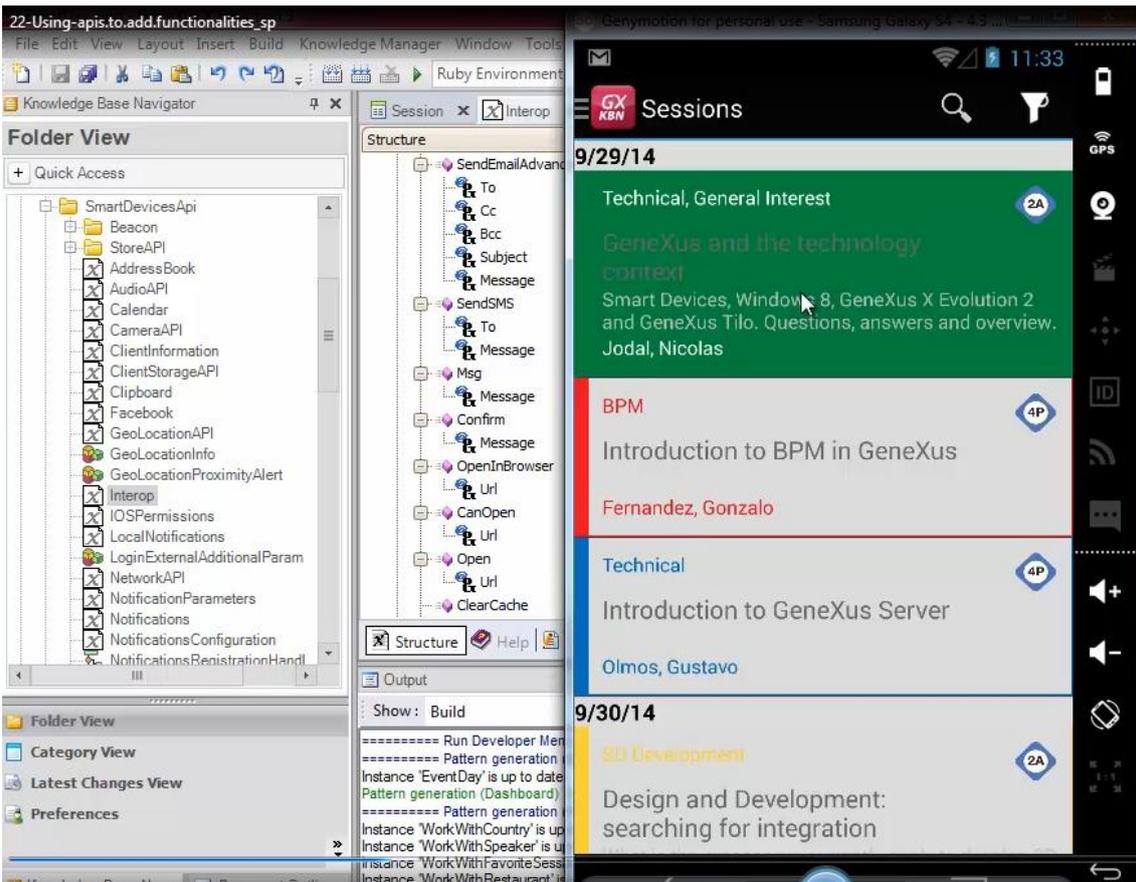


enviar mensajes, escanear códigos de barras, ejecutar videos o audios, enviar un email... un sms... desplegar mensajes al usuario o pedirle confirmación... entre otras cosas.

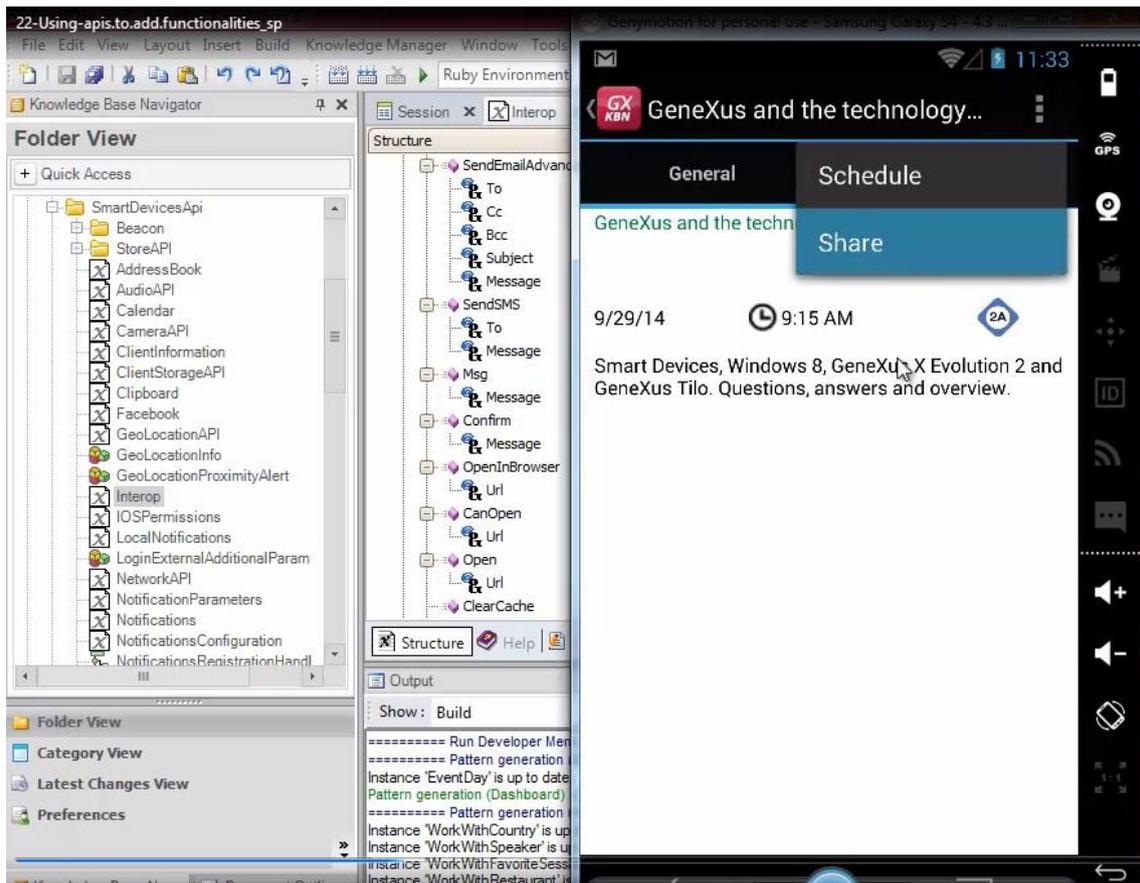
Por ejemplo



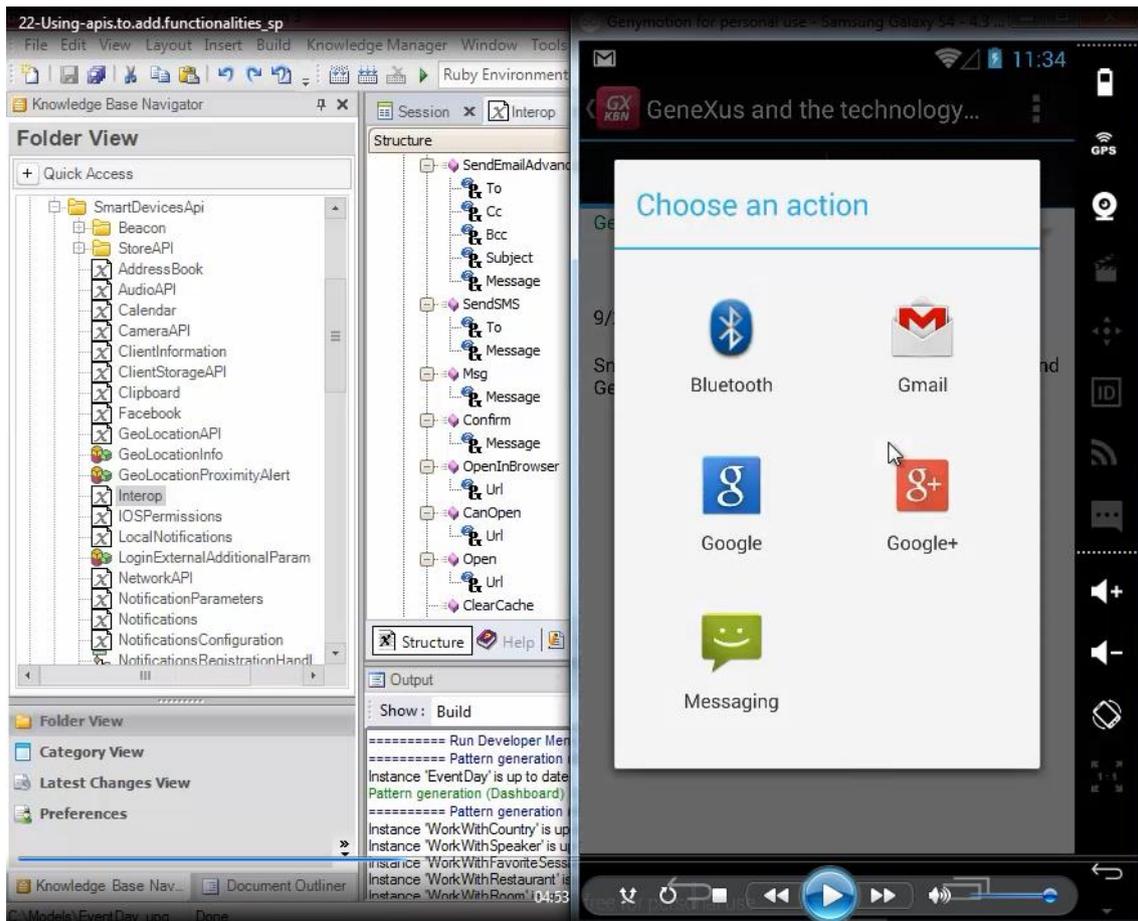
si vamos al detalle de una conferencia



vemos que podemos



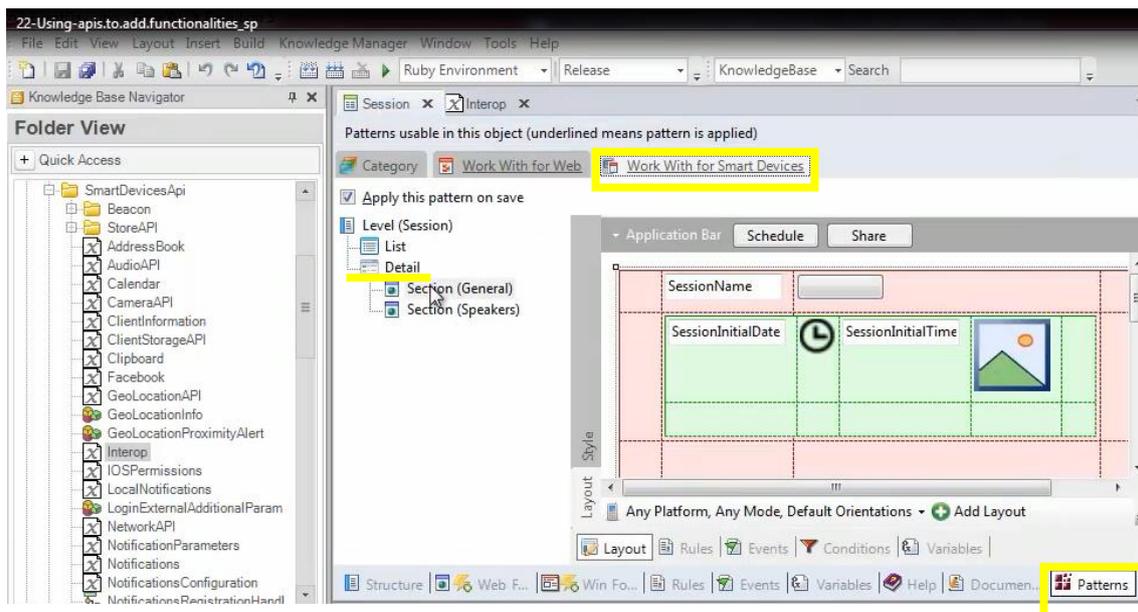
compartir parte de su información



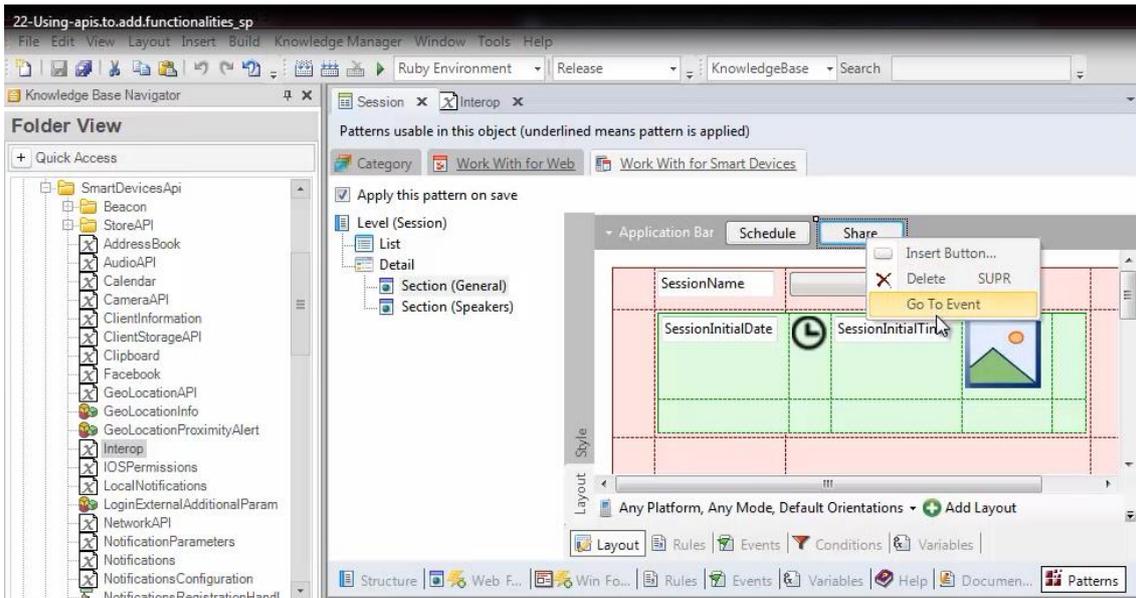
Observemos que se nos ofrece hacerlo a través de los programas que tenemos instalados en el dispositivo, que permiten enviar mensajes.

¿Cómo se implementó?

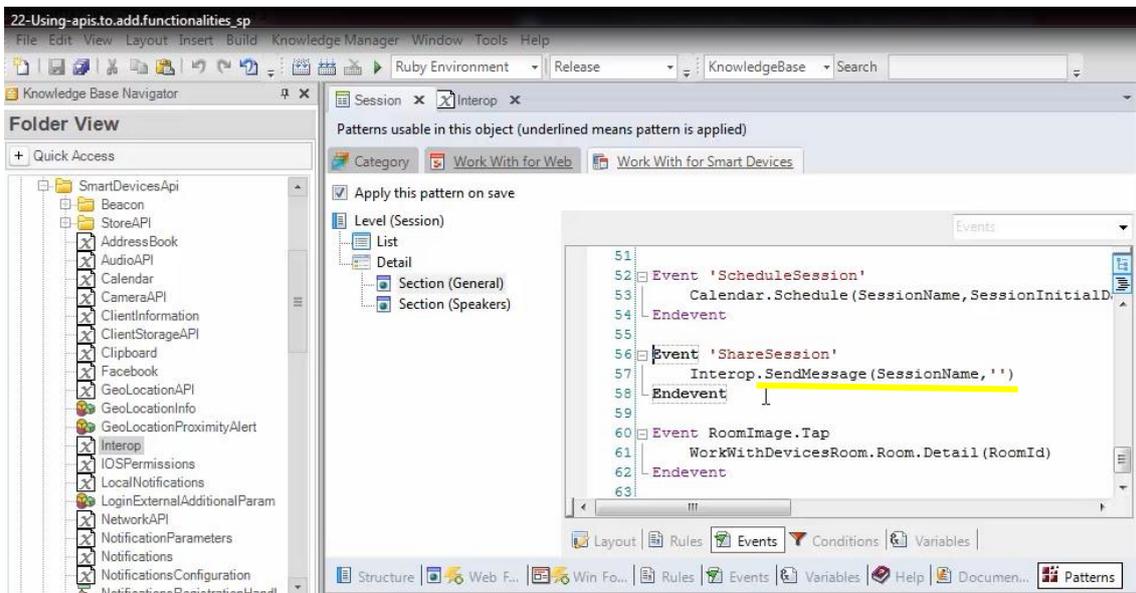
Si vamos al Detail del work with de Session



y buscamos el código del evento correspondiente al botón

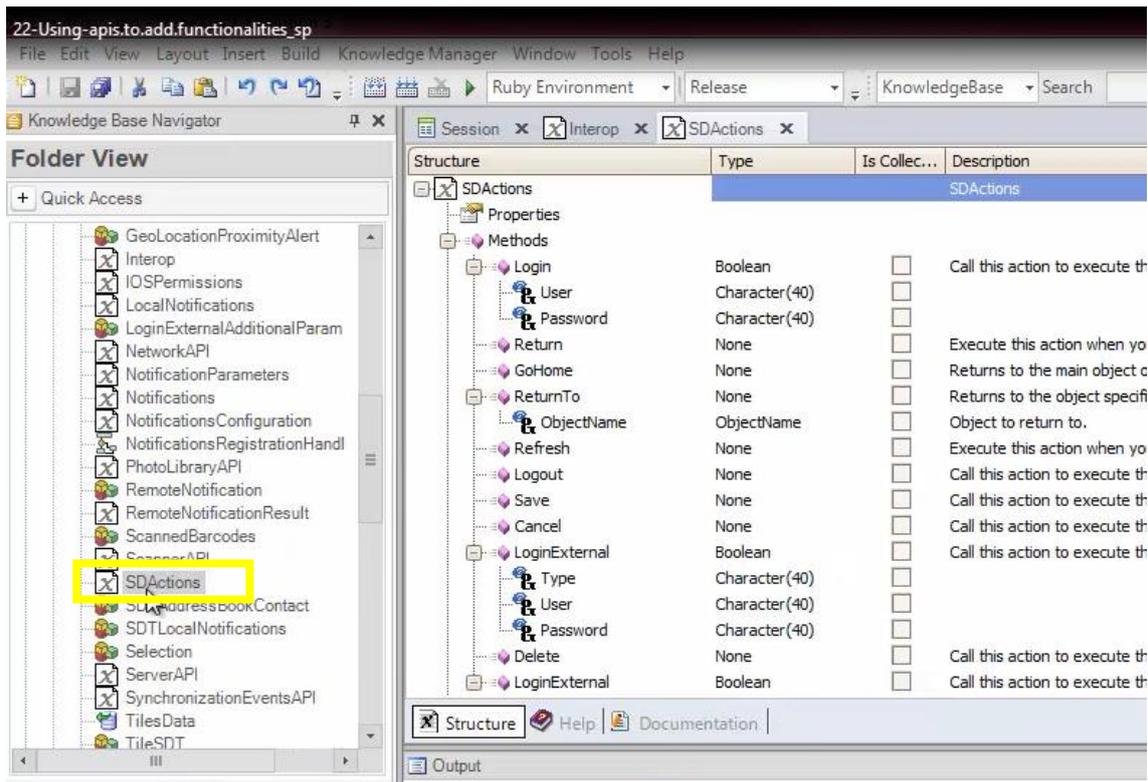


Share, en la Application Bar



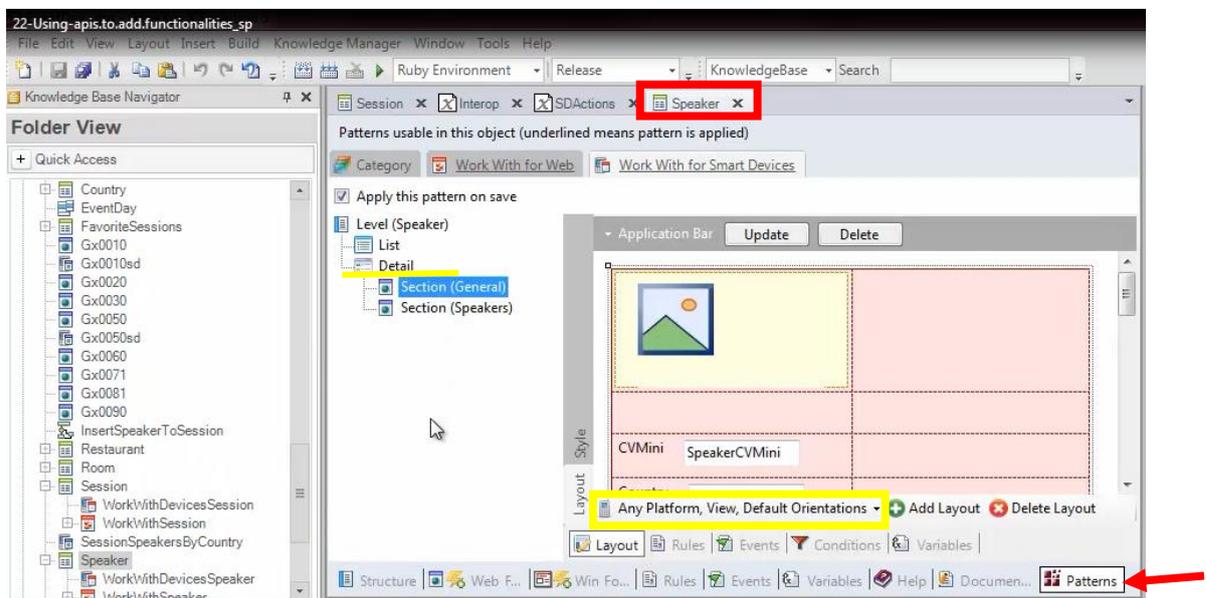
vemos que está utilizando la api Interop con su método SendMessage

Otra api que ya había aparecido anteriormente, era la SDActions

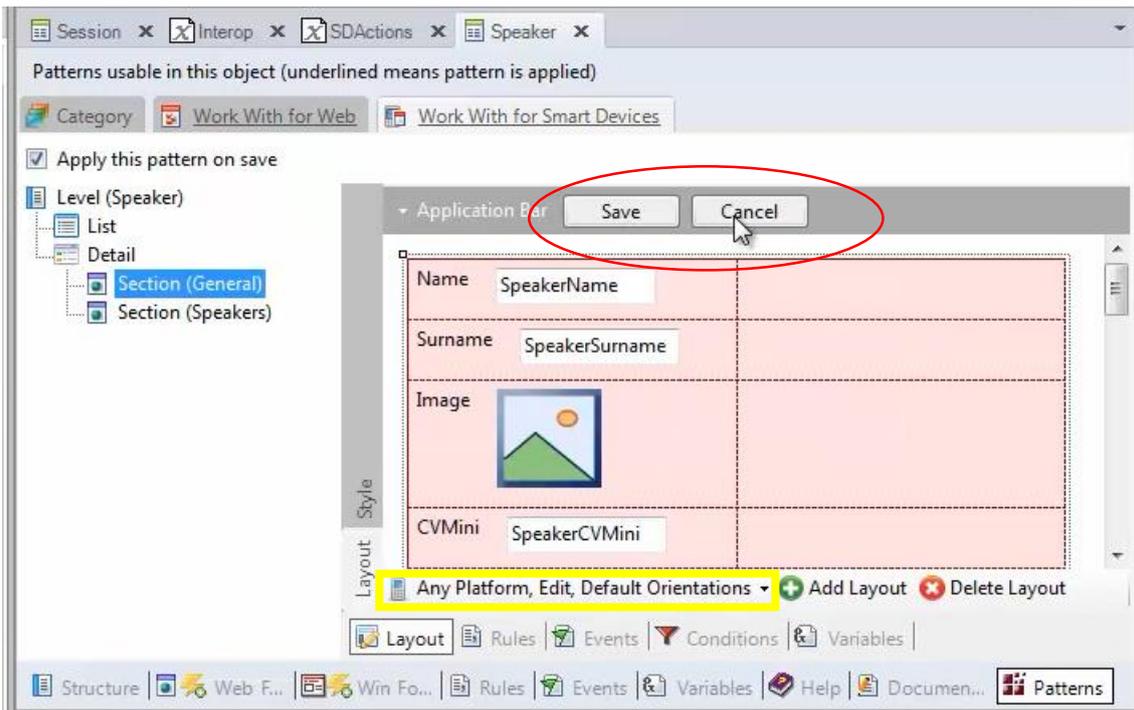


que permite entre otras cosas abstraer la funcionalidad de login, return, refresh, save, cancel... métodos estos últimos, que ya habíamos visto aparecer en los eventos que implementaba automáticamente el pattern work with a nivel del Detail.

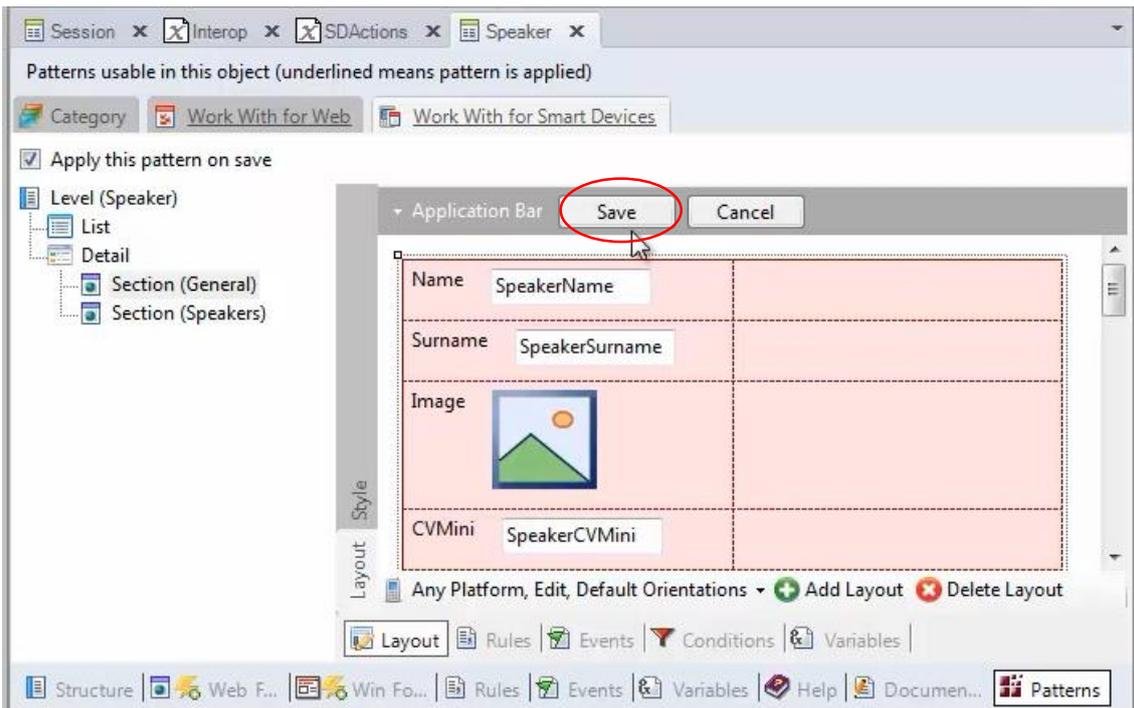
Por ejemplo, vamos a ver el Detail... la Section General de Speakers

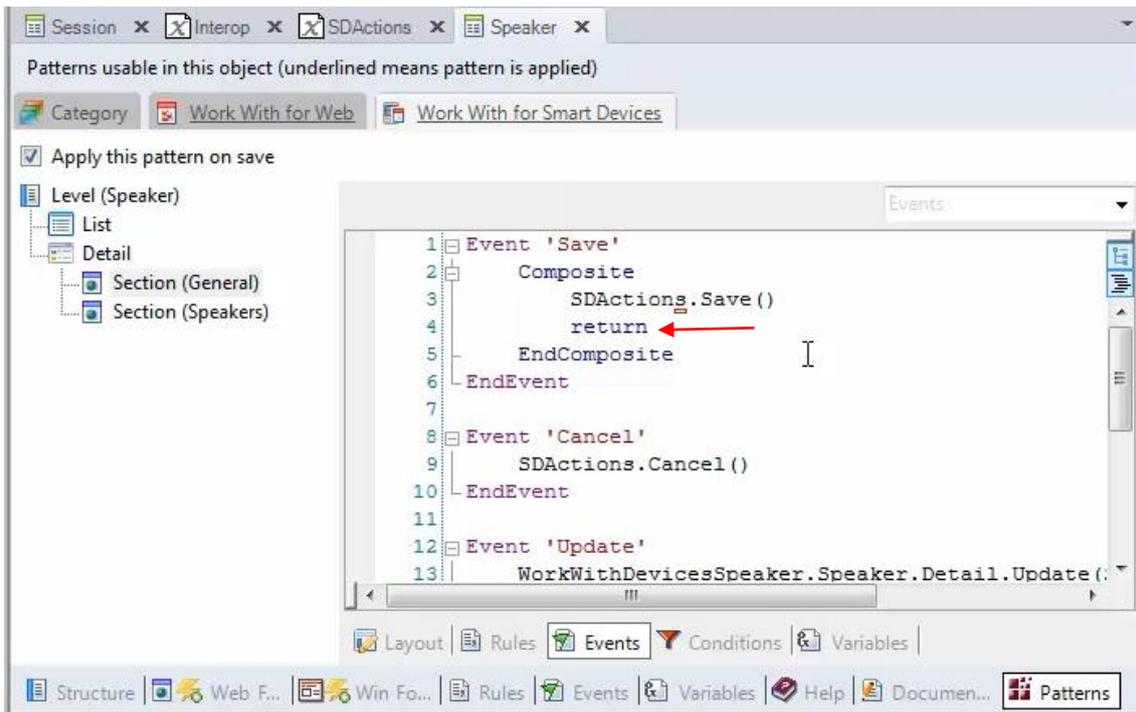


y vemos que tenemos los botones Update y Delete... para el modo View y para el Edit, los Save y Cancel



que si vamos a observar su programación

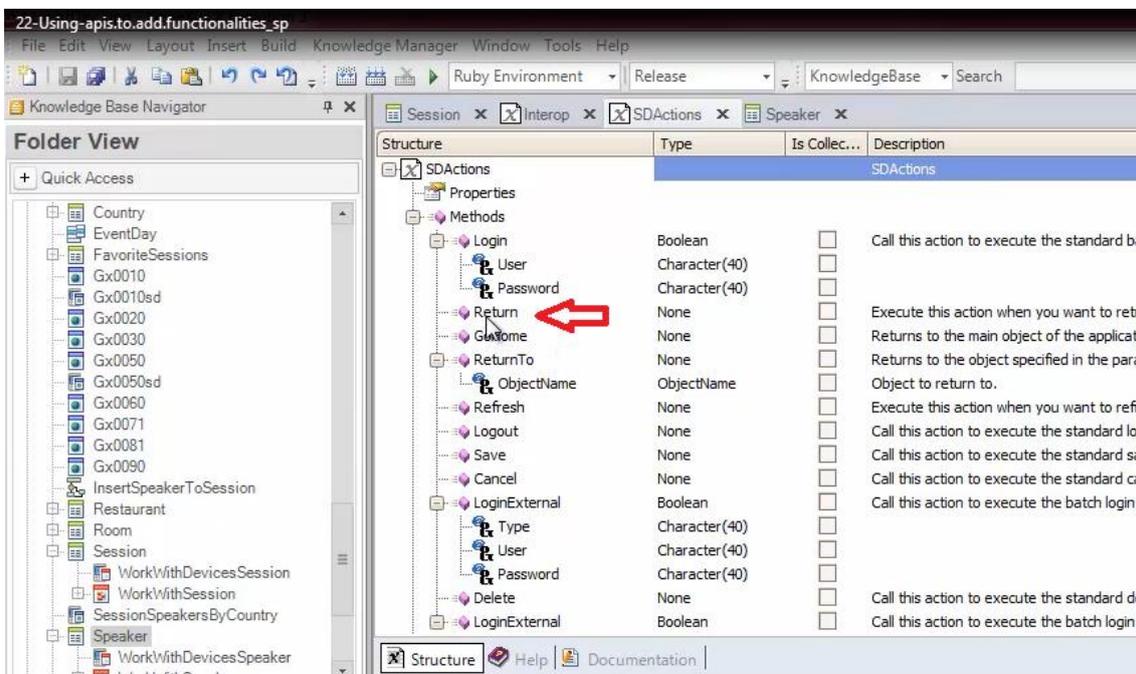




están utilizando la api que veíamos.

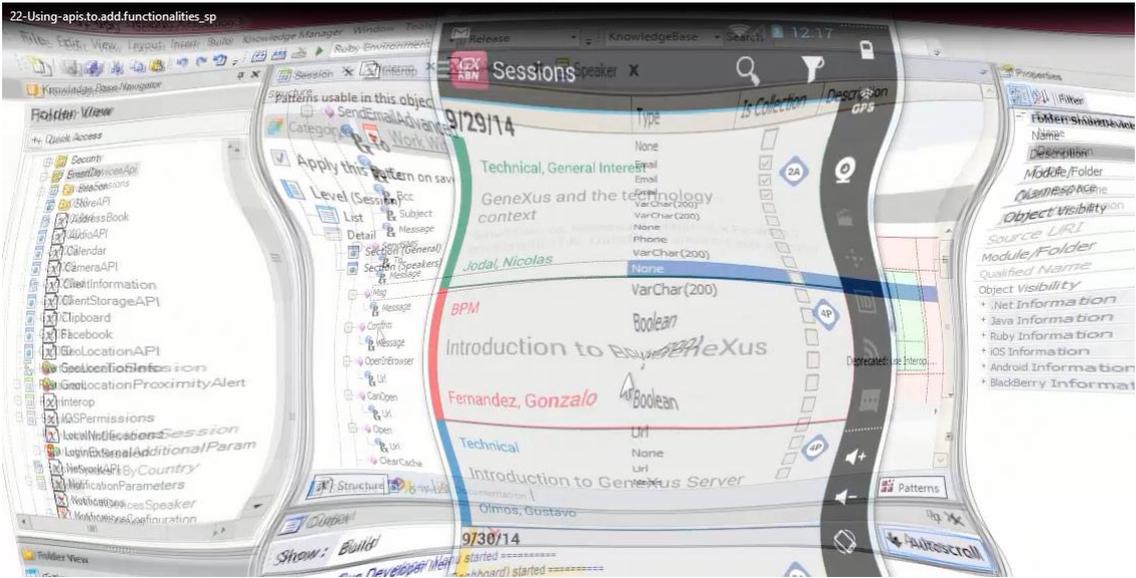
Este evento Save, corresponde al caso en el que estamos editando la información de un orador y estamos queriendo grabar esa información en la base de datos.

El método Save de esta api, lo que hace es encapsular la invocación al business component correspondiente. Luego está retornando al llamador. Este **return** corresponde al método: Return de la api

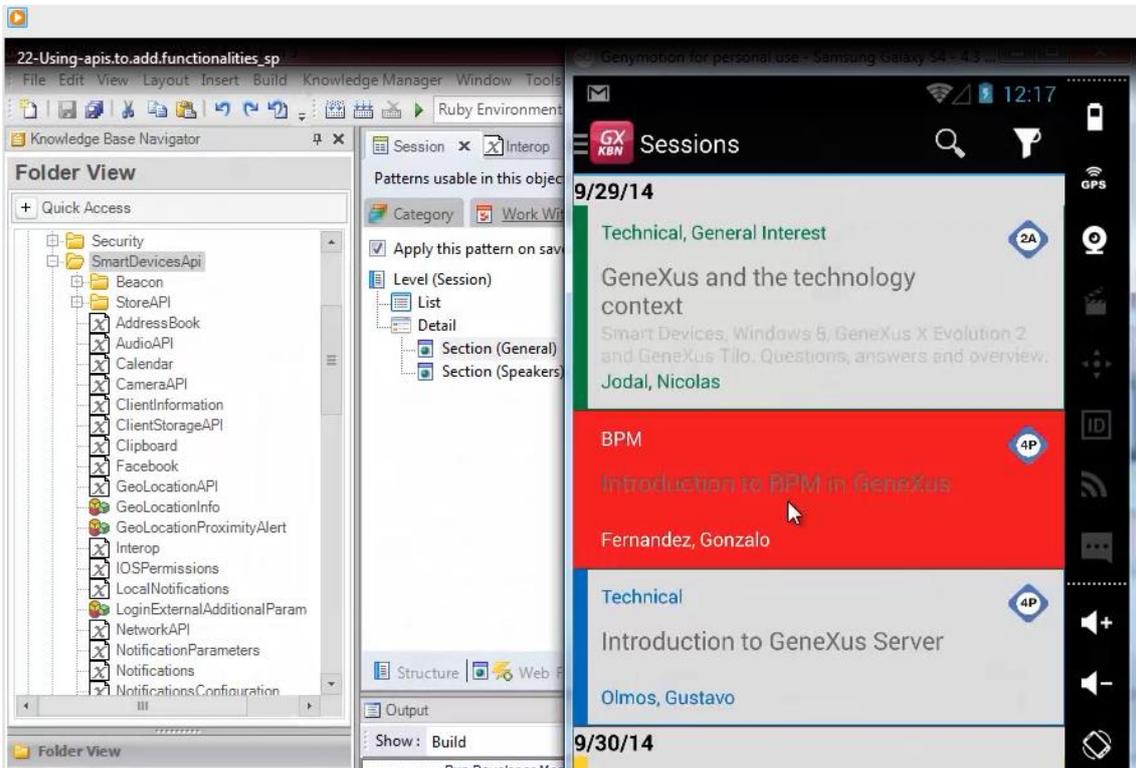


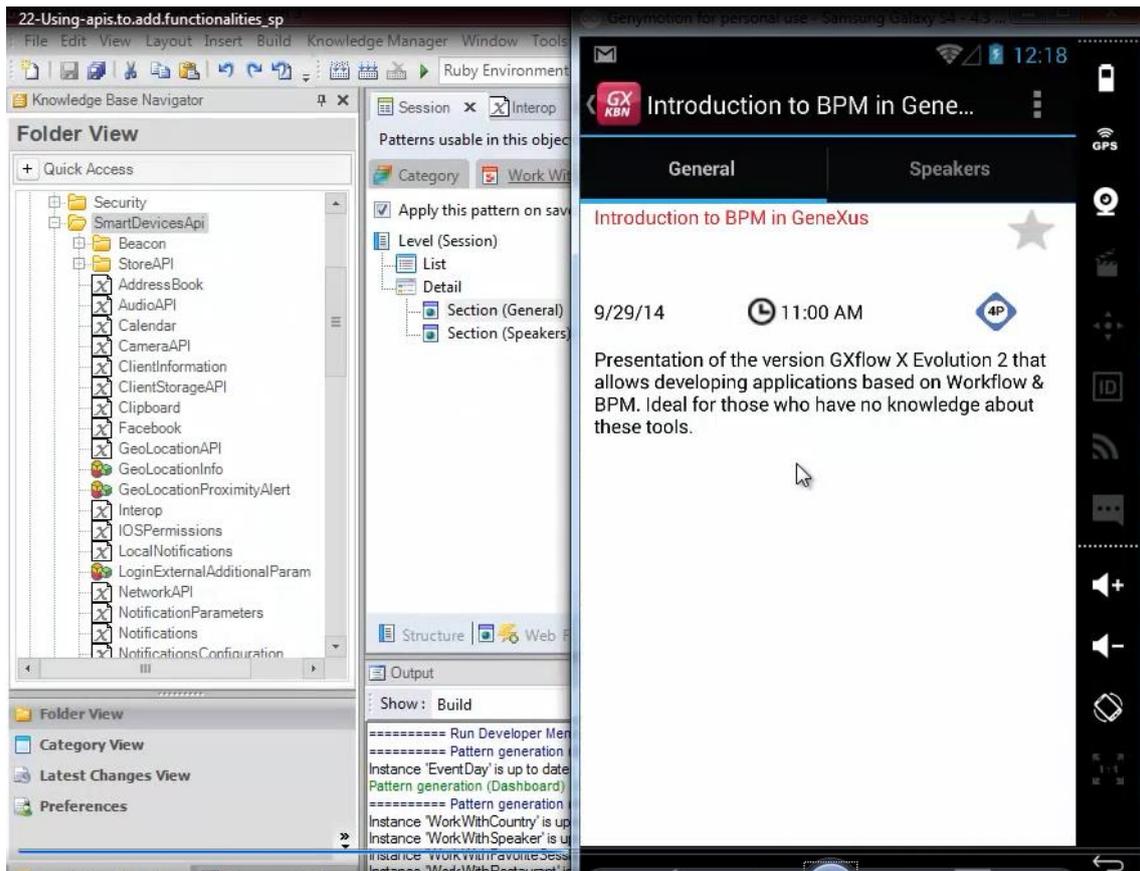
Aunque no se haya especificado como prefijo, el nombre de la api.

Algunos métodos de esta api y de la Interop, como el msg o el confirm, admiten omitir entonces el nombre de la api debido a la frecuencia con la que se utilizan.

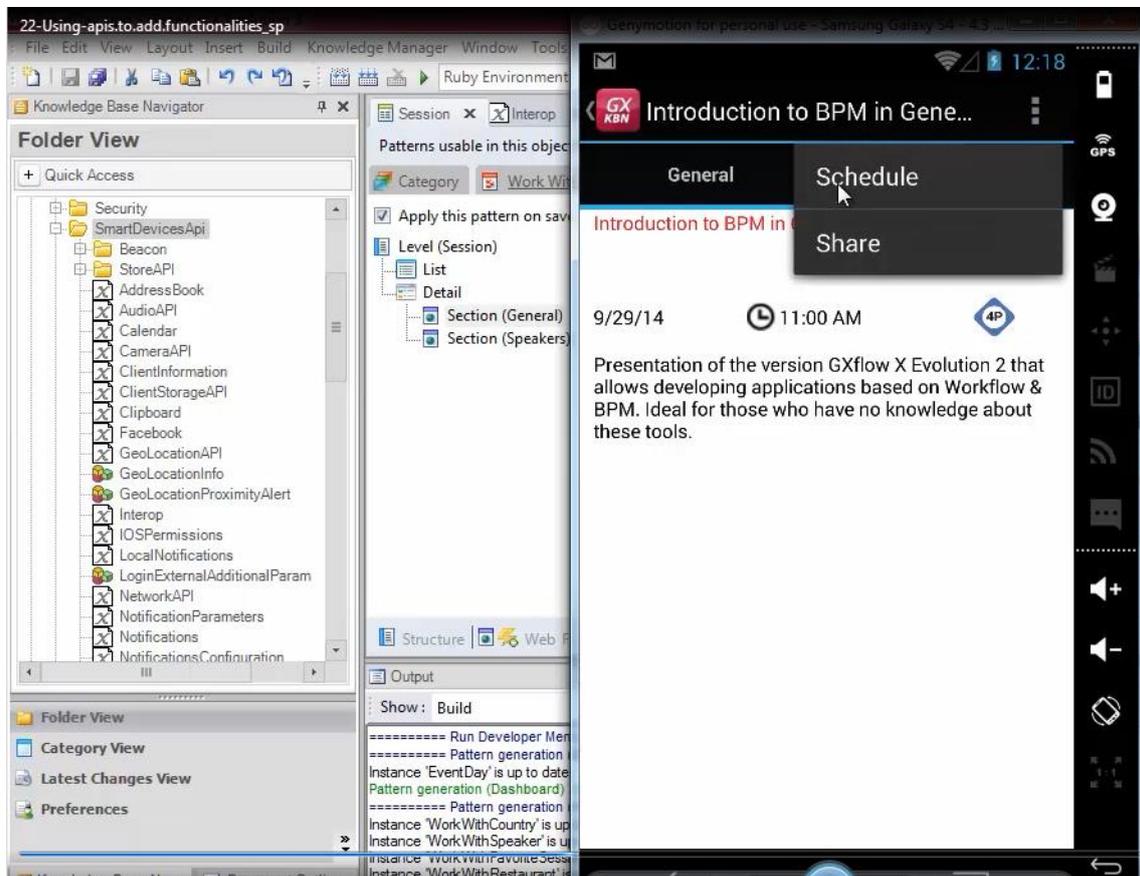


Volvamos a visualizar el detalle de una conferencia

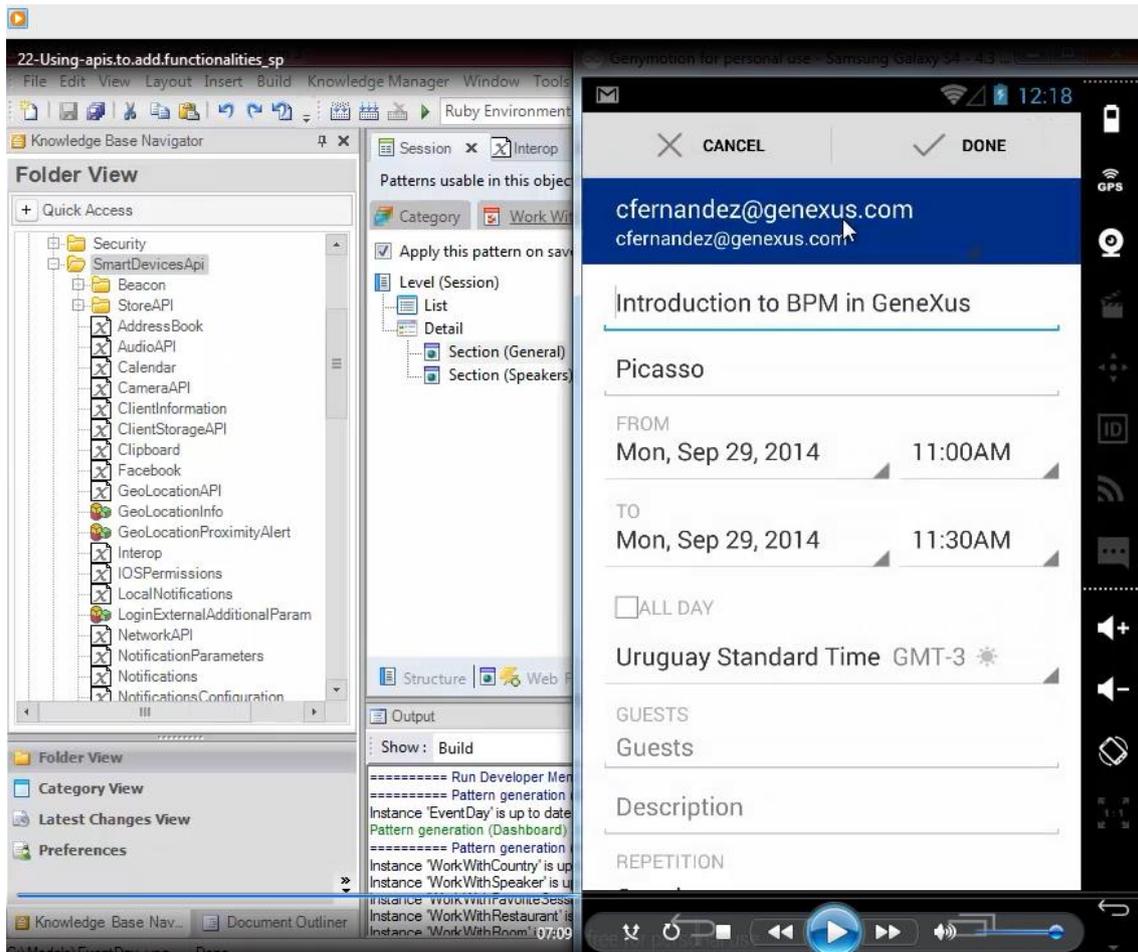




Tenemos también la posibilidad



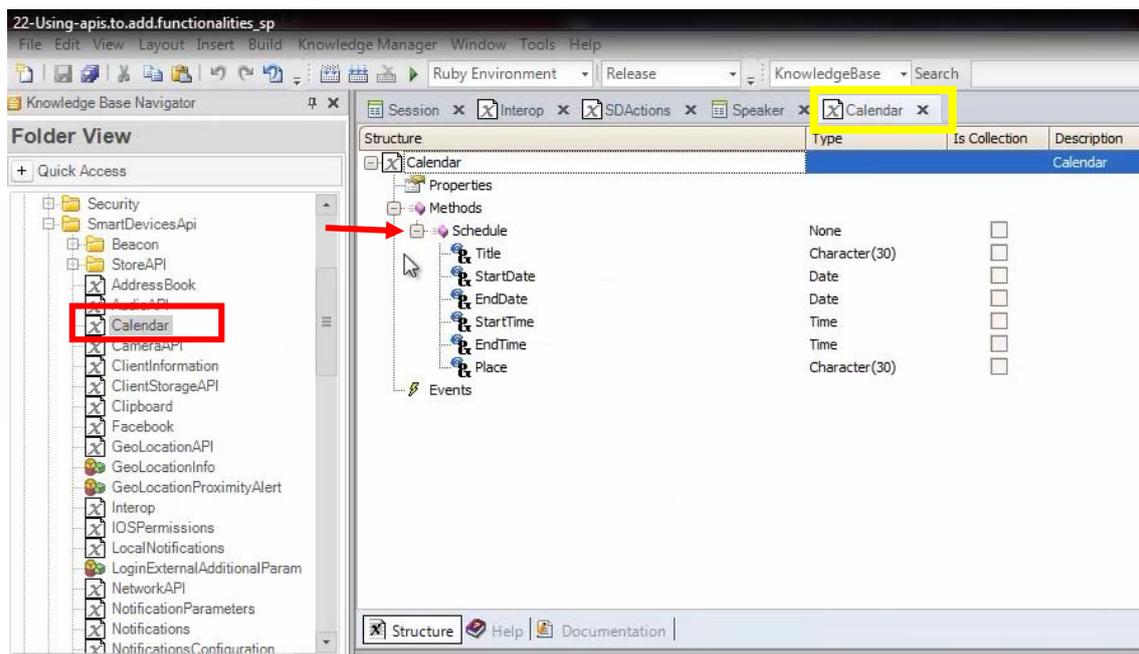
de agendarla en el calendario de nuestro dispositivo



Aquí vemos el título o nombre de la conferencia.. la sala donde se va a realizar.. y las fechas.

¿Cómo logramos esta integración?

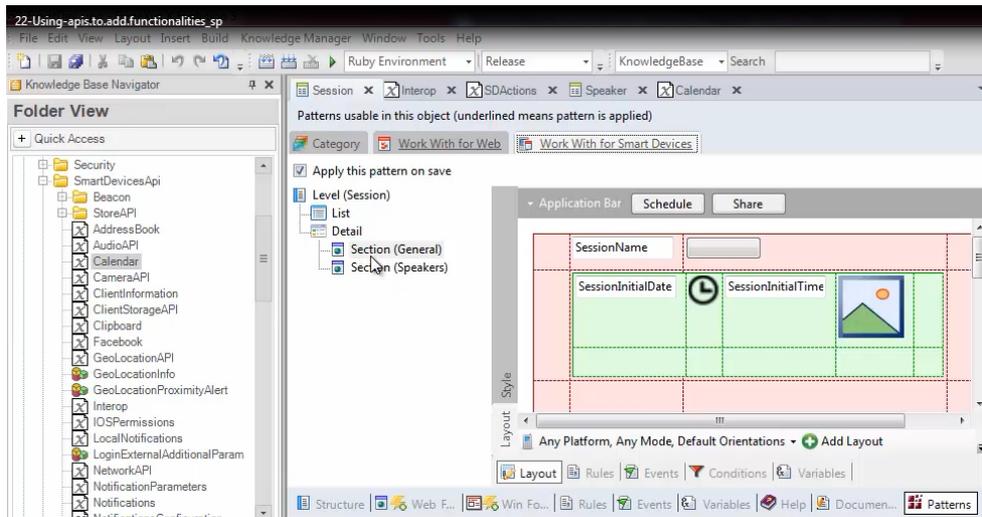
A través de la api: calendar



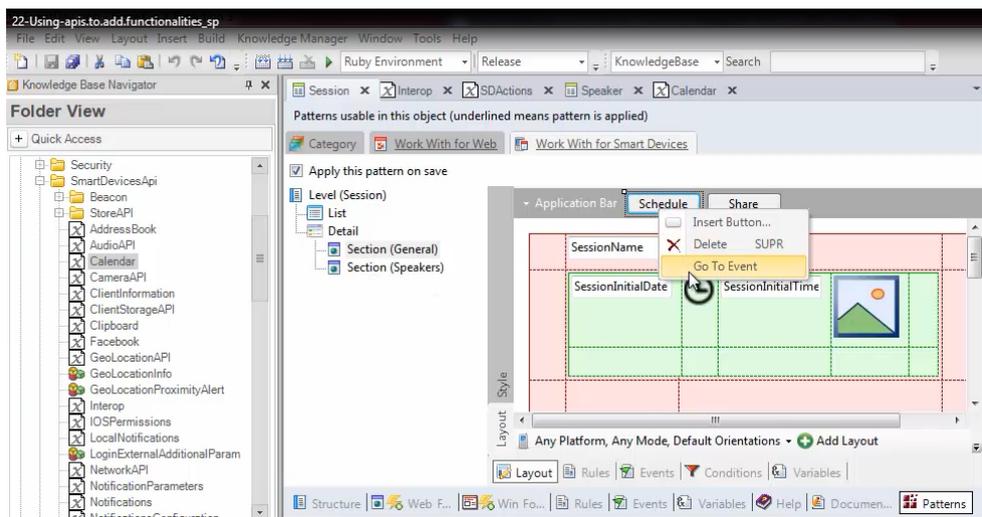
Video filmado con GeneXus X Evolution 3

que ofrece el método: Schedule ... al que hay que pasarle estos parámetros que aquí aparecen

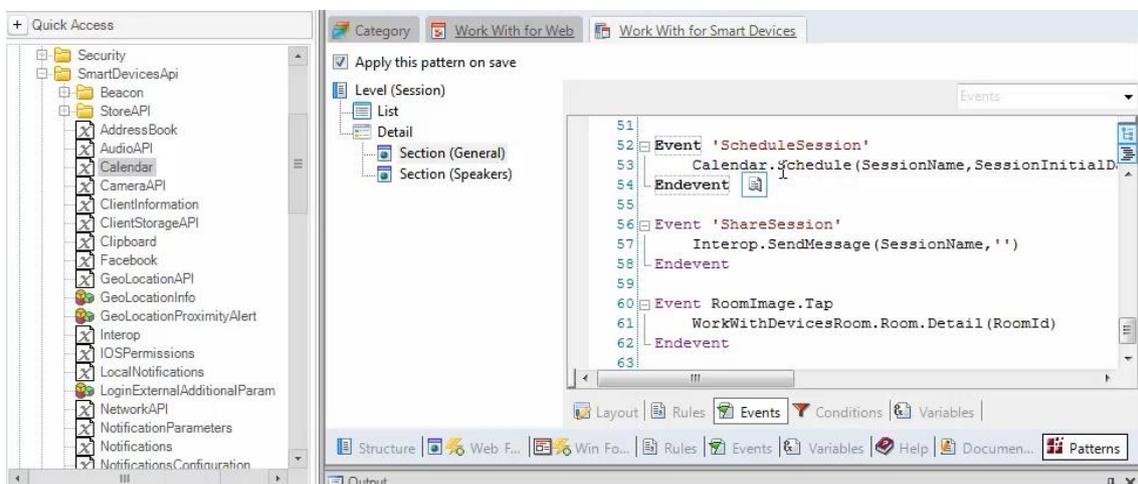
Si vamos al Detail de Session



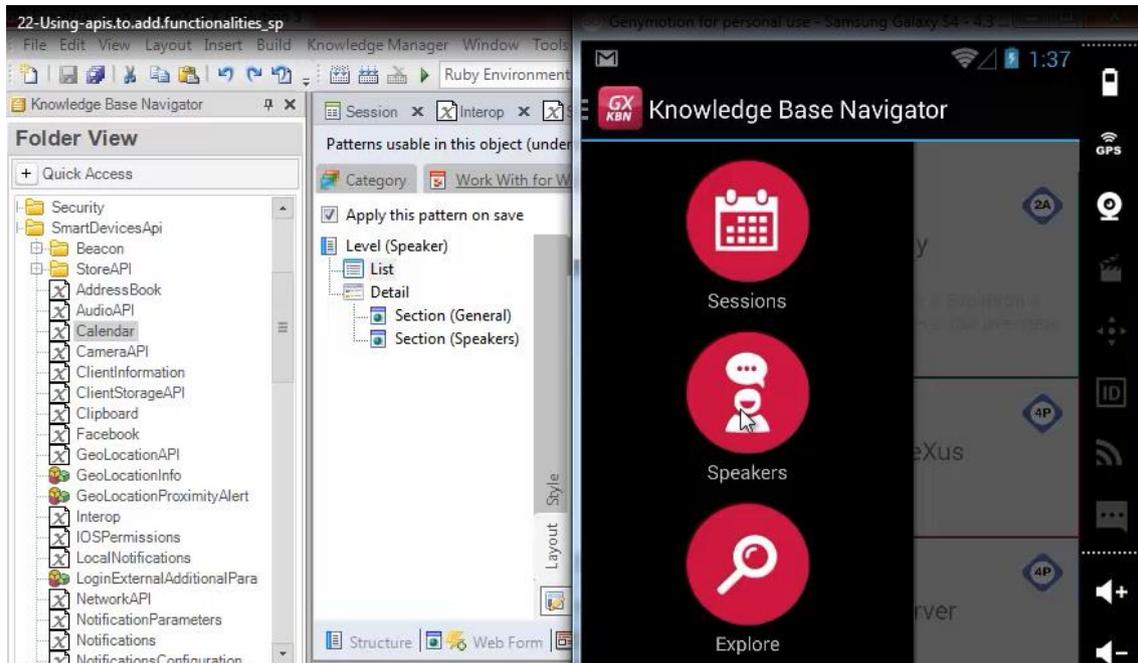
vemos el botón correspondiente



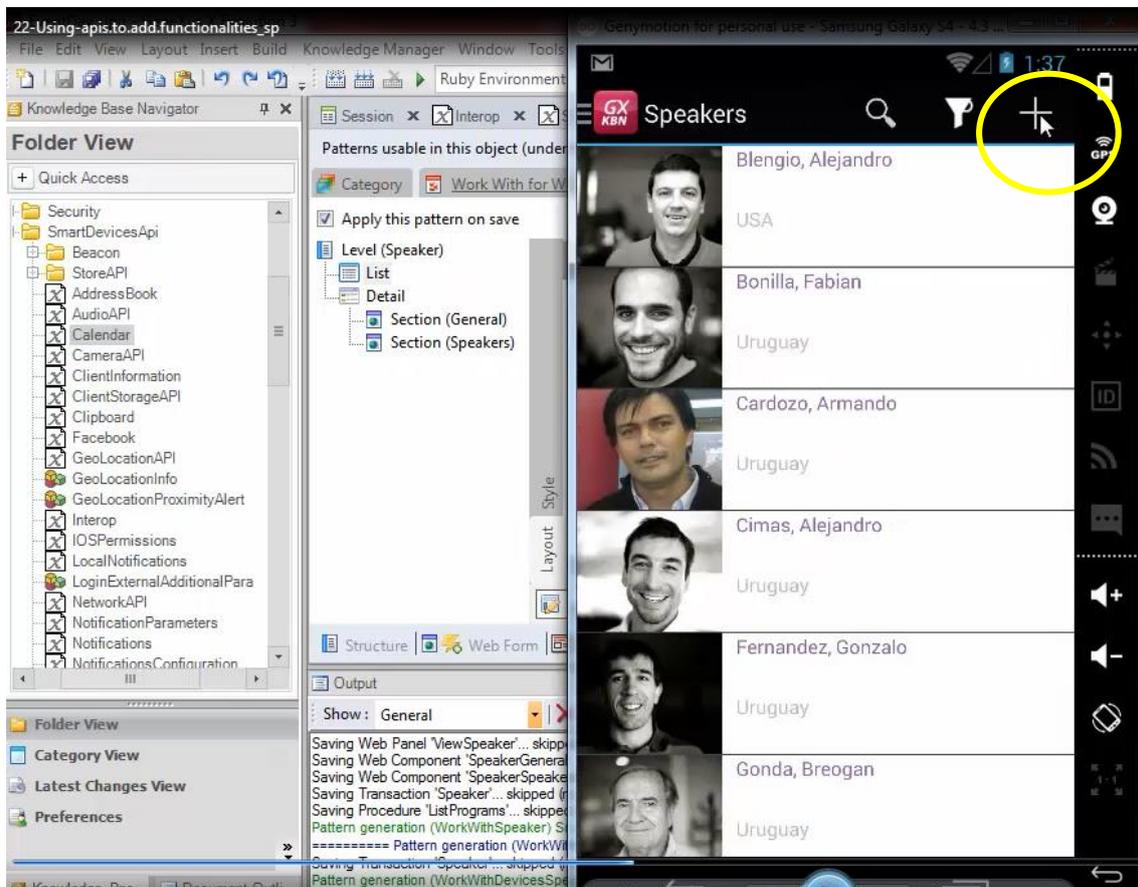
y vamos al código del evento



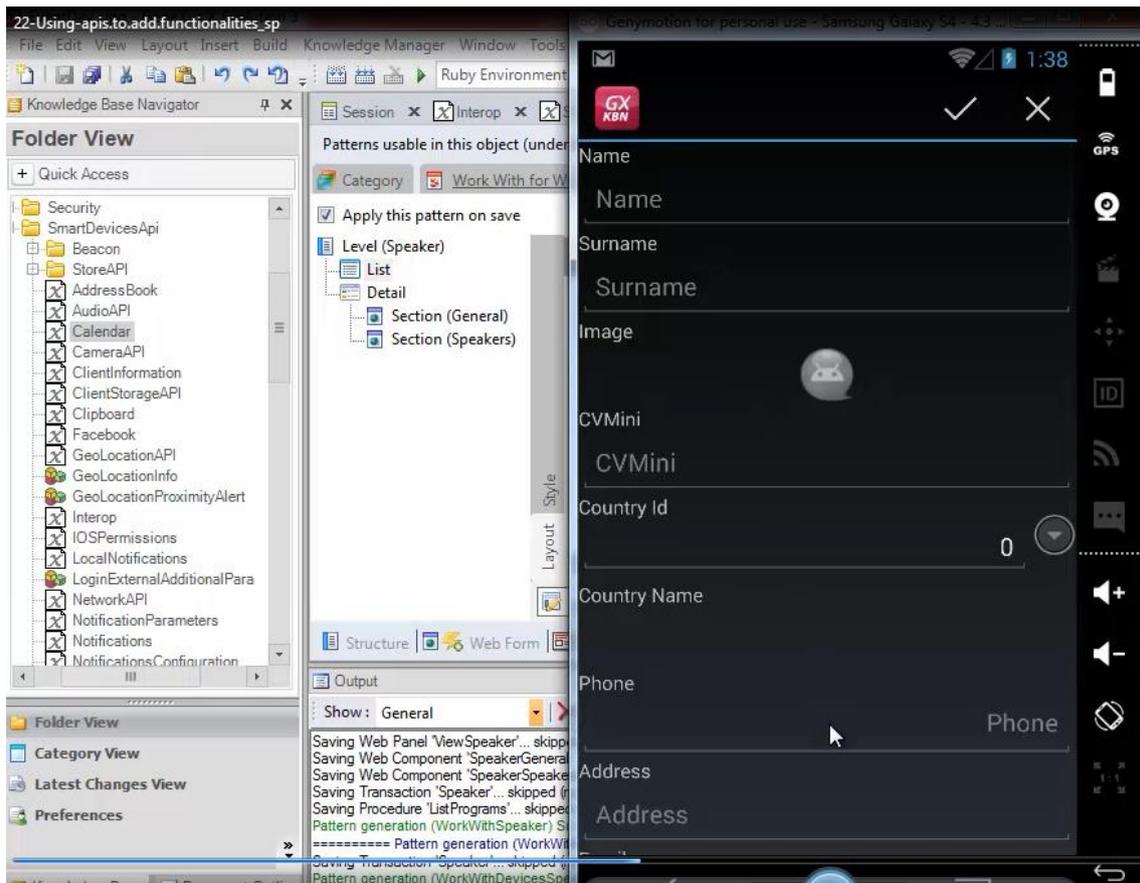
y efectivamente estamos llamando entonces a la api: Calendar, con el método Schedule (pasándole la información: SessionName, SessionInitialDate, FinalDate, InitialTime, EndTime y el nombre de la sala).



Supongamos ahora que desde el List de Speakers, queremos

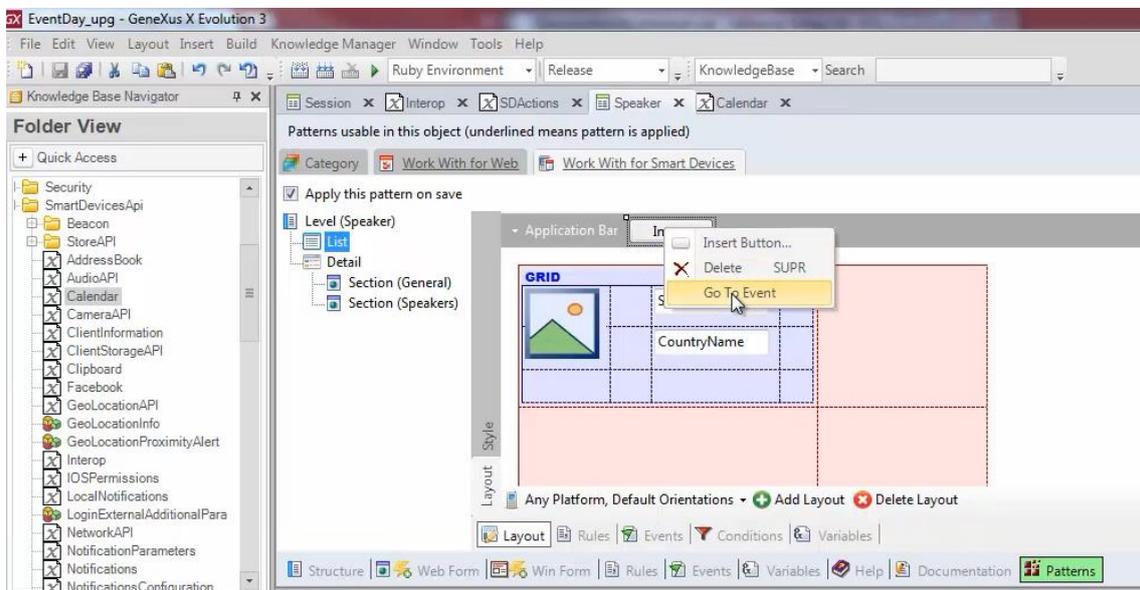


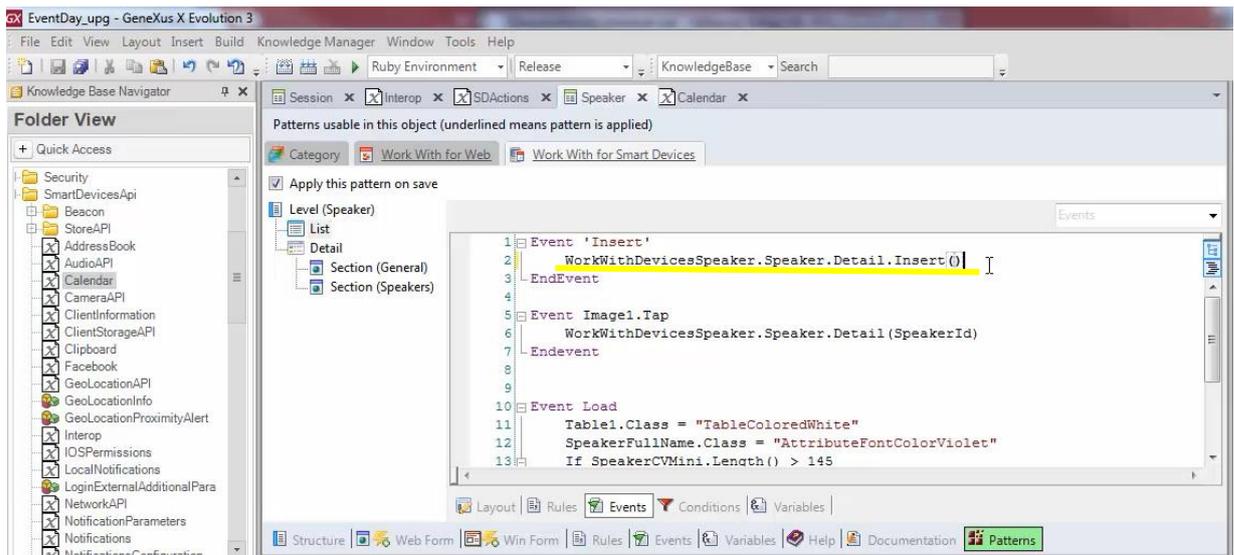
que tras ingresar un nuevo orador en la base de datos



se lo agregue también como contacto en la libreta de direcciones del dispositivo.

Para ello

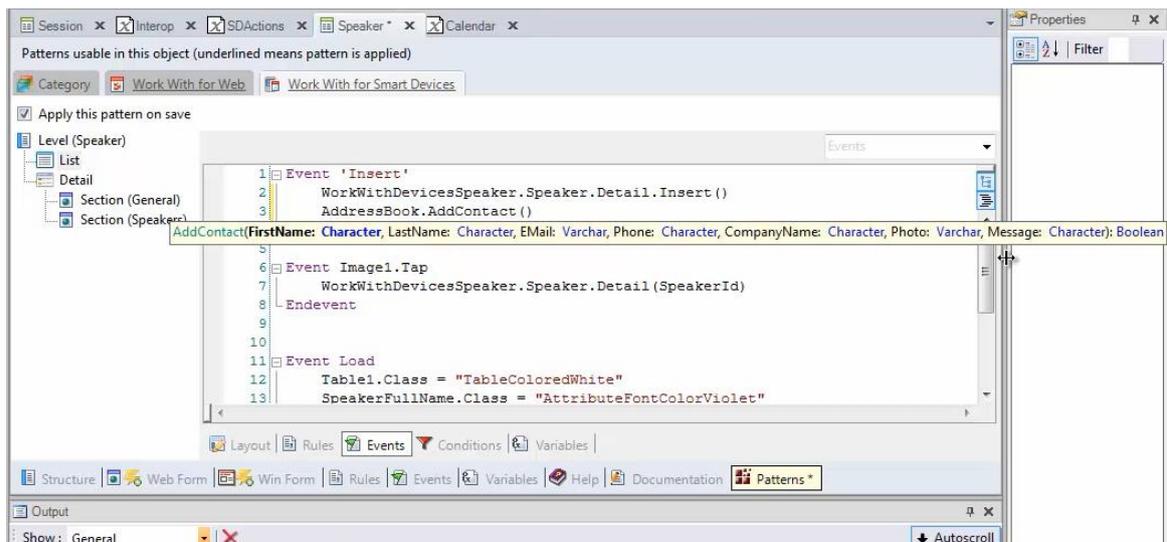




después de invocar al Detail en modo Insert, debemos invocar a la api AddressBook

Método: AddContact

Al que debemos de pasarle esta serie de parámetros

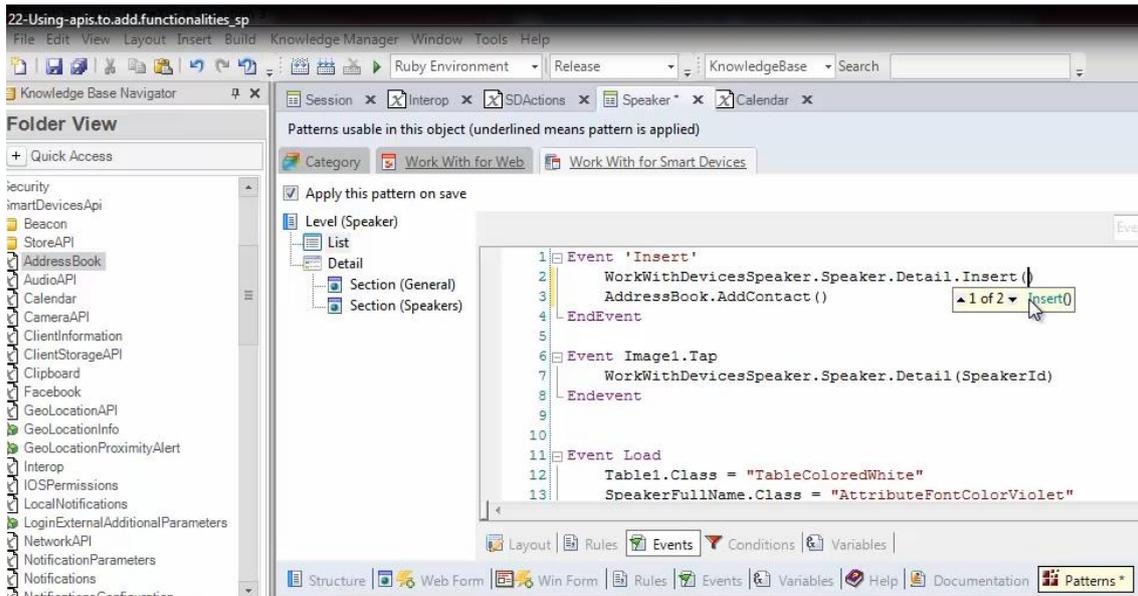


y en este orden

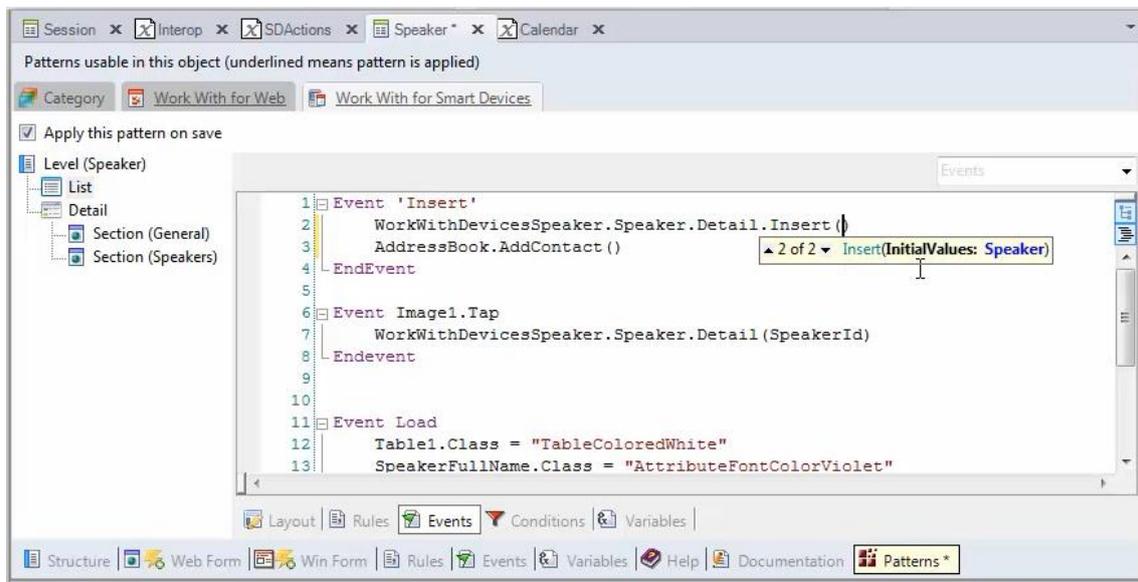
Estos parámetros van a corresponder a datos del orador insertado por el usuario, en este panel.

¿Cómo recuperamos esos datos para poder enviarlos como parámetro a este método?

Recordemos que el insert ofrecía 2 opciones. La primera era no pasar ningún parámetro.



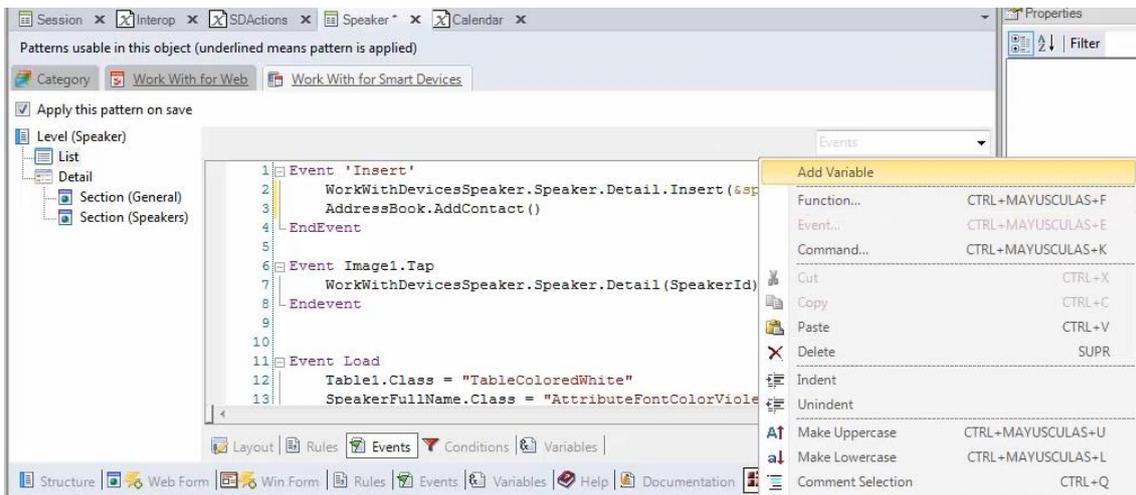
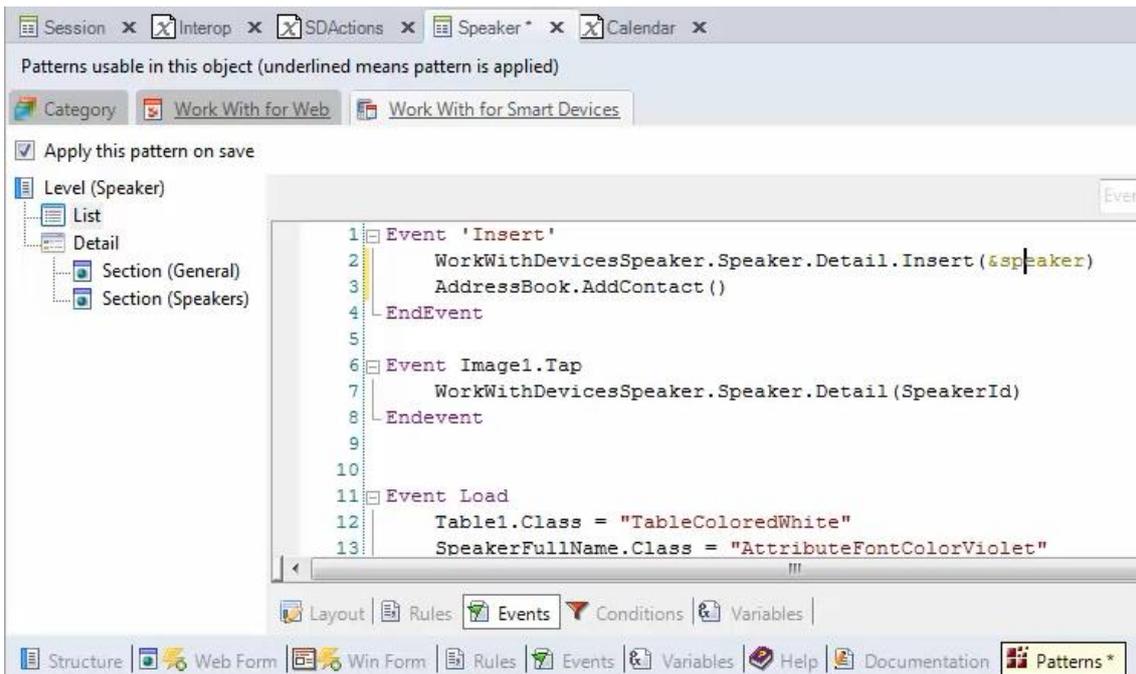
Y la segunda



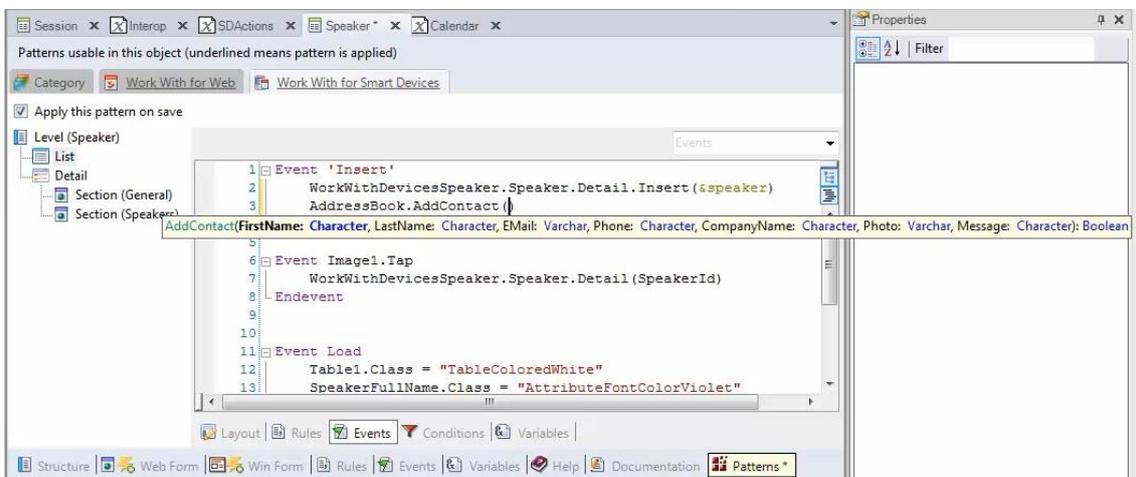
era pasar un único parámetro del tipo de datos el business component asociado al nivel en el que se está queriendo hacer la inserción.

Este business component nos permitía inicializar valores en la pantalla a la que estamos invocando, y a la vez, al finalizar la inserción, va a venir cargado con todos los valores insertados; y esa es la manera que vamos a tener para poder recuperar esos datos para enviárselos al método AddContact.

Por tanto vamos a definir una variable &speaker



que por defecto, como se llama igual que el business component, va a quedar inicializada con ese tipo de datos y esa variable es la que vamos a utilizar, para en el método AddContact



ir agregando los parámetros:

El primero -> FirstName, va a corresponder a → &Speaker – punto - Name

El segundo, LastName, va a corresponder a → &Speaker – punto - SpeakerSurname

El tercero nos está advirtiéndolo que es el email → &Speaker – punto – SpeakerEmail

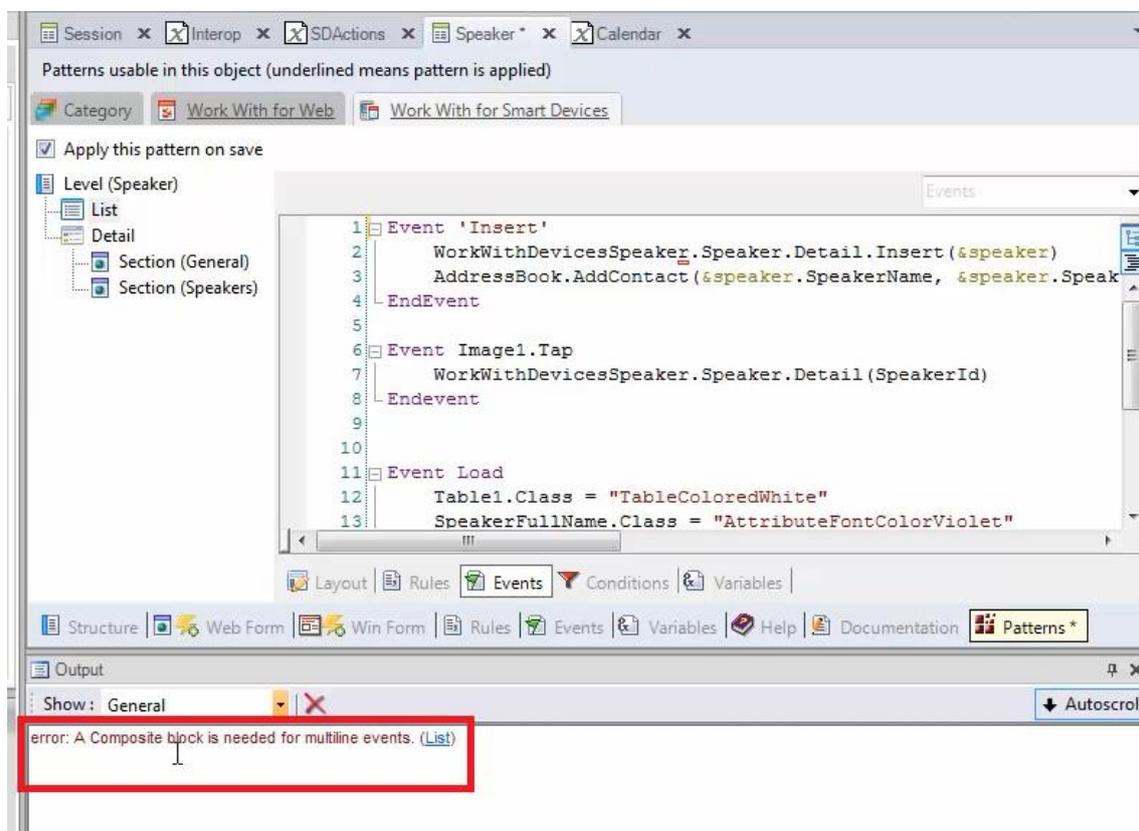
Luego el teléfono..

El que le sigue.. CompanyName.. y como no tenemos ese dato, vamos a pasarle la cadena vacía, puesto que hay que pasar el parámetro aunque no lo tengamos

La foto, será &Speaker.SpeakerImage

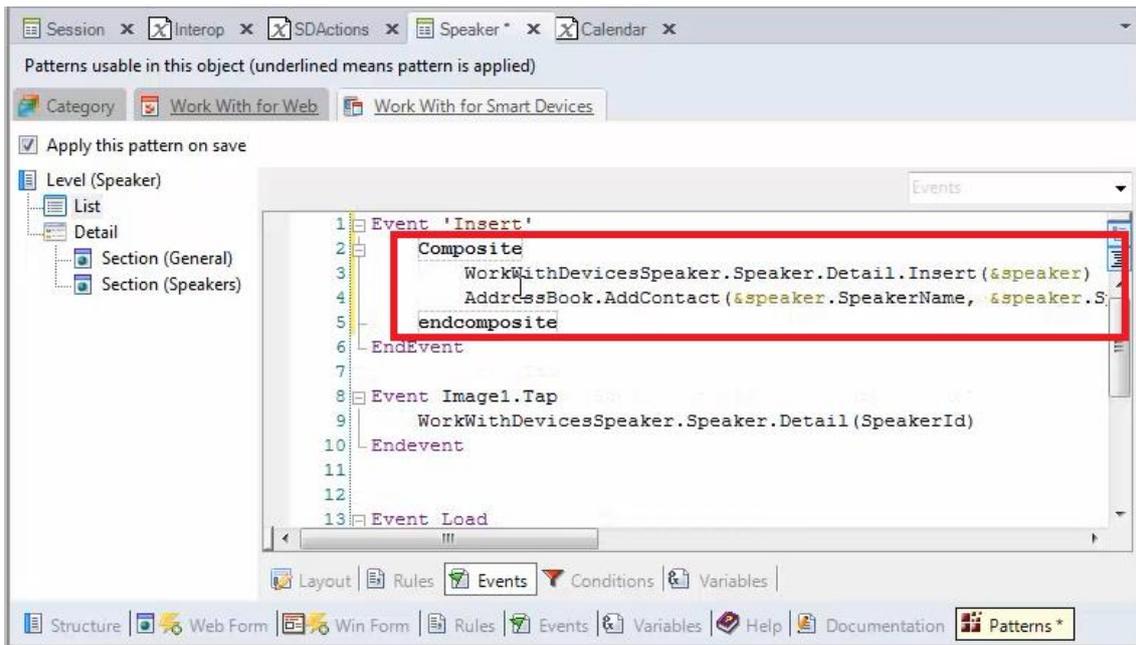
Y el mensaje: cadena vacía

Si ahora intentamos grabar, advirtiéndonos



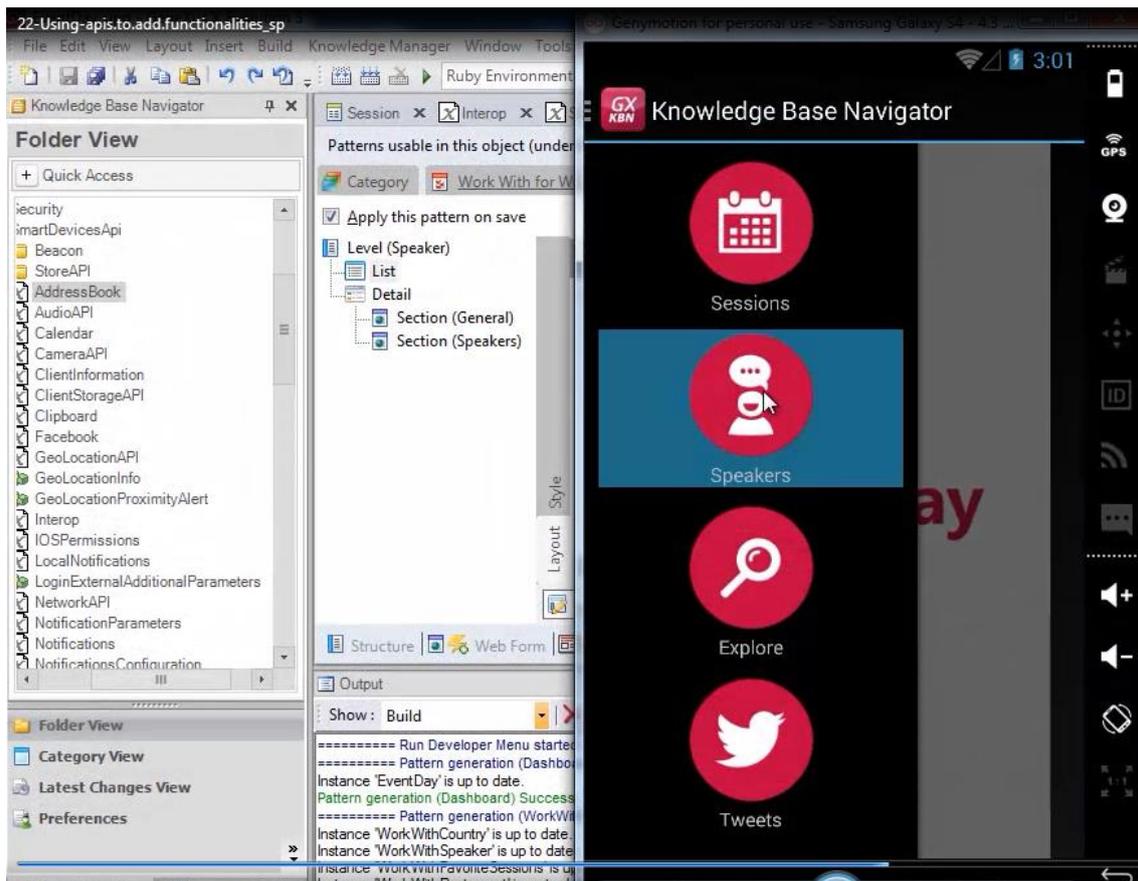
que nos está faltando el bloque Composit.

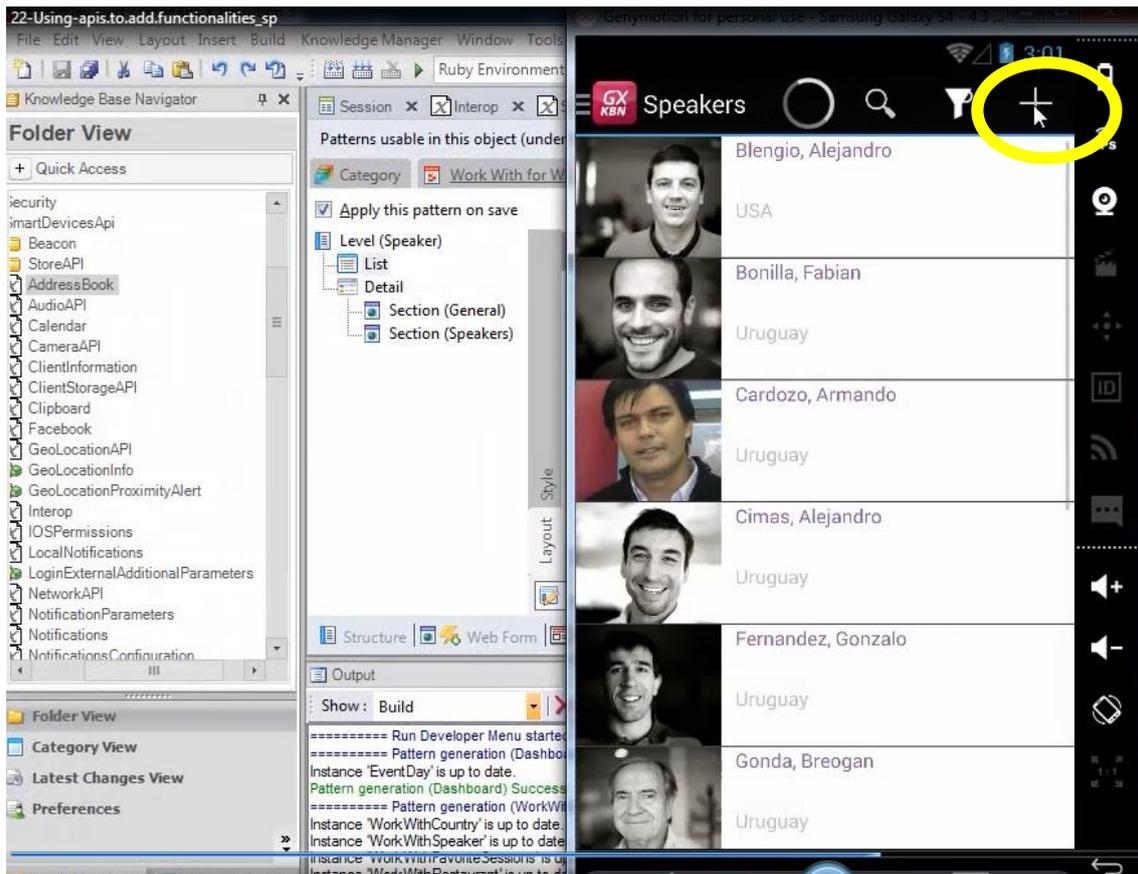
Veremos este comando en otro video, pero ya podemos advertir que toda vez que tengamos 2 invocaciones dentro de un mismo evento, debemos encapsularlas dentro de ese comando.



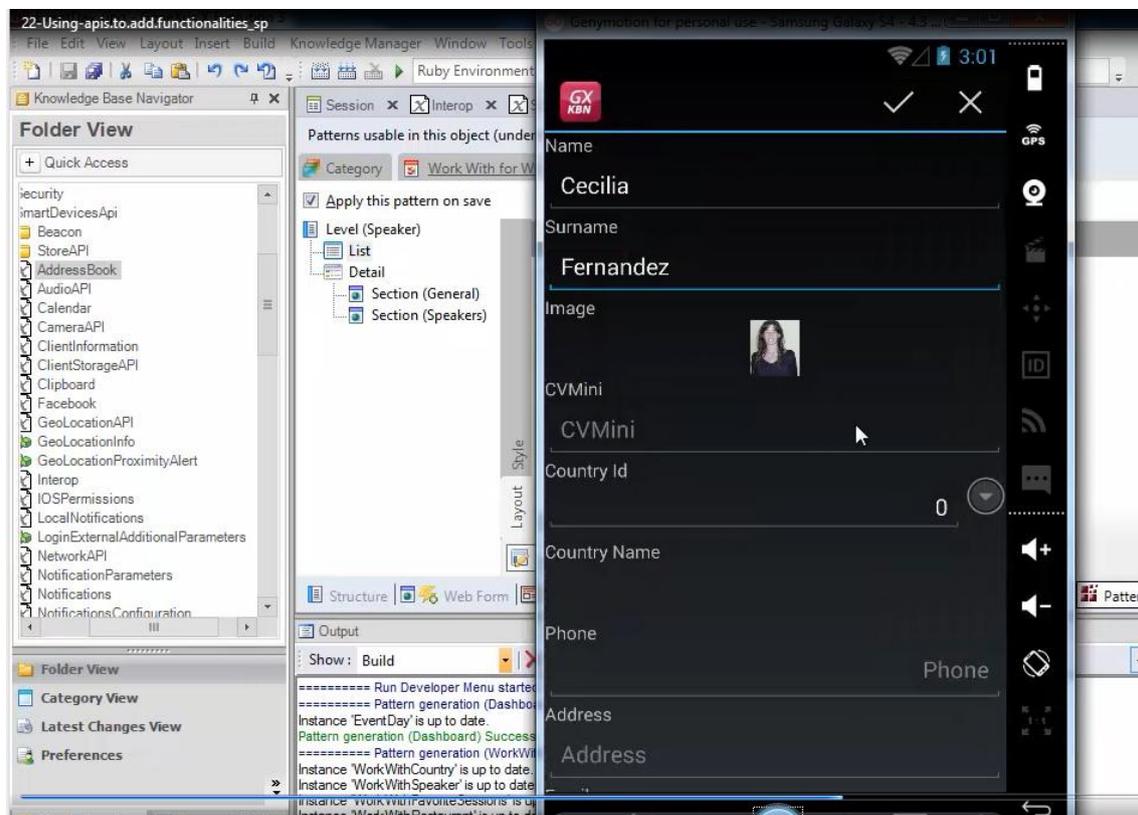
Si ahora grabamos, no vamos a tener problema.

Hagamos ahora un F5 y probemos.





Agregamos un nuevo orador..



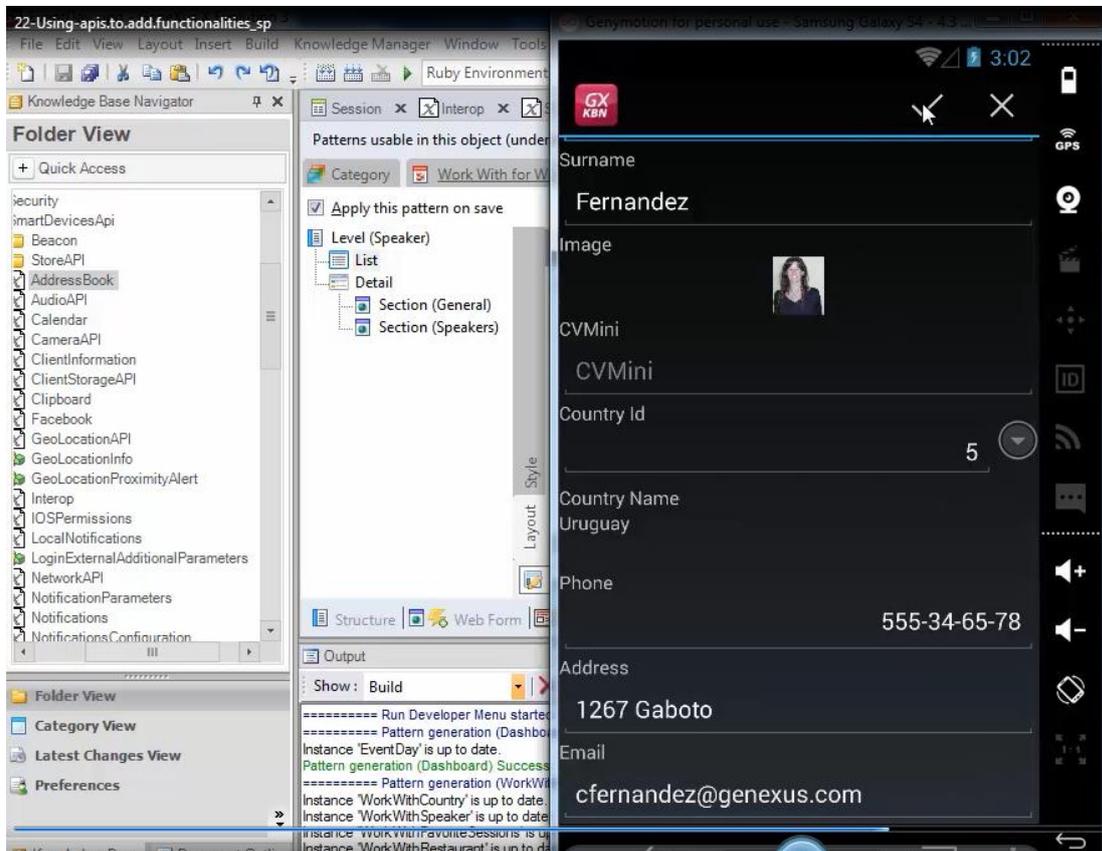
seleccionamos la imagen de la galería de imágenes que tenemos..

vamos a dejar vacío el currículum vitae

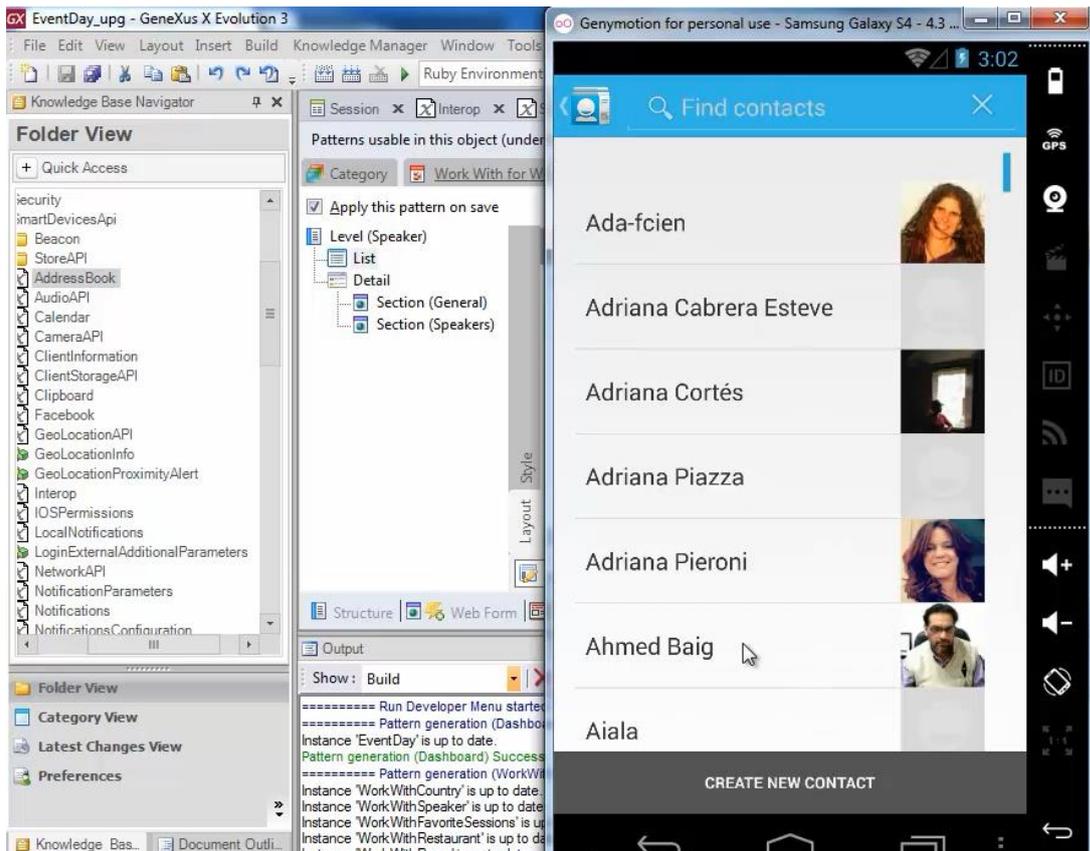
agregamos el país: Uruguay

teléfono.. dirección.. email..

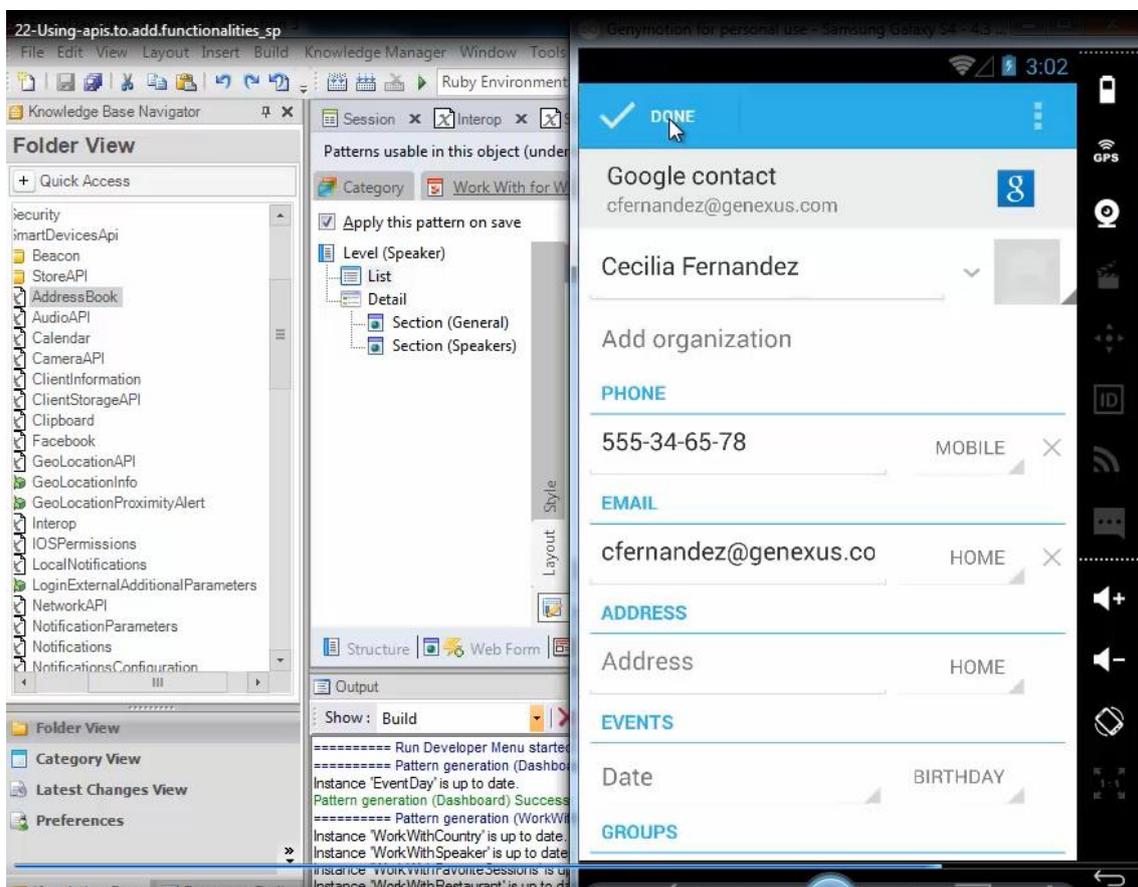
y grabamos



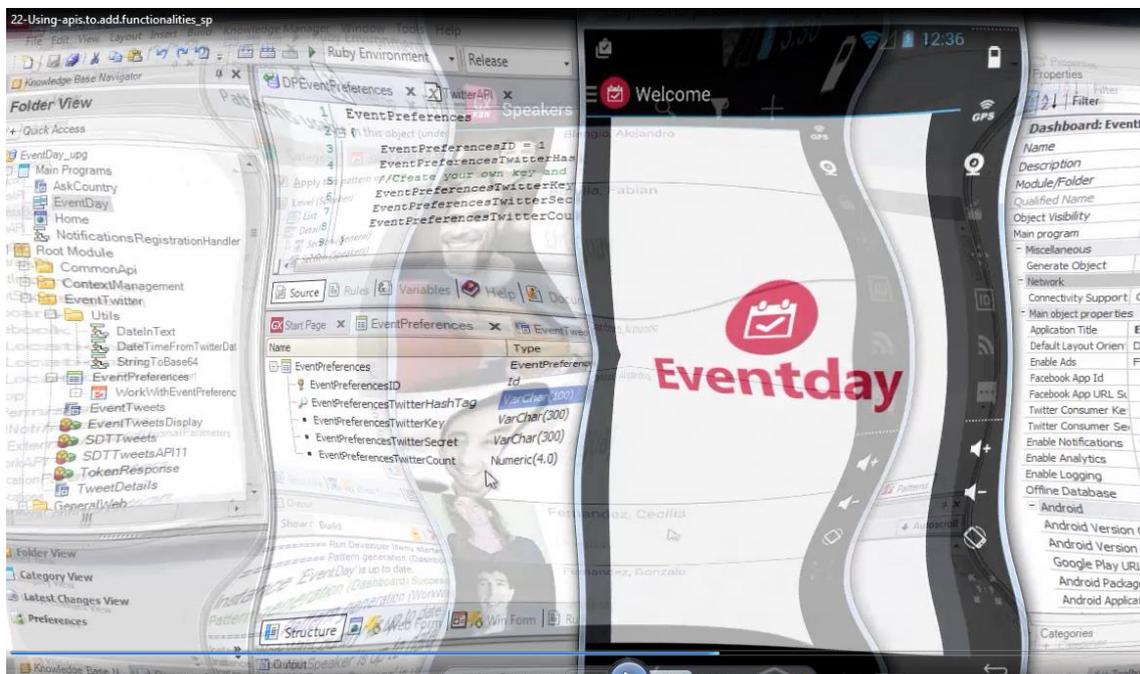
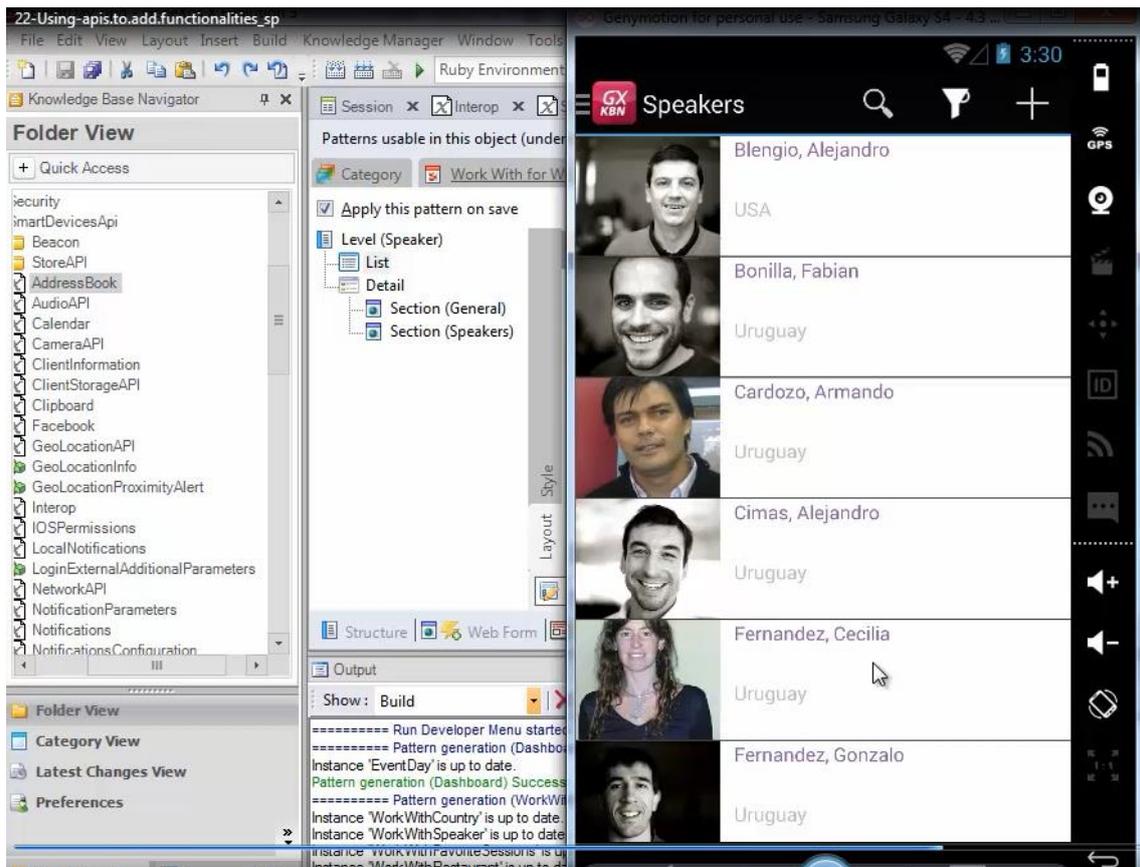
Vemos que fue insertada la información en la base de datos y se nos abre la libreta de direcciones



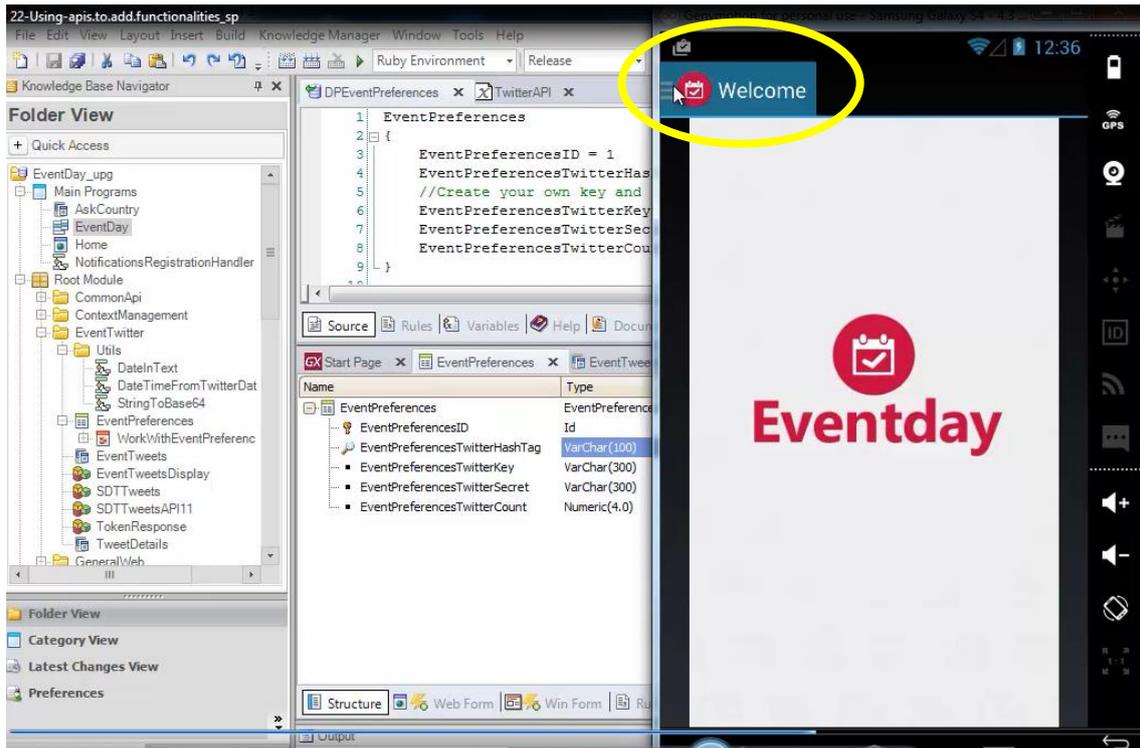
para crear el nuevo contacto con los datos correspondientes



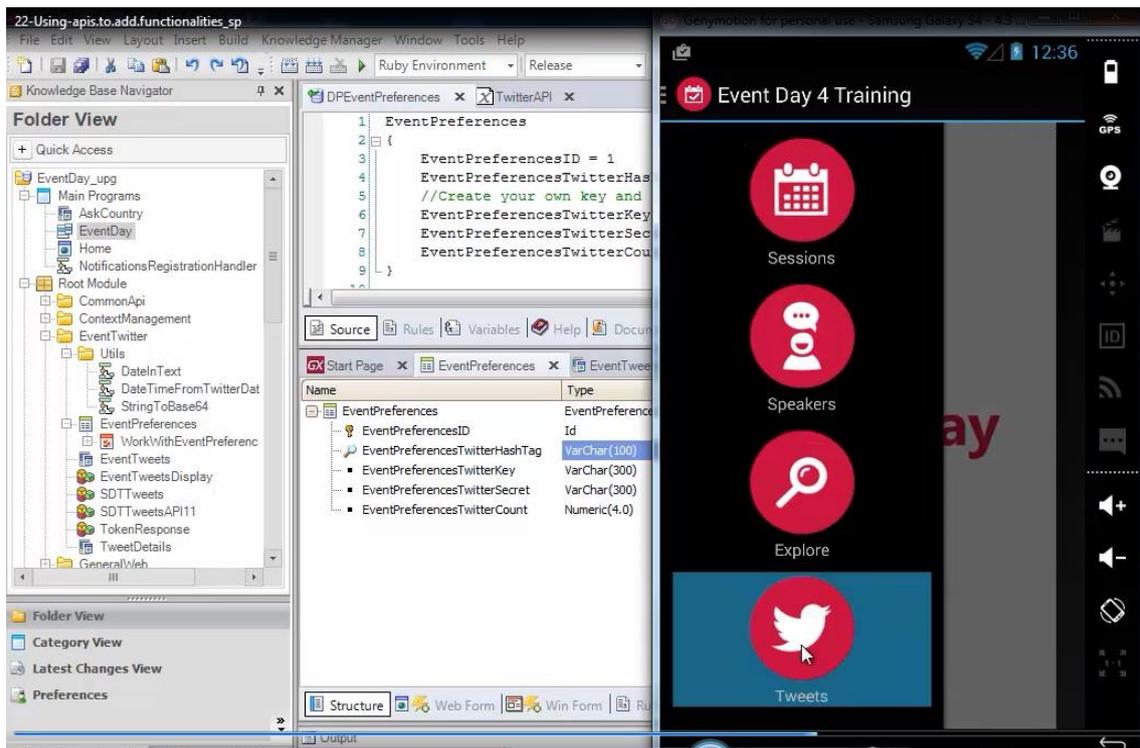
Vemos que se ha agregado el nuevo orador.



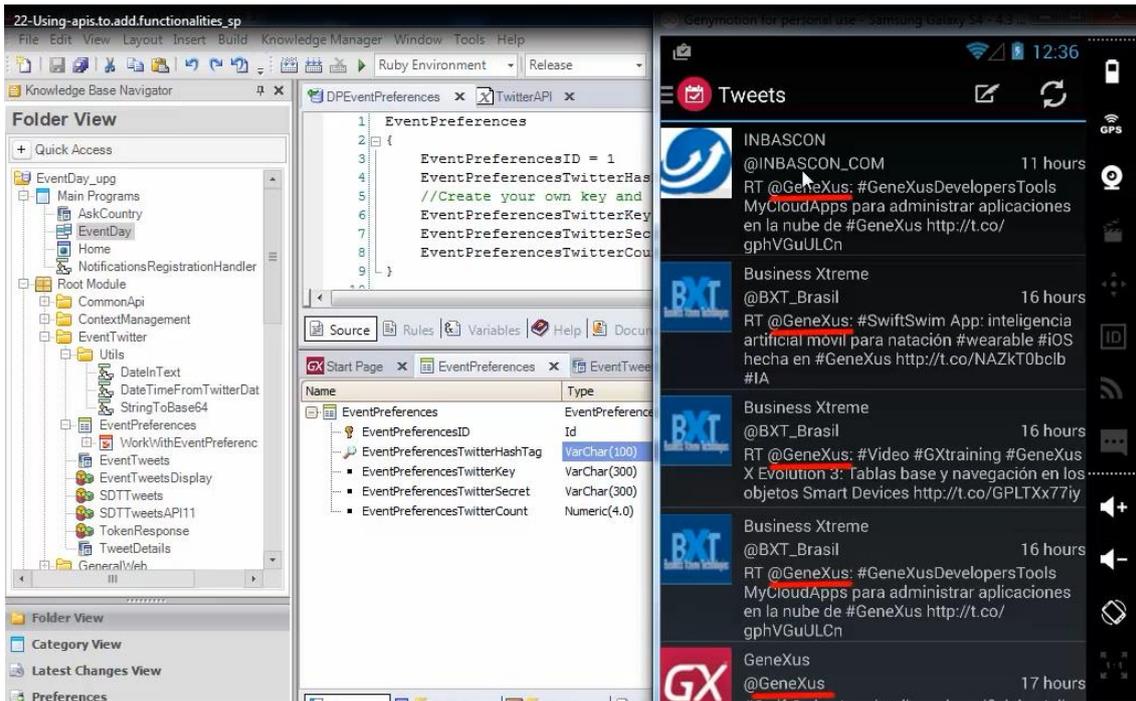
Hemos agregado en nuestra aplicación..



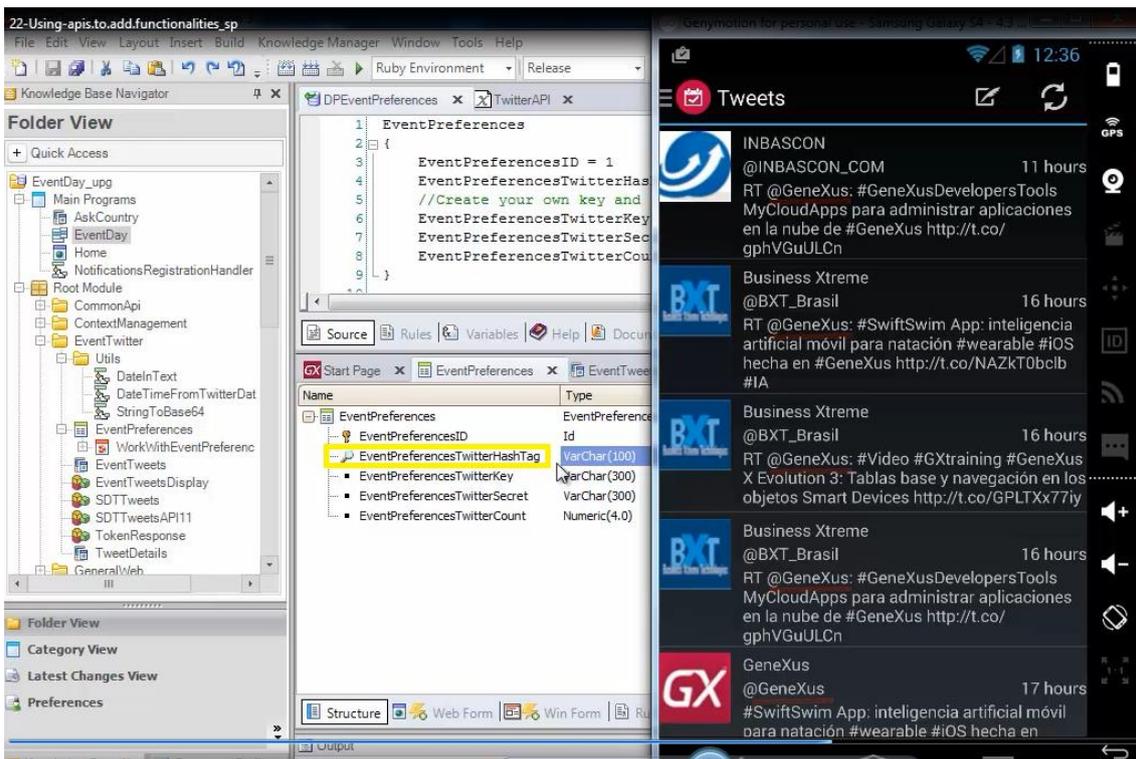
un panel



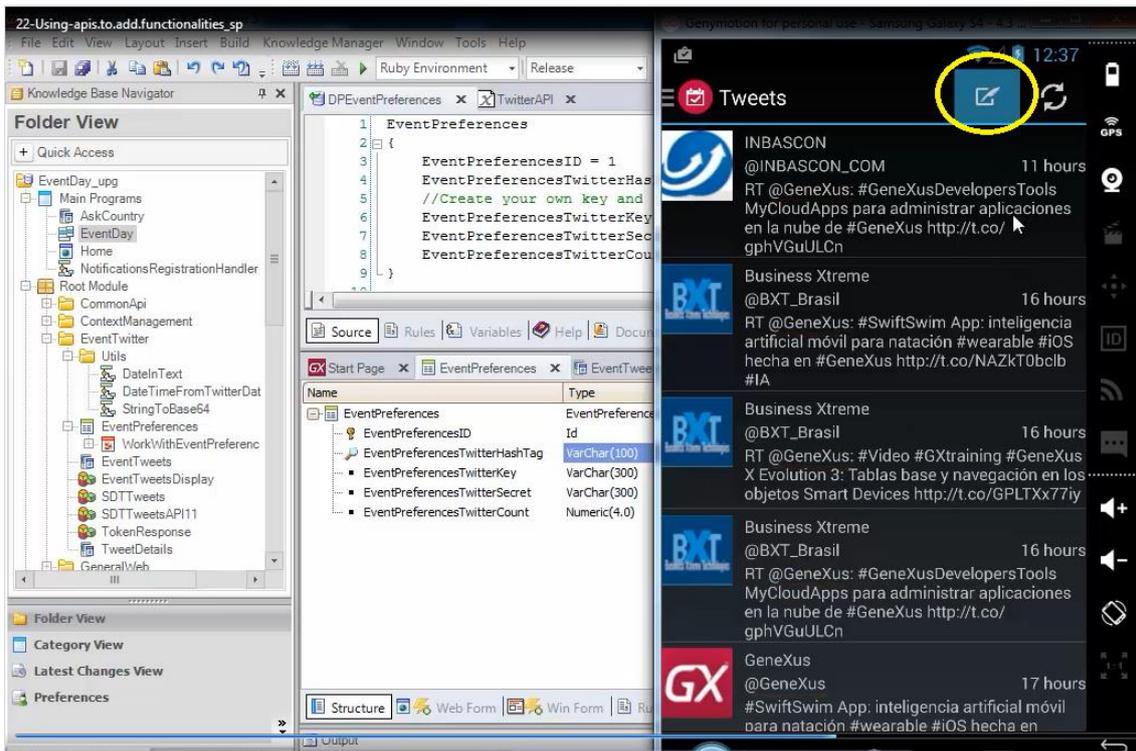
que muestra los últimos tweets



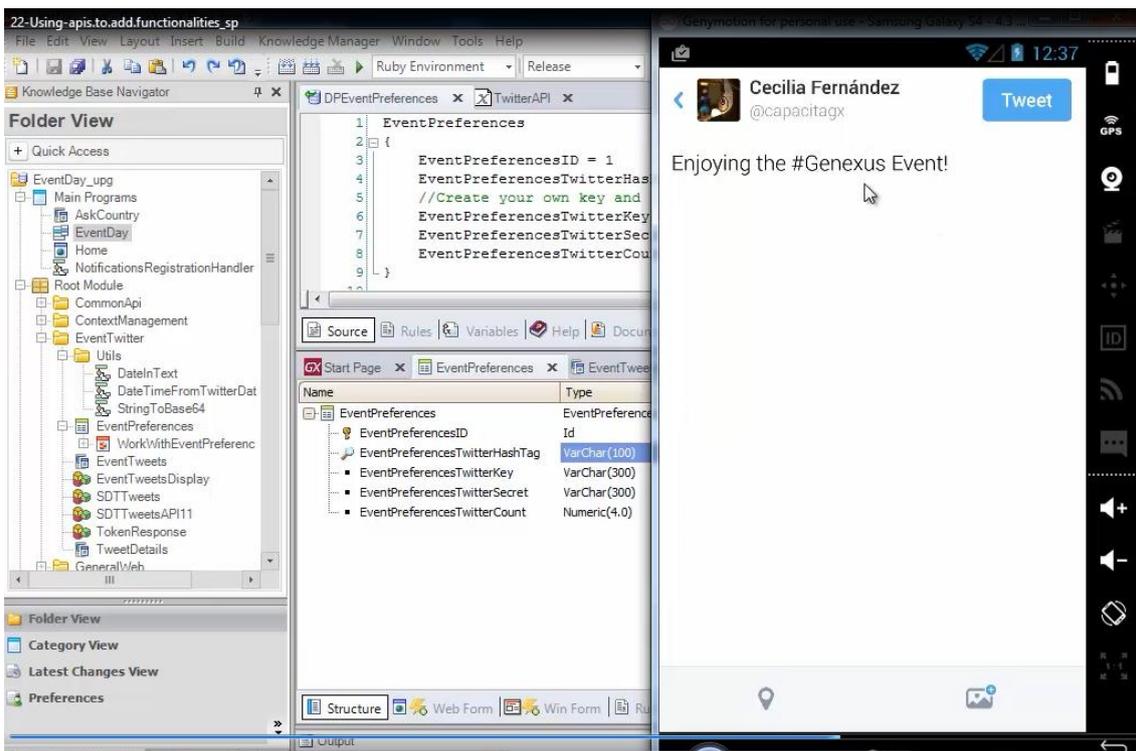
del HashTag que hemos definido en una transacción Preferences de nuestro evento, en el atributo EventPreferencesTwitterHashTag



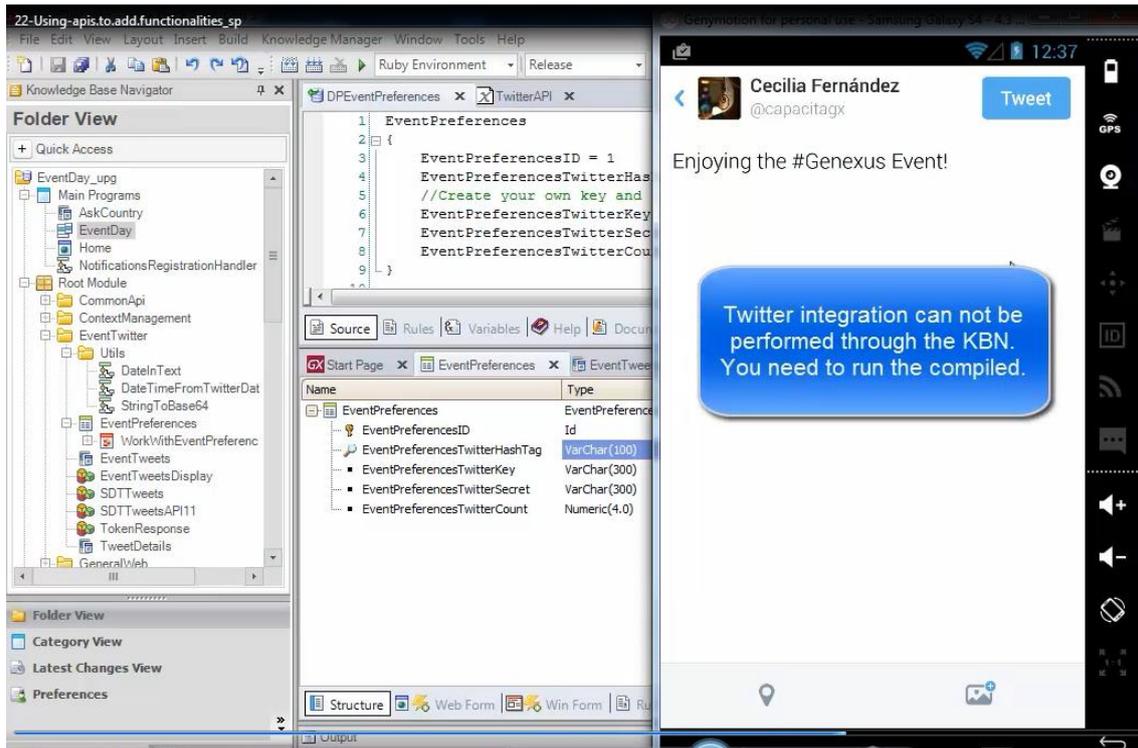
Desde este panel vamos a poder ingresar



un nuevo tweet con ese HashTag definido

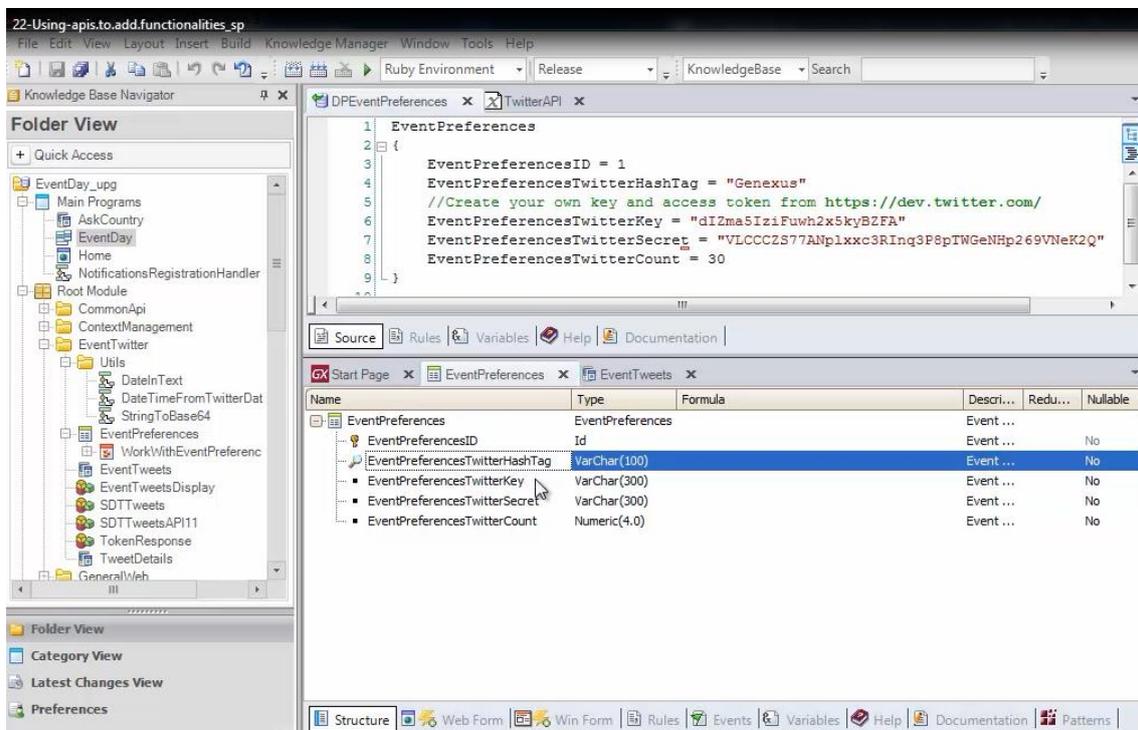


Observemos por tanto que nos estamos integrando con la aplicación de Twitter instalada en el dispositivo.

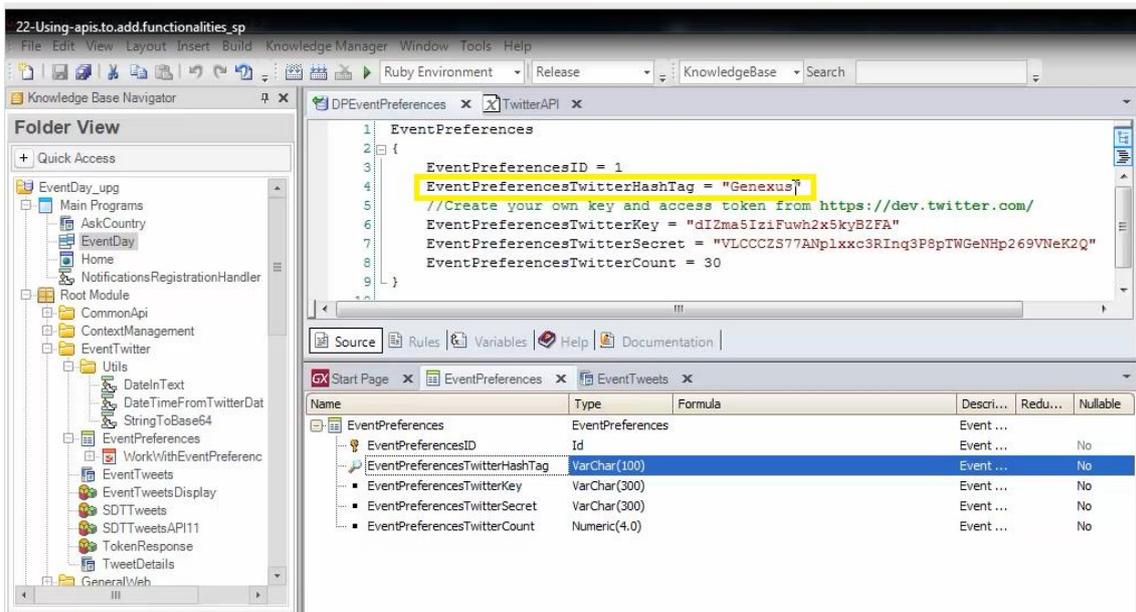


¿Cómo se implementó todo esto?

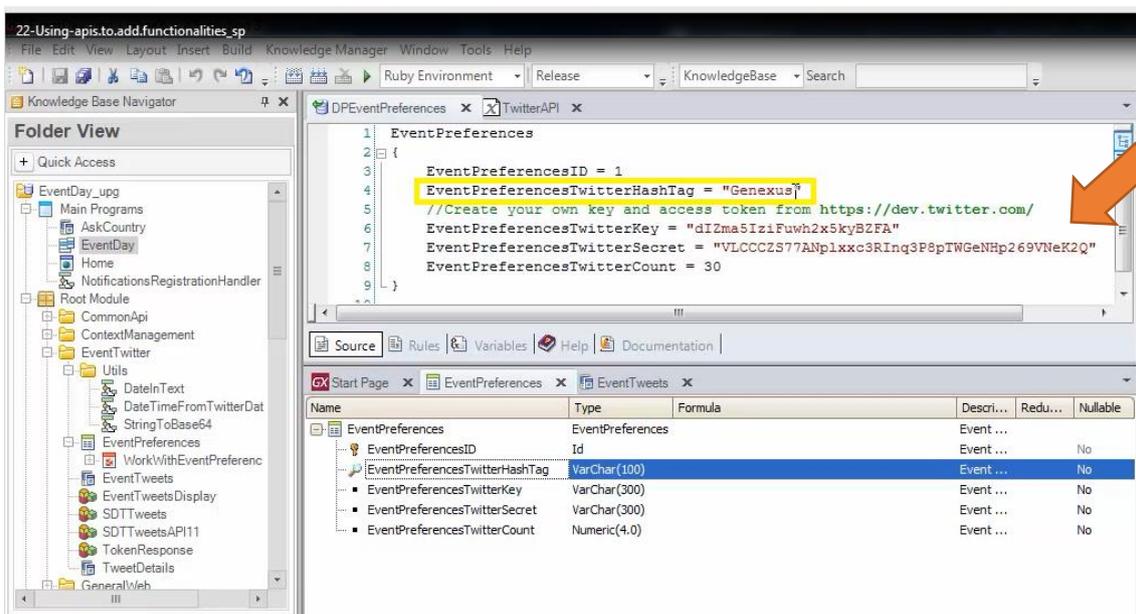
Vayamos a GeneXus.. Veamos entonces que tenemos la transacción donde debemos ingresar estos datos



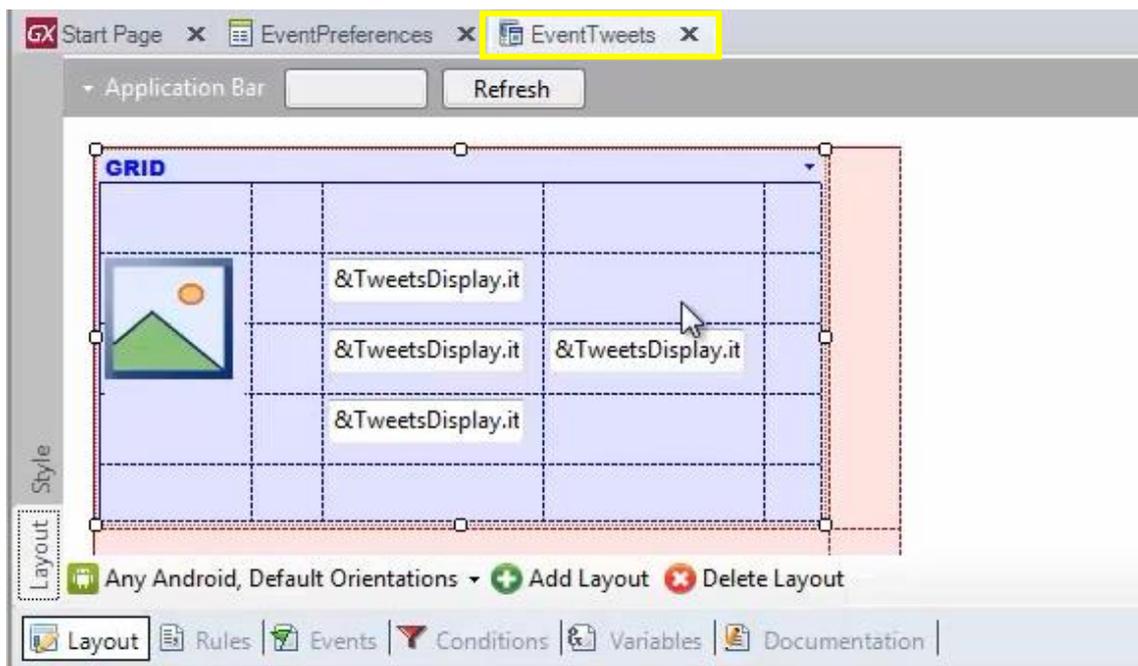
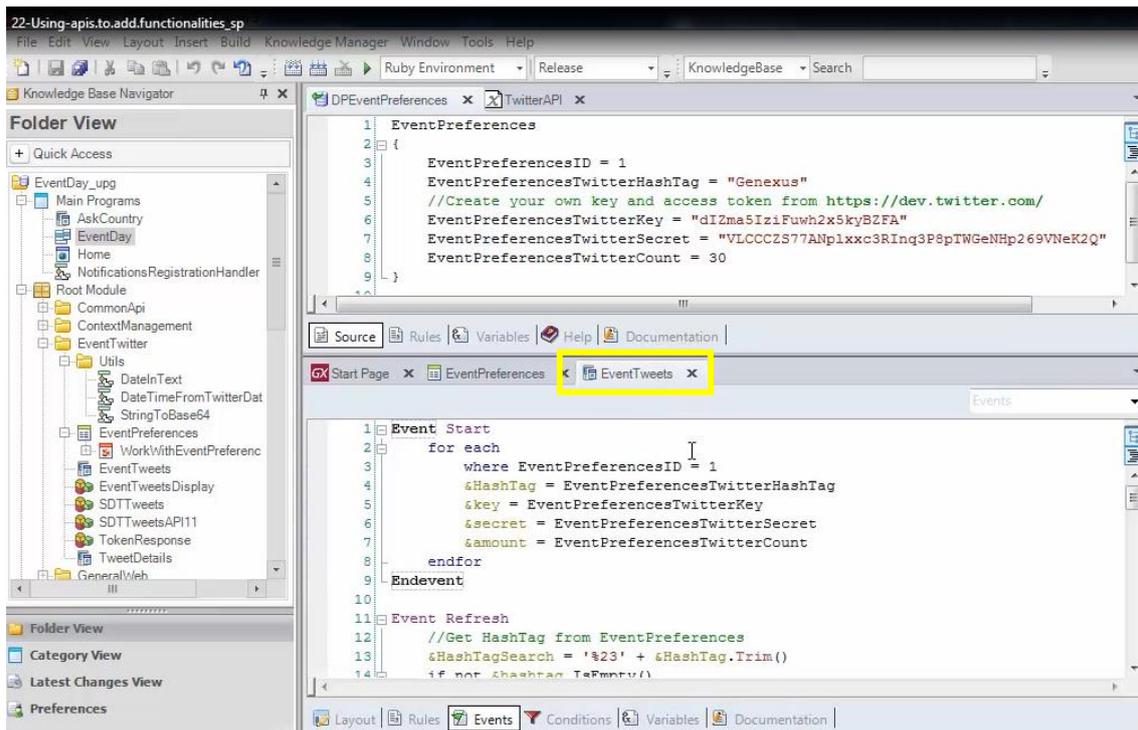
Lo hacemos a través de un data provider



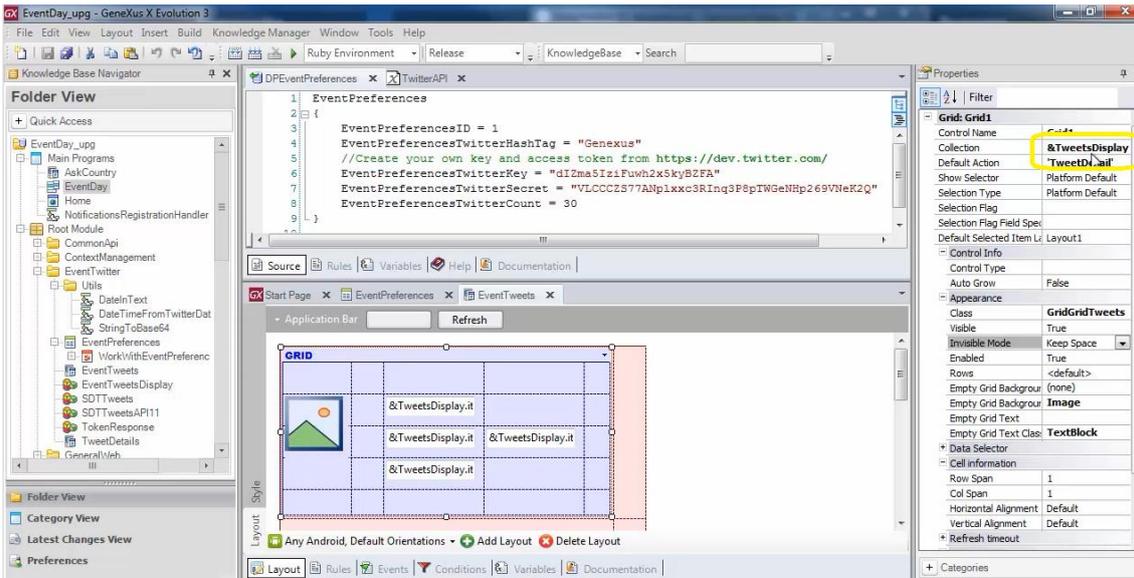
donde definimos como HashTag, GeneXus.. pero aquí sería el HashTag correspondiente al evento que se esté realizando.. y vamos a tener que ir a este sitio de Twitter para los desarrolladores, a pedir una clave y un token para poder comunicarnos con la aplicación. Aquí estamos usando el de GeneXus.



Luego, vamos al panel que implementa todo esto

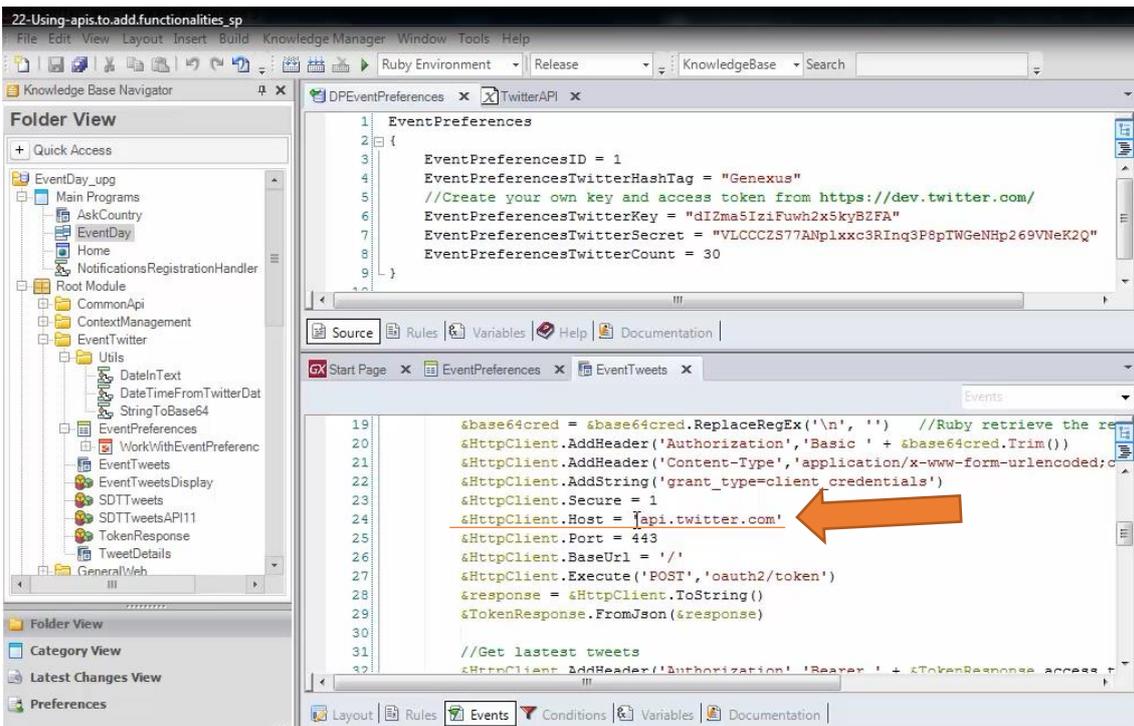


que hemos definido con un grid basado en una variable de tipo colección &TweetsDisplay



que es la que va a tener que ser cargada entonces con los últimos 30 tweets de ese hashtag.

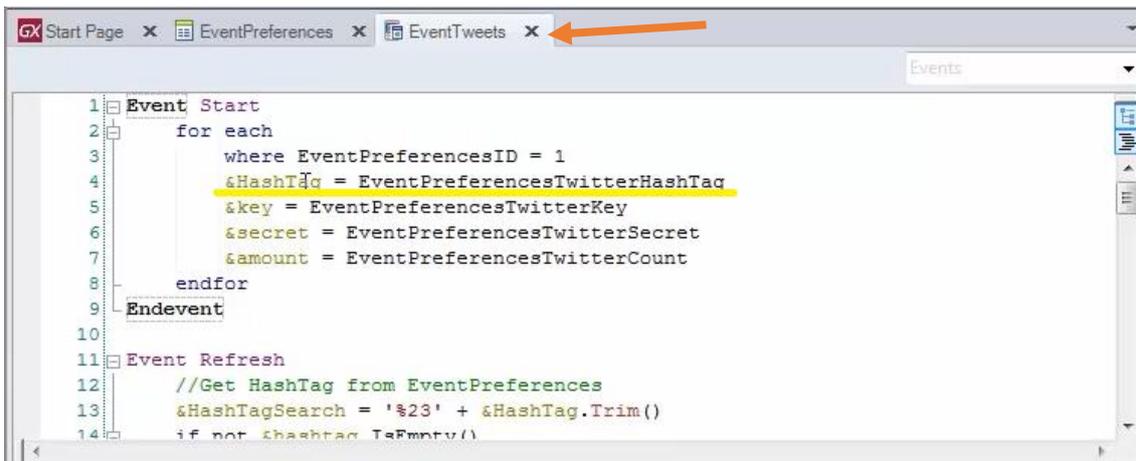
Eso se realiza en el evento Refresh, donde tenemos que comunicarnos



con este host --> api.twitter.com

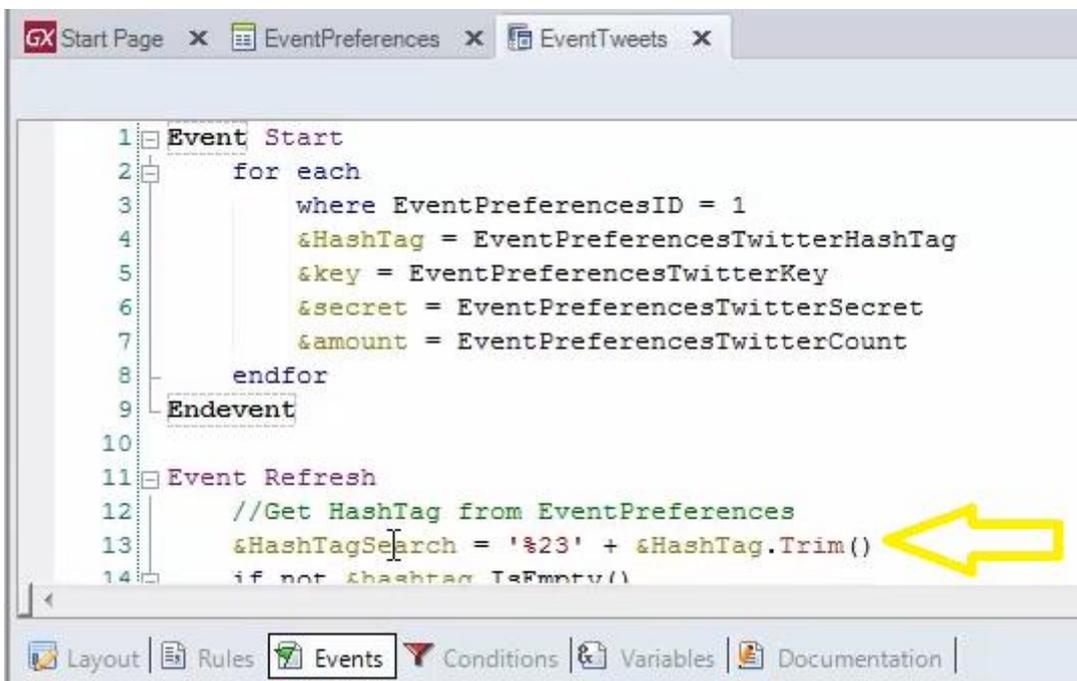
para obtener el token

y luego, con el hashtag correspondiente, que fue obtenido en el evento start



```
1 Event Start
2   for each
3     where EventPreferencesID = 1
4     &HashTag = EventPreferencesTwitterHashTag
5     &key = EventPreferencesTwitterKey
6     &secret = EventPreferencesTwitterSecret
7     &amount = EventPreferencesTwitterCount
8   endfor
9 Endevent
10
11 Event Refresh
12   //Get HashTag from EventPreferences
13   &HashTagSearch = '%23' + &HashTag.Trim()
14   if not &hashtag IsEmpty()
```

tomándolo del registro de la tabla correspondiente.. y aquí vemos que fue reformateado de esta manera:



```
1 Event Start
2   for each
3     where EventPreferencesID = 1
4     &HashTag = EventPreferencesTwitterHashTag
5     &key = EventPreferencesTwitterKey
6     &secret = EventPreferencesTwitterSecret
7     &amount = EventPreferencesTwitterCount
8   endfor
9 Endevent
10
11 Event Refresh
12   //Get HashTag from EventPreferences
13   &HashTagSearch = '%23' + &HashTag.Trim()
14   if not &hashtag IsEmpty()
```

Decíamos, con eso entonces: el token, el hashtag y la cantidad de tweets que queremos recuperar, lo hacemos en esta variable &tweets

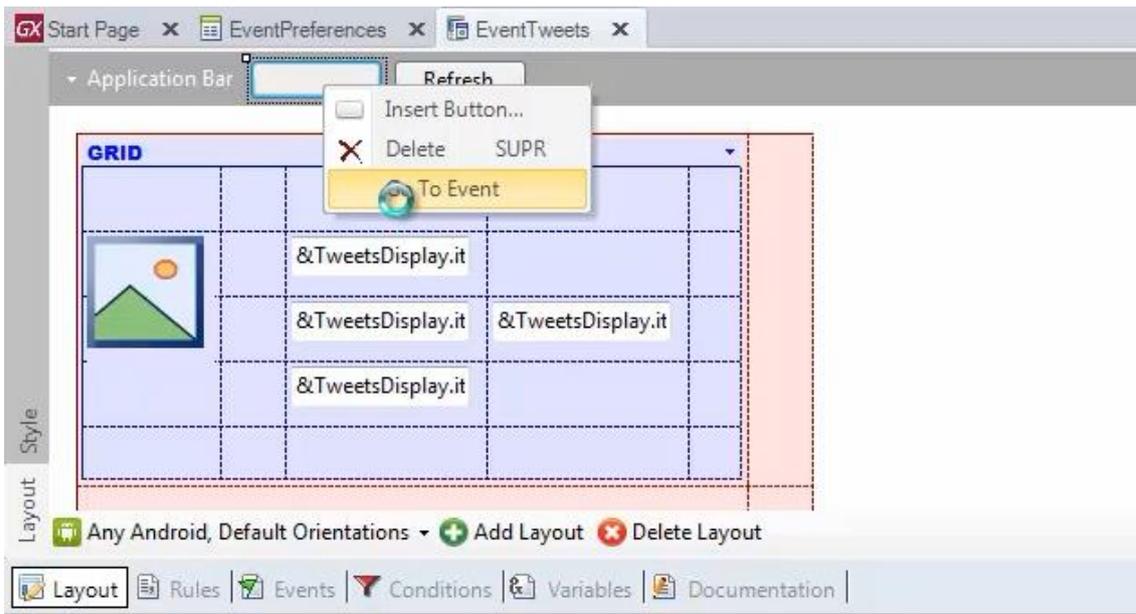
```
31 //Get lastest tweets
32 &HttpClient.AddHeader('Authorization','Bearer ' + &TokenResponse.access_t
33 &HttpClient.AddHeader('Content-Type','application/x-www-form-urlencoded;c
34 &HttpClient.Secure = 1
35 &HttpClient.Port = 443
36 &HttpClient.Host = 'api.twitter.com'
37 &HttpClient.BaseUrl = '/'
38 &HttpClient.Execute('GET','1.1/search/tweets.json?q=' + &HashTagSearch.Tr
39 &jsontext = &HttpClient.ToString()
40 &tweets.FromJson(&jsontext)
41
42 &TweetsDisplay = new()
43 for &itemresult in &tweets.statuses
44     &TweetsDisplayItem = new()
```

Variable que recorremos aquí

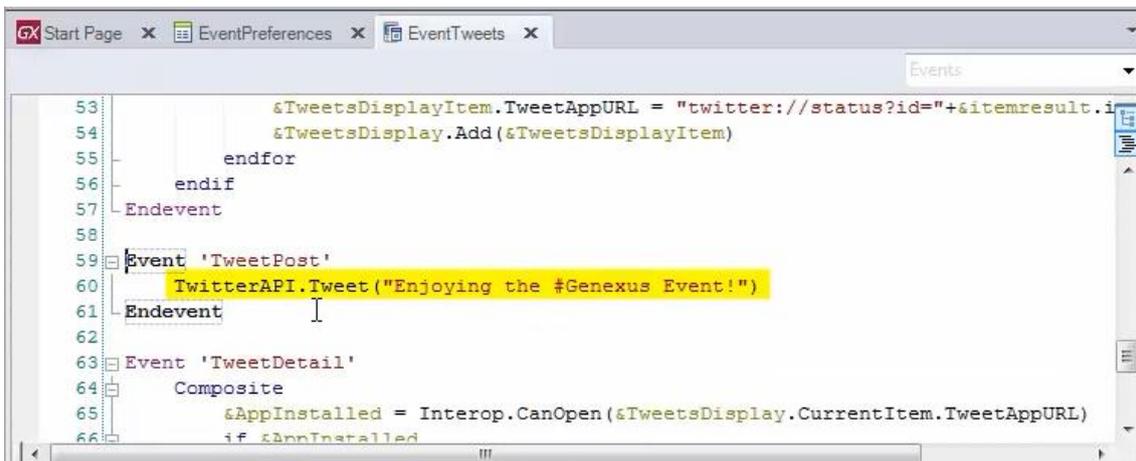
```
40 &tweets.FromJson(&jsontext)
41
42 &TweetsDisplay = new()
43 for &itemresult in &tweets.statuses
44     &TweetsDisplayItem = new()
45     &TweetsDisplayItem.TweetUser = '@' + &itemresult.user.screen_name
46     &TweetsDisplayItem.TweetUserName = &itemresult.user.name
47     &TweetsDisplayItem.TweetProfileImage.FromURL(&itemresult.user.profile
48     &TweetsDisplayItem.TweetText = &itemresult.text
49     &DateTime = DateTimeFromTwitterDate(&itemresult.created_at)
50     &TweetsDisplayItem.TweetTime = DateInText(&DateTime)
51     &TweetsDisplayItem.TweetID = &itemresult.id_str.ToNumeric()
52     &TweetsDisplayItem.TweetURL = 'http://twitter.com/' + &itemresult.use
53     &TweetsDisplayItem.TweetAppURI = "twitter://status?id="+&itemresult.i
```

cargando entonces la variable colección que se va a desplegar en el grid.

Y luego, tenemos el botón

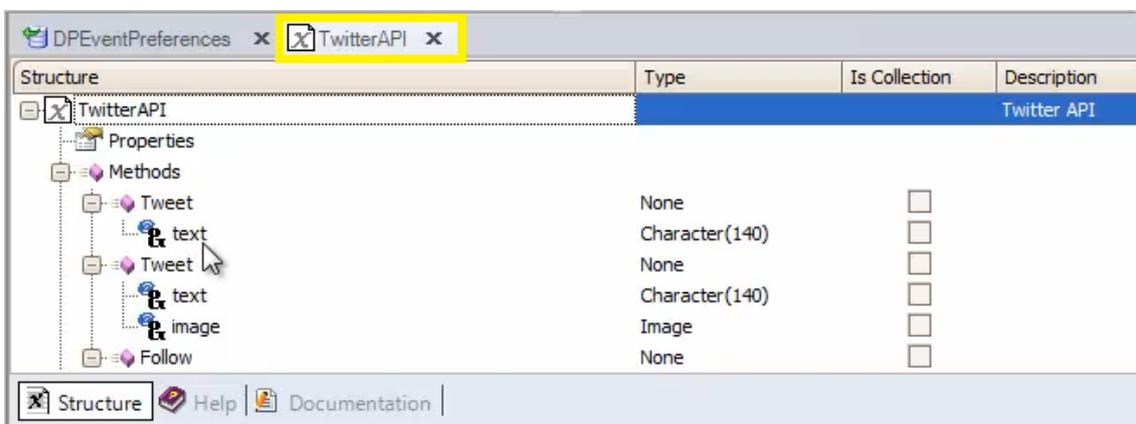


que permite que el usuario ingrese un tweet



Observemos que allí estamos usando la api TwitterAPI-método Tweet.

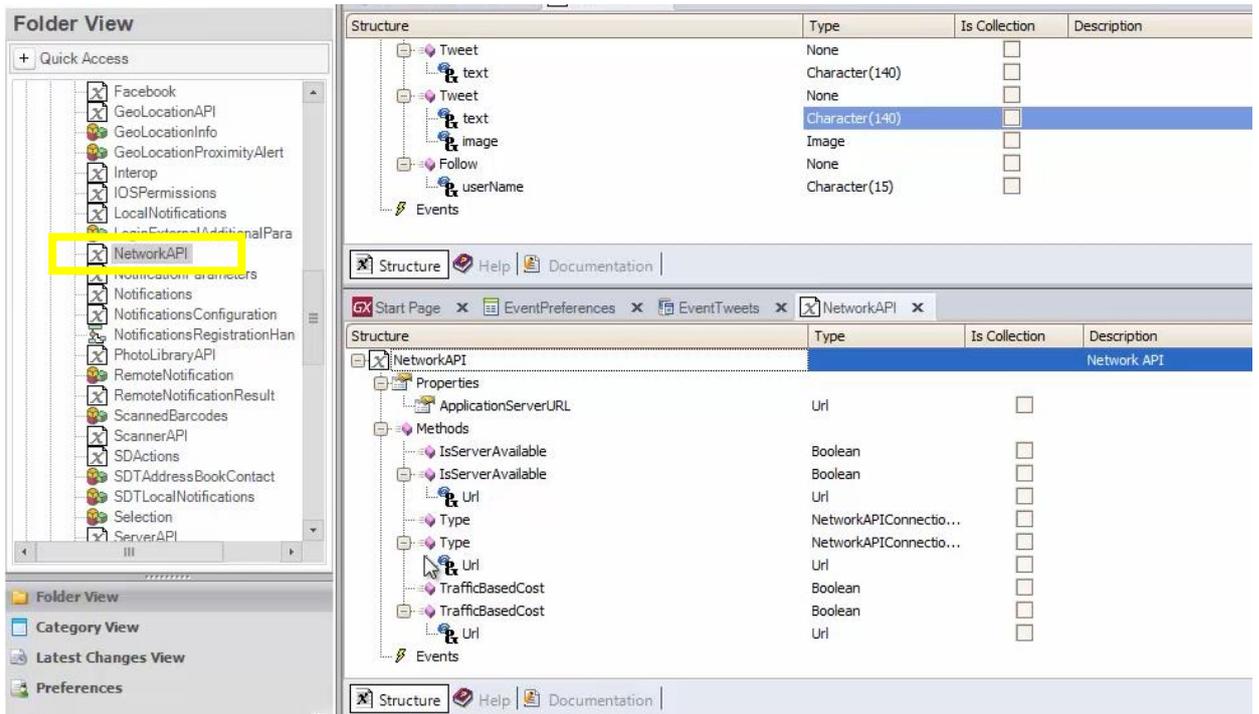
Si observamos la api, nos ofrece 2 métodos tweet



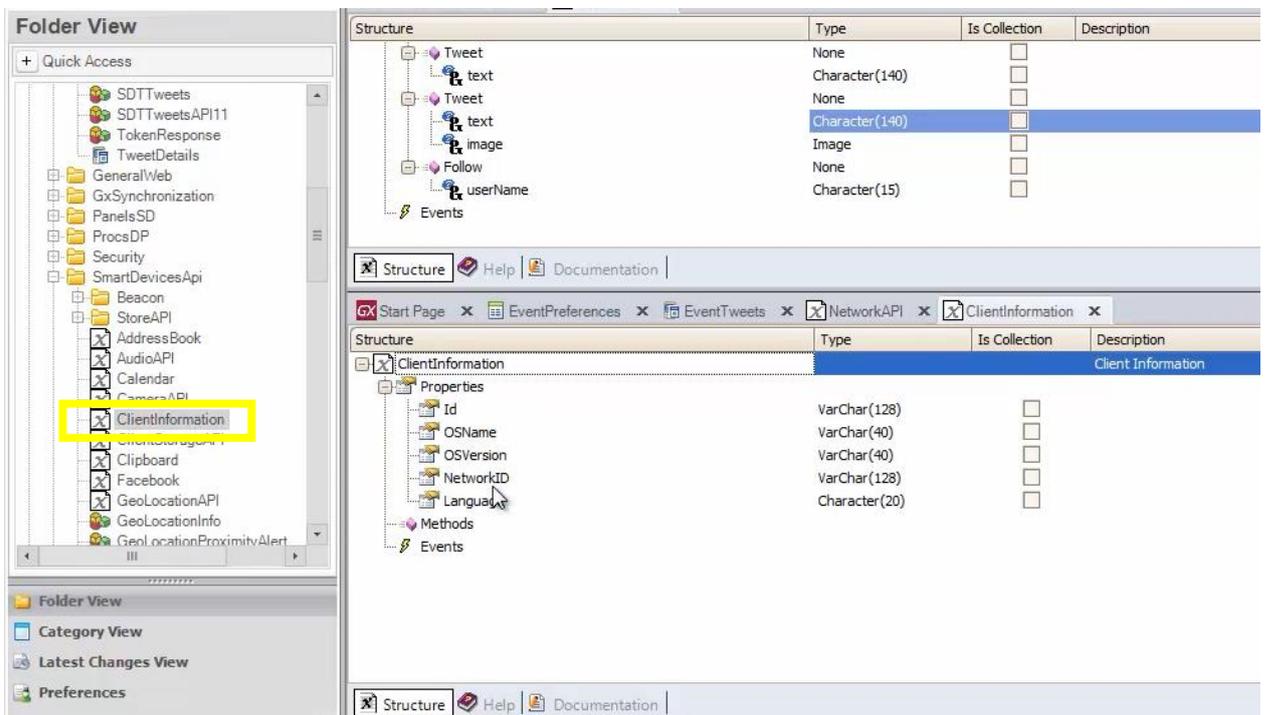
en uno solamente enviamos el mensaje a ser tweeteado y en el otro el texto y la imagen.

Y por otro lado tenemos el método Follow.

Tenemos unas cuantas apis más. Por ejemplo, una que nos permite saber si hay conexión con el servidor en un momento dado.. y el tipo de conexión..

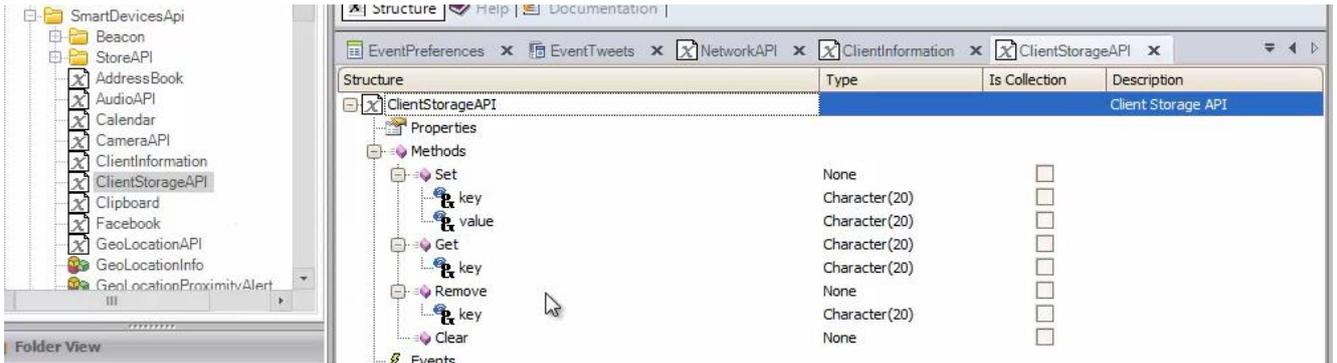


Otra para obtener la información del dispositivo que está ejecutando la aplicación



Su Id, sistema operativo, versión, lenguaje..

Una que permite definir variables globales para el dispositivo



Una para integrarnos con Facebook, otra muy utilizada para trabajar con la geolocalización
 Ya usábamos el dominio semántico Geolocation

Device Resources GeneXus™

Semantic Domains + Control type Smart Devices APIs

Name	Type
Restaurant	Restaurant
RestaurantId	Id
RestaurantName	Name
RestaurantImage	Image
RestaurantAddress	Address
RestaurantGeolocation	Geolocation
RestaurantPhone	Phone
RestaurantWeb	Url
RestaurantRating	Numeric(1.0)

cuando queríamos presentar los restaurants, no como un List standard

22-Using apis.to.add.functionalities_sp
 Device Resources GeneXus™

Semantic Domains + Control type Smart Devices APIs

Name	Type
Restaurant	Restaurant
RestaurantId	Id
RestaurantName	Name
RestaurantImage	Image
RestaurantAddress	Address
RestaurantGeolocation	Geolocation
RestaurantPhone	Phone
RestaurantWeb	Url
RestaurantRating	Numeric(1.0)

sino como puntos en un mapa

22-Using-apis.to.add.functionalities_sp
Device Resources

Semantic Domains + Control type

Name	Type
Restaurant	Restaurant
RestaurantId	Id
RestaurantName	Name
RestaurantImage	Image
RestaurantAddress	Address
RestaurantGeolocation	Geolocation
RestaurantPhone	Phone
RestaurantWeb	Url
RestaurantRating	Numeric(1.0)

Smart Devices APIs

Restaurants

- Don Peperone
- Jacinto Café y Restaurant
- La corte
- Los lenos
- PV Lounge

Restaurants Map

Control Type: SD Maps

Para ello nos alcanzaba con cambiar el tipo de control del grid por: SDMaps

22-Using-apis.to.add.functionalities_sp
Device Resources

Semantic Domains + Control type

Name	Type
Restaurant	Restaurant
RestaurantId	Id
RestaurantName	Name
RestaurantImage	Image
RestaurantAddress	Address
RestaurantGeolocation	Geolocation
RestaurantPhone	Phone
RestaurantWeb	Url
RestaurantRating	Numeric(1.0)

Smart Devices APIs

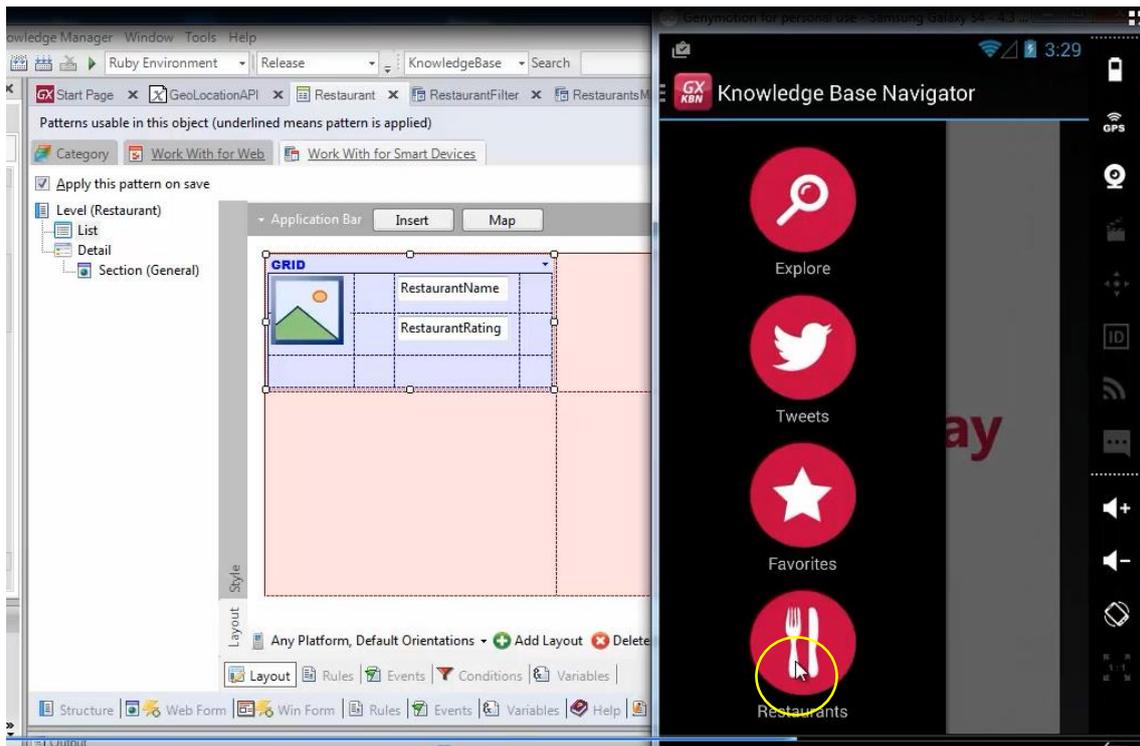
Restaurants

- Don Peperone
- Jacinto Café y Restaurant
- La corte
- Los lenos
- PV Lounge

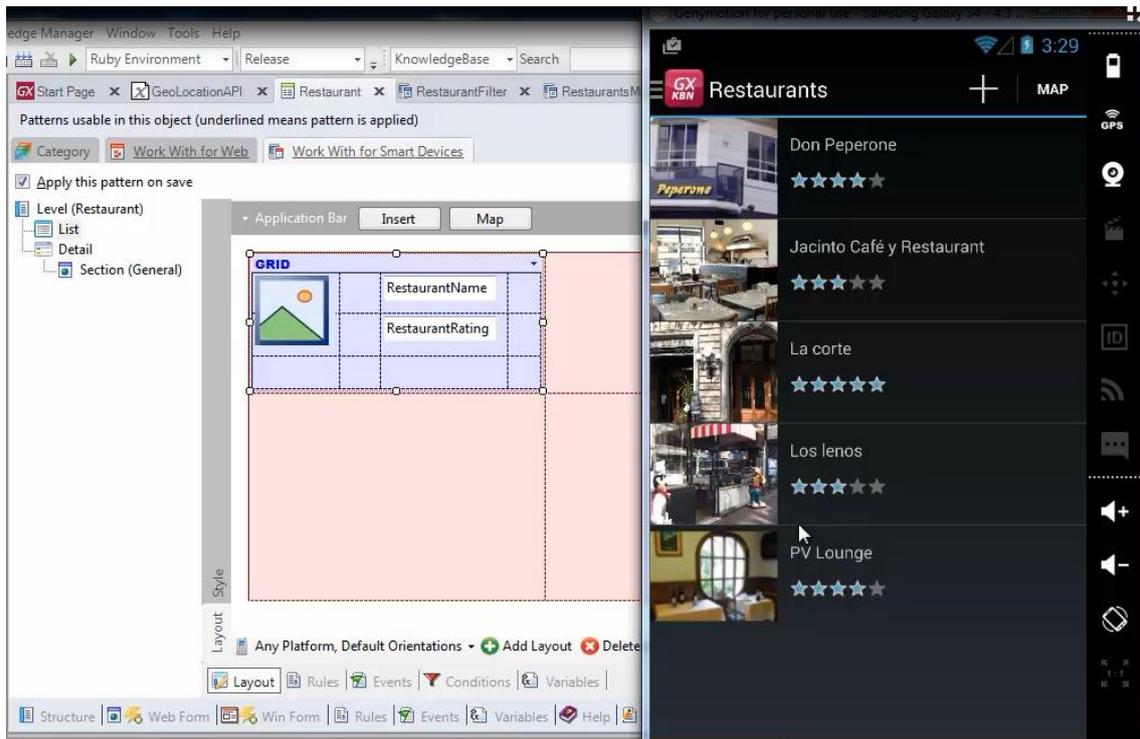
Restaurants Map

Control Type: SD Maps

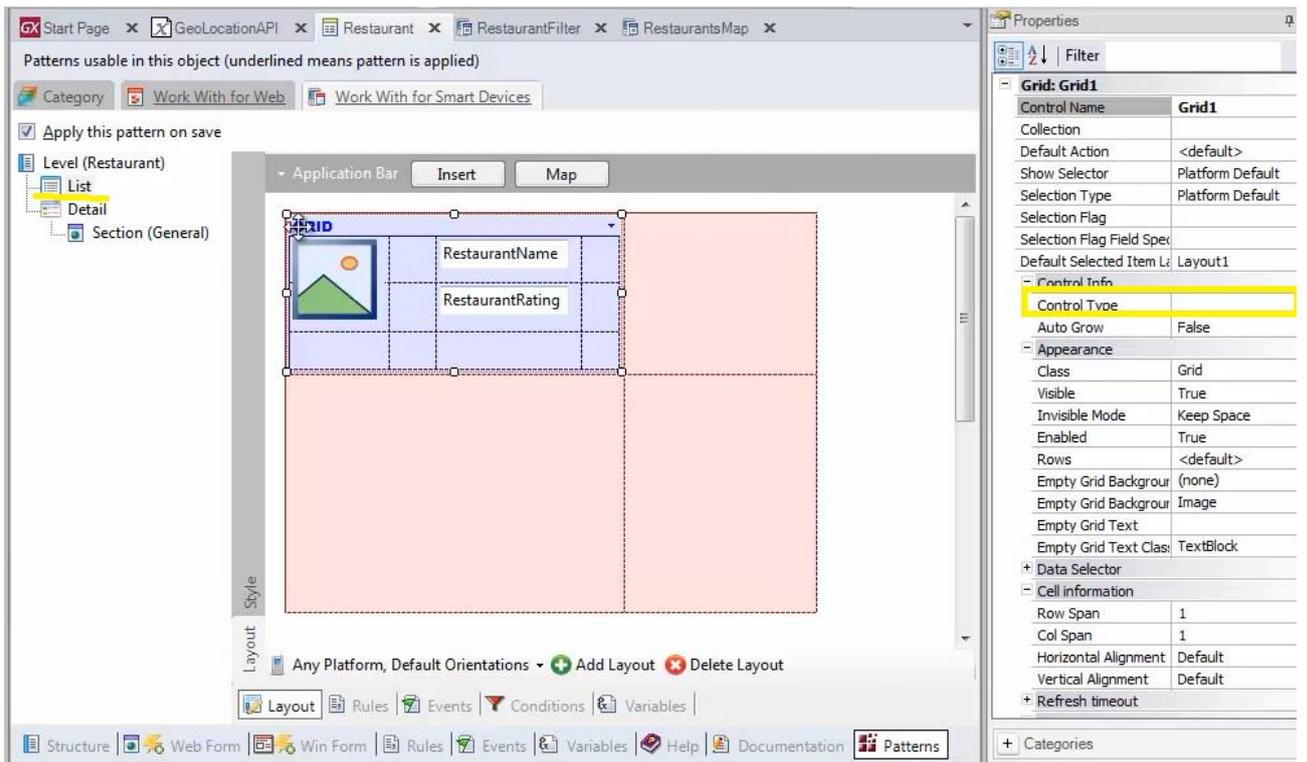
En nuestra aplicación, tenemos



Tenemos implementado el List de restaurants

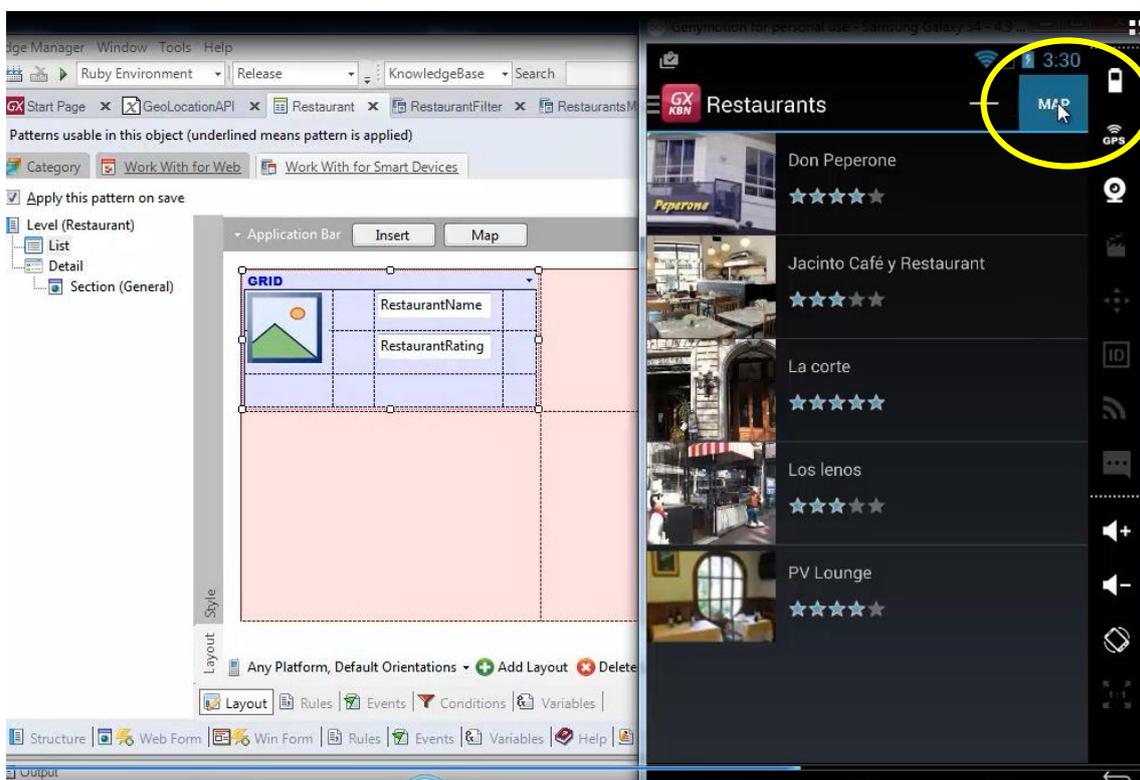


que corresponde al nodo del work with

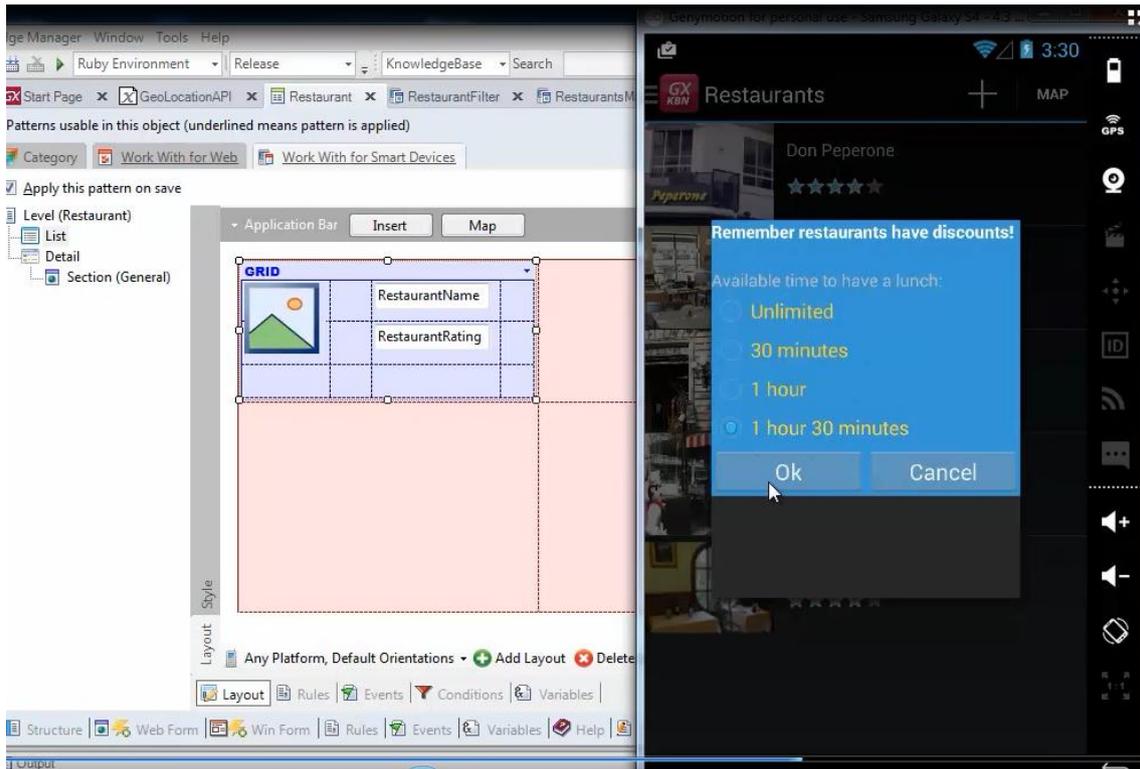


Donde el grid tiene el Control Type default

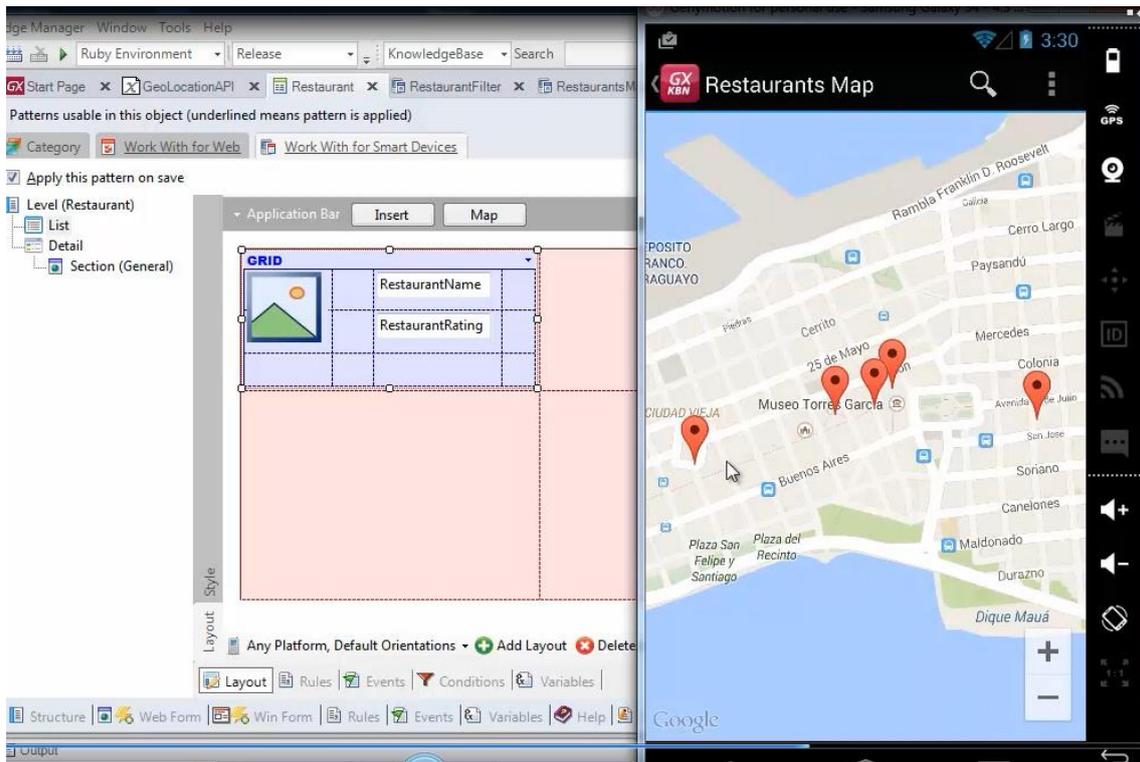
Y por otro lado, ofrecemos al usuario



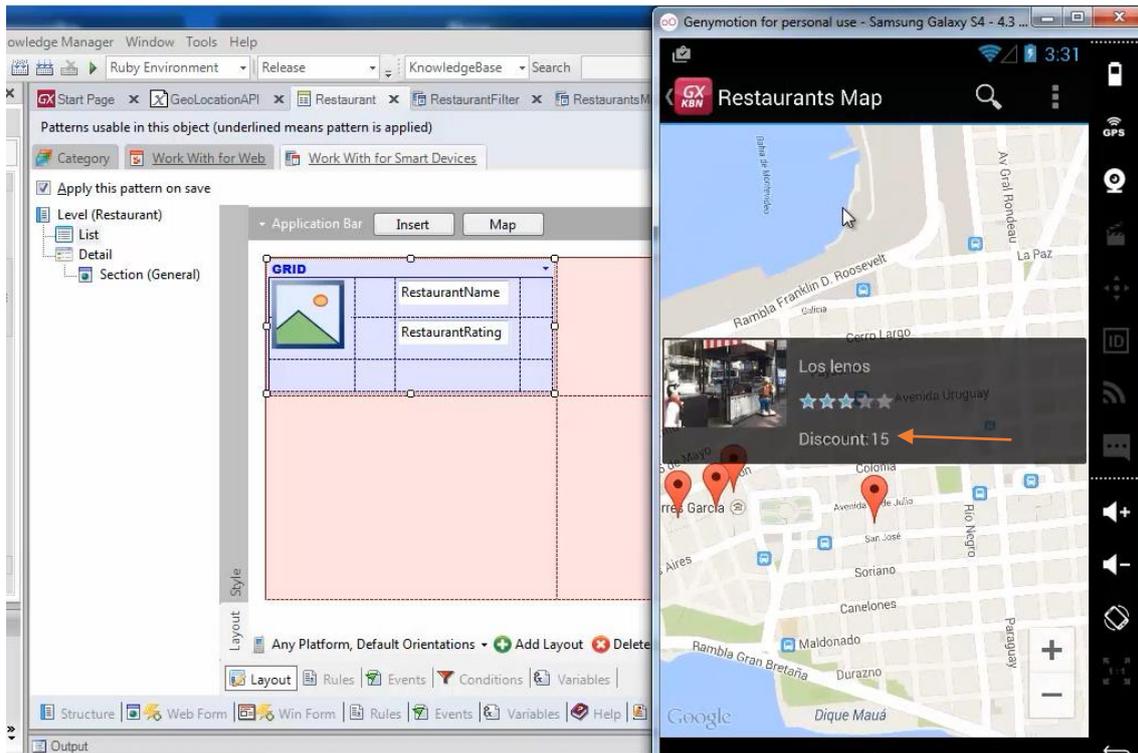
la posibilidad de eligiendo el tiempo del que dispone para almorzar



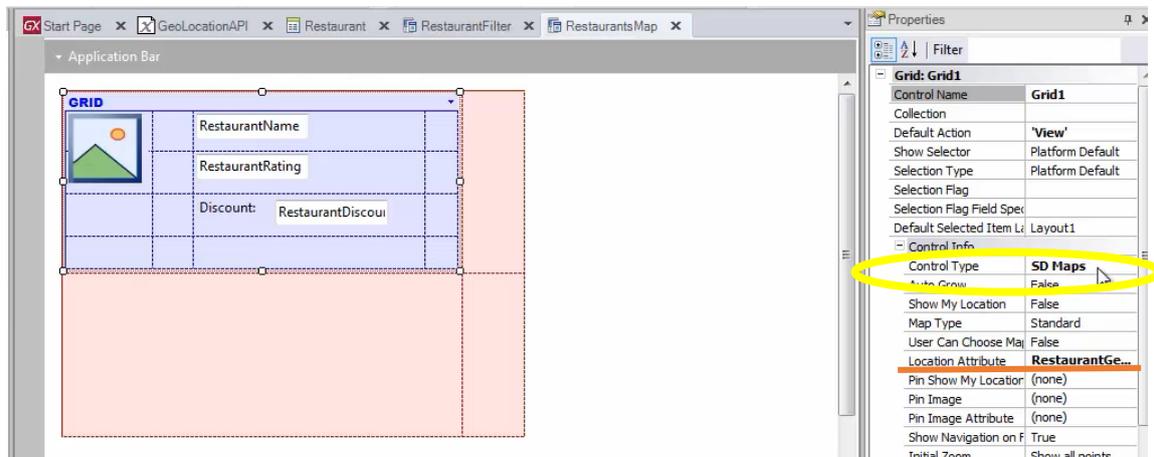
mostrarle los restaurants que se comprometen a brindarle el servicio dentro de ese tiempo



Este es un panel:

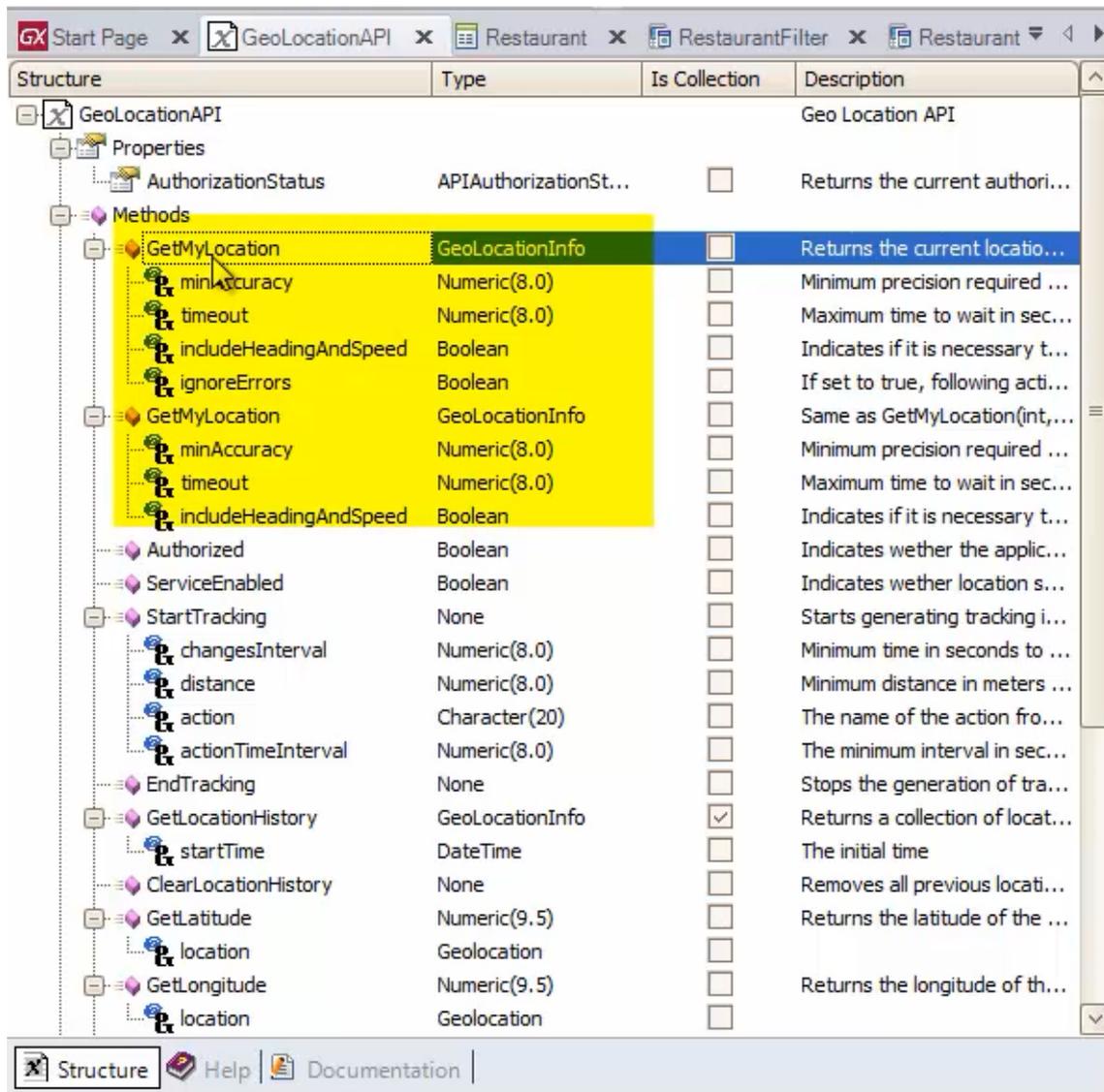


cuyo Layout es muy similar al del List del work with, con la diferencia de este atributo de aquí .. y de la manera de mostrar entonces la información del grid, que será como puntos en un mapa



Debido a que tenemos el atributo: RestaurantGeolocation

Ahora bien, podríamos solamente querer mostrar al usuario, los restaurants que se encuentran a menos de cierta distancia de la localización actual del dispositivo. Allí es donde interviene la GeoLocationAPI



que ofrece métodos para por ejemplo obtener la Geolocation actual, para lo cual necesita el GPS del dispositivo..

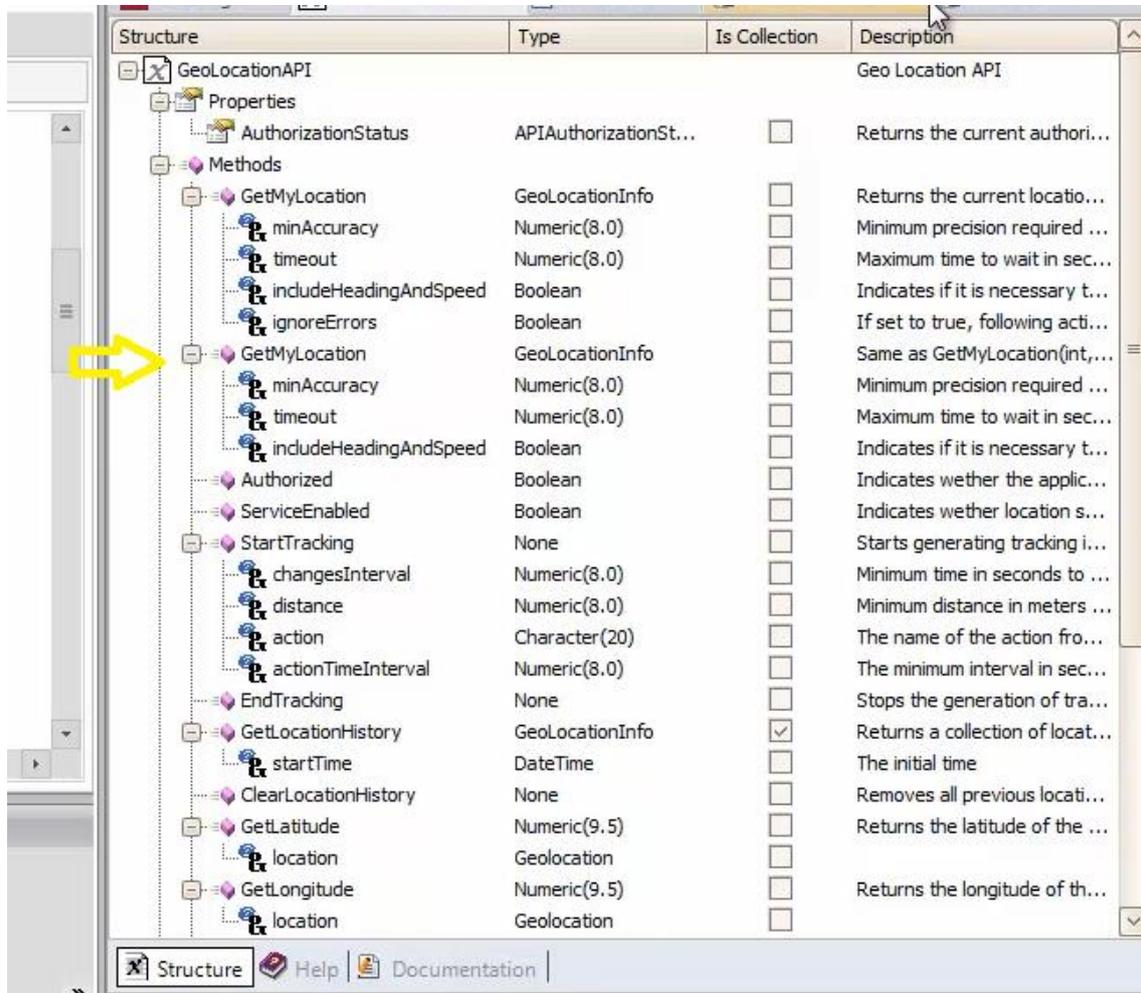
Structure	Type	Is Col...	Description
GeoLocationAPI			Geo Location API
Properties			
AuthorizationStatus	APIAuthoriza...	<input type="checkbox"/>	Returns the current ...
Methods			
GetMyLocation	GeoLocationI...	<input type="checkbox"/>	Returns the current l...
minAccuracy	Numeric(8.0)	<input type="checkbox"/>	Minimum precision re...
timeout	Numeric(8.0)	<input type="checkbox"/>	Maximum time to wait...
includeHeadingAndSpeed	Boolean	<input type="checkbox"/>	Indicates if it is neces...
ignoreErrors	Boolean	<input type="checkbox"/>	If set to true, followi...
GetMyLocation	GeoLocationI...	<input type="checkbox"/>	Same as GetMyLocati...
minAccuracy	Numeric(8.0)	<input type="checkbox"/>	Minimum precision re...
timeout	Numeric(8.0)	<input type="checkbox"/>	Maximum time to wait...
includeHeadingAndSpeed	Boolean	<input type="checkbox"/>	Indicates if it is neces...
Authorized	Boolean	<input type="checkbox"/>	Indicates wether the ...
ServiceEnabled	Boolean	<input type="checkbox"/>	Indicates wether loca...
StartTracking	None	<input type="checkbox"/>	Starts generating tra...
changesInterval	Numeric(8.0)	<input type="checkbox"/>	Minimum time in seco...
distance	Numeric(8.0)	<input type="checkbox"/>	Minimum distance in ...
action	Character(20)	<input type="checkbox"/>	The name of the acti...
actionTimeInterval	Numeric(8.0)	<input type="checkbox"/>	The minimum interval ...
EndTracking	None	<input type="checkbox"/>	Stops the generation...
GetLocationHistory	GeoLocationI...	<input checked="" type="checkbox"/>	Returns a collection o...
startTime	DateTime	<input type="checkbox"/>	The initial time
ClearLocationHistory	None	<input type="checkbox"/>	Removes all previous...
GetLatitude	Numeric(9.5)	<input type="checkbox"/>	Returns the latitude ...
location	Geolocation	<input type="checkbox"/>	

realizar un tracking del camino por el que va pasando el dispositivo a través del tiempo, finalizar el tracking, obtener la latitud a partir de una geolocalización, o la longitud, así como obtener la distancia entre 2 puntos

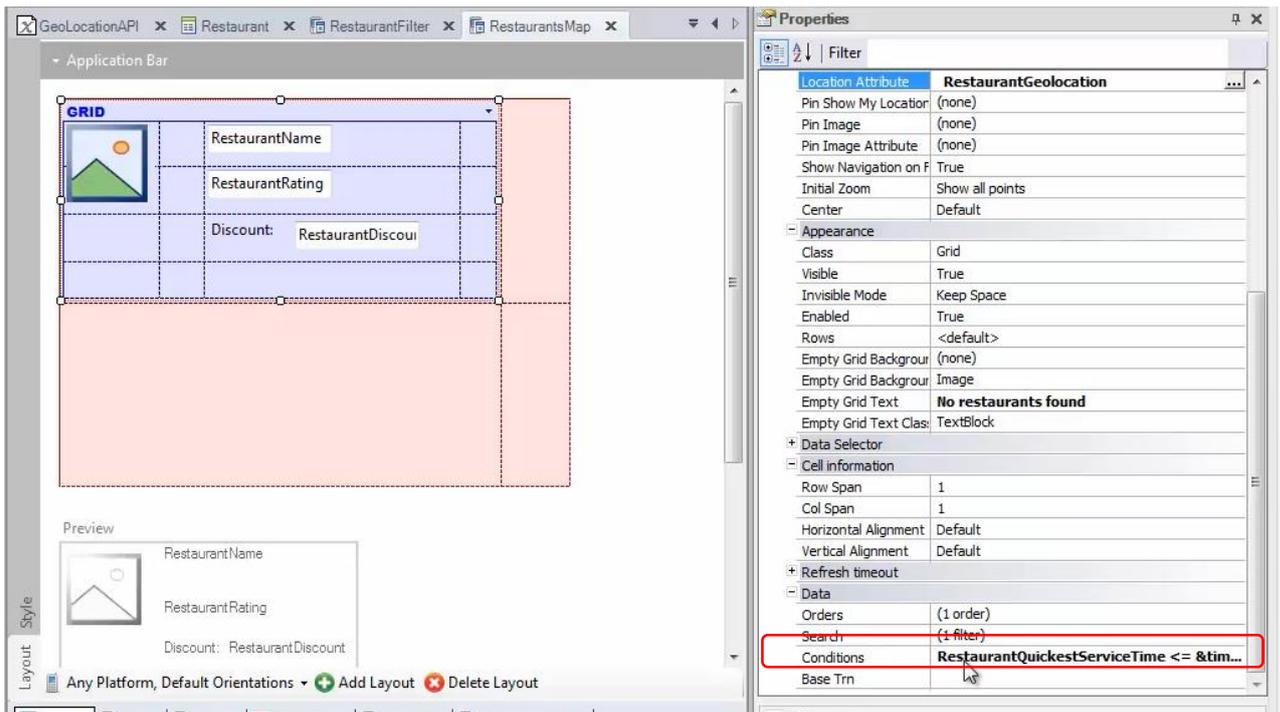
Structure	Type	Is Collection	Description
Authorized	Boolean	<input type="checkbox"/>	Indicates wether the applic...
ServiceEnabled	Boolean	<input type="checkbox"/>	Indicates wether location s...
StartTracking	None	<input type="checkbox"/>	Starts generating tracking i...
changesInterval	Numeric(8.0)	<input type="checkbox"/>	Minimum time in seconds to ...
distance	Numeric(8.0)	<input type="checkbox"/>	Minimum distance in meters ...
action	Character(20)	<input type="checkbox"/>	The name of the action fro...
actionTimeInterval	Numeric(8.0)	<input type="checkbox"/>	The minimum interval in sec...
EndTracking	None	<input type="checkbox"/>	Stops the generation of tra...
GetLocationHistory	GeoLocationInfo	<input checked="" type="checkbox"/>	Returns a collection of locat...
startTime	DateTime	<input type="checkbox"/>	The initial time
ClearLocationHistory	None	<input type="checkbox"/>	Removes all previous locati...
GetLatitude	Numeric(9.5)	<input type="checkbox"/>	Returns the latitude of the ...
location	Geolocation	<input type="checkbox"/>	
GetLongitude	Numeric(9.5)	<input type="checkbox"/>	Returns the longitude of th...
location	Geolocation	<input type="checkbox"/>	
GetDistance	Numeric(8.0)	<input type="checkbox"/>	Returns the distance betwe...
fromLocation	Geolocation	<input type="checkbox"/>	
toLocation	Geolocation	<input type="checkbox"/>	
GetAddress	Address	<input checked="" type="checkbox"/>	Returns a collection of addr...
location	Geolocation	<input type="checkbox"/>	
GetLocation	Geolocation	<input checked="" type="checkbox"/>	Returns a collection of locat...
address	Address	<input type="checkbox"/>	
SetProximityAlerts	Boolean	<input type="checkbox"/>	
proximityAlerts	GeoLocationProximi...	<input checked="" type="checkbox"/>	
GetProximityAlerts	GeoLocationProximi...	<input checked="" type="checkbox"/>	
GetCurrentProximityAlert	GeoLocationProximi...	<input type="checkbox"/>	
ClearProximityAlerts	None	<input type="checkbox"/>	
Events			

U obtener la dirección a partir de una geolocalización o viceversa.. configurar alertas de proximidad u obtenerlas.. etc.

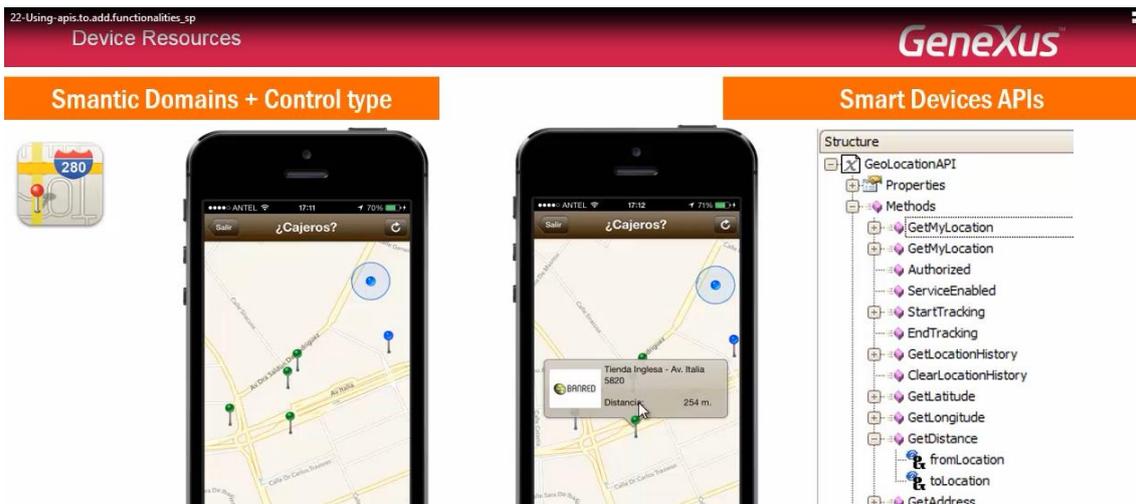
Por tanto, para mostrar solamente los restaurants a menos de –supongamos- 400 metros de nuestro dispositivo, tendremos que utilizar el método GetDistance y el método GetMyLocation



en el grid, a la hora de filtrar la información que mostramos

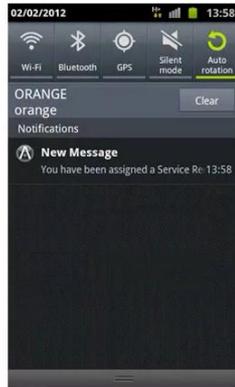


No es ni más ni menos que lo que hace una aplicación para obtener los cajeros automáticos cercanos al punto donde nos encontramos



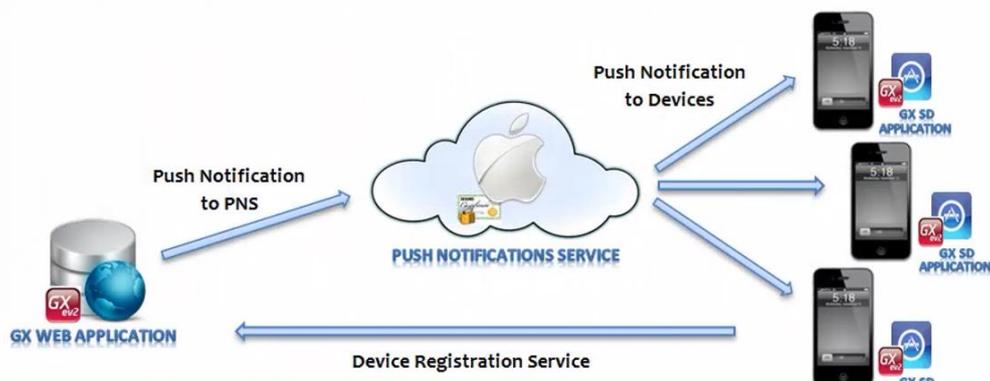
Otra funcionalidad muy utilizada corresponde a las push notifications, notificaciones que pueden enviarse desde el servidor web a los distintos dispositivos que tienen instalados nuestra aplicación móvil, incluso cuando esta no está corriendo

Push Notifications



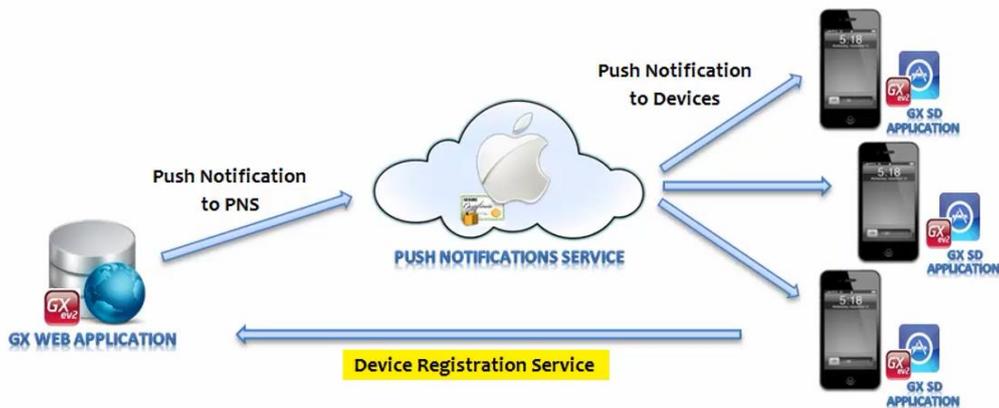
De esta forma, el usuario recibirá la notificación que le envía el servidor y podrá realizar la acción que corresponda.

Push Notifications



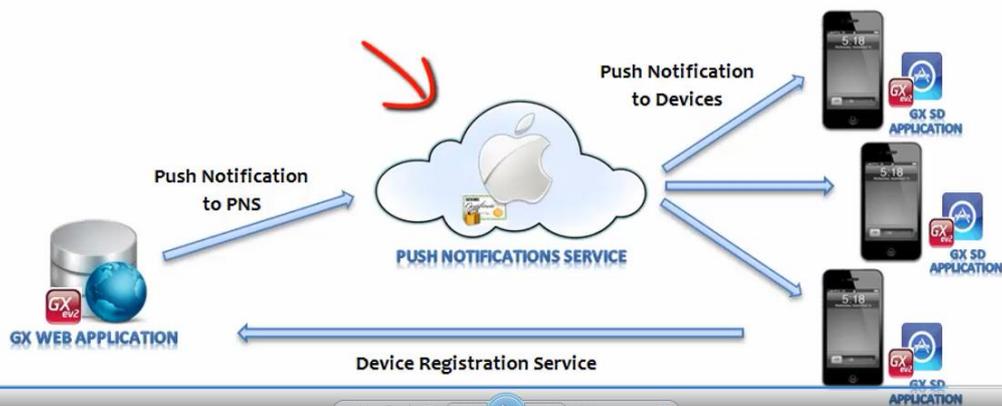
La idea básica es que cuando el usuario se instala la aplicación en el dispositivo y la ejecuta, este se registra en una tabla en el server

Push Notifications



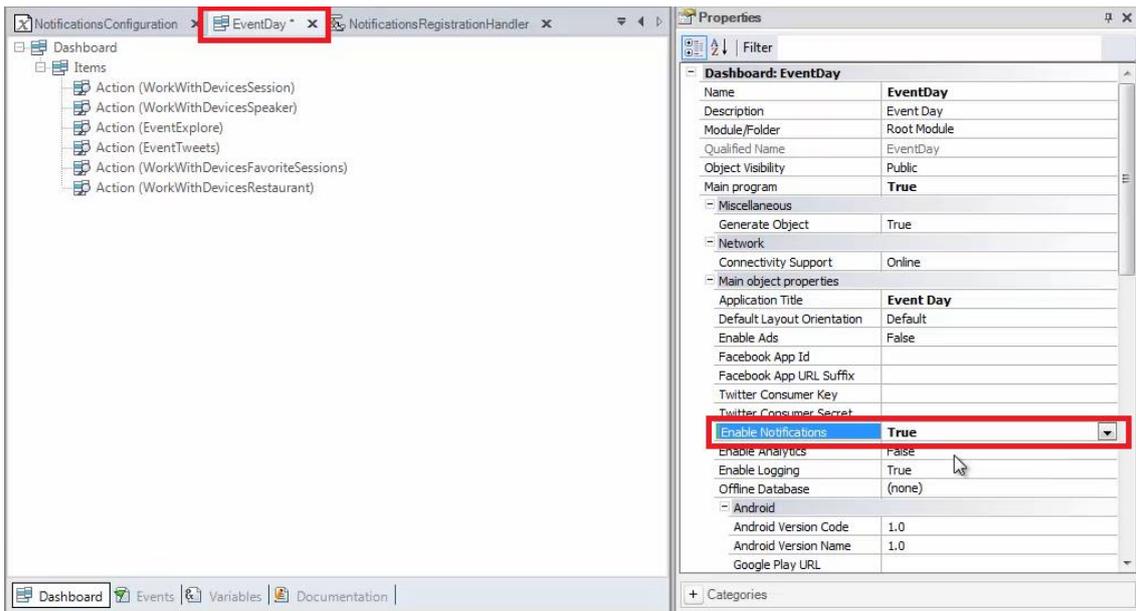
De esta manera, el servidor tiene identificados a los devices para poder mandarles notificaciones. Pero quien envía efectivamente las notificaciones, es un proveedor que depende de cada tecnología

Push Notifications

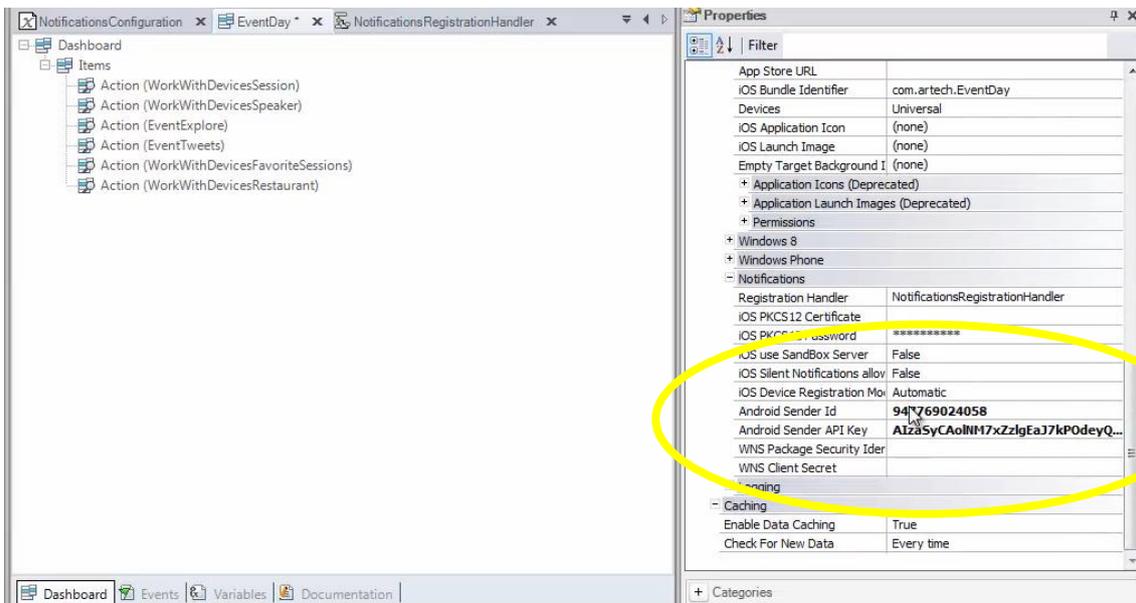


Así el de Apple, se llama Apple Push Notifications Service.. el de Google, Google Cloud Messaging.. el de Blackberry, Blackberry Pushing Service.. y el de Windows, Windows Push Notifications Services.

Lo que debe hacerse es ir a cada uno de estos servicios a crear un proyecto y obtener las credenciales, para luego ir al objeto main de nuestra aplicación móvil y prender –habilitar- las notificaciones

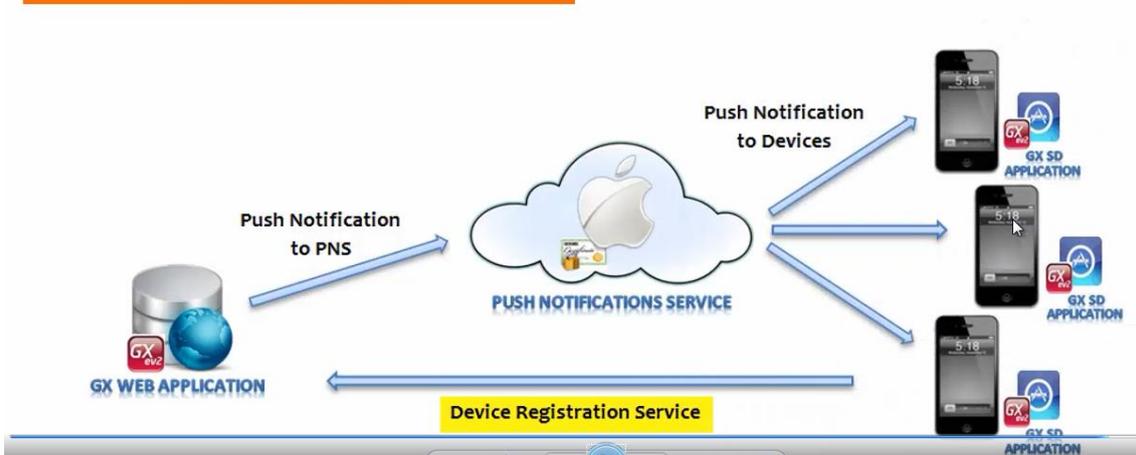


ingresando en las propiedades correspondientes, las credenciales obtenidas

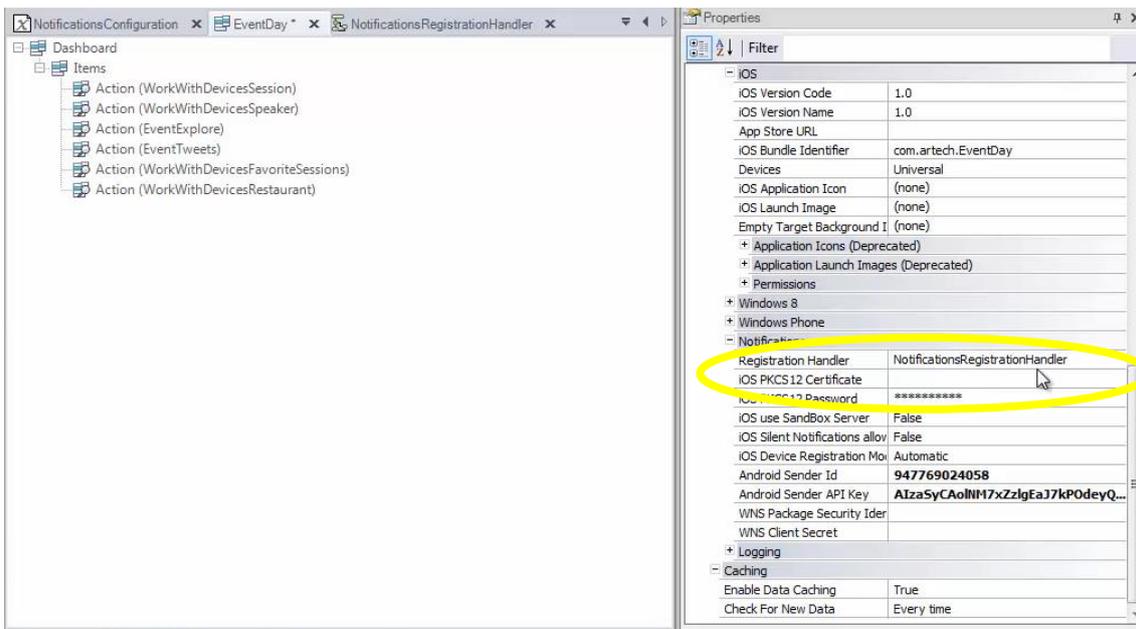


Debemos luego implementar la registraci3n de los dispositivos

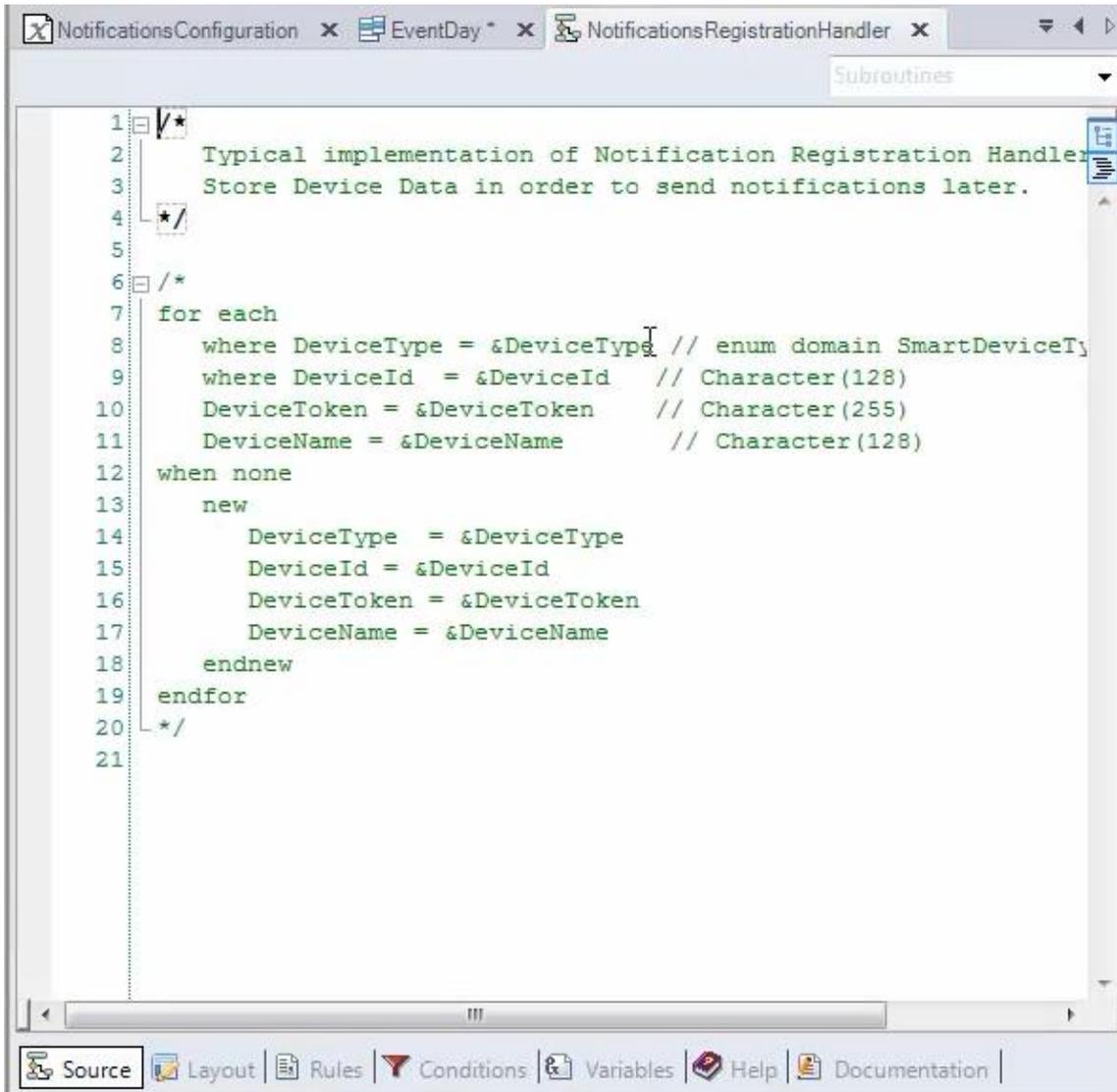
Push Notifications



Para ellos ya se cuenta con el procedimiento predefinido de este nombre



Podemos crear uno nuevo y colocarlo aquí... o utilizar el predefinido



The screenshot shows a code editor window with three tabs: 'NotificationsConfiguration', 'EventDay *', and 'NotificationsRegistrationHandler'. The active tab is 'NotificationsRegistrationHandler'. The code is written in a structured query language (likely Genexus's internal language) and is enclosed in a multi-line comment block. The code defines a 'for each' loop that iterates over device data. Inside the loop, it uses 'where' clauses to define variables for DeviceType, DeviceId, DeviceToken, and DeviceName. It then uses a 'when none' block to create a new record with the specified values. The code is as follows:

```
1  /*
2  Typical implementation of Notification Registration Handler
3  Store Device Data in order to send notifications later.
4  */
5
6  /*
7  for each
8      where DeviceType = &DeviceType // enum domain SmartDeviceType
9      where DeviceId = &DeviceId // Character(128)
10     DeviceToken = &DeviceToken // Character(255)
11     DeviceName = &DeviceName // Character(128)
12 when none
13     new
14         DeviceType = &DeviceType
15         DeviceId = &DeviceId
16         DeviceToken = &DeviceToken
17         DeviceName = &DeviceName
18     endnew
19 endfor
20 */
21
```

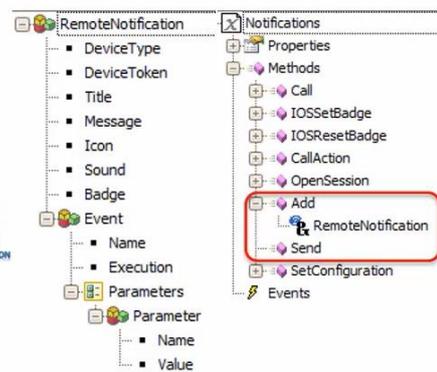
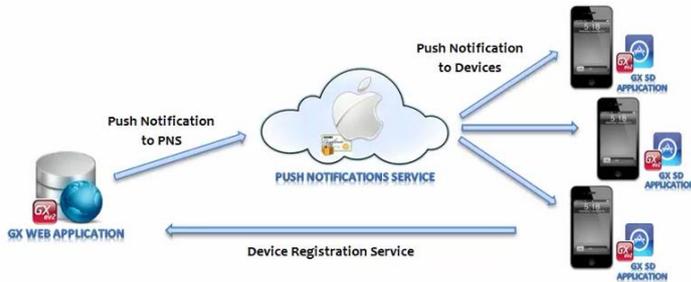
The editor interface includes a 'Subroutines' dropdown menu at the top right and a toolbar at the bottom with icons for Source, Layout, Rules, Conditions, Variables, Help, and Documentation.

creando una transacción que tenga estos atributos, para registrar entonces los dispositivos en el sistema.

Este proc será invocado cada vez que se ejecute la aplicación en el dispositivo.

Smart Devices APIs

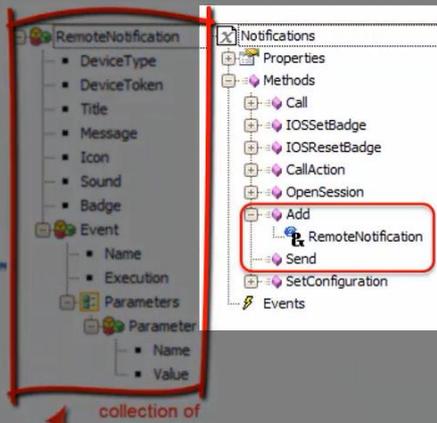
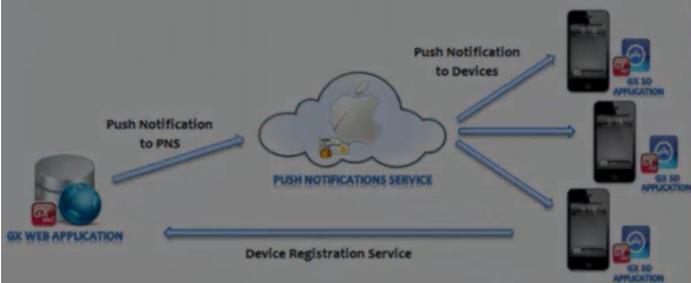
Push Notifications



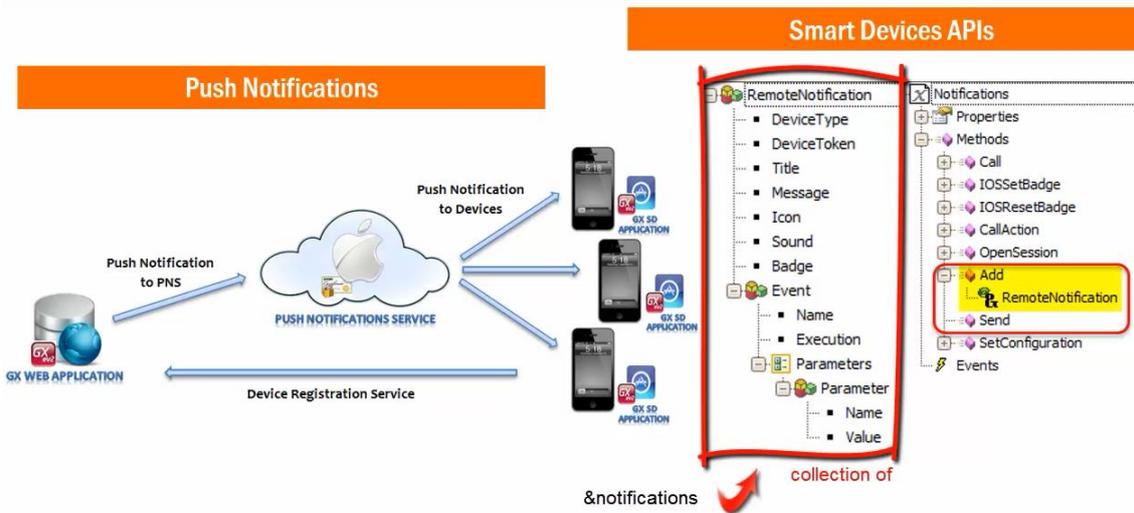
Luego desde el objeto que necesitamos que inicie las push notifications, nos creamos una variable de tipo colección del sdt que viene con la api notification

Smart Devices APIs

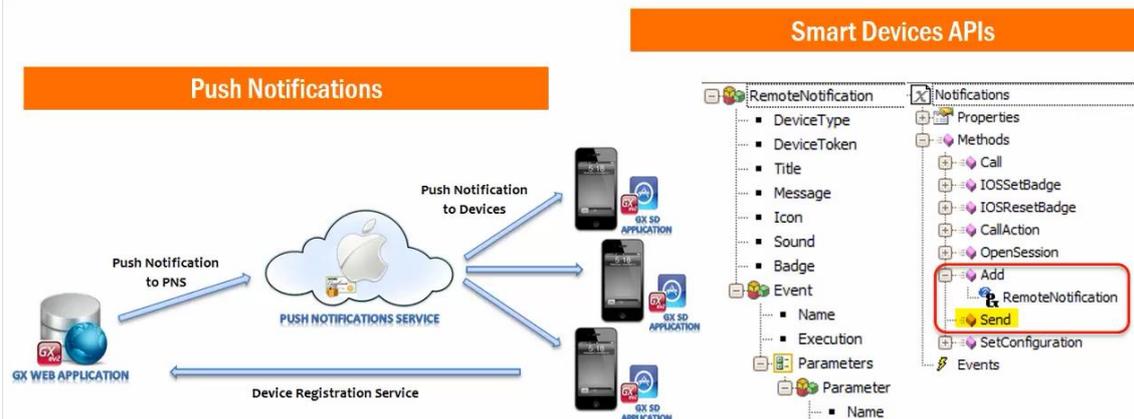
Push Notifications



(sdt de nombre: RemoteNotification) y la vamos cargando a través del método Add de la api:



Con las notificaciones a ser enviadas a cada dispositivo registrado. Luego de cargada esta variable colección, enviamos las notificaciones haciendo uso del método Send

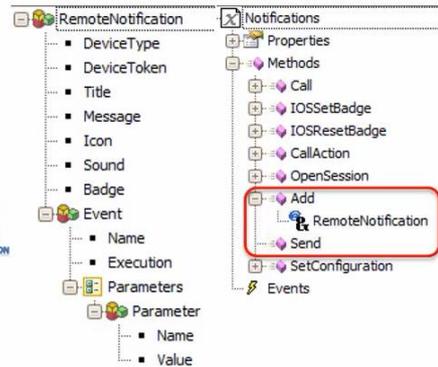
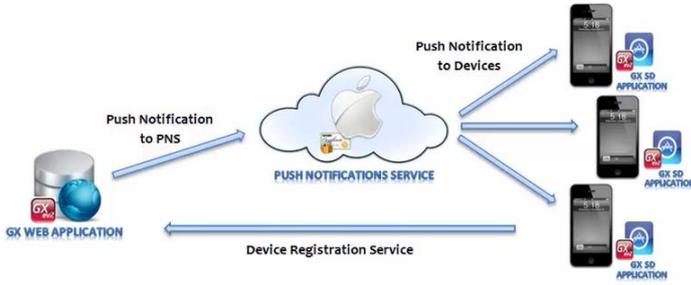


Este método en capsulará la invocación a los servicios de notificaciones de cada tecnología.

Vea más sobre push notifications en nuestro wiki:

Smart Devices APIs

Push Notifications

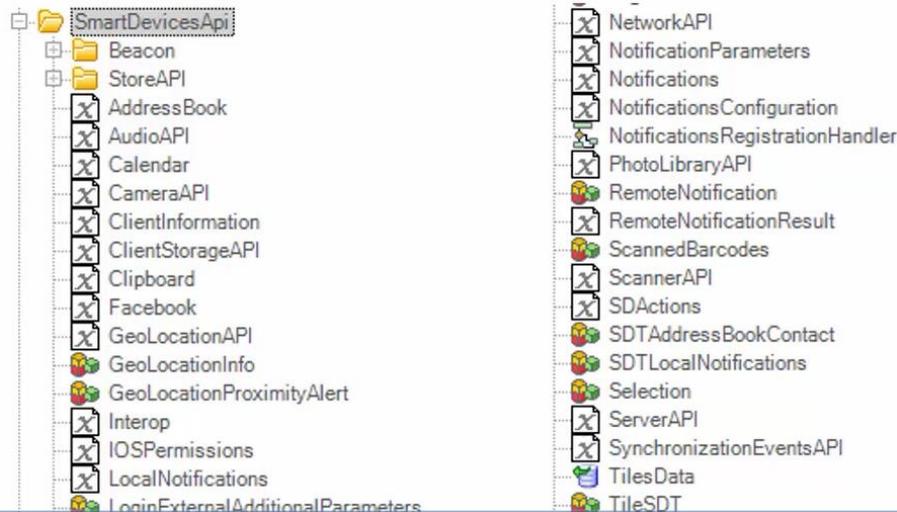


<http://wiki.genexus.com/commwiki/servlet/hwikibypageid?19945>

Hemos presentado en este video

22-Using-apis.to.add.functionalities.sp
 Device Resources GeneX

Smart Devices APIs



Sólo algunas de las muchas apis que ofrece geneXus, y que van aumentando día a día.

Para obtener información de las demás api's o acceder a detalles de las apis aquí vistas, lo invitamos a dirigirse a nuestro wiki.

