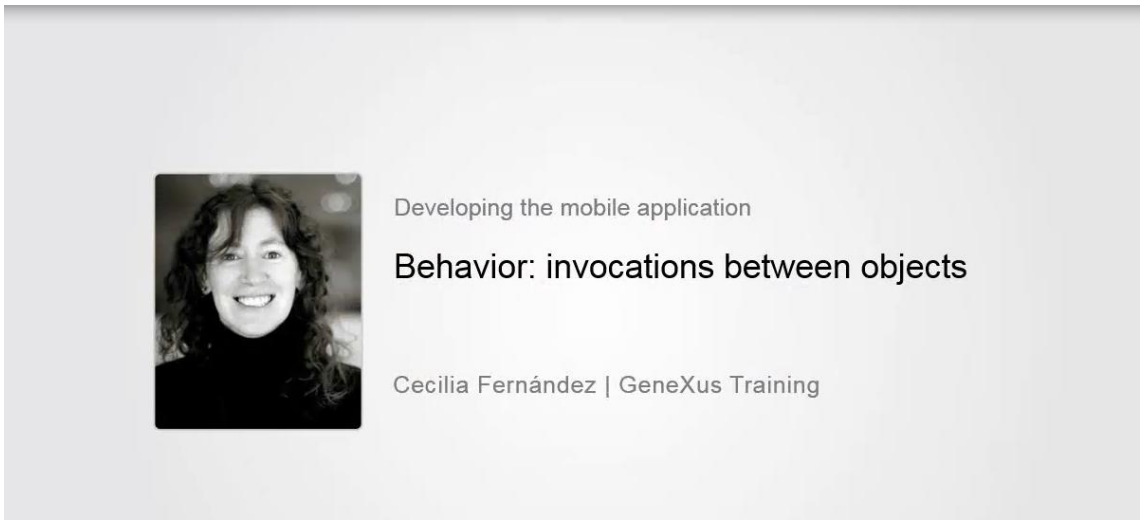
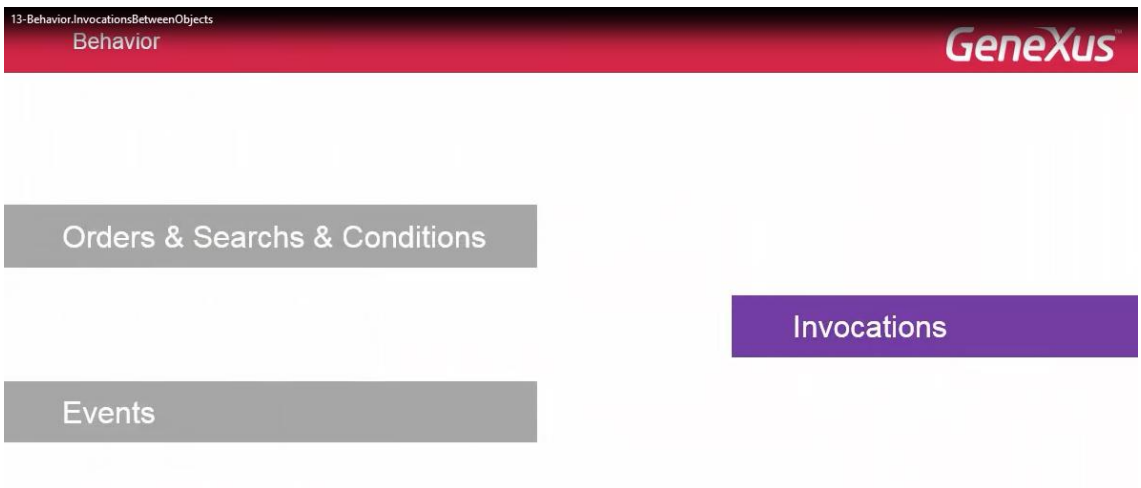
BEHAVIOUR : INVOCATION BETWEEN OBJECTS

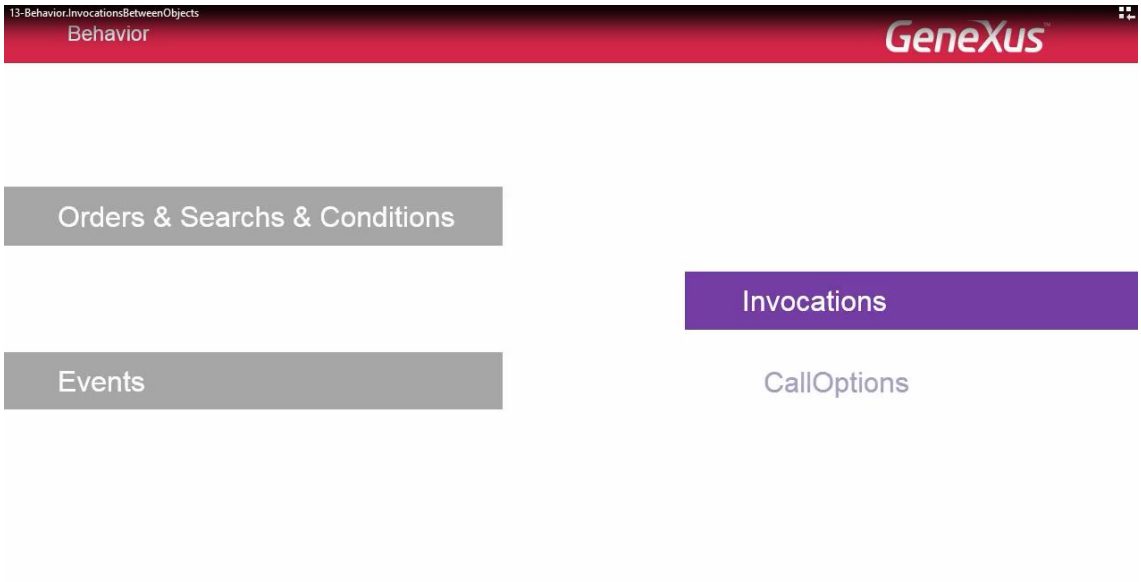




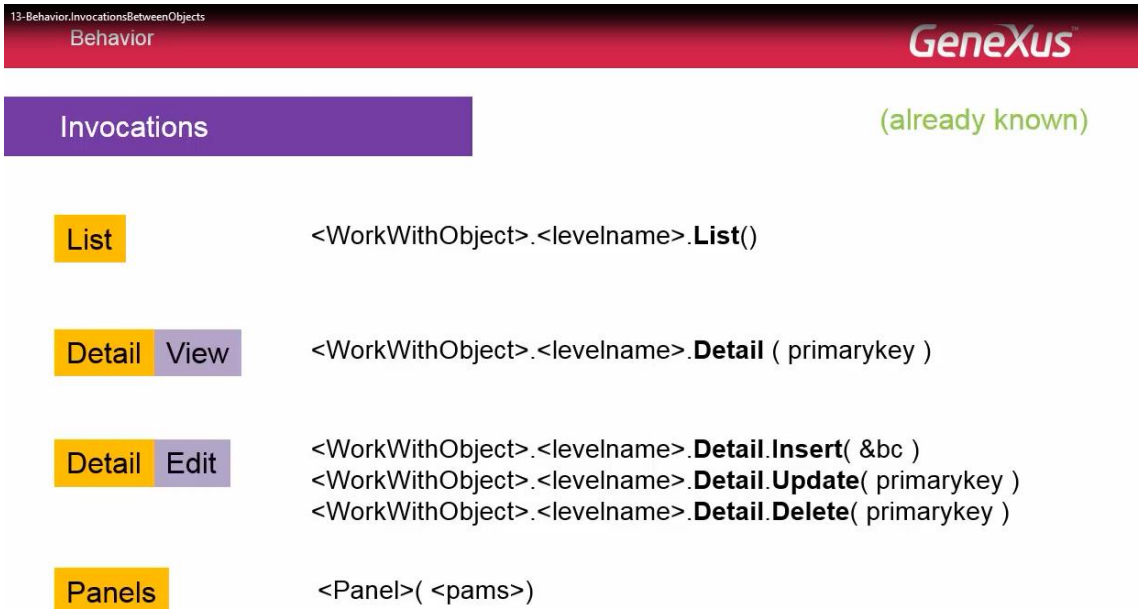
En este video nos detendremos en las opciones que tenemos al invocar desde un objeto a otro con interfaz.

Video filmado con GeneXus X Evolution 3

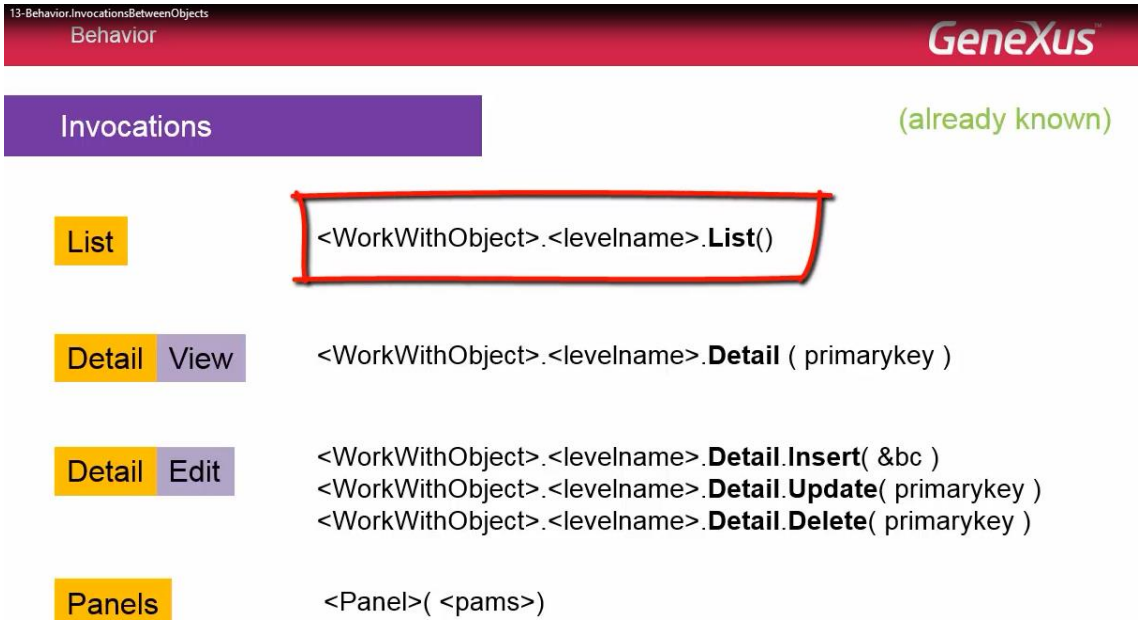
Veremos que podemos hacer la invocación, especificando en runtime, la transición con la que se abrirá y cerrará el objeto llamado, el comportamiento respecto al tipo de call.. y la posición en la pantalla que ocupará el objeto llamado. Esto se realiza a través de lo que conocemos como **CallOptions, opciones de invocación**



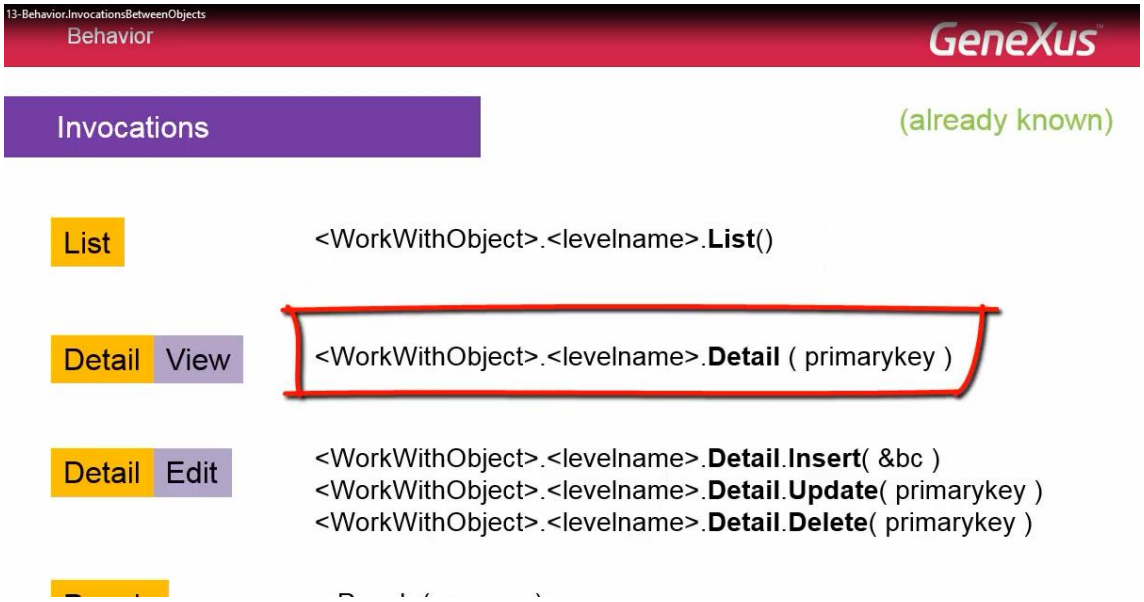
Repasemos las sintaxis de las invocaciones a los work with y a los paneles que son los principales objetos con interfaz



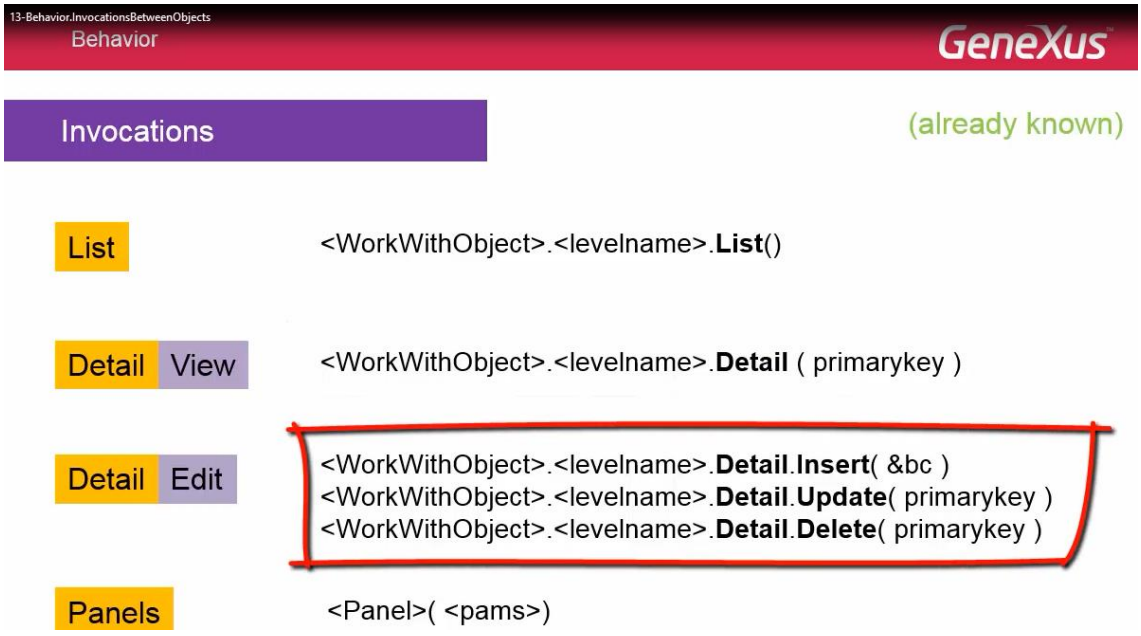
Al List de un WorkWith:



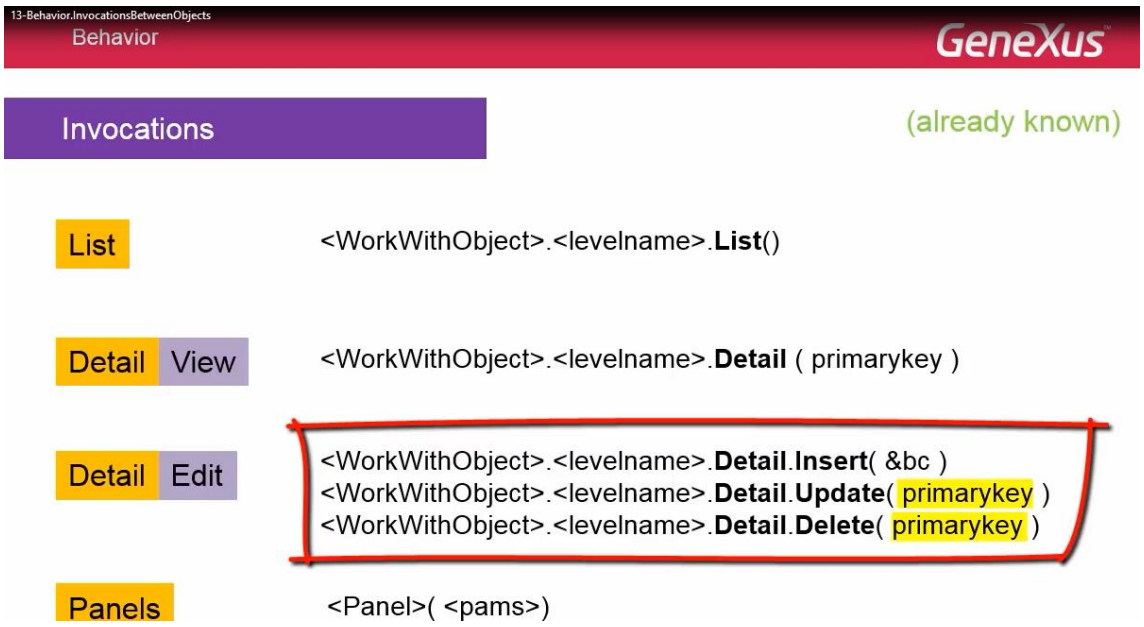
Al Detail en modo View (se le pasa por parámetro la clave primaria):



Al Detail en modo Edit (aquí debemos especificar en la invocación si se desea insertar, actualizar o eliminar):



Si la operación que vamos a realizar es update o delete, pasamos los parámetros correspondientes a la clave primaria



El caso de insert es especial.

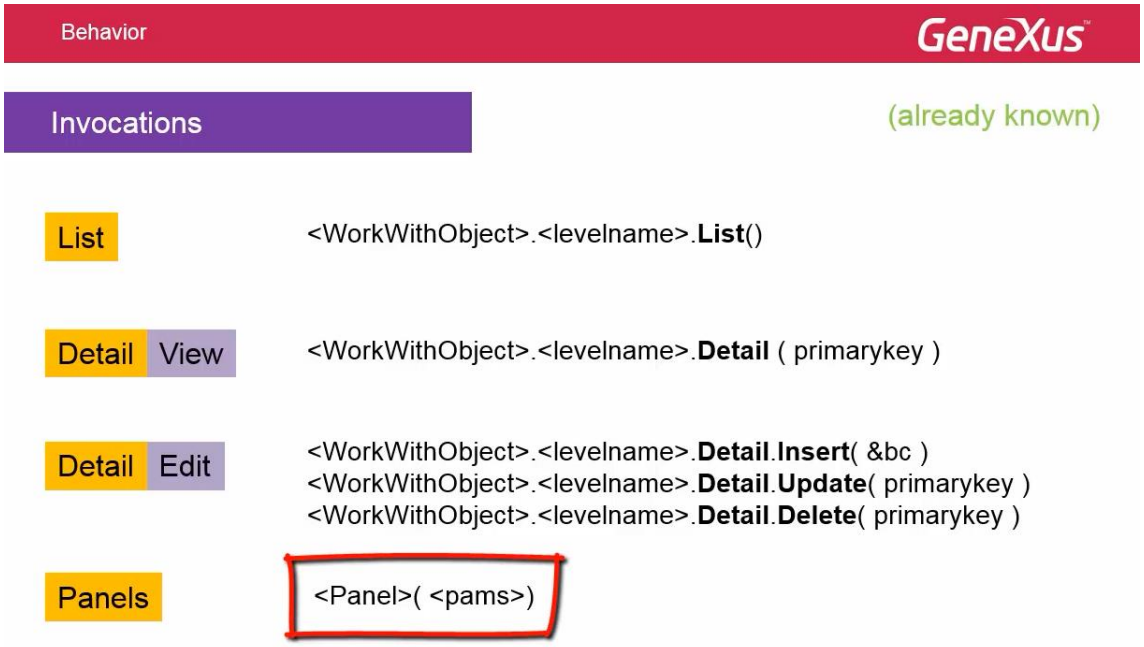
Podemos no pasar parámetros, en cuyo caso el usuario ingresará todos los datos en la pantalla Detail-Edit invocada… pero si necesitamos pasar algún parámetro para que se inicialicen los valores al invocar a la pantalla de Detail-Edit, debemos hacerlo a través de una variable de tipo de datos: el business component asociado.

Son in-out.

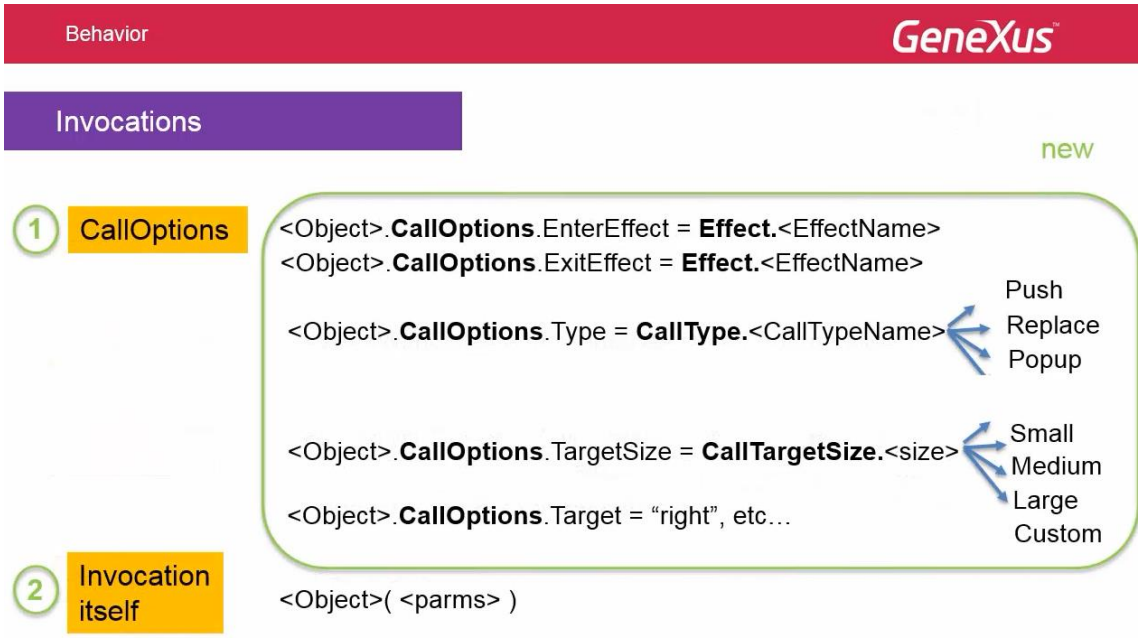
Out para después quedarse con la clave primaria si es autonumber por ejemplo. Recordemos que podemos inicializar cualquiera de los valores. No sólo los correspondientes a la clave primaria.

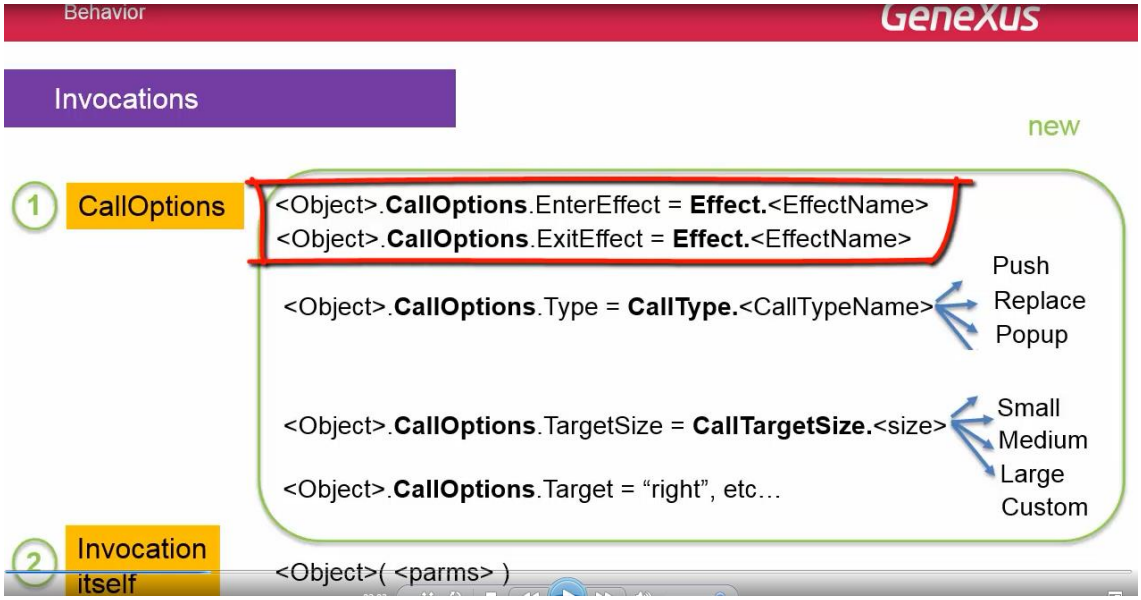
El business component volverá cargado entonces, con los datos ingresados.

La invocación a un panel es igual a cualquier otra invocación

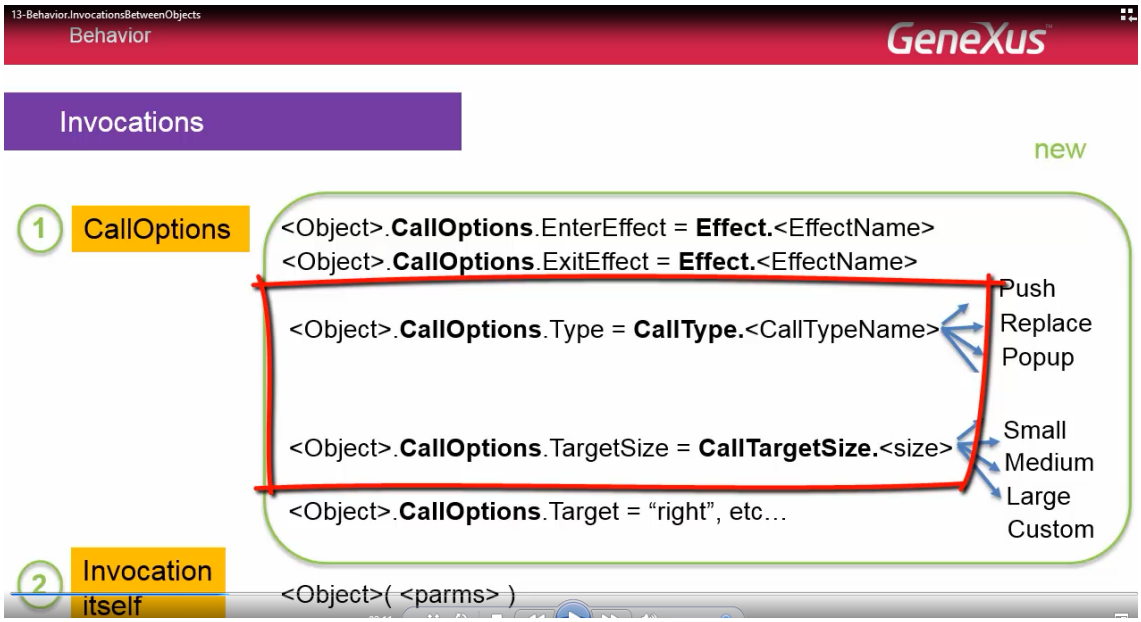


Pero además de invocar al objeto, podemos hacerlo, decíamos, especificando en runtime, la transición de entrada y salida para la interfaz de usuario de la pantalla llamada

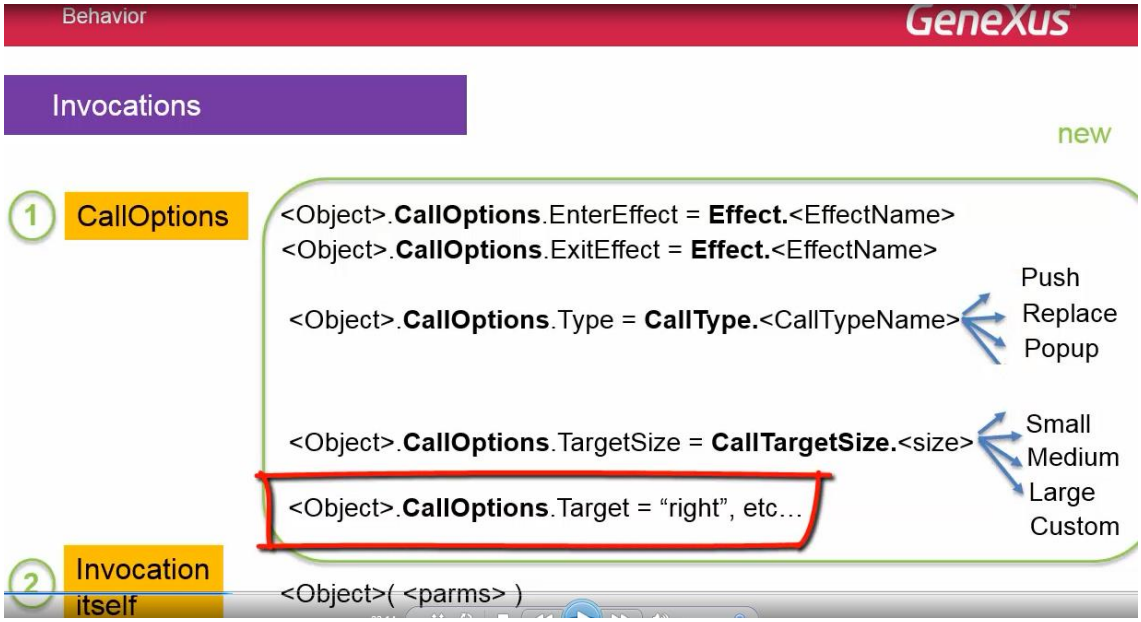




el comportamiento respecto al tipo de call

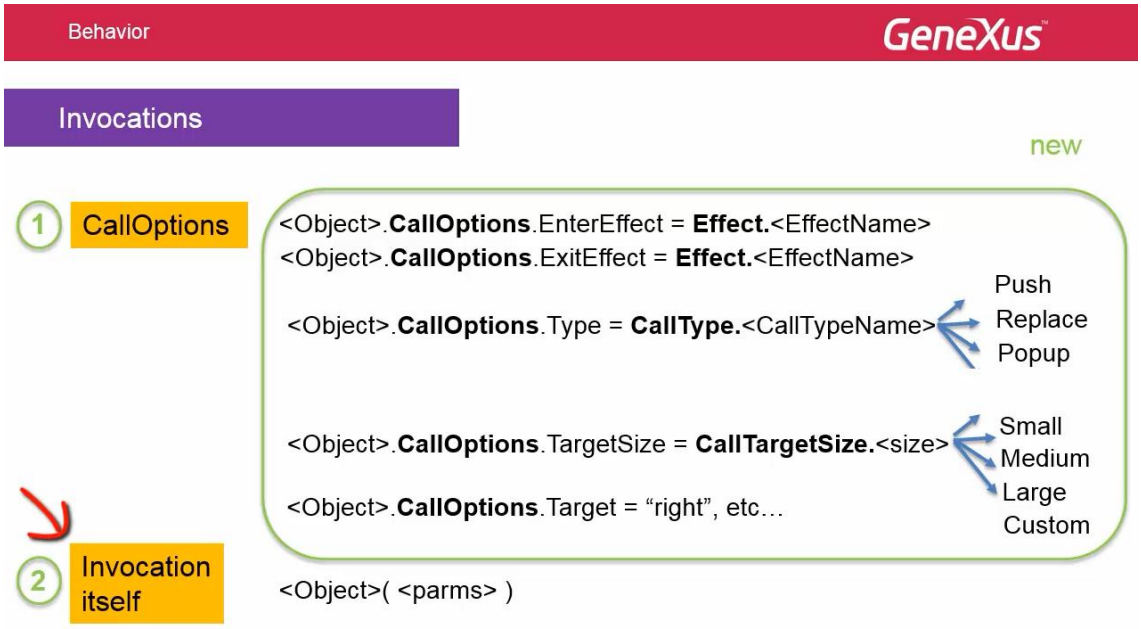


y la posición (target)

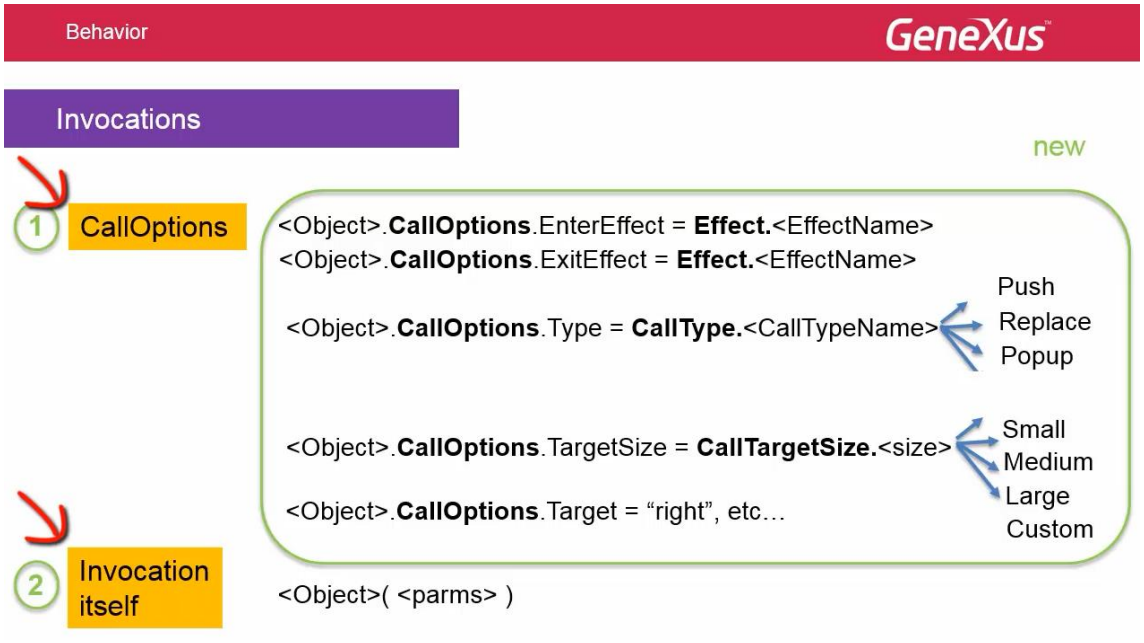


que ocupará el objeto llamado, en el área de pantalla del dispositivo.

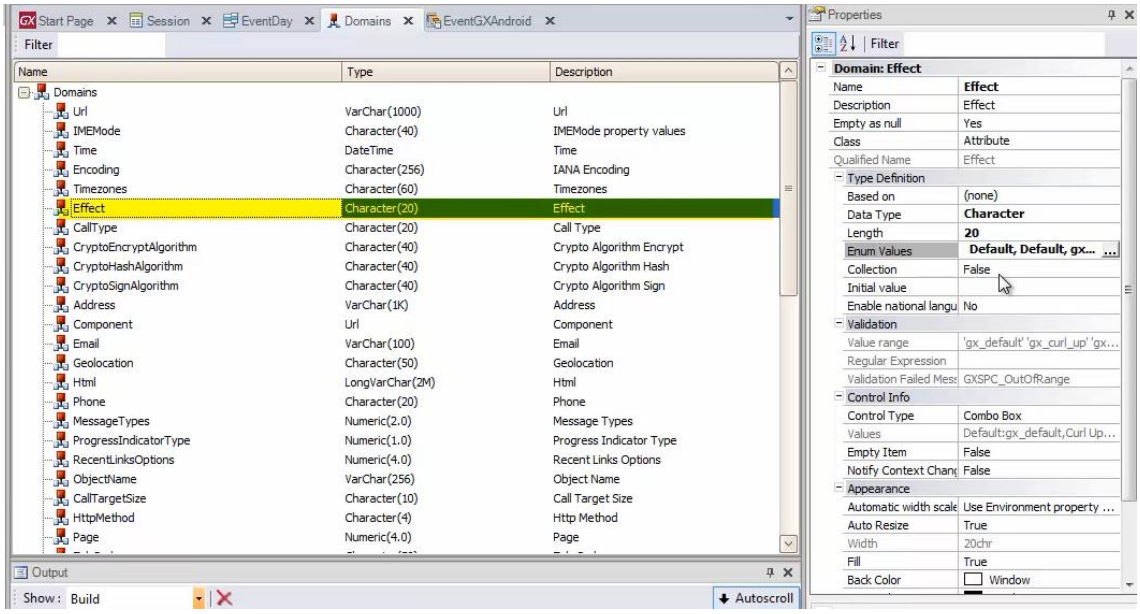
Es por eso que antes de hacer la invocación (con la sintaxis que vimos antes)



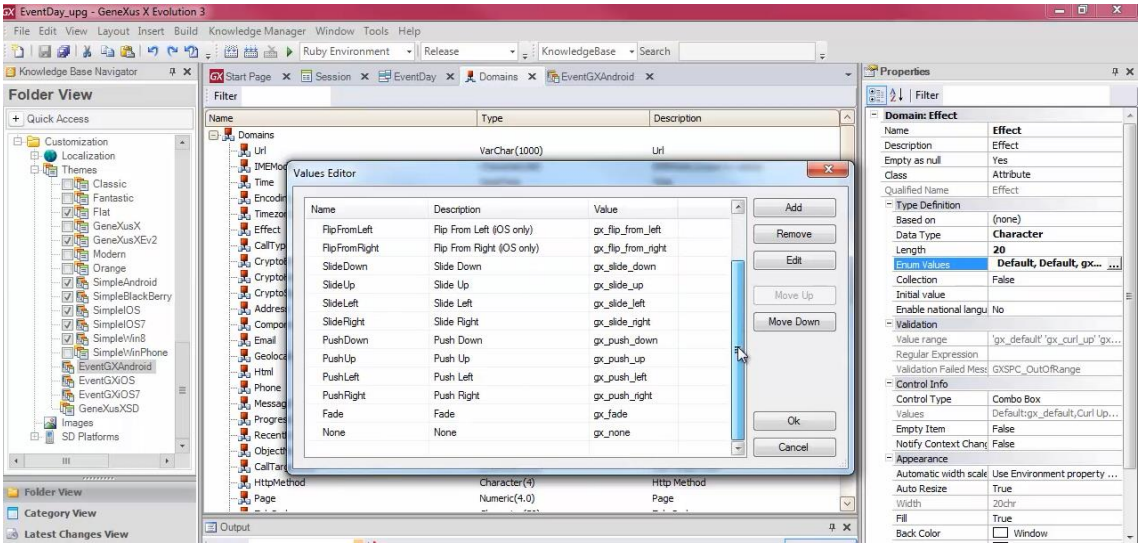
hay que ejecutar las CallOptions deseadas



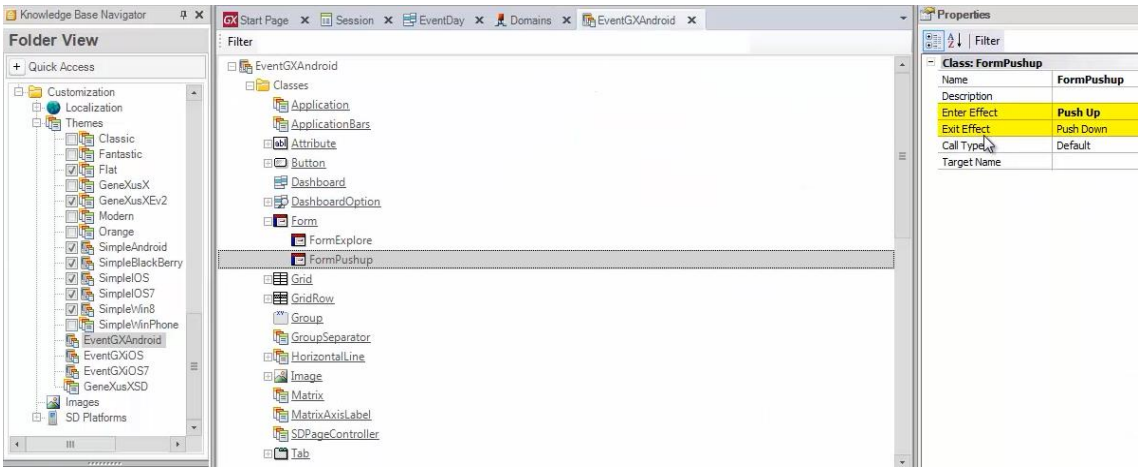
Por ejemplo, existe el dominio predefinido: Effect



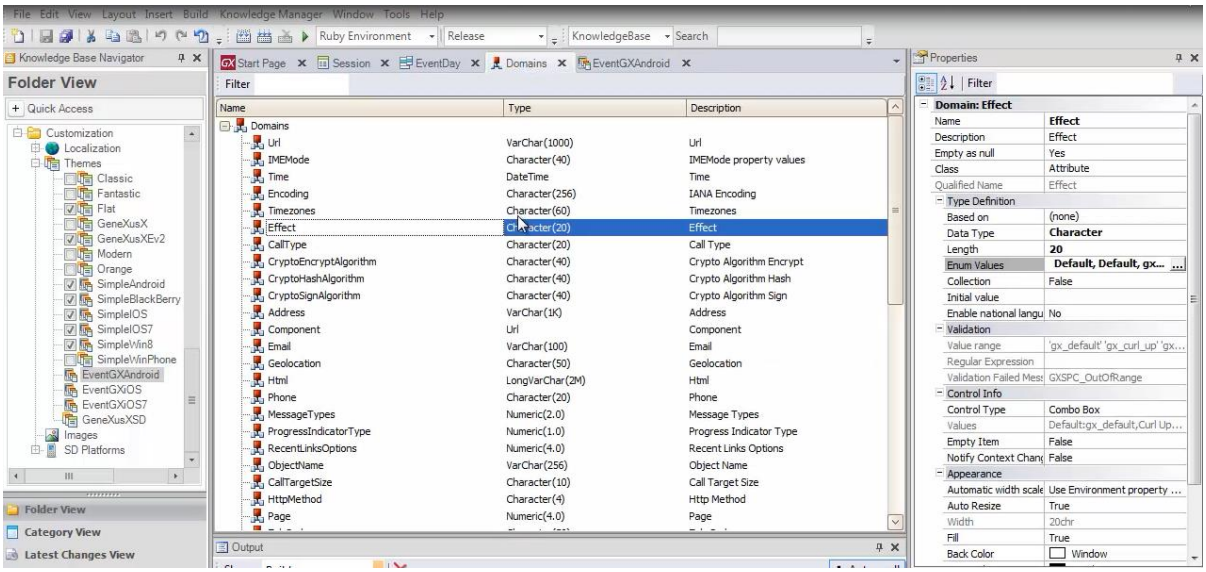
Enumerado.. que ofrece como vemos..



las mismas posibilidades de transición, que las propiedades Enter Effect y Exit Effect de las clases Form del theme

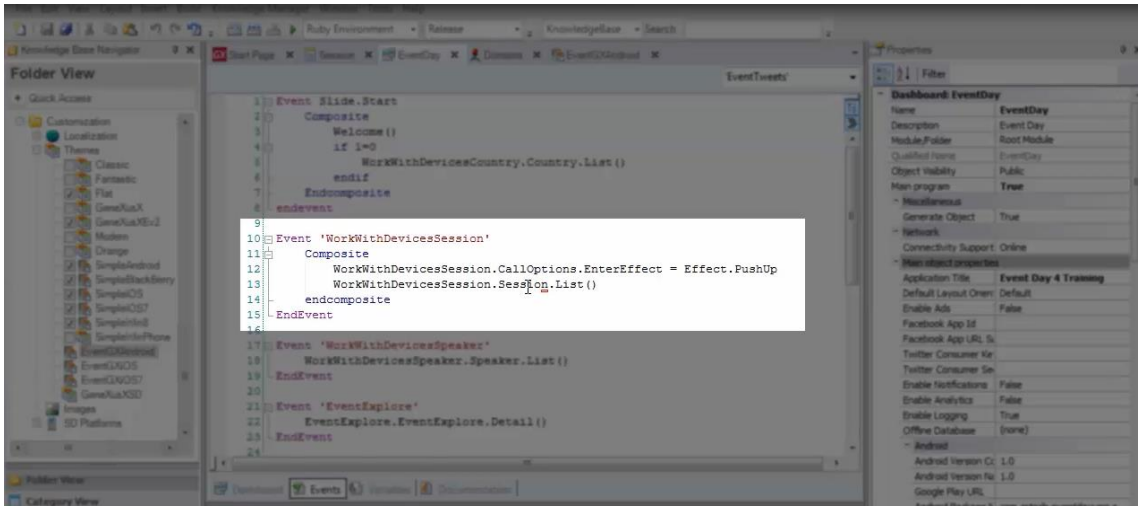


que usábamos para definir en tiempo de diseño el efecto de entrada y salida que tendría el layout de un objeto.



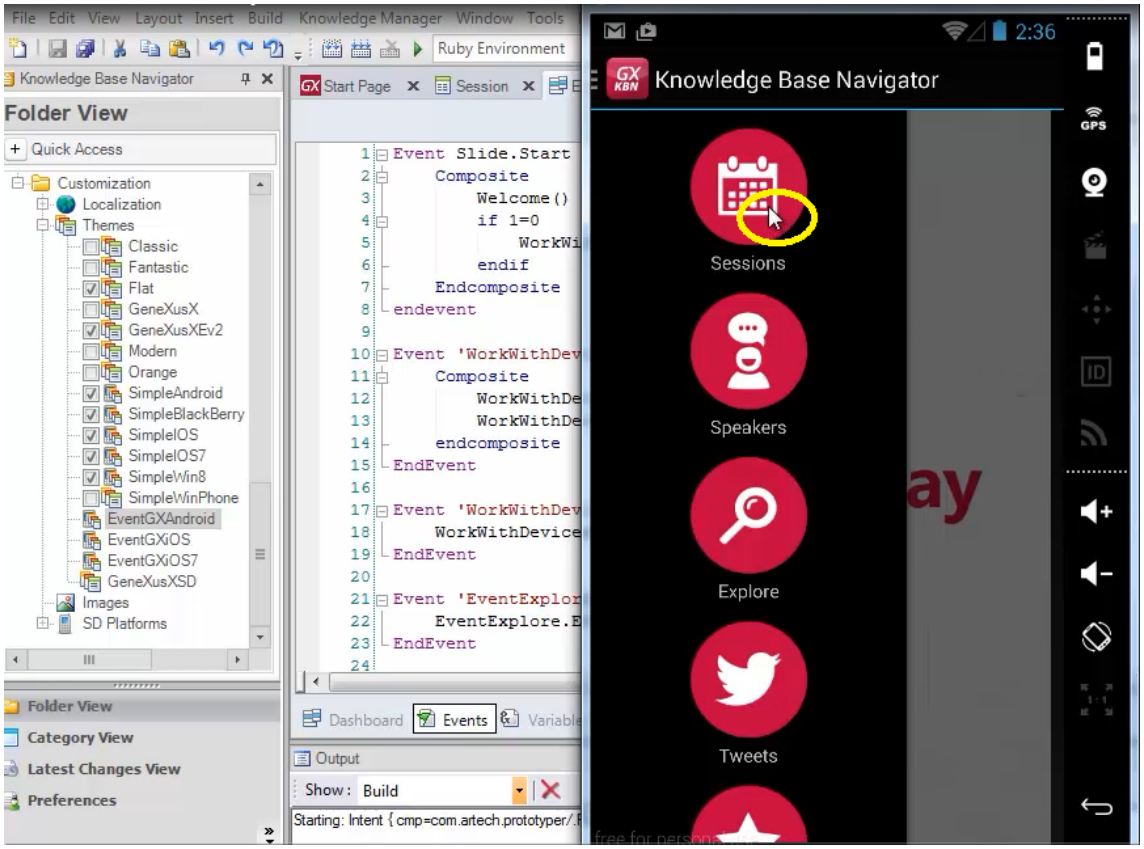
De esta forma, entonces, podemos antes de realizar la invocación a un objeto, definir sus call options Enter Effect y Exit Effect.

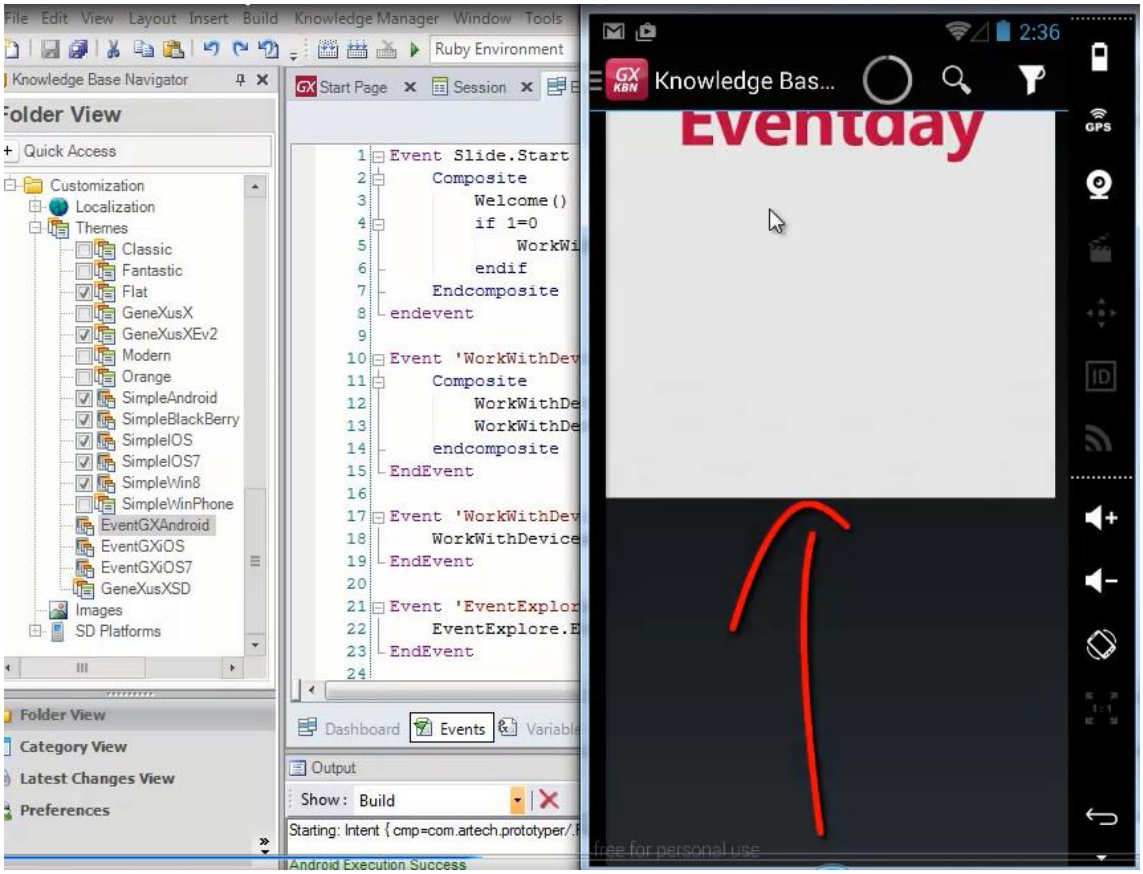
Lo hemos implementado dentro del Dashboard para hacer la invocación al List de Session



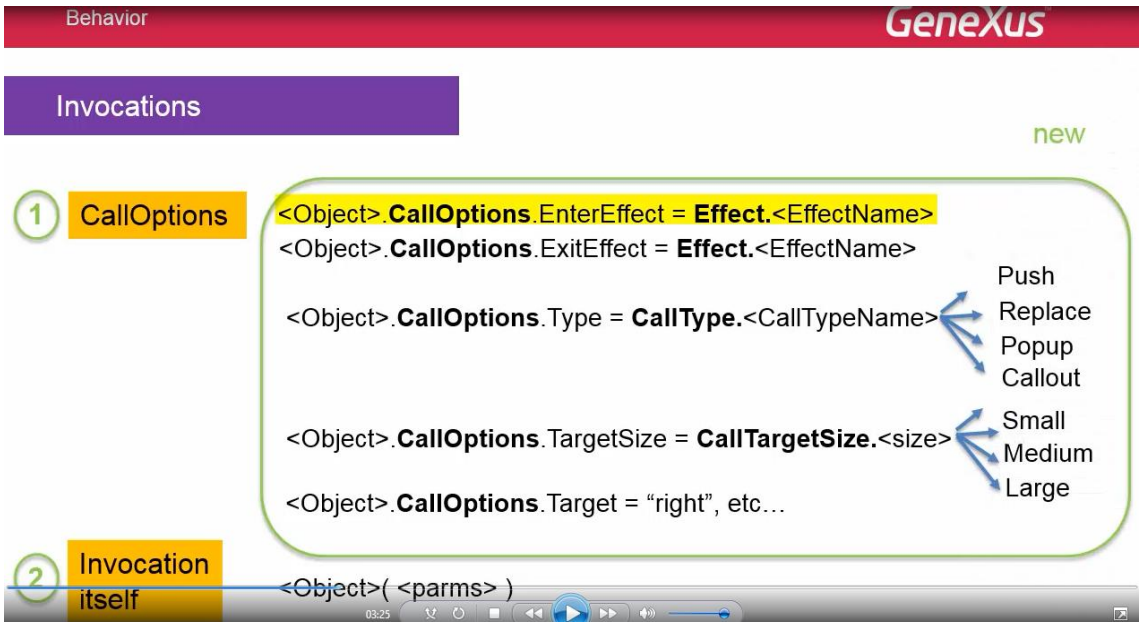
Vemos que hemos especificado entonces entre sus CallOptions, el EnterEffect = PushUp.

Si lo vemos en ejecución

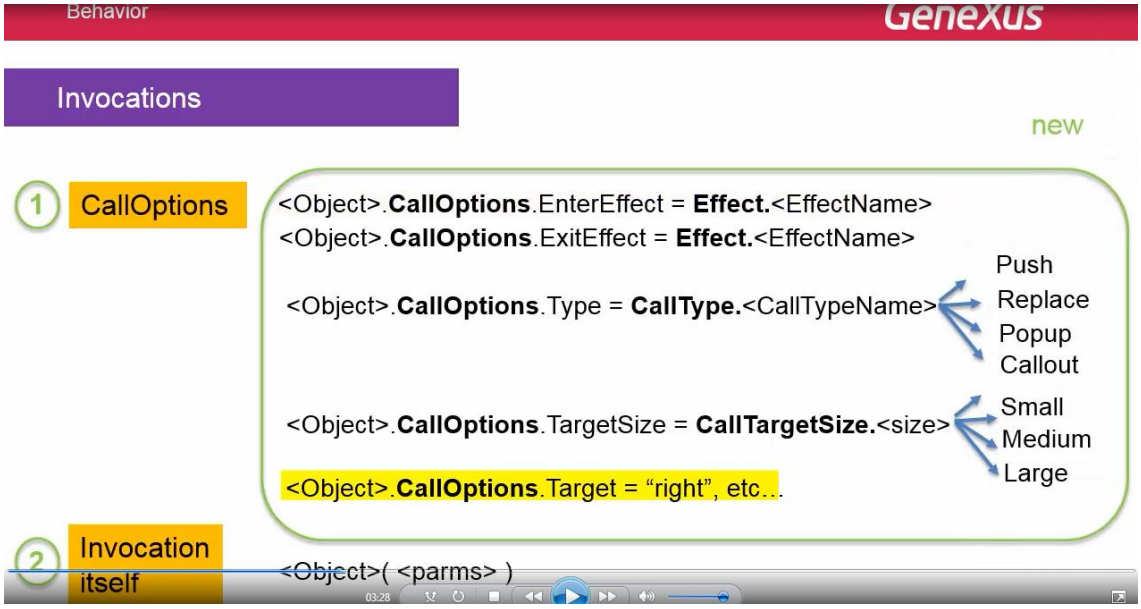




lo vemos operativo.

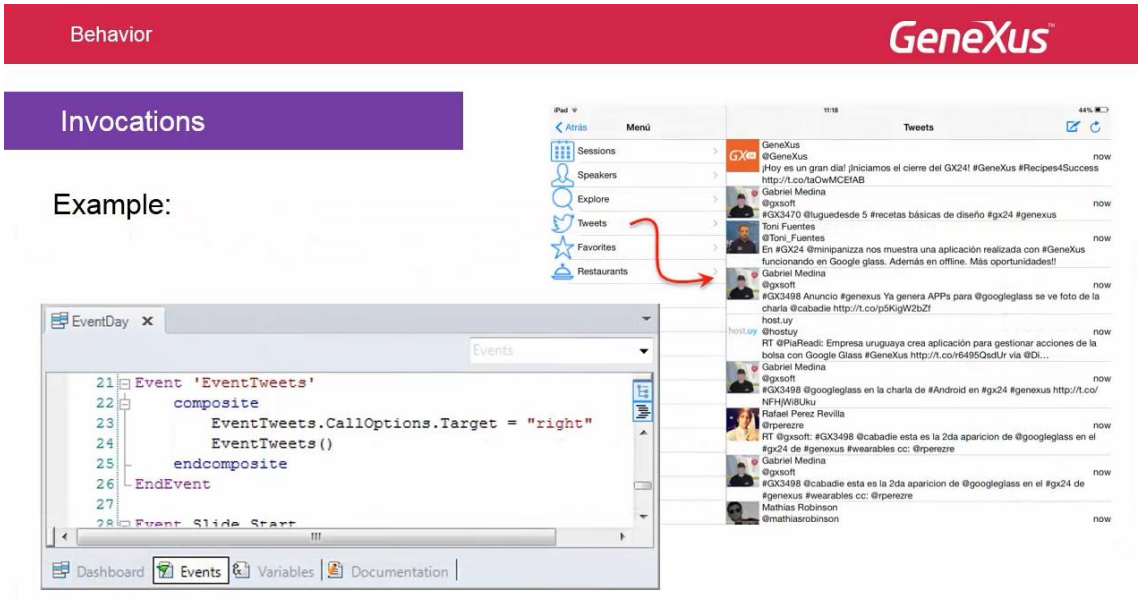


Tenemos también implementada en nuestra aplicación

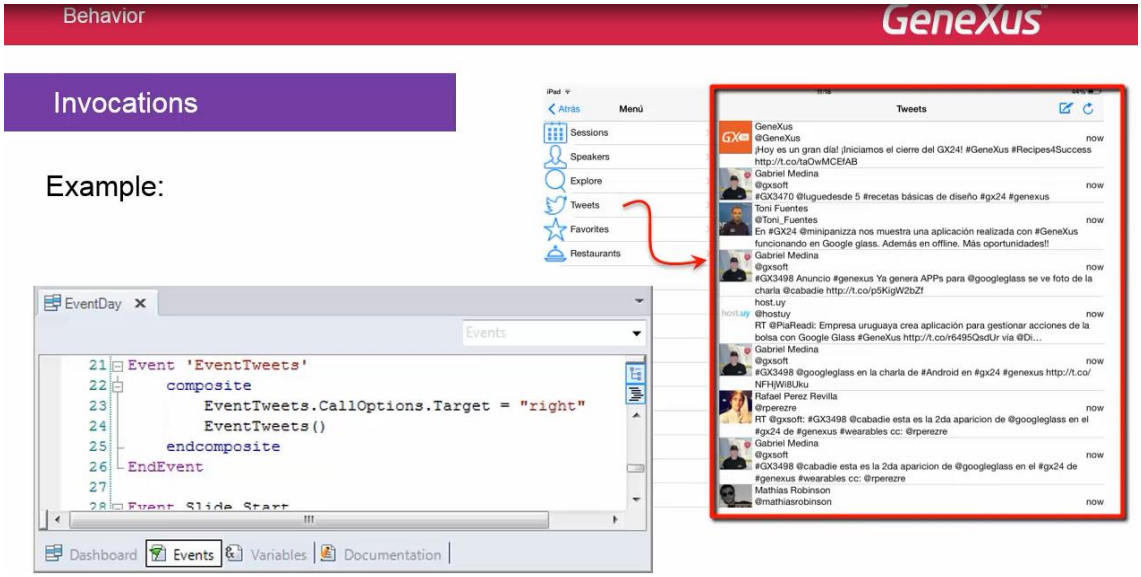


La CallOption : Target

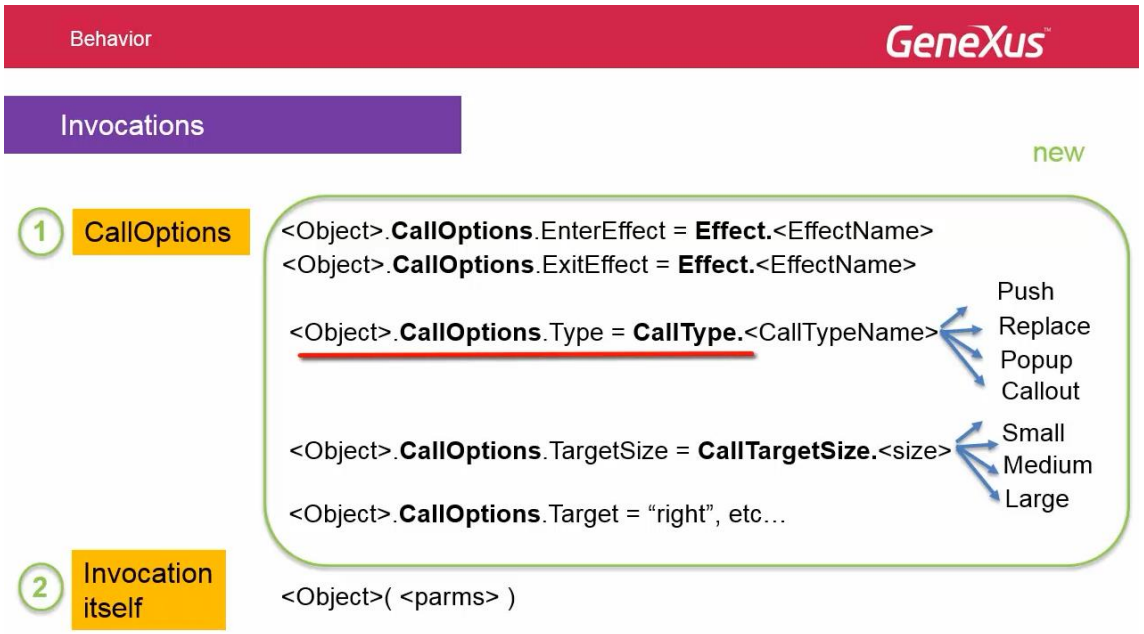
Aquí vemos el ejemplo:



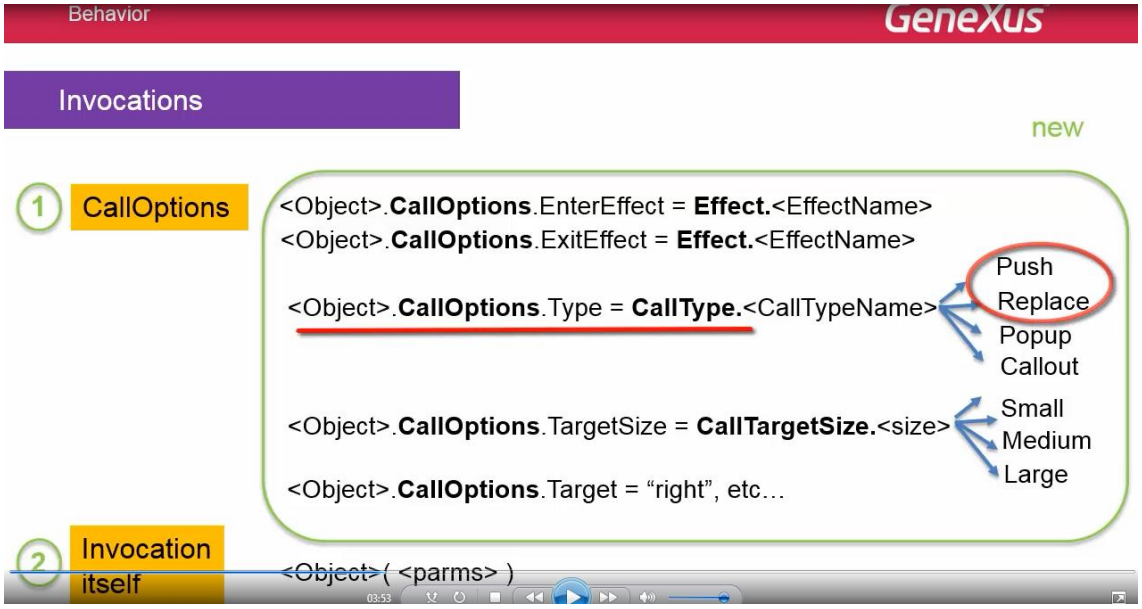
donde queremos que el panel que muestra los tweets, se abra en el área de la derecha



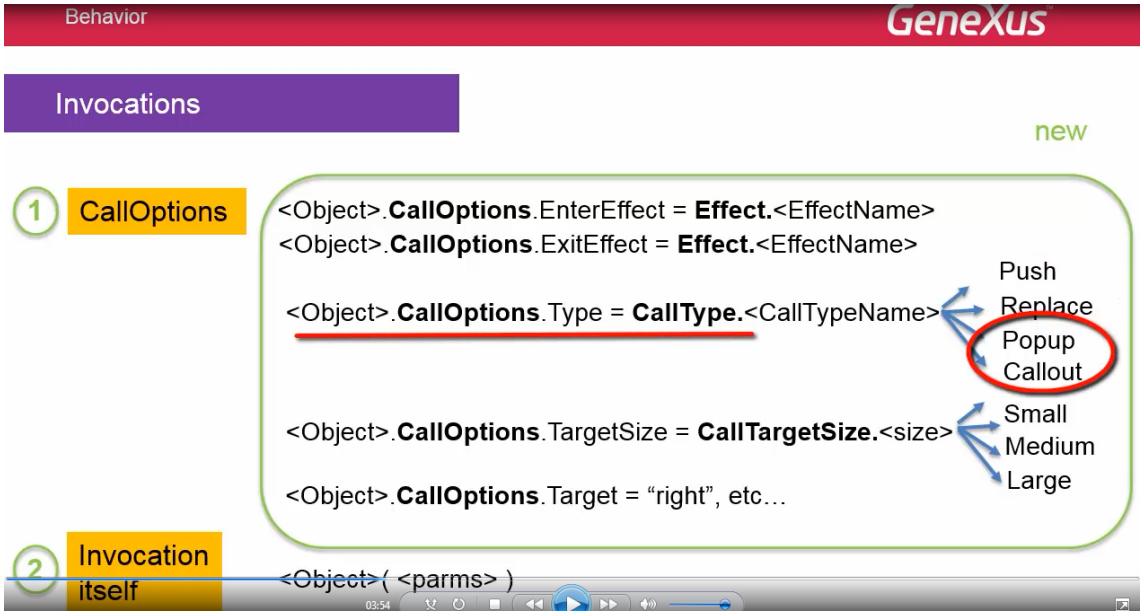
Sólo tiene sentido cuando se trata de estilos de navegación donde existen múltiples targets para una misma pantalla.



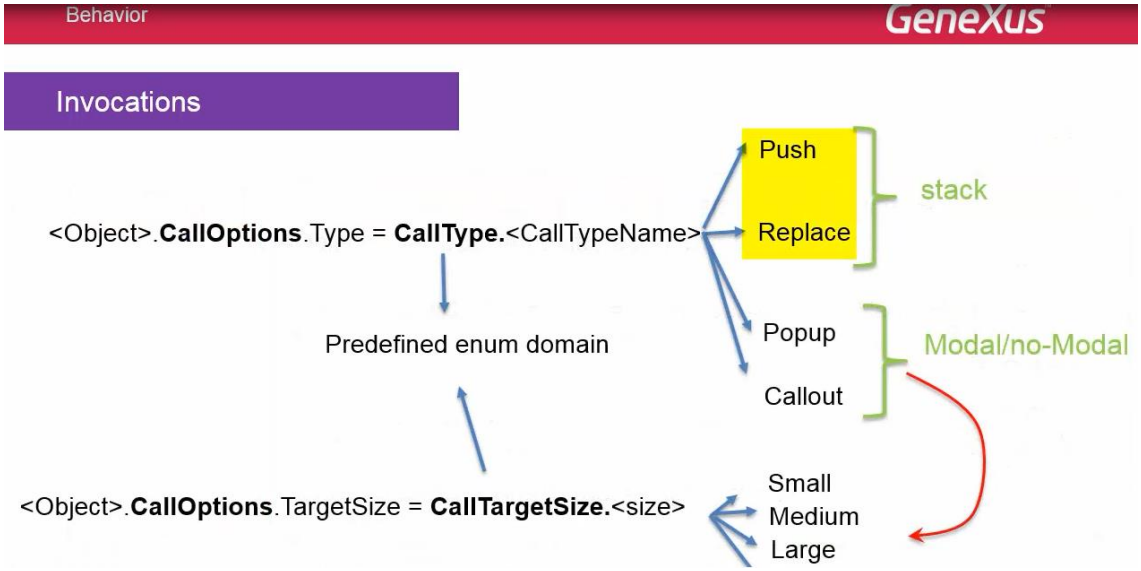
También podemos modificar el comportamiento de la llamada, **respecto al tipo de call**, que tendrá que ver con el stack de invocaciones:



y con el funcionamiento del objeto llamado:

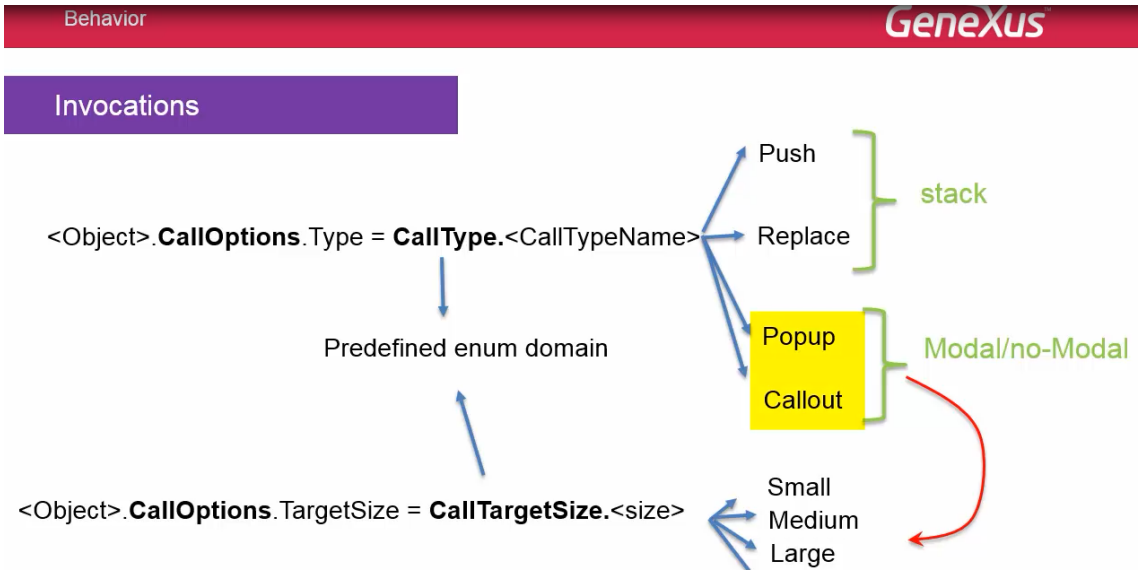


Los primeros dos tipos presentados



definen qué va a suceder con el stack de invocaciones cuando se haga la llamada… que tendrá que ver **con a qué objeto se vuelve al finalizar la ejecución del llamado** o **al hacer back.**

Los últimos dos tipos, definirán si el objeto llamado

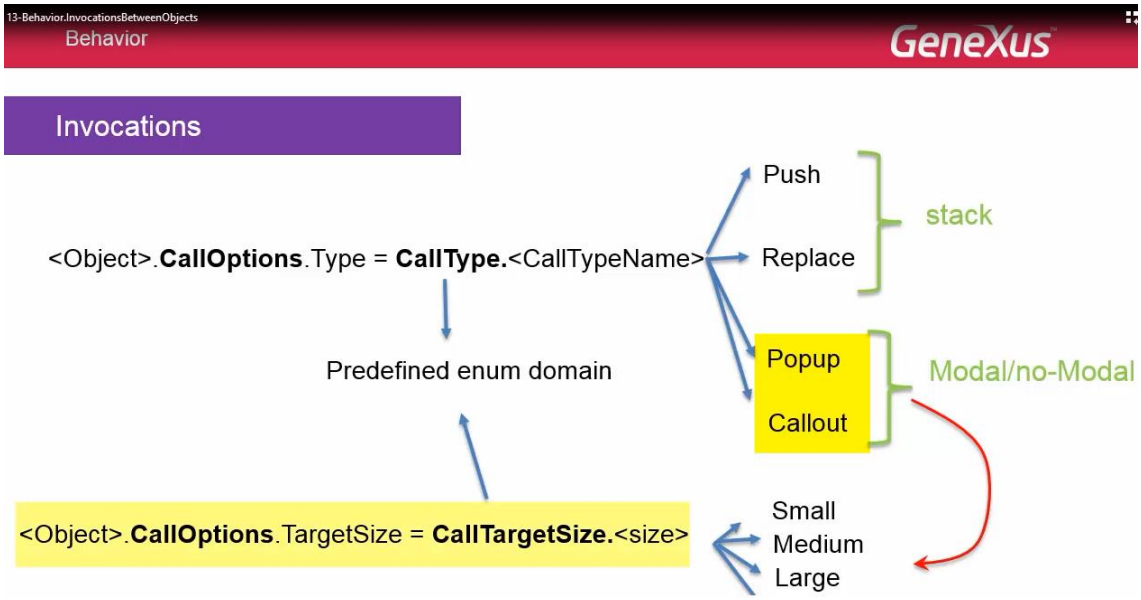


Funcionarán como una ventana popup o como callout.

Para popup además, la pantalla será modal o no, dependiendo de si hay parámetros devueltos o no los hay.

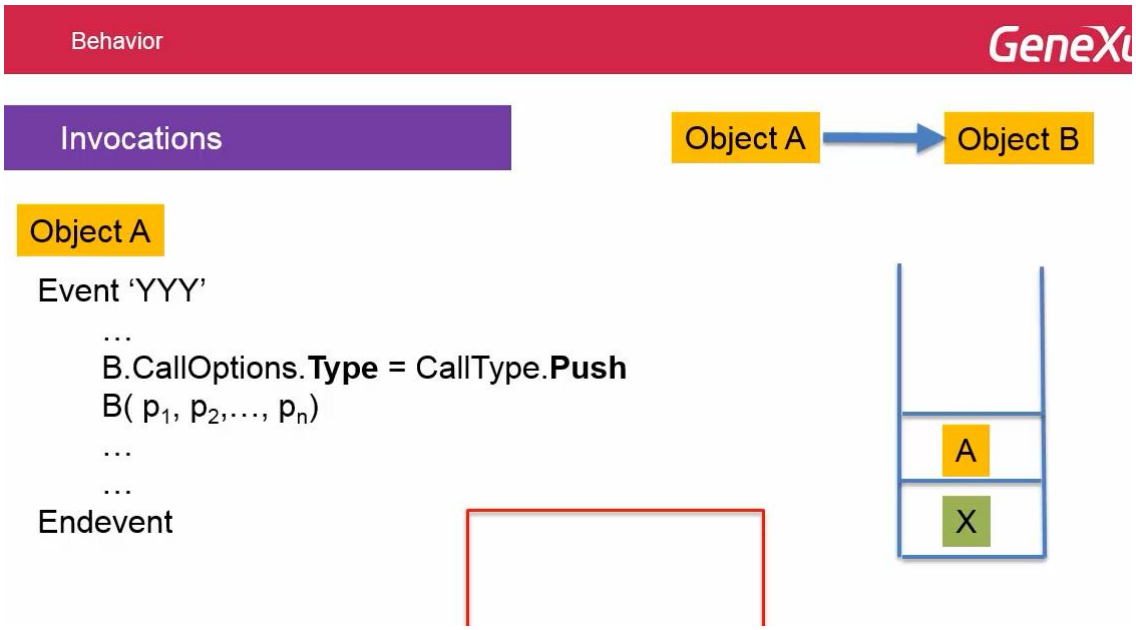
En el caso de tipo Callout, será no modal. Haciendo TAP fuera del área del callout, se habilitará al llamador.

Es justamente para el caso de Popup o Callout que aparece la otra CallOption: TargetSize



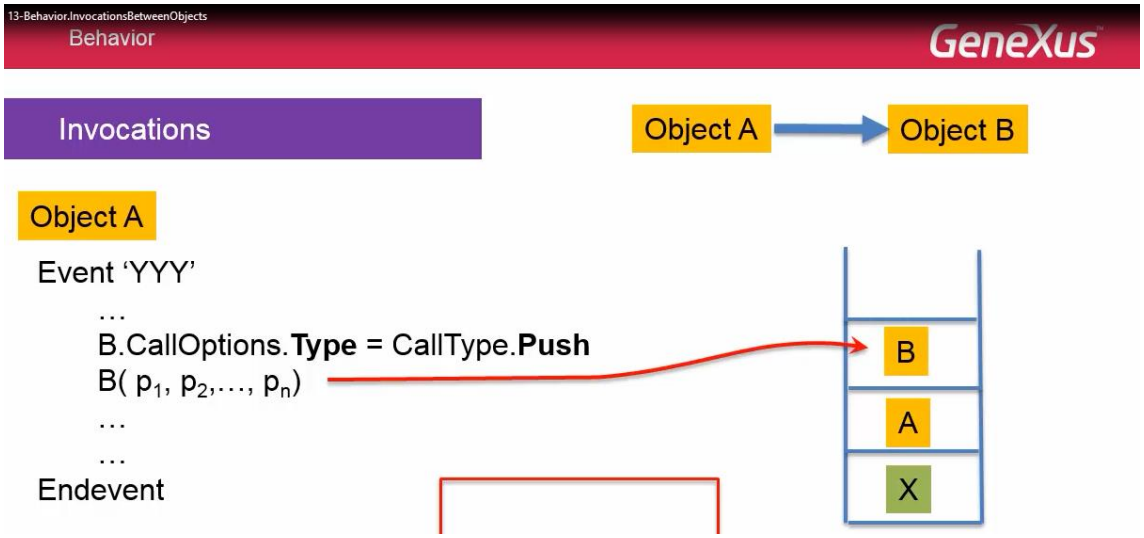
para indicar el tamaño de la pantalla Popup o Callout.

Respecto a los tipos Push y Replace, Push es el default.

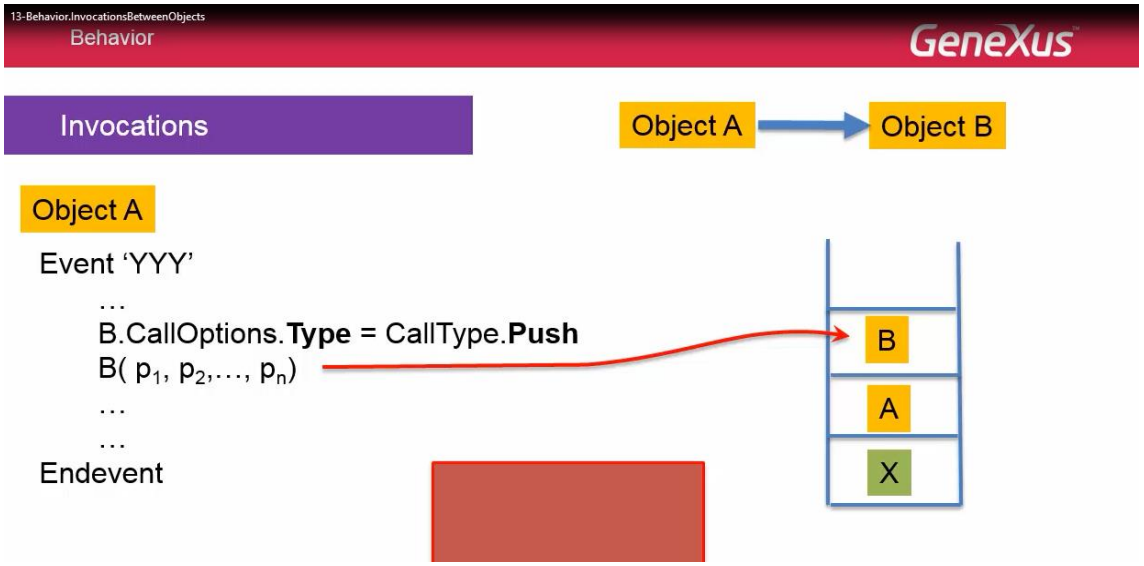


Supongamos que un objeto X, llamó a un objeto A.

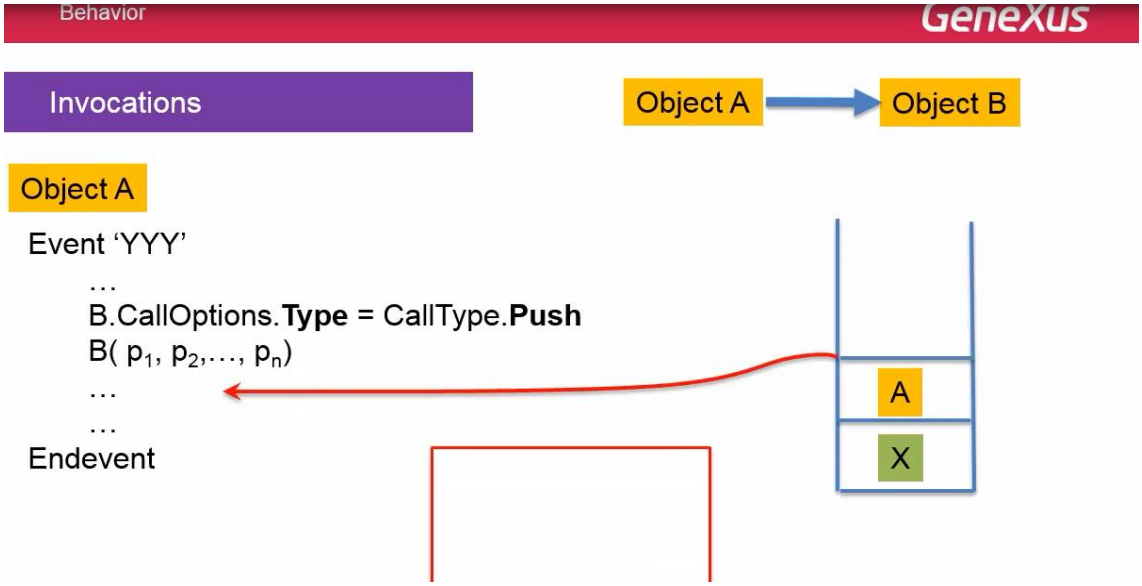
Si ahora desde A en un evento, llamamos a un objeto B, con el tipo de call **Push**, el objeto llamado es colocado arriba en el stack:



Su pantalla se abre sobre la pantalla del llamador:

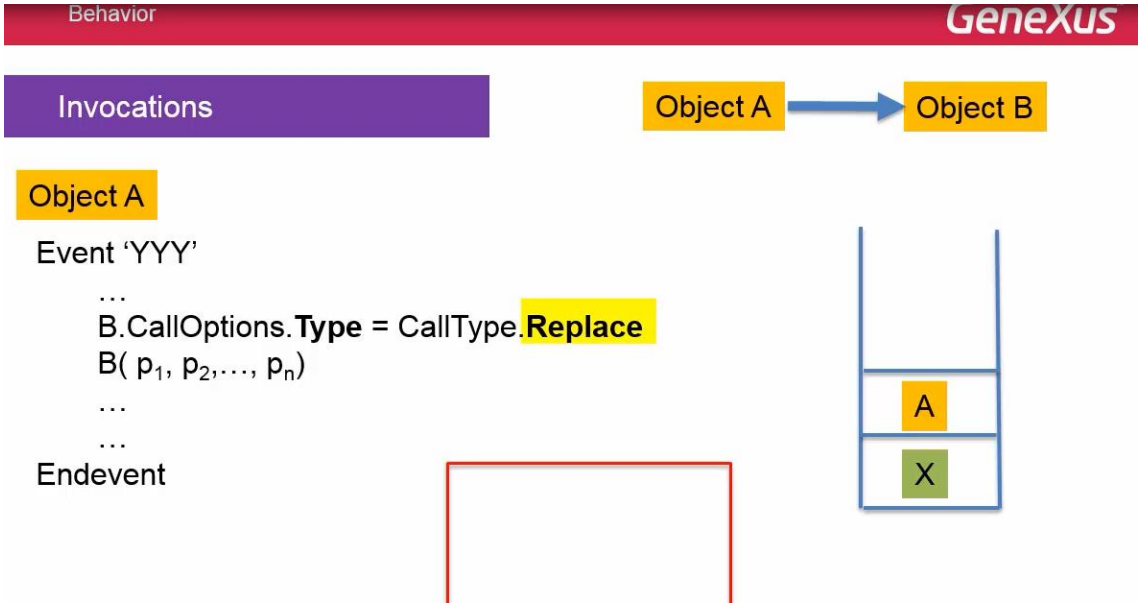


ocupando exactamente el mismo lugar… y el llamador espera para continuar su ejecución, a que termine la ejecución del objeto llamado: B

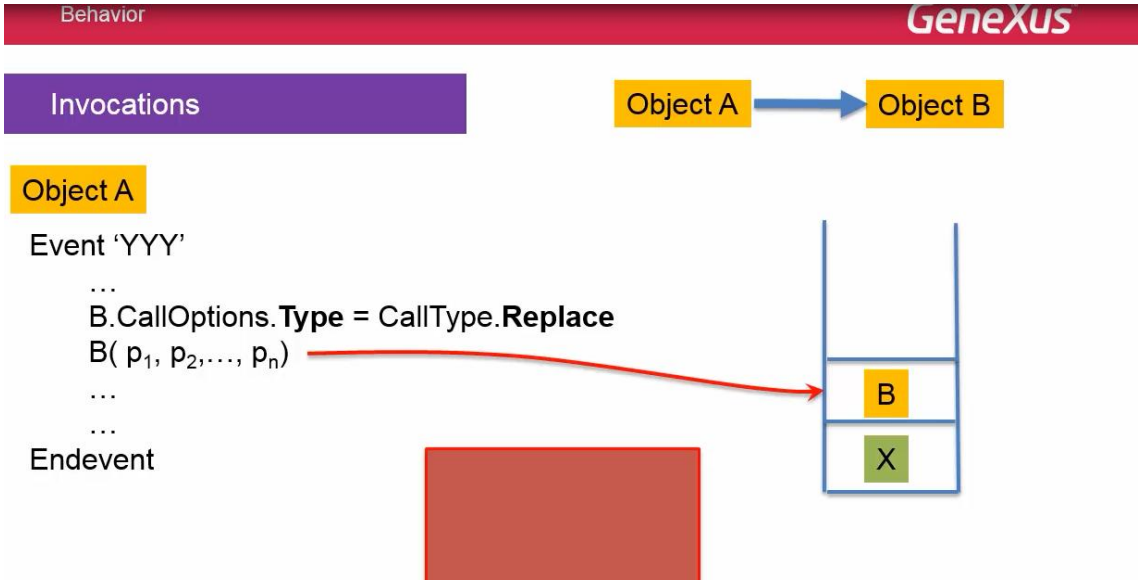


que es así eliminado del stack.

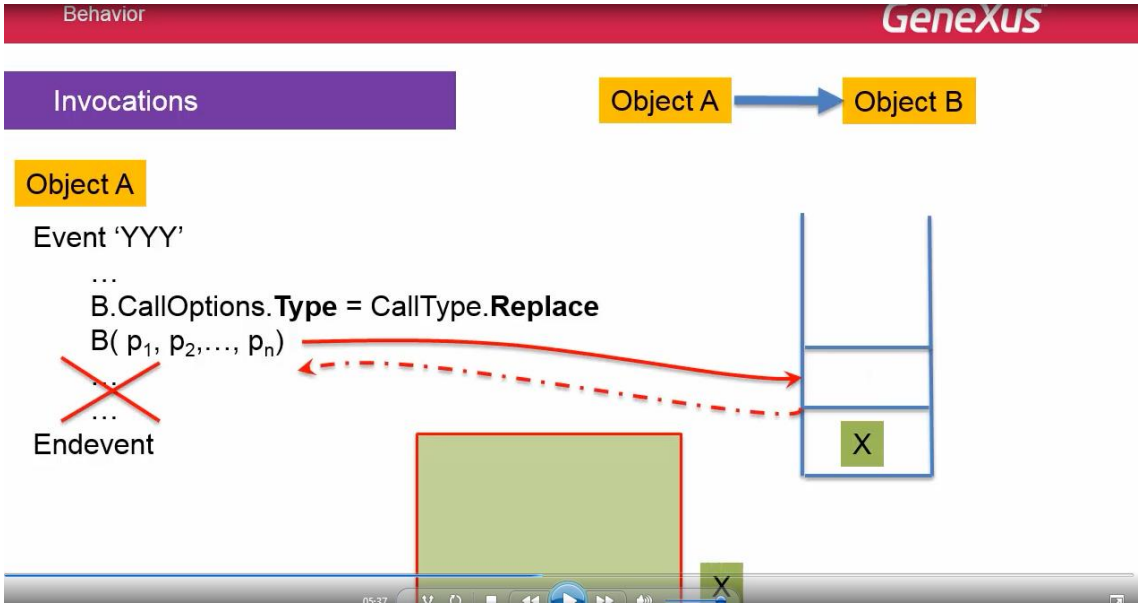
Si en cambio desde el objeto A, llamamos a B con el tipo de call **Replace**,



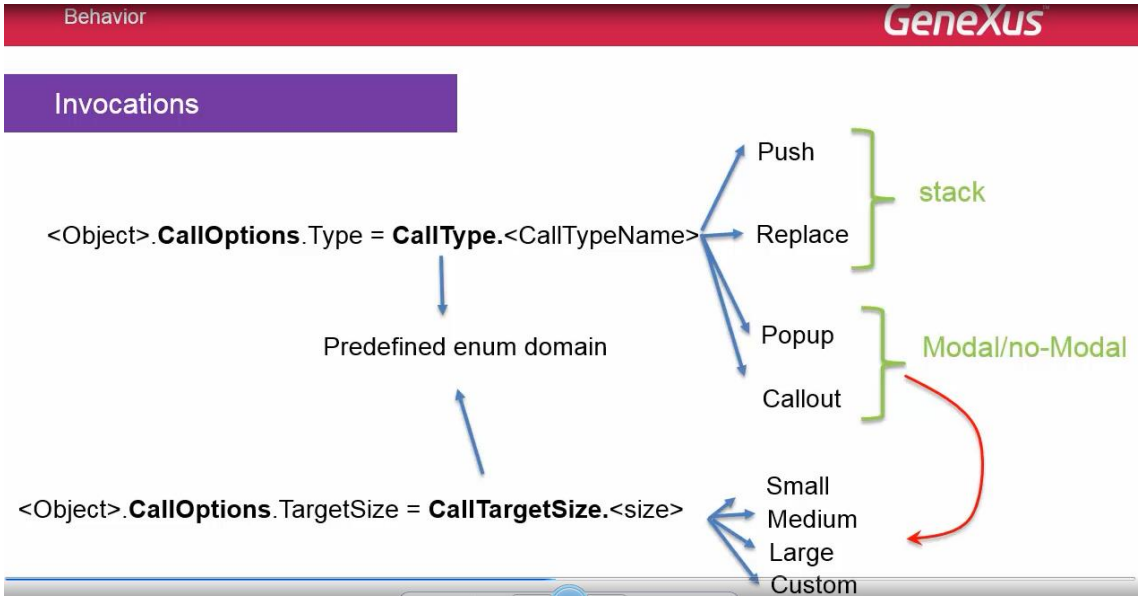
el objeto llamado también se abrirá ocupando exactamente la misma área de pantalla que el llamador, **pero va a sustituir en el stack al objeto llamador**



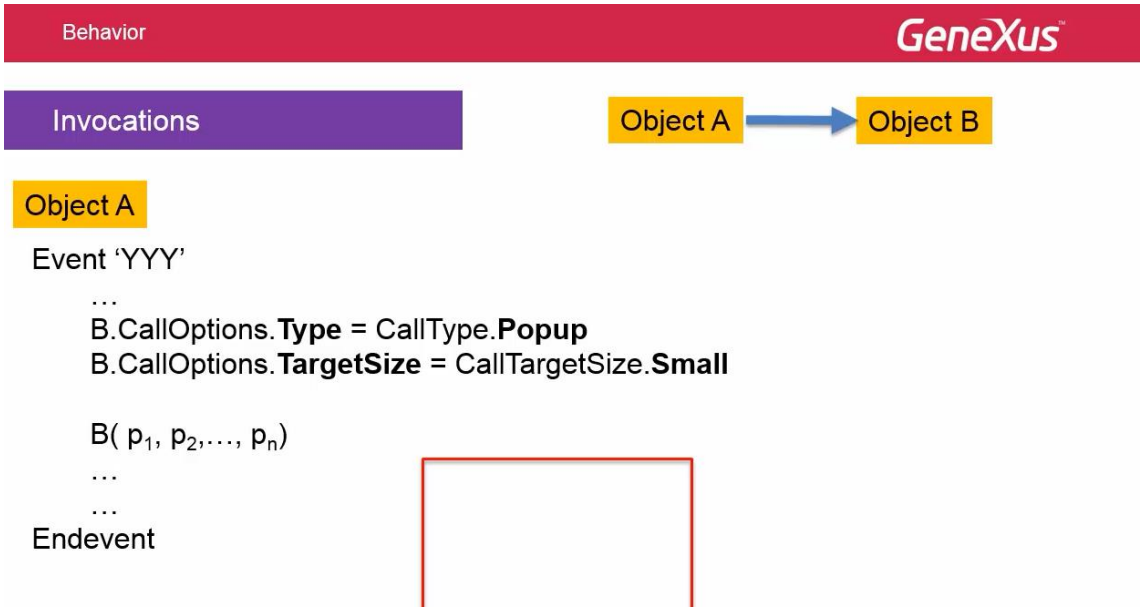
por lo que, cuando termine su ejecución



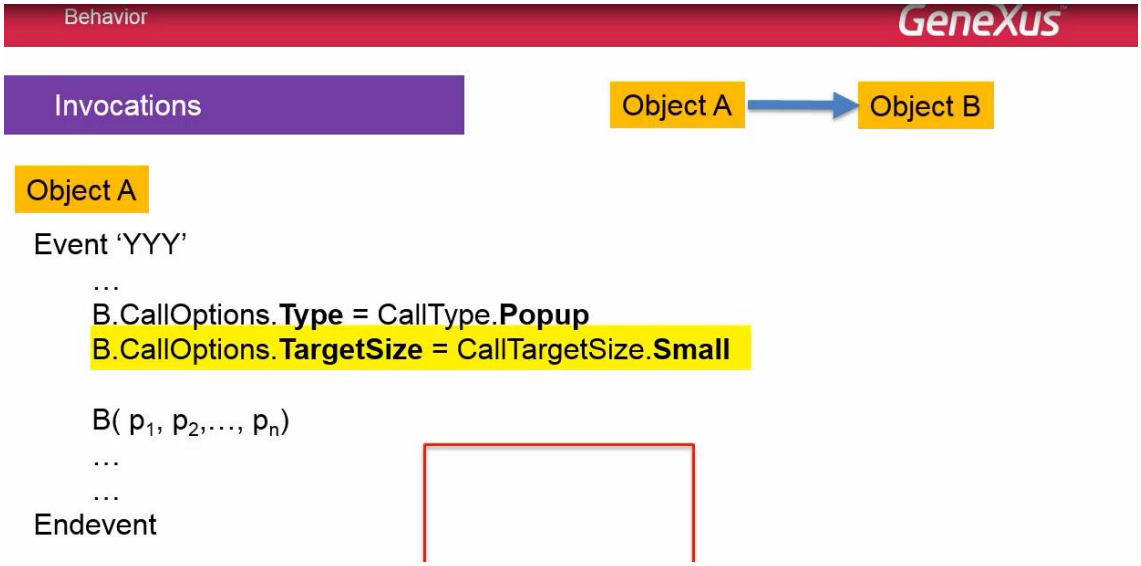
no volverá a continuar la ejecución del evento de A, sino que volverá al objeto que estuviera antes en el stack; en este caso, el objeto: X.



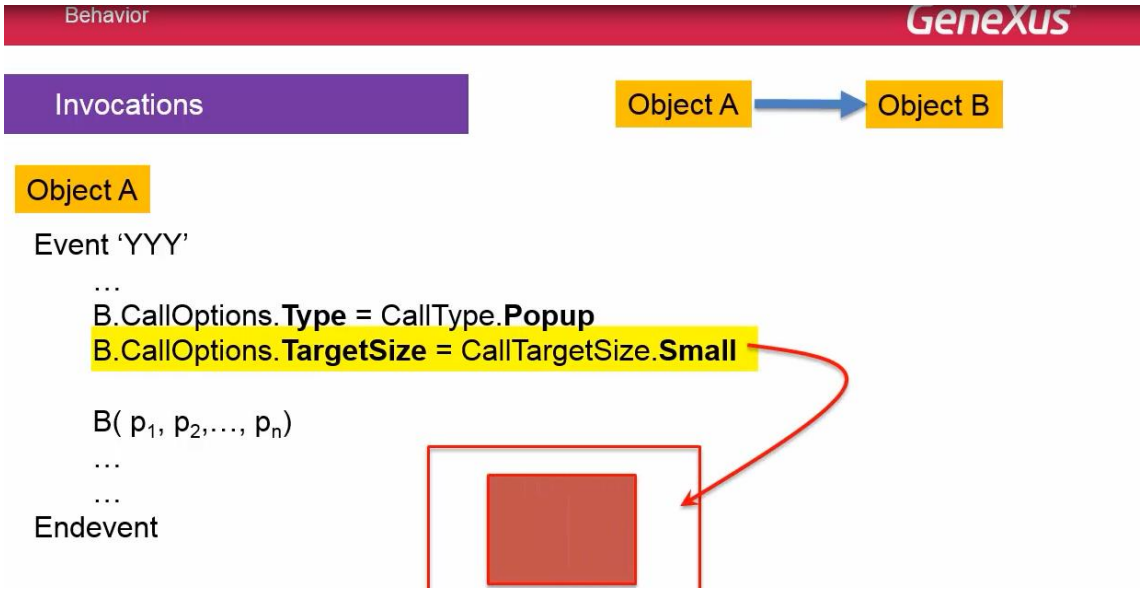
Respecto a Popup y Callout… Veamos el tipo Popup:



Si no se modifica el TargetSize, ocupará la misma área de pantalla que el llamador.

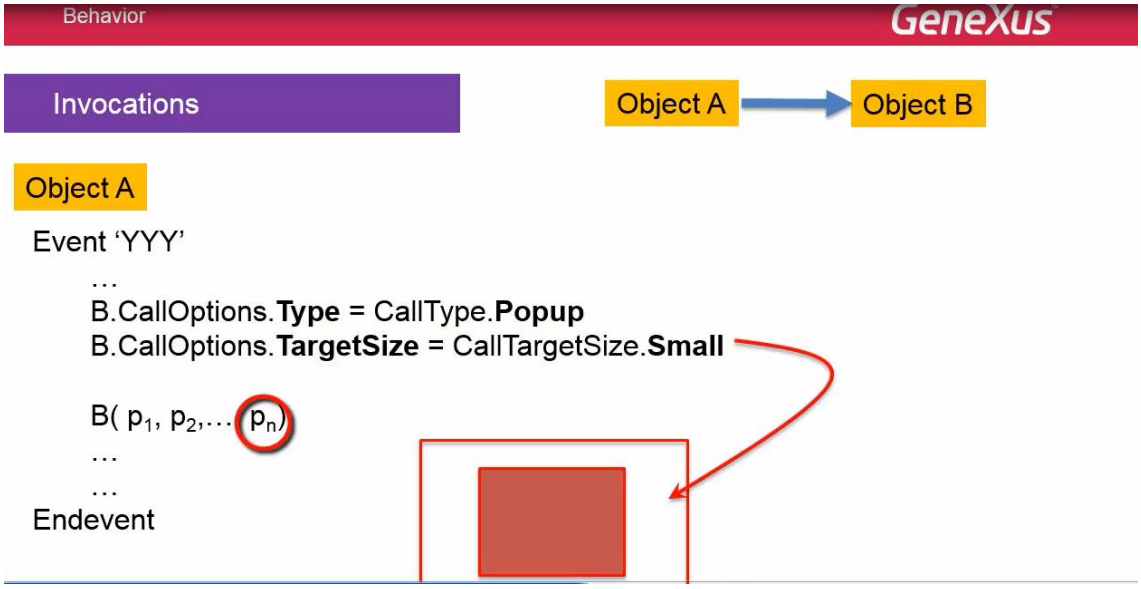


En caso contrario

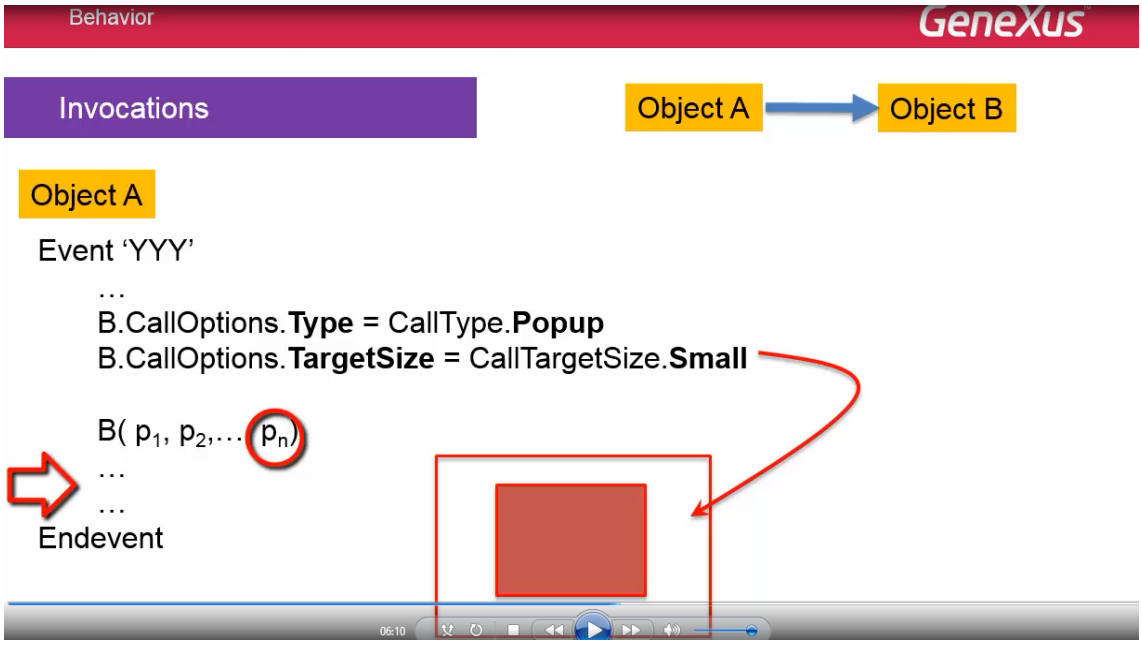


ocupará el área que hayamos especificado.

Si entre los parámetros de invocación



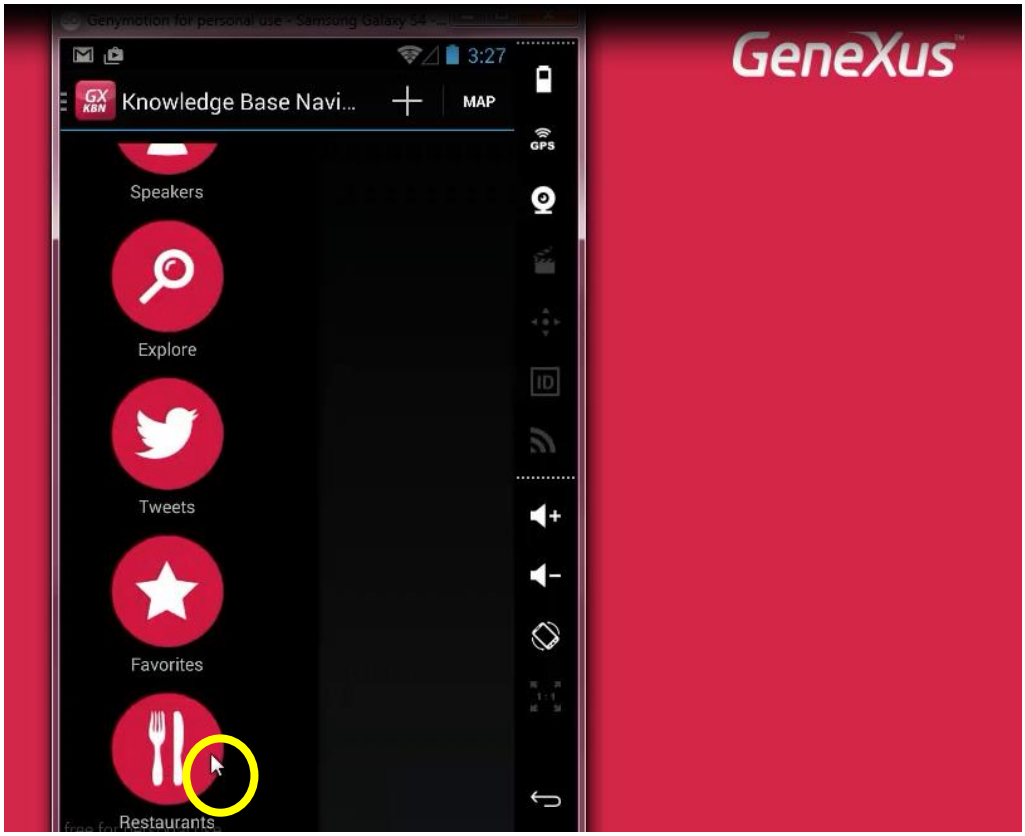
alguno es output, entonces el diálogo será modal, es decir, el llamador va a esperar el retorno de la ejecución de B para continuar



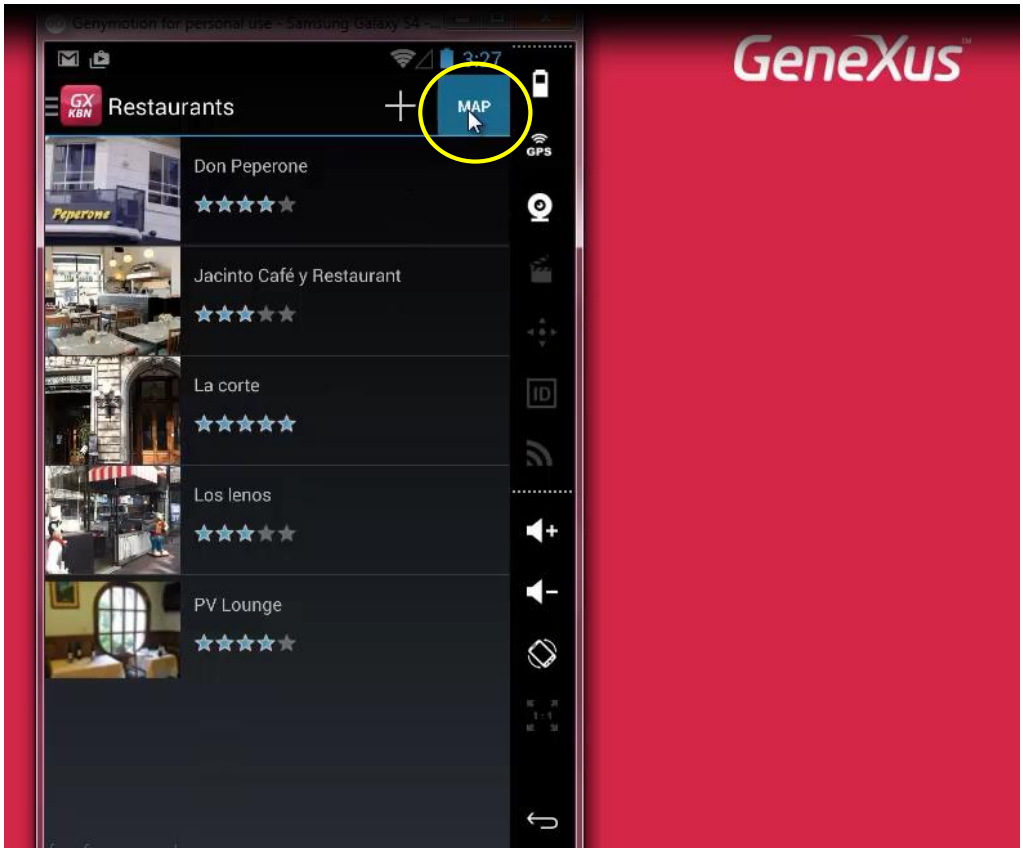
Si ninguno de los parámetros es de output, entonces el diálogo será no modal.

Veamos un ejemplo.

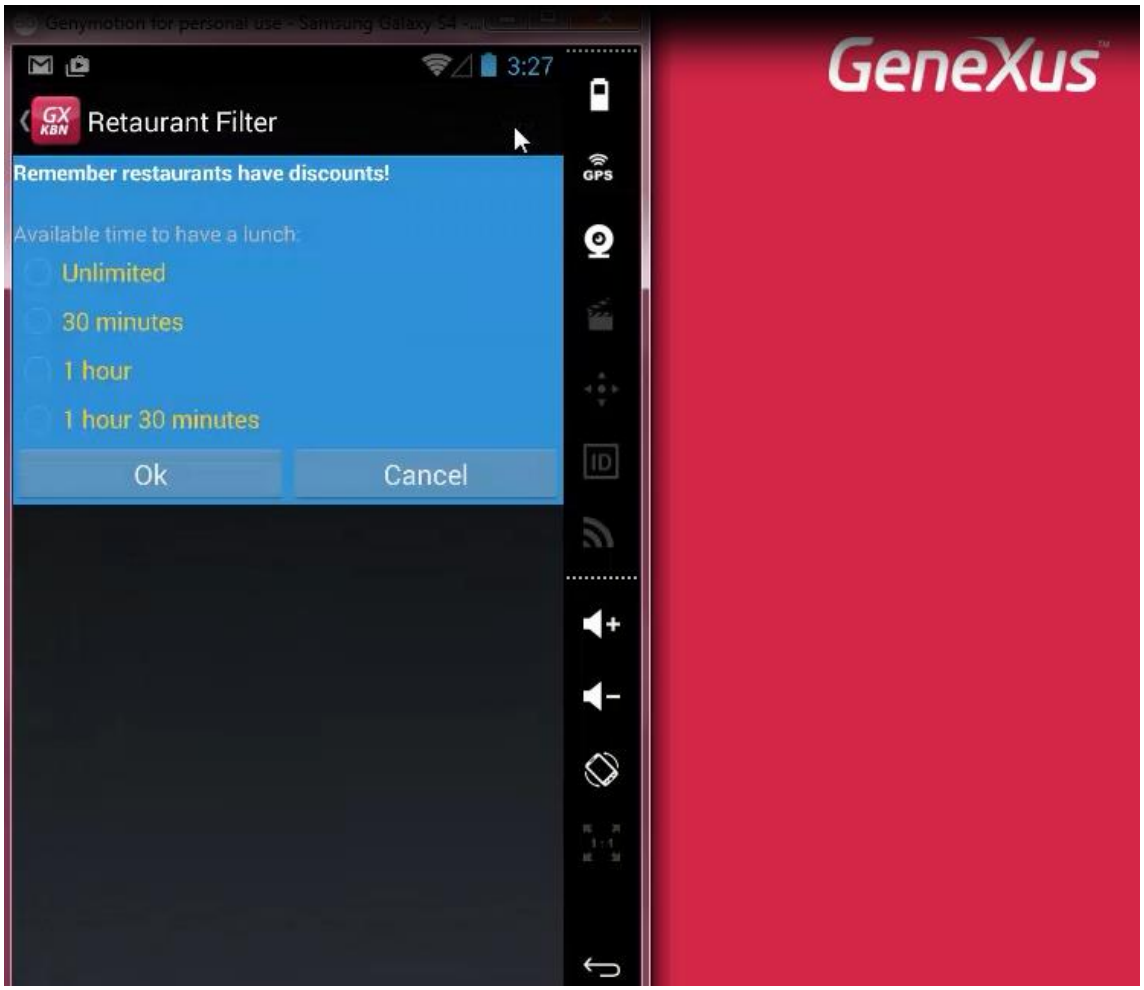
Cuando el usuario elige ver la lista de restaurants,



queremos darle la opción

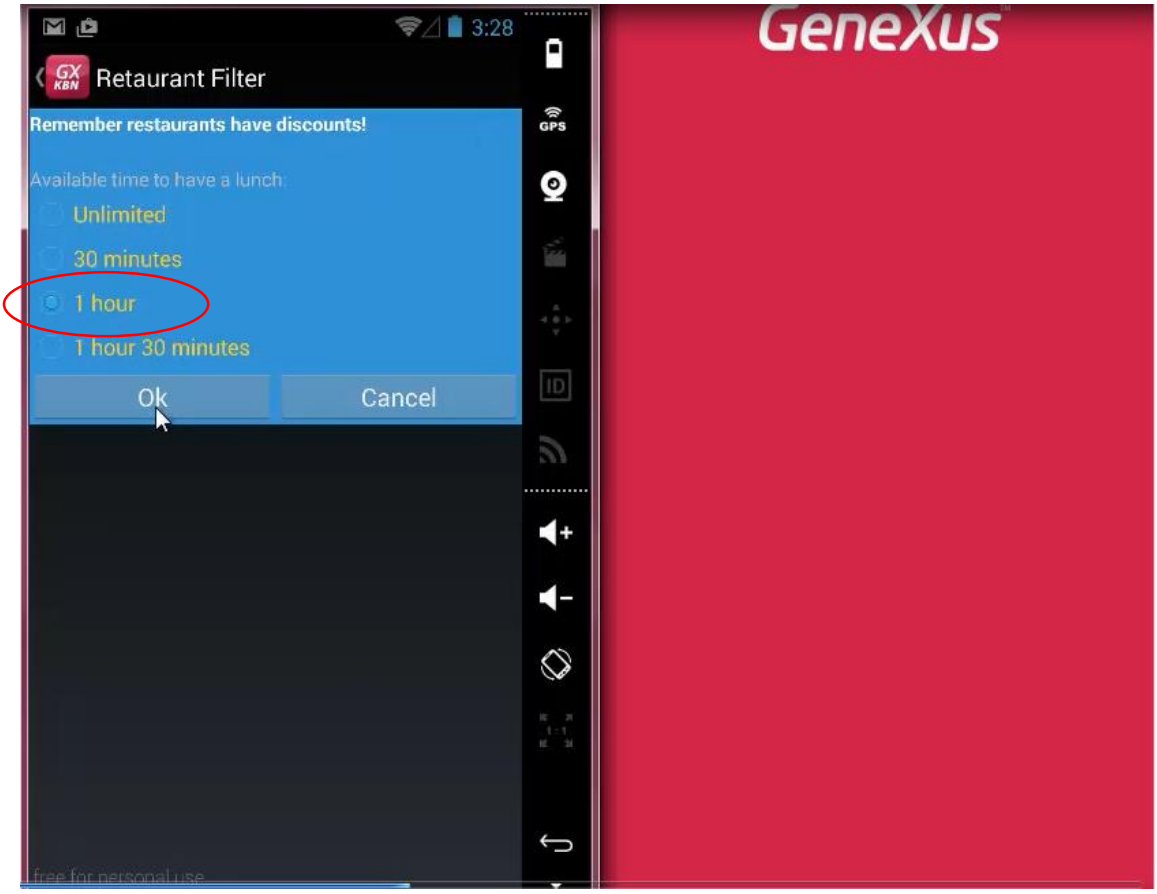


a través del botón MAP

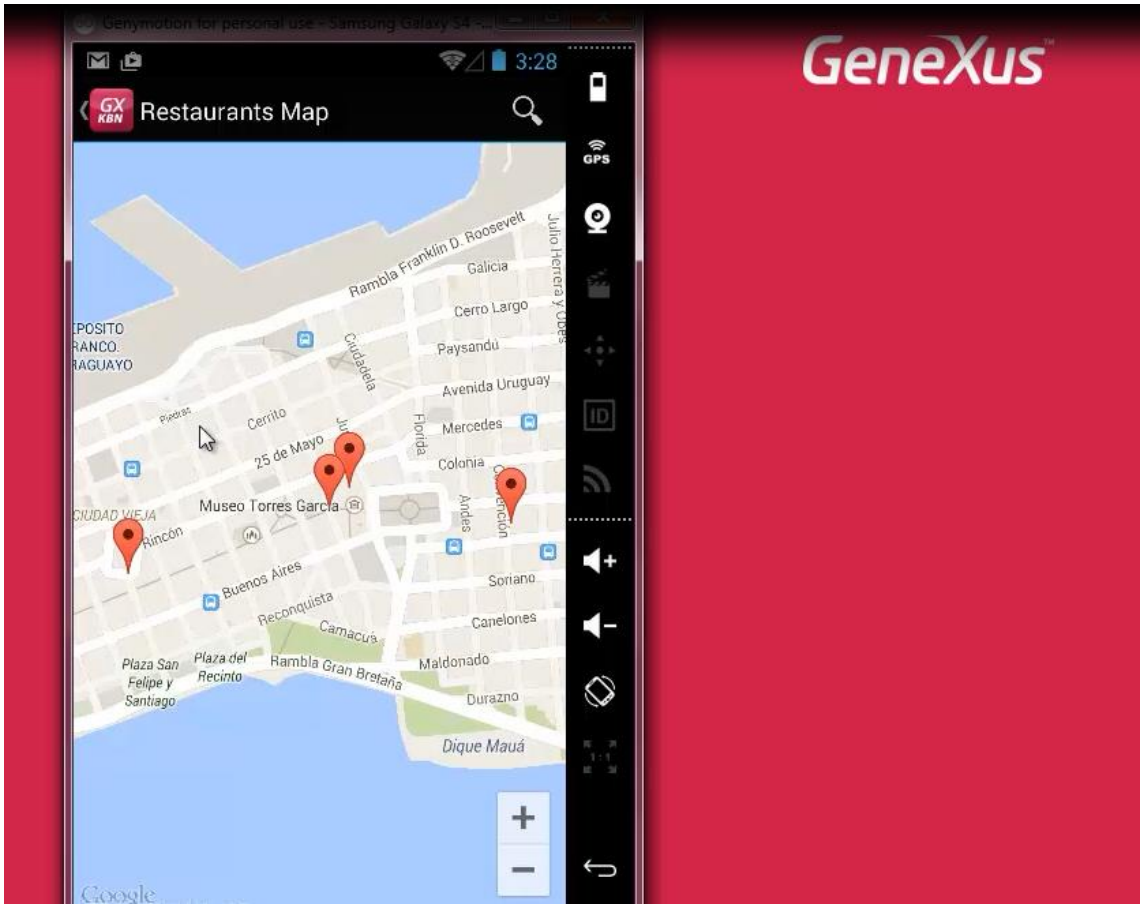


de que los pueda ver en un mapa. Pero antes de mostrárselos, queremos recordarle que los restaurants tienen diferentes descuentos para los participantes del evento. Y queremos darle la posibilidad de mostrar sólo aquellos restaurants que se comprometen a que el cliente almuerce en no más del tiempo del que dispone.

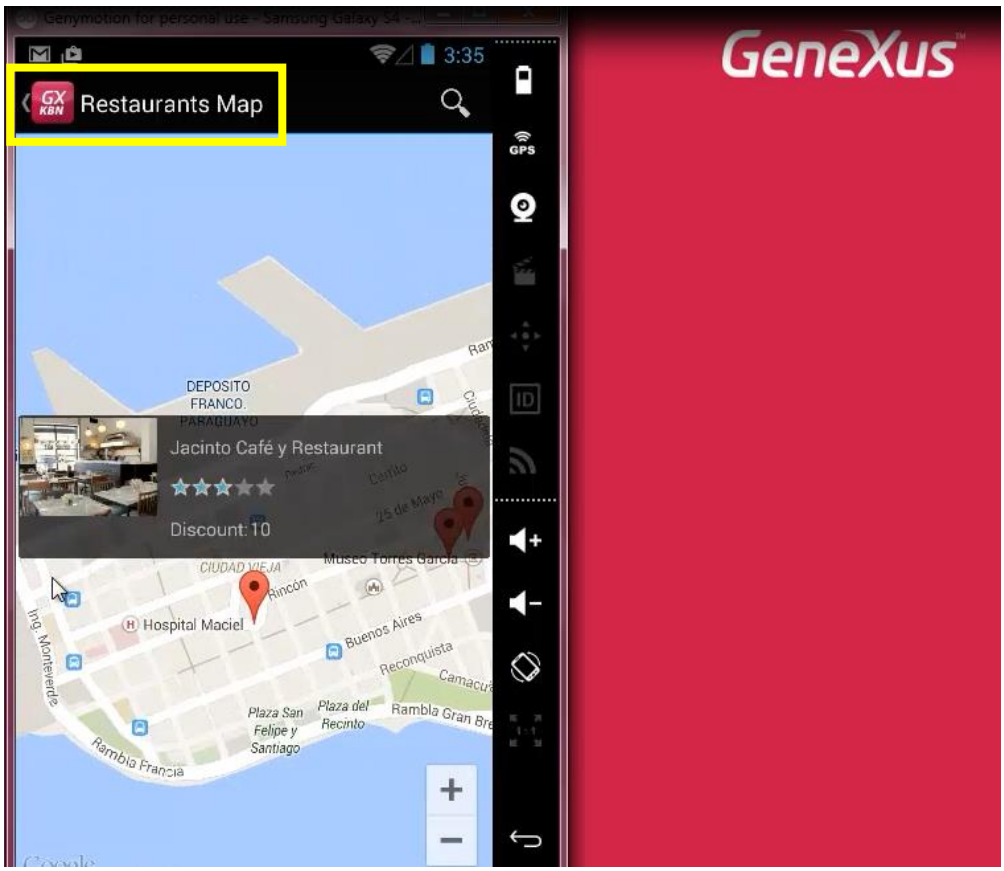
Así por ejemplo, si disponemos de una hora:



queremos ver en el mapa los restaurants que se comprometen a que se almuerce en una hora de tiempo

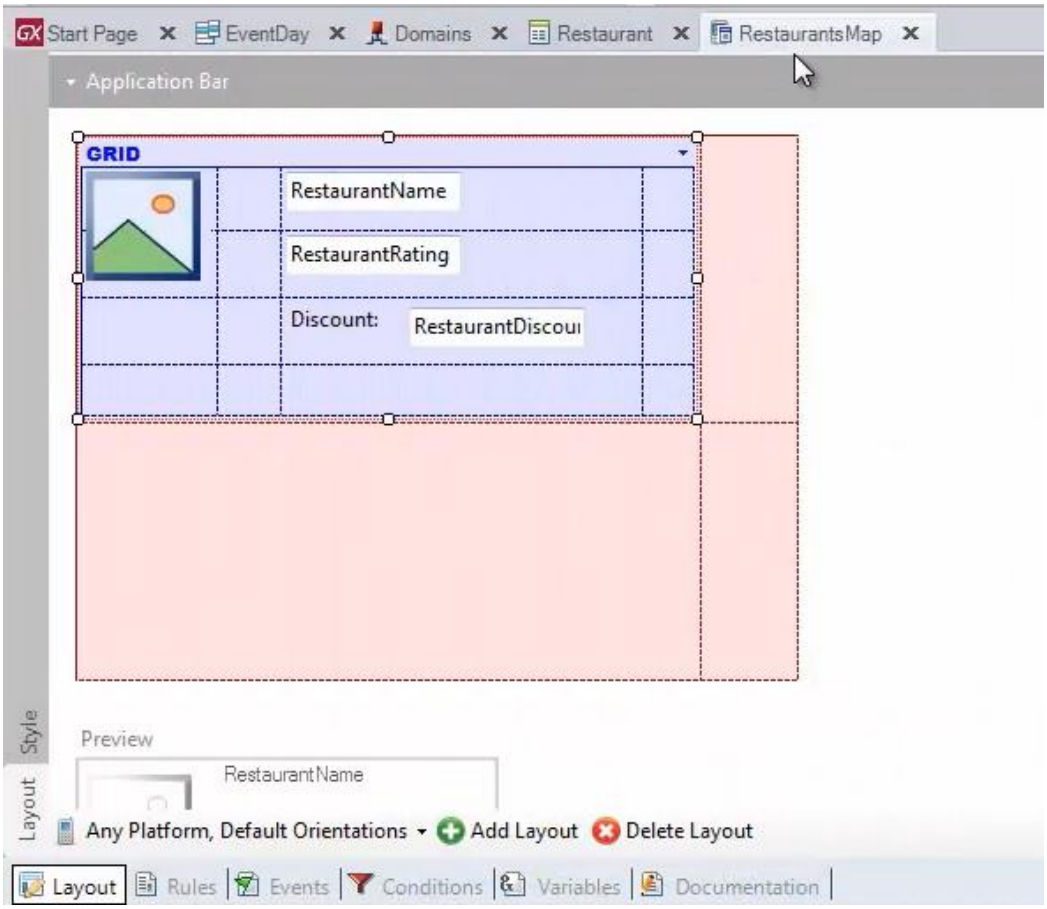


Observemos que esta pantalla

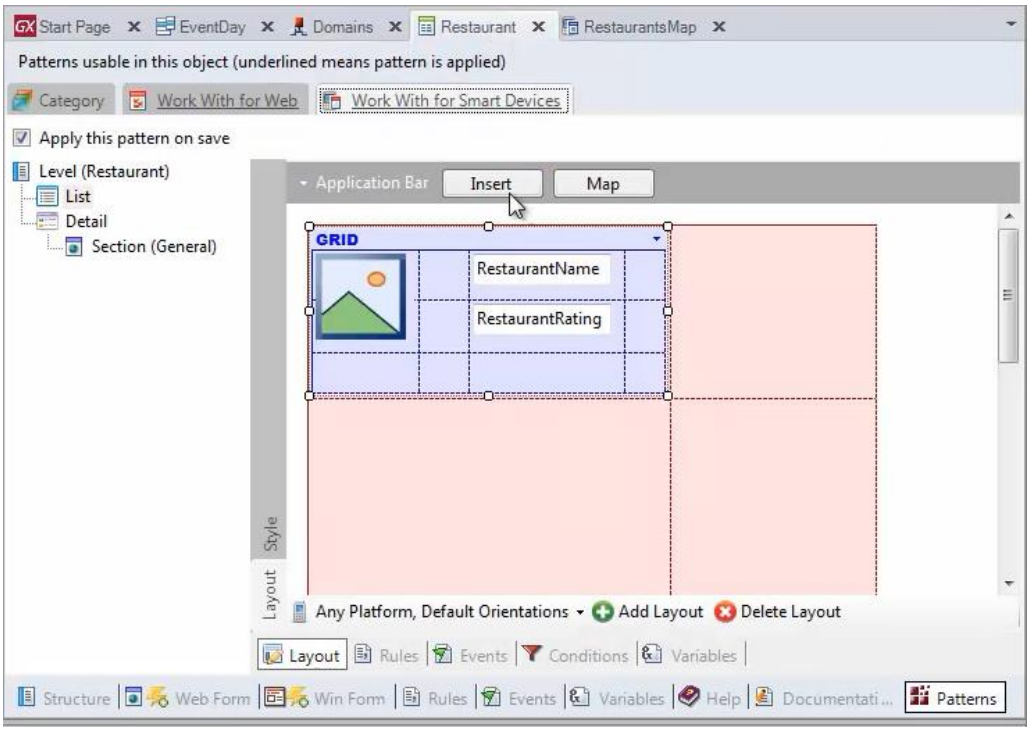


corresponde a un panel “Restaurants Map”, que va a ser muy similar al List del WorkWith de restaurants.

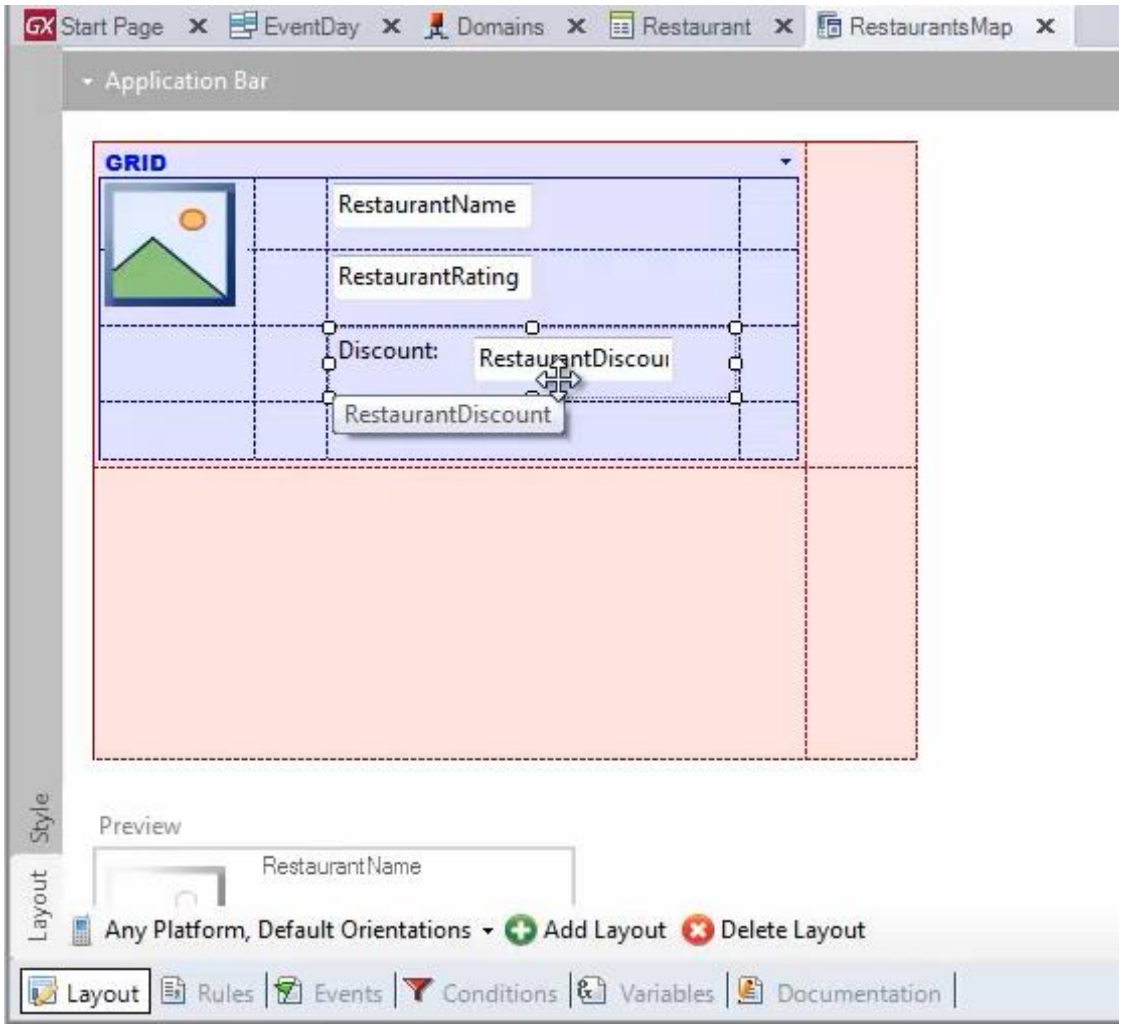
La principal diferencia entre uno:



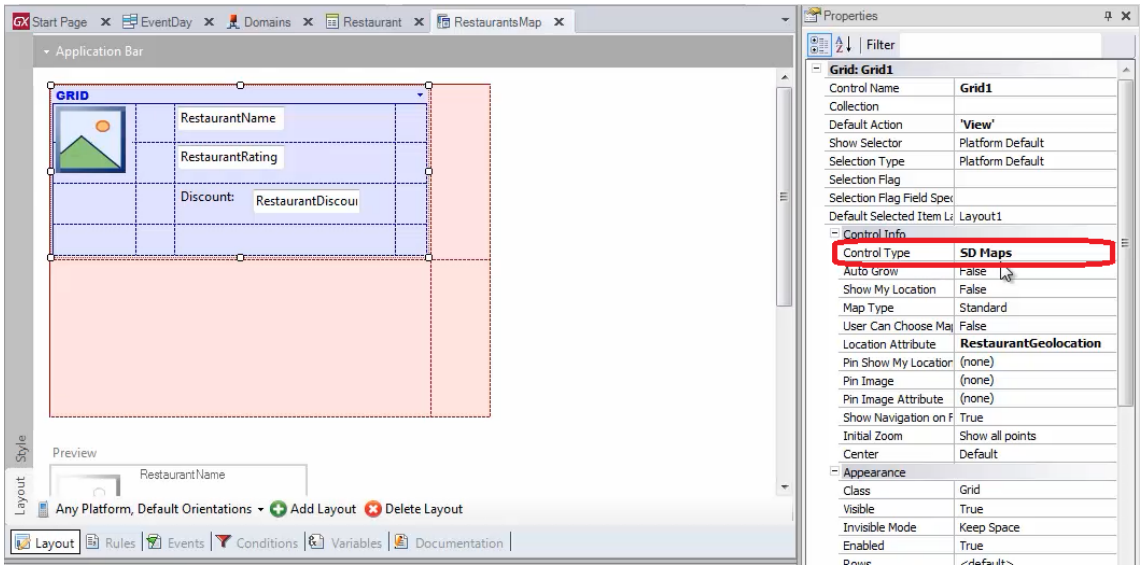
y otro:



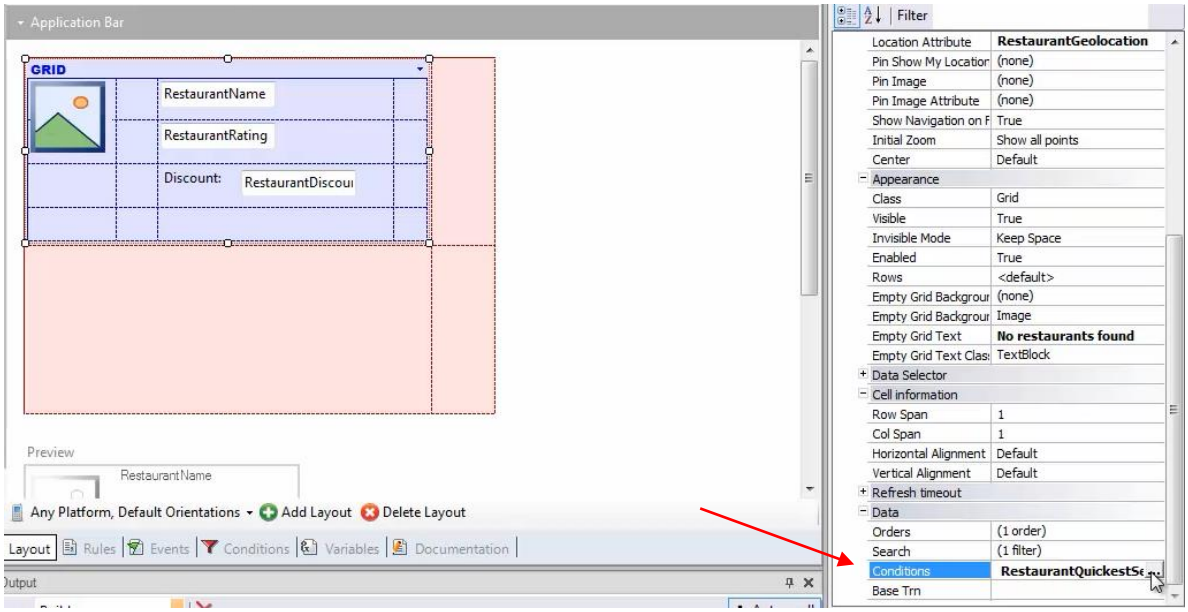
radicará no sólo en que en este caso se está agregando el atributo: Discount



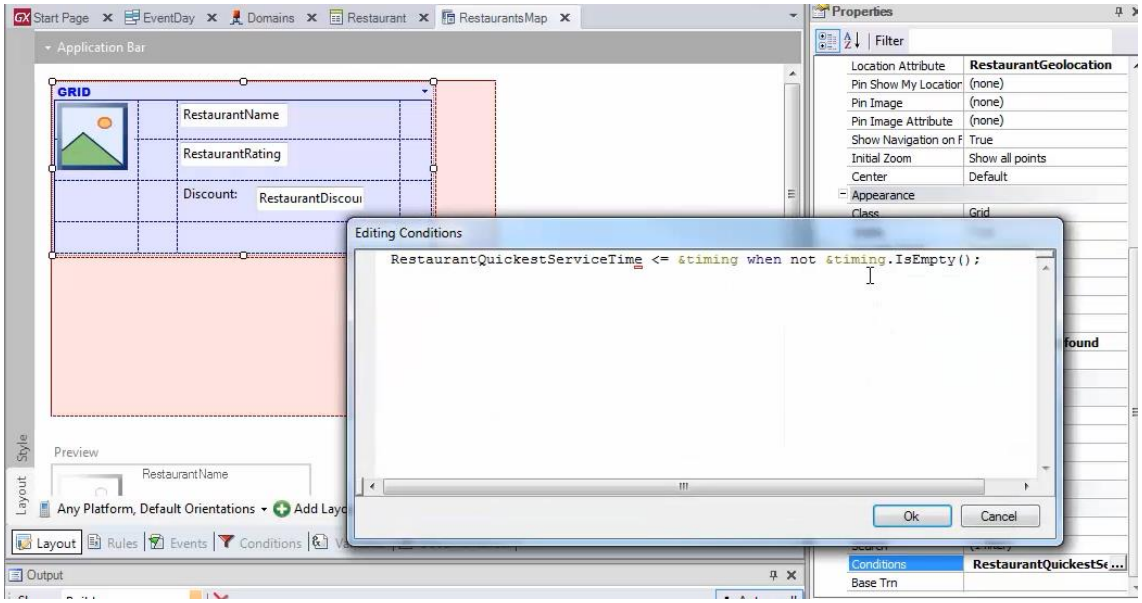
sino que además, este grid, tiene el Control Type: SD Maps



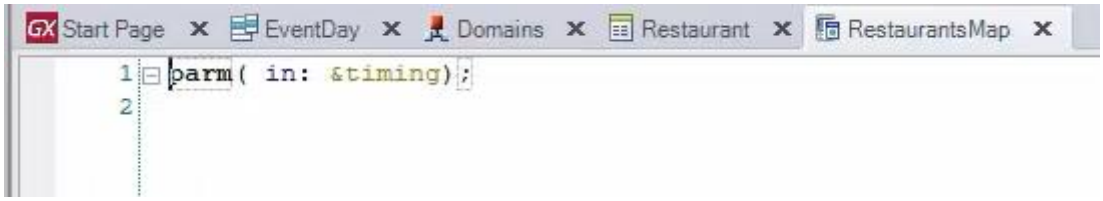
y está filtrando los datos que se cargan



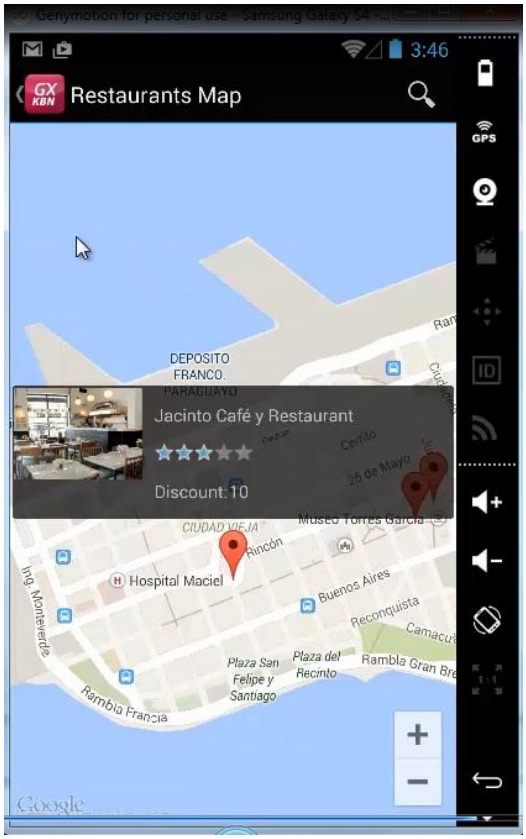
de acuerdo a esta condition:



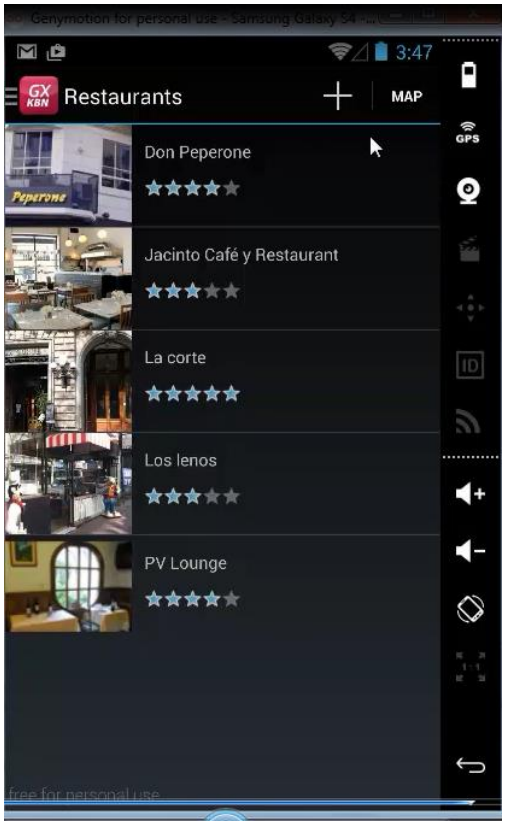
Es decir, de acuerdo al tiempo que se dispone para almorzar, tiempo que viene dado en una variable en la regla parm



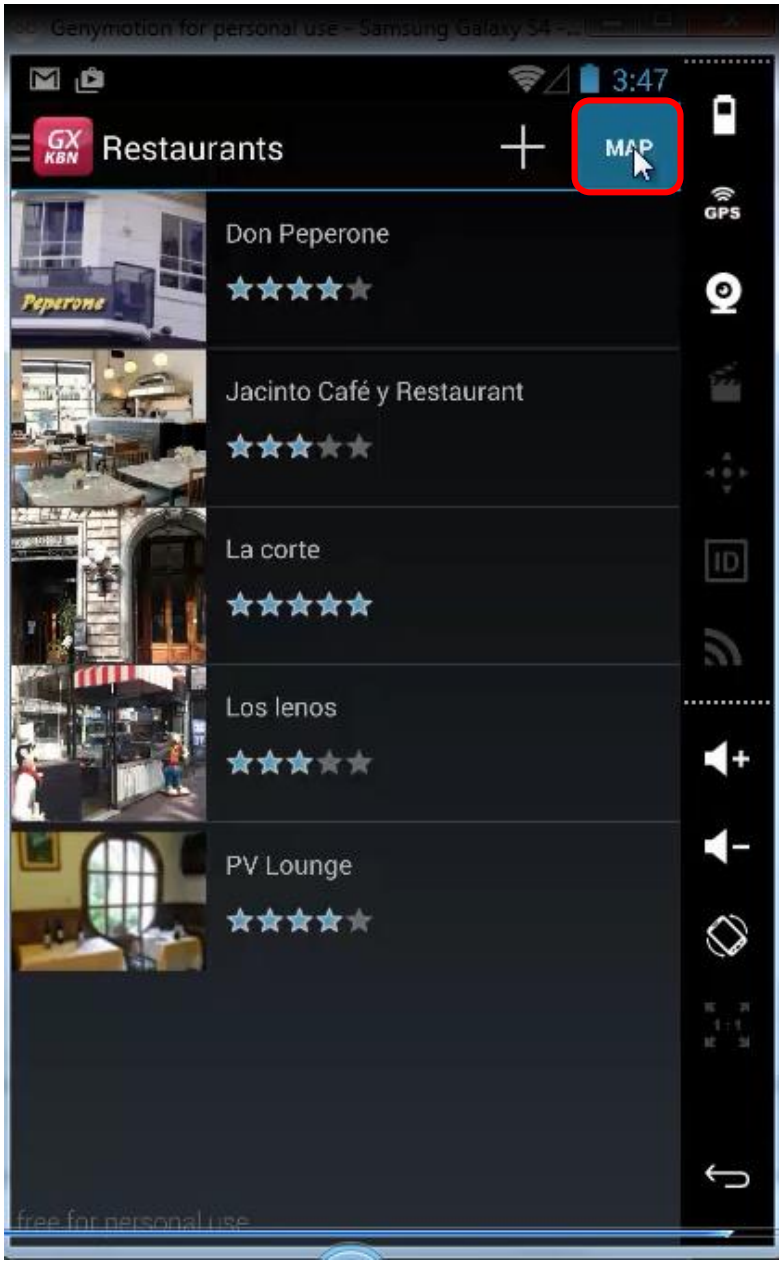
Quien llama a este panel:



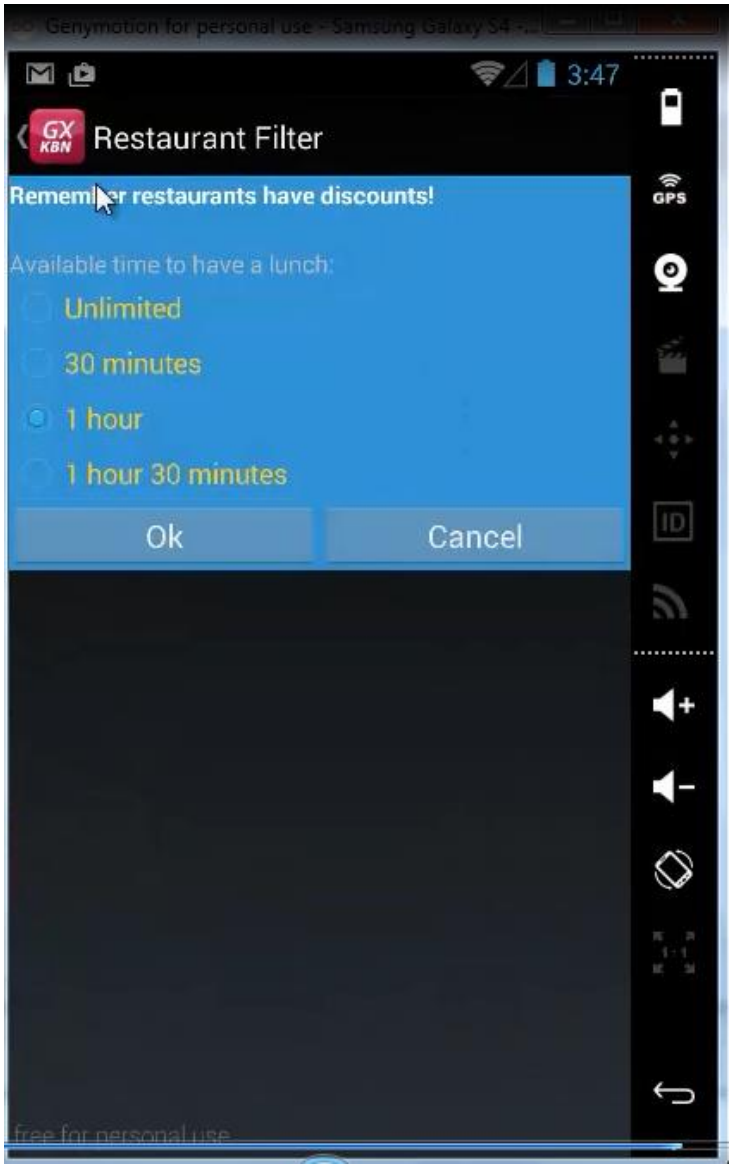
no es directamente el List del WorkWith



sino que

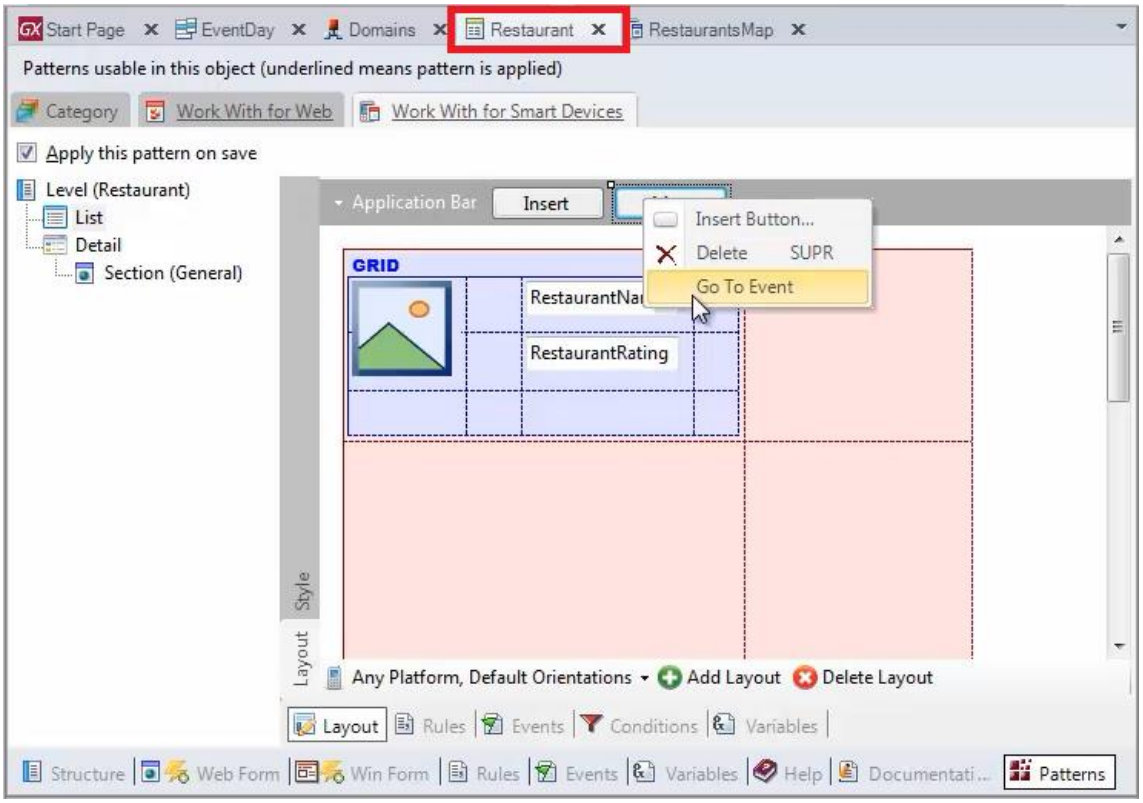


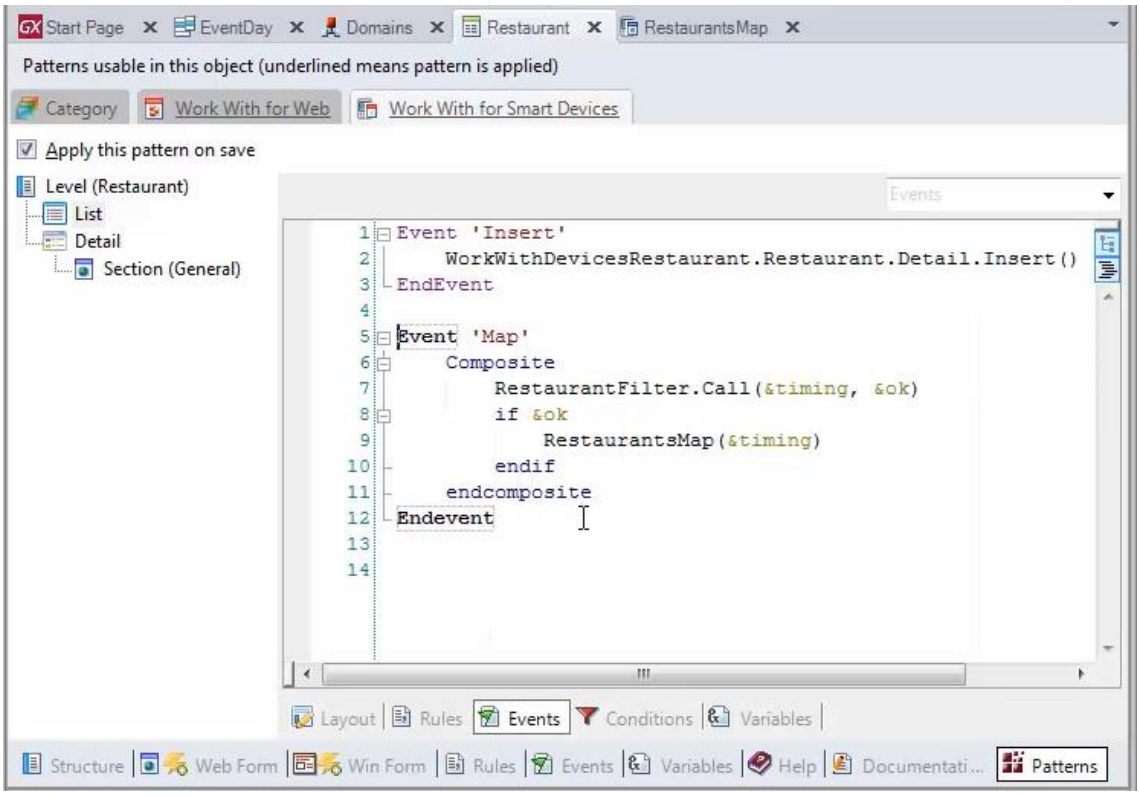
es este panel intermedio: “Restaurant Filter” →



que se está abriendo como una pantalla independiente sobre la anterior.

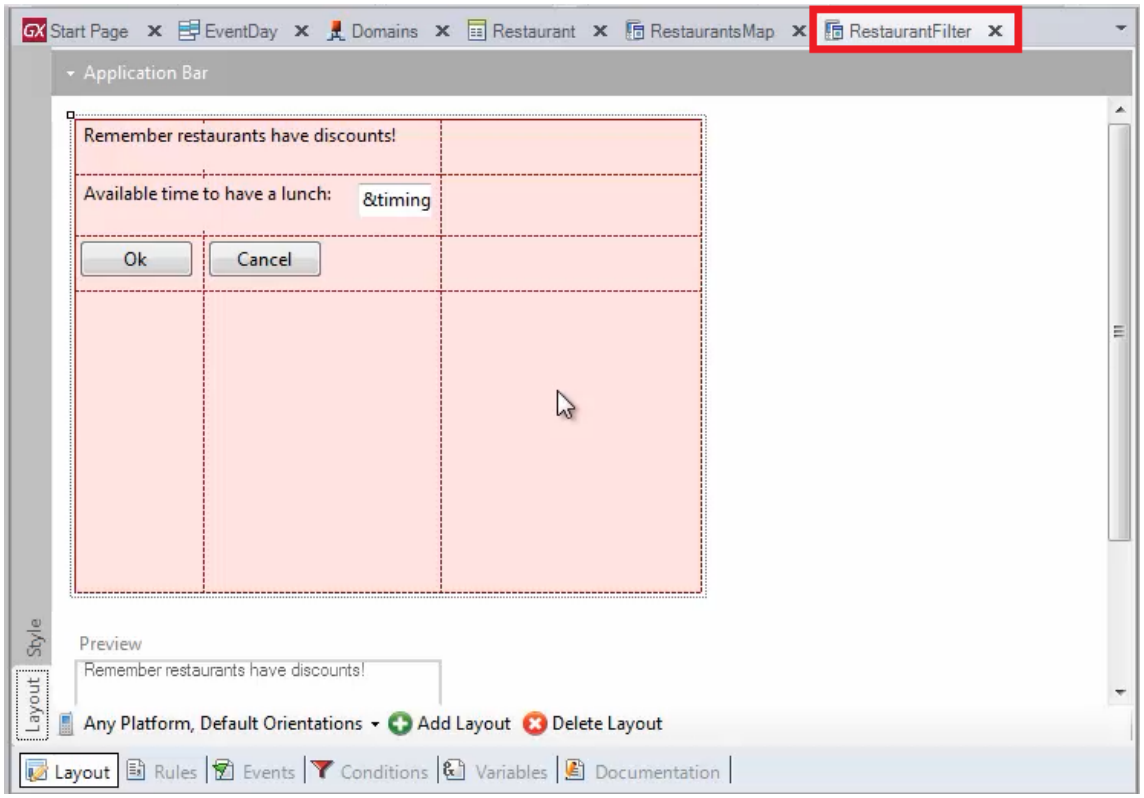
Si vamos a ver la invocación, sobre el botón Map, vamos al evento asociado..



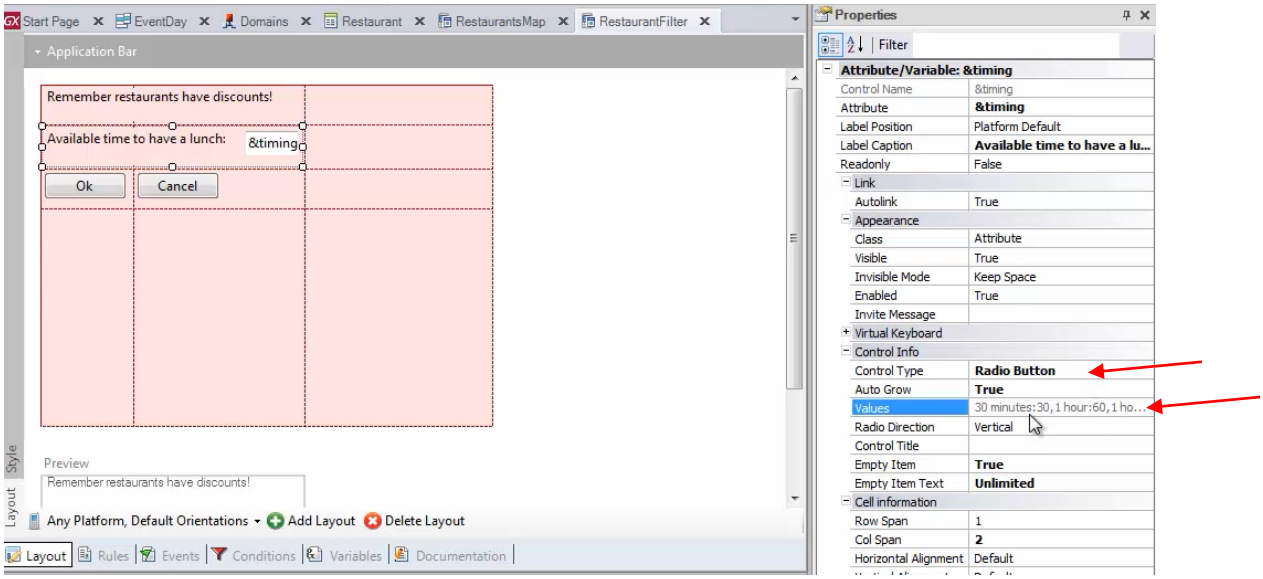


y vemos acá que se está invocando al panel RestaurantFilter, pasándole 2 parámetros: &timing y &ok.

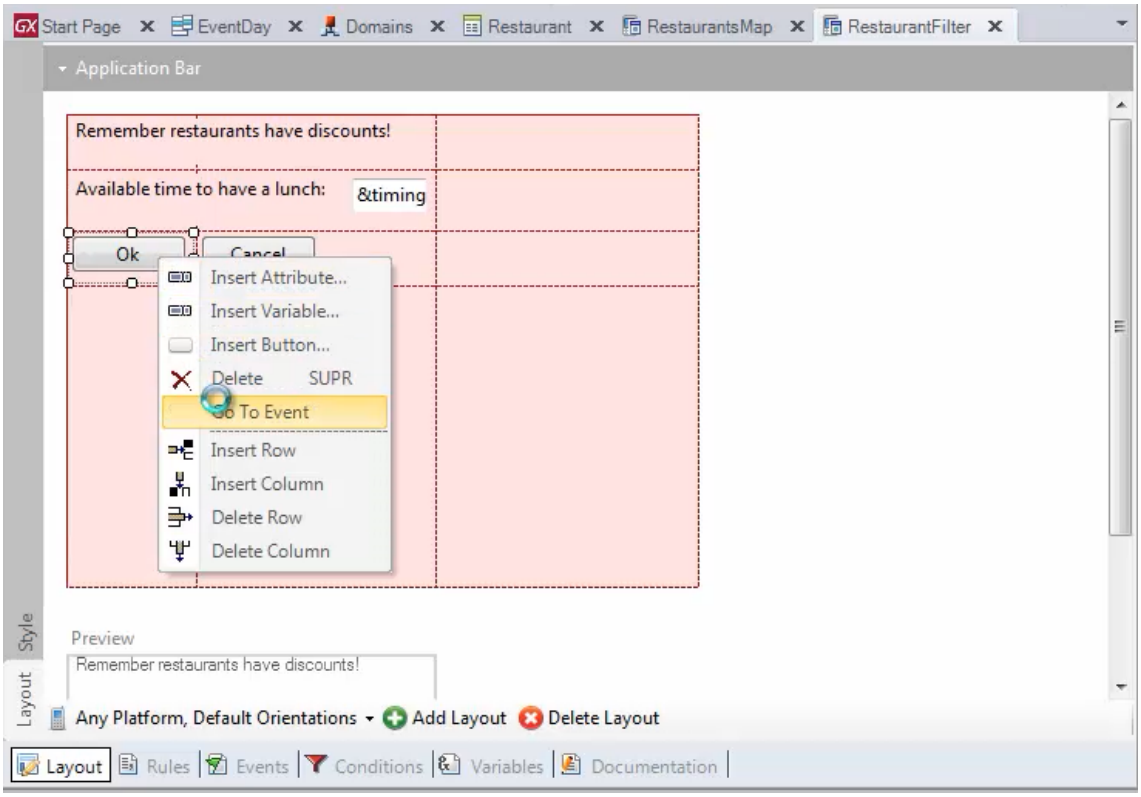
Vamos a abrir ese panel

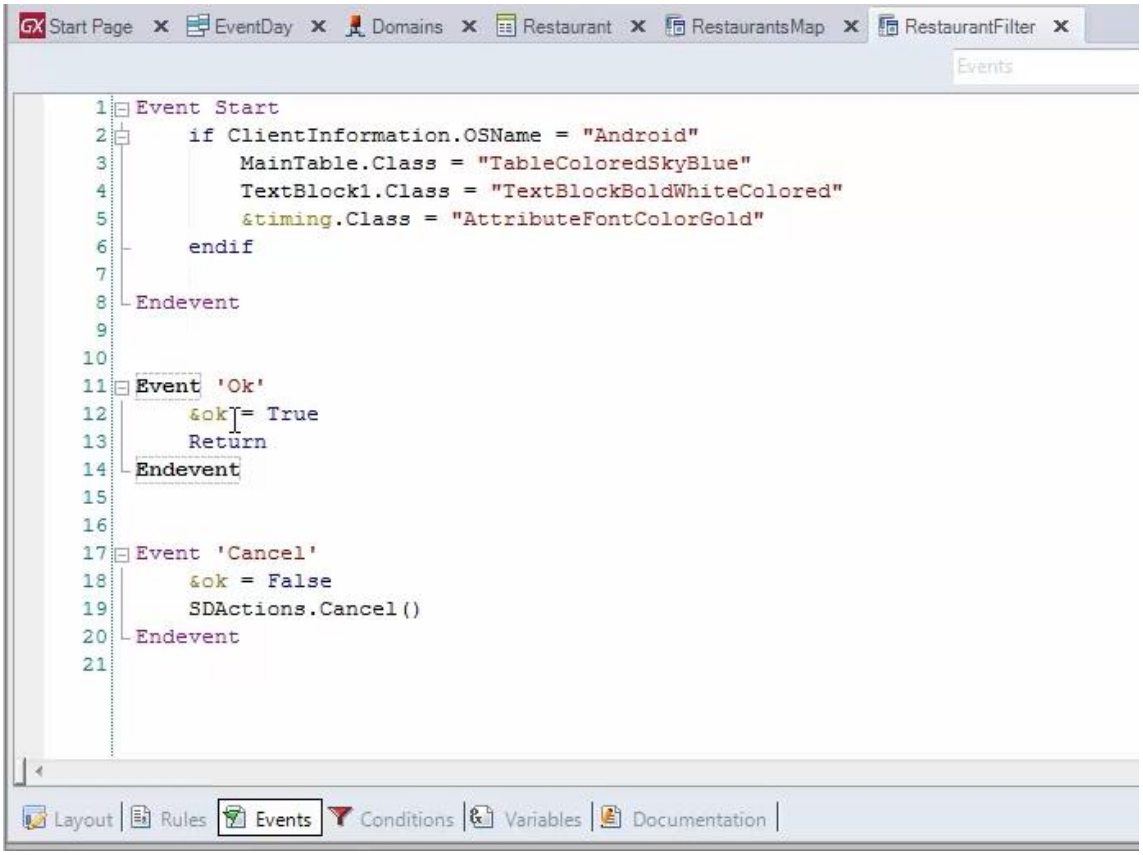


Vemos que tiene la variable &timing, un radio button, que es el que ofrece los valores



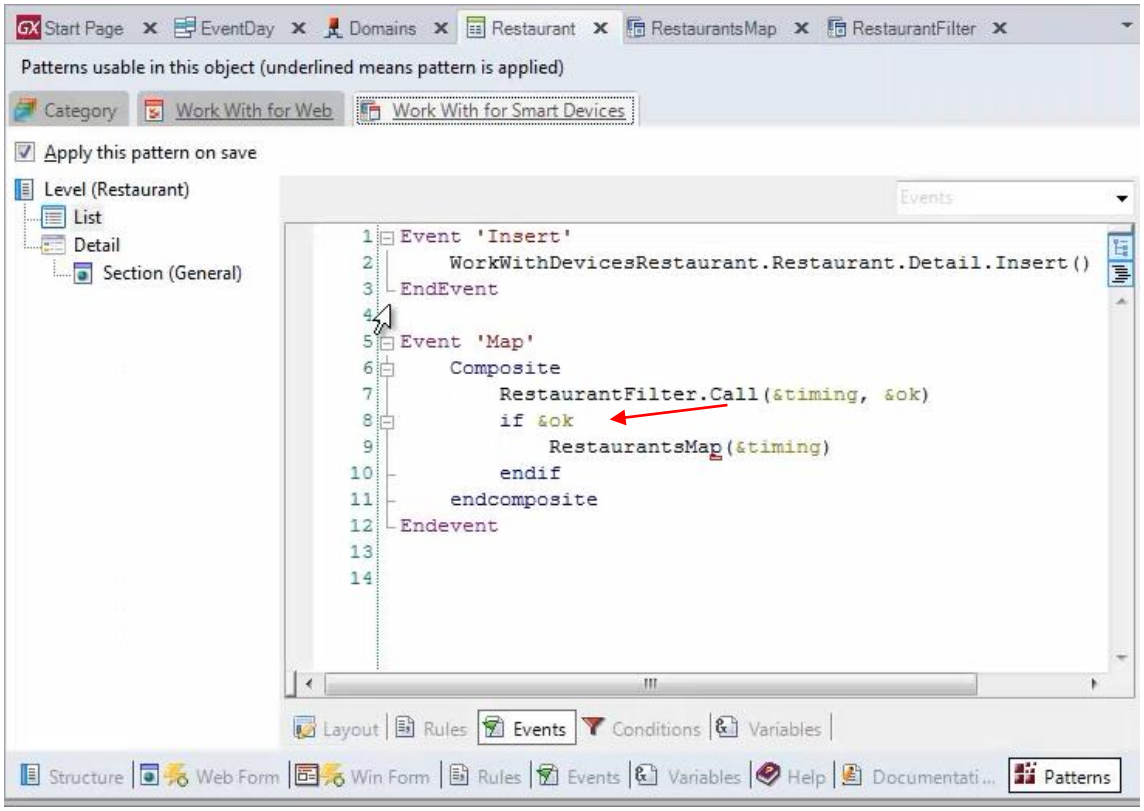
Y si vamos a ver el evento asociado al botón OK





Vemos que asigna True a la variable &ok … en caso contrario, si se presiona el evento asociado al botón Cancel, se deja en False.

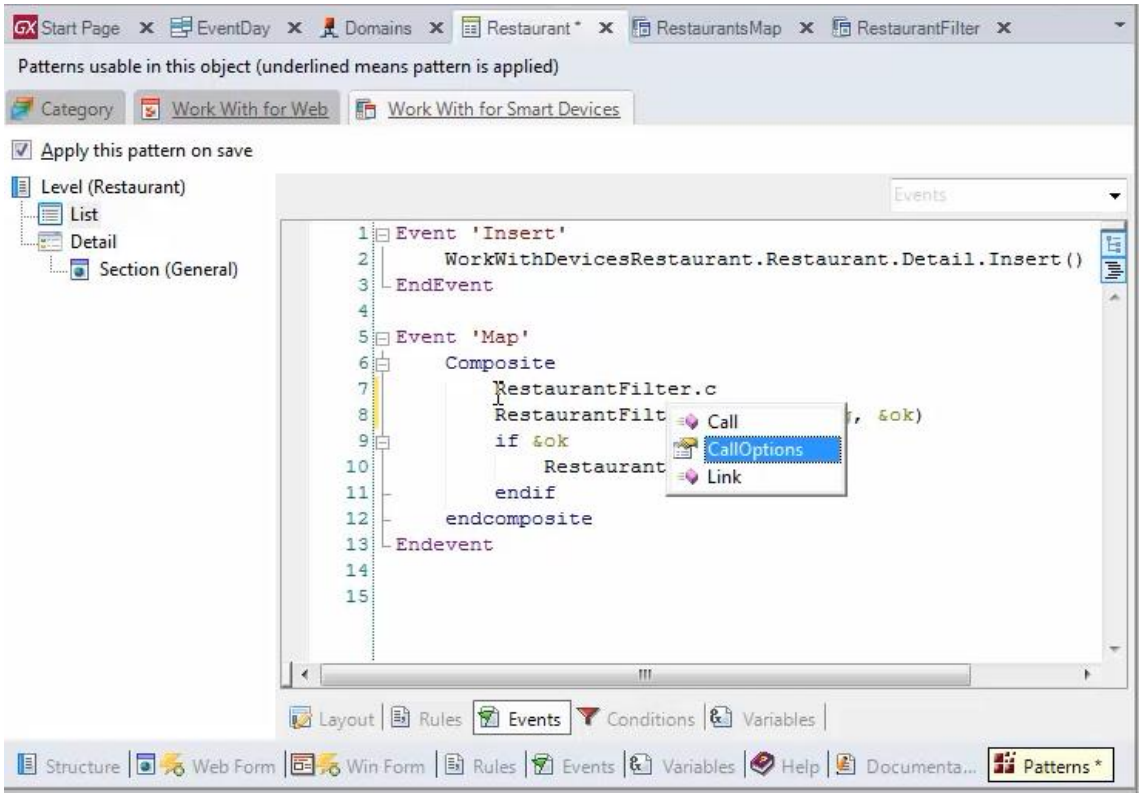
Vemos que en el 'Ok' se retorna al llamador, esto es al List del WorkWith



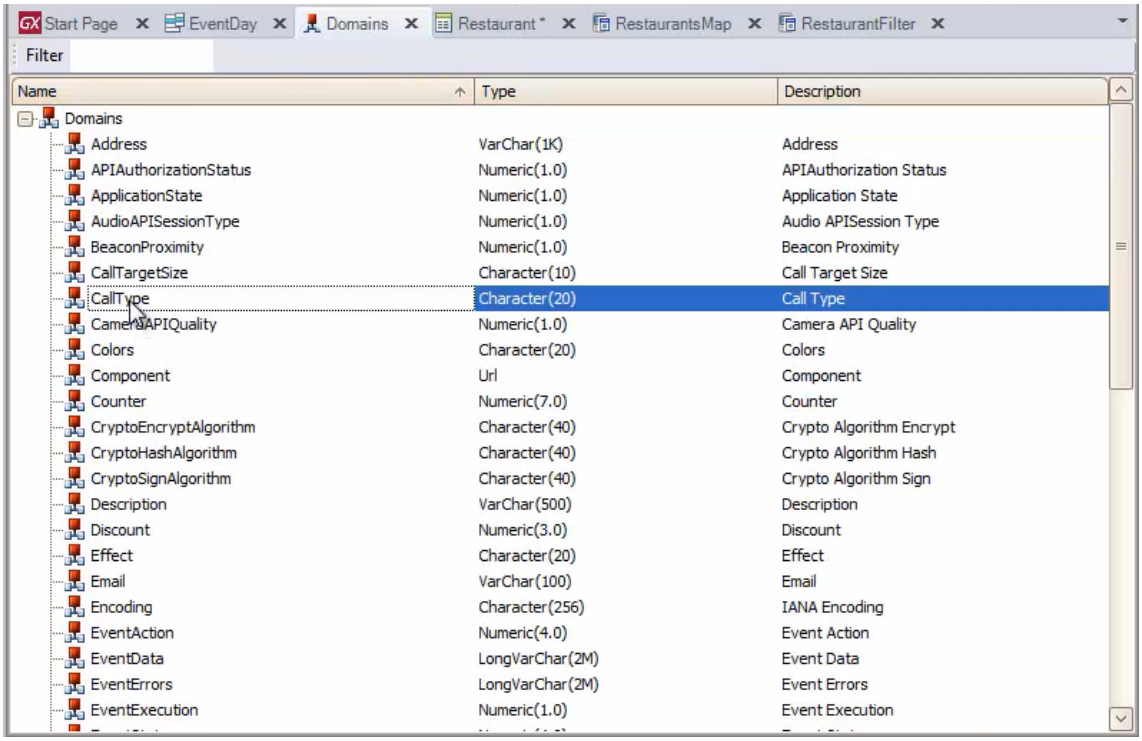
Y aquí, si efectivamente se eligió un valor de tiempo, entonces es que se invoca al panel RestairantsMaps, pasándole esa variable &timing.

Sin embargo, si lo pensamos un poco, la pantalla de RestaurantFilter, corresponde más a una pantalla Popup de tipo modal; es decir la ejecución de lo que sigue a la invocación, deberá esperar a que esta termine para continuar.

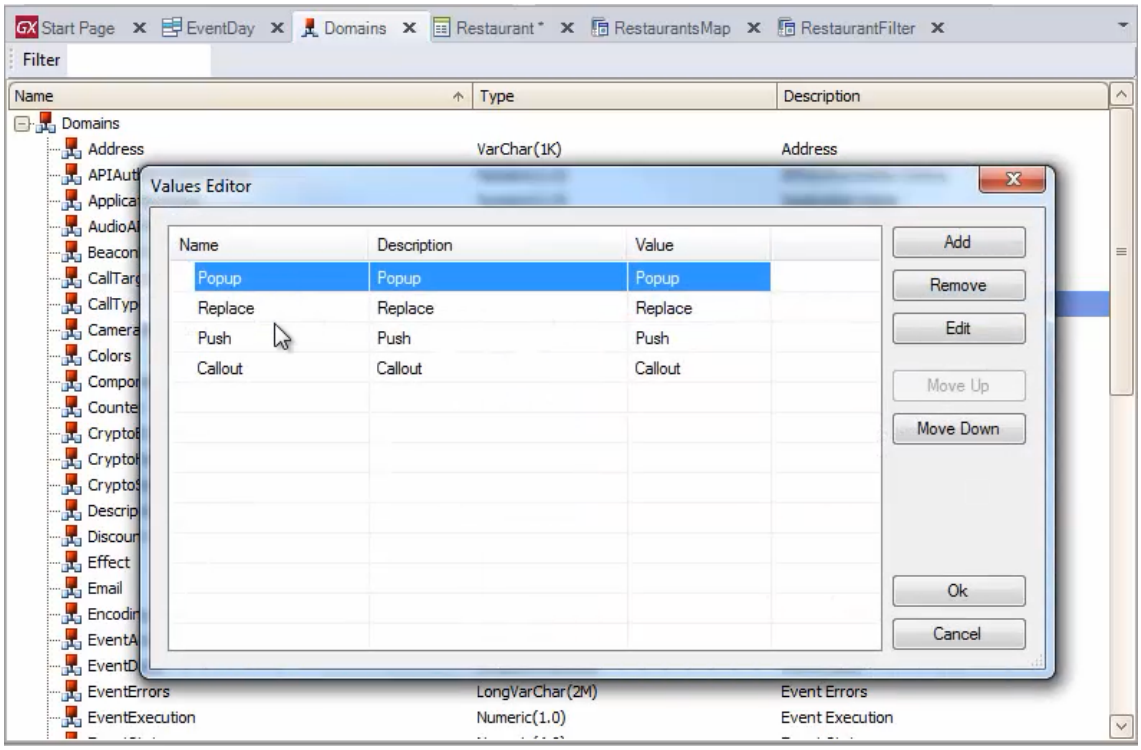
Para lograrlo, lo único que debemos hacer es anteceder la invocación con la definición de las CallOptions .. Type..



Para definir el tipo de call, tenemos el dominio predefinido → CallType

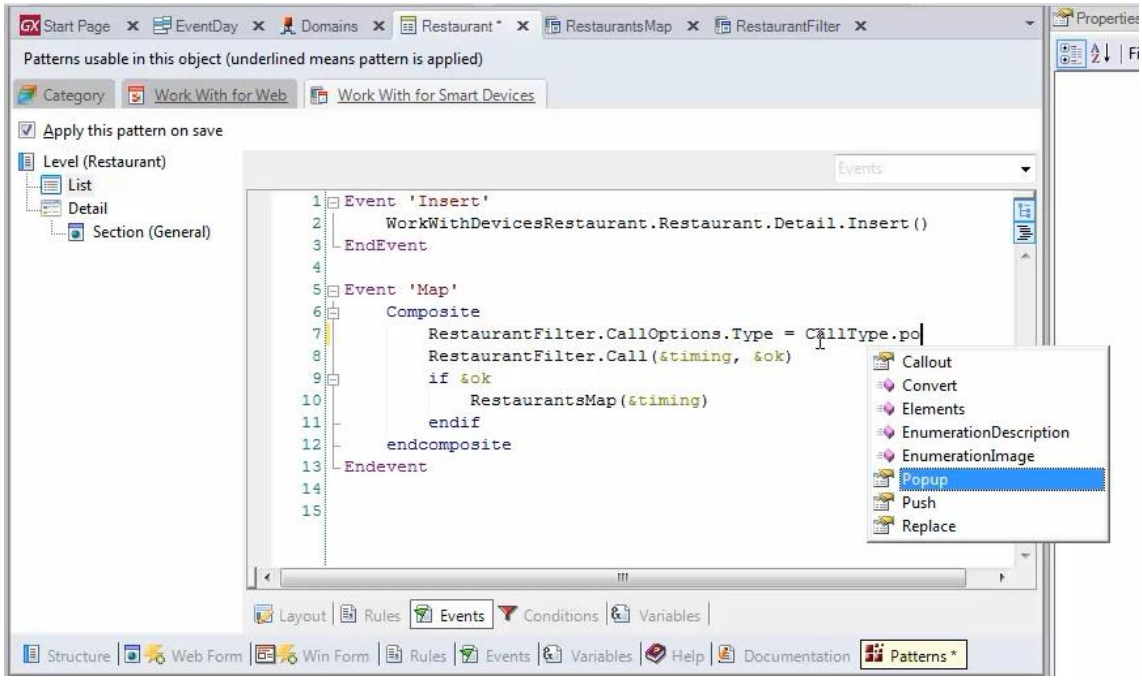


que como vemos es un enumerado



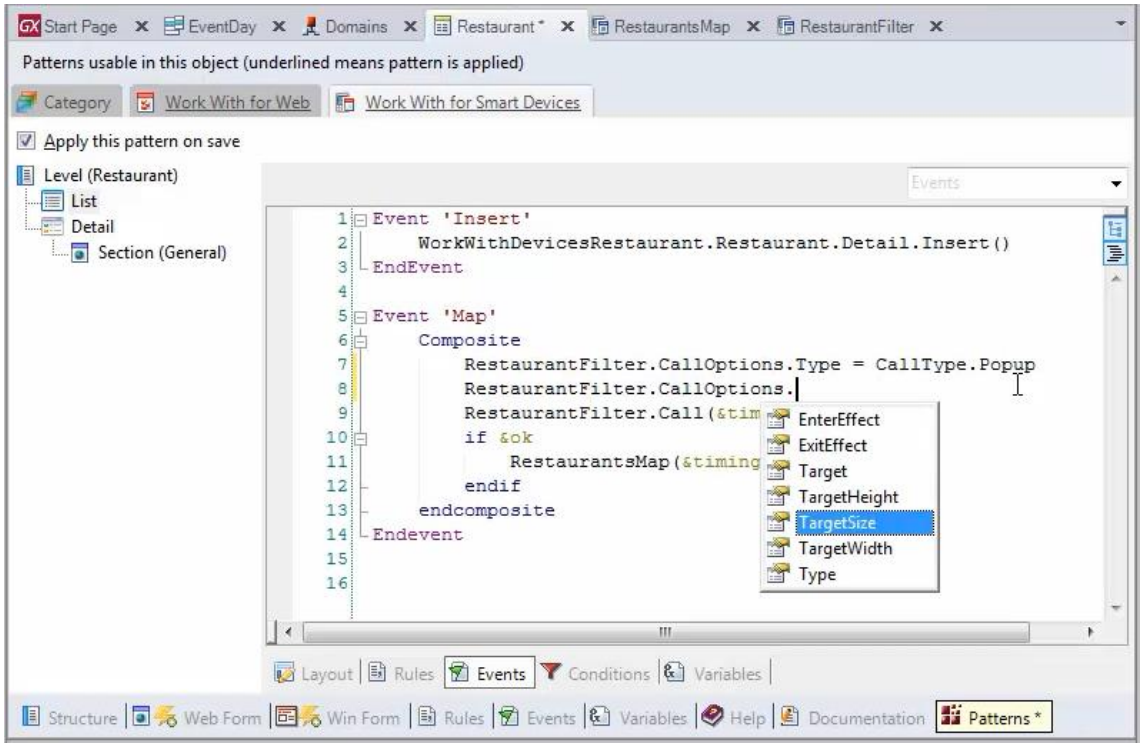
con las 4 opciones que habíamos visto.

Vamos entonces a elegir: CallType.Popup

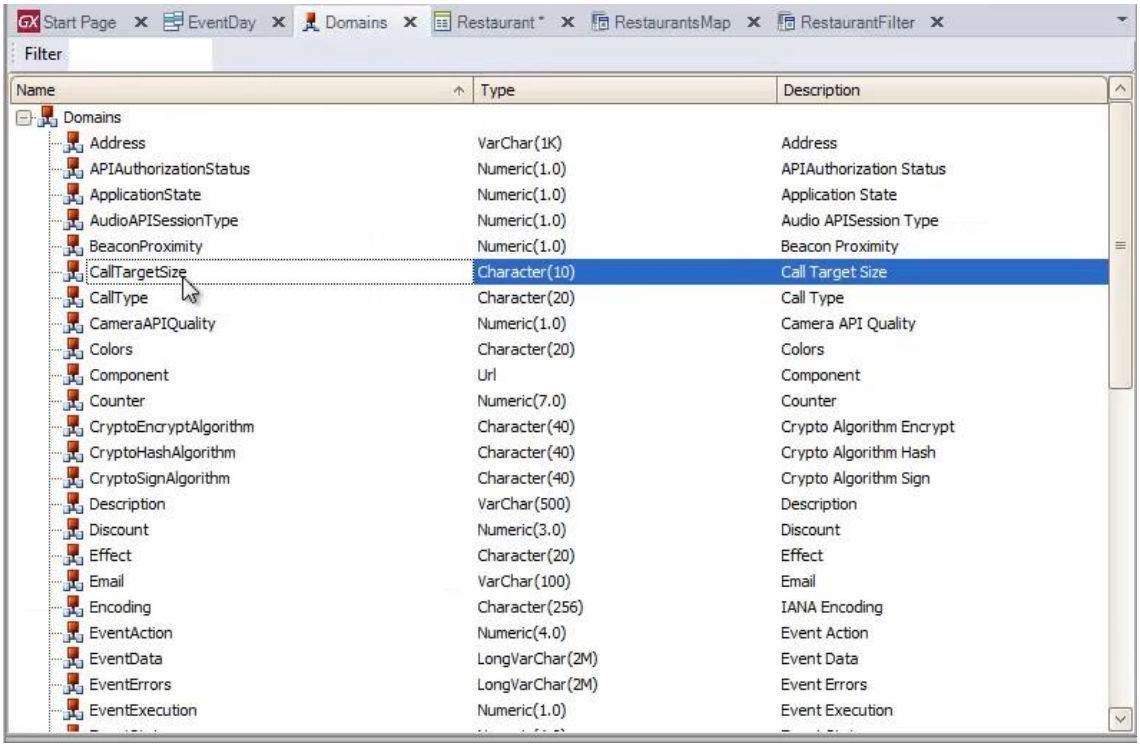


Si además queremos que este panel se abra ocupando un área de pantalla reducida (Small) respecto al List que lo está llamando, entonces también tendremos que definir el TargetSize.

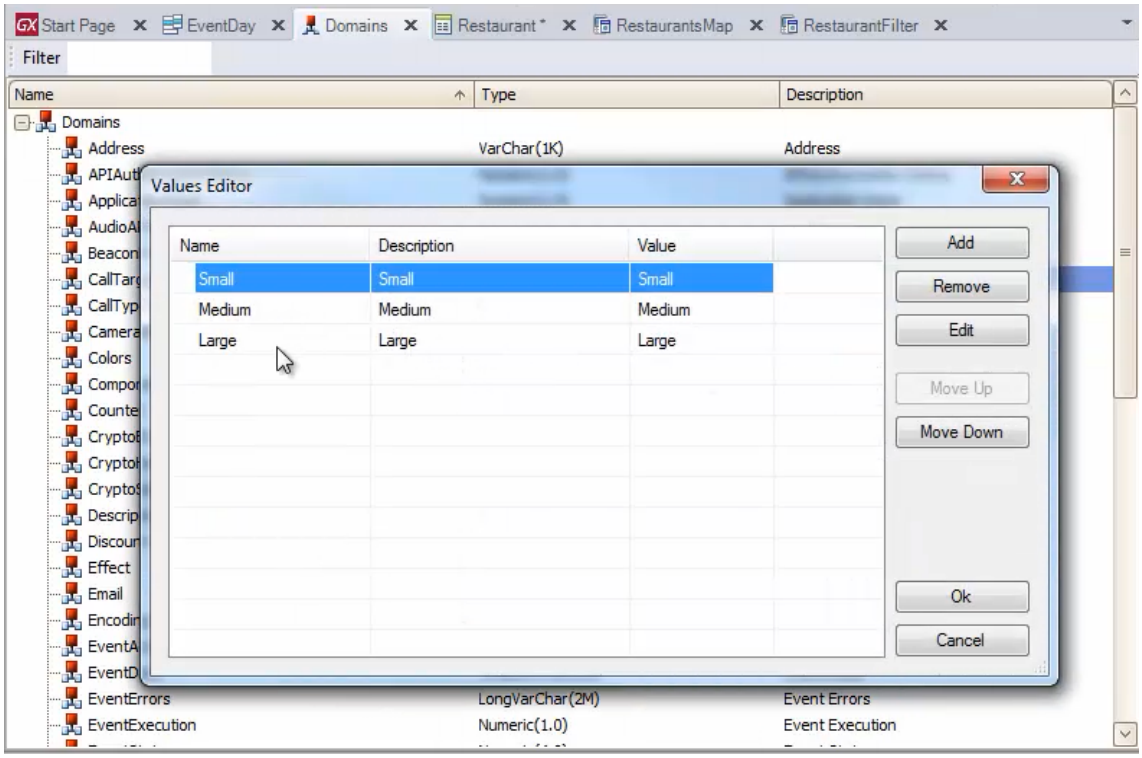
Otra vez haciendo uso de las CallOptions… punto… TargetSize..



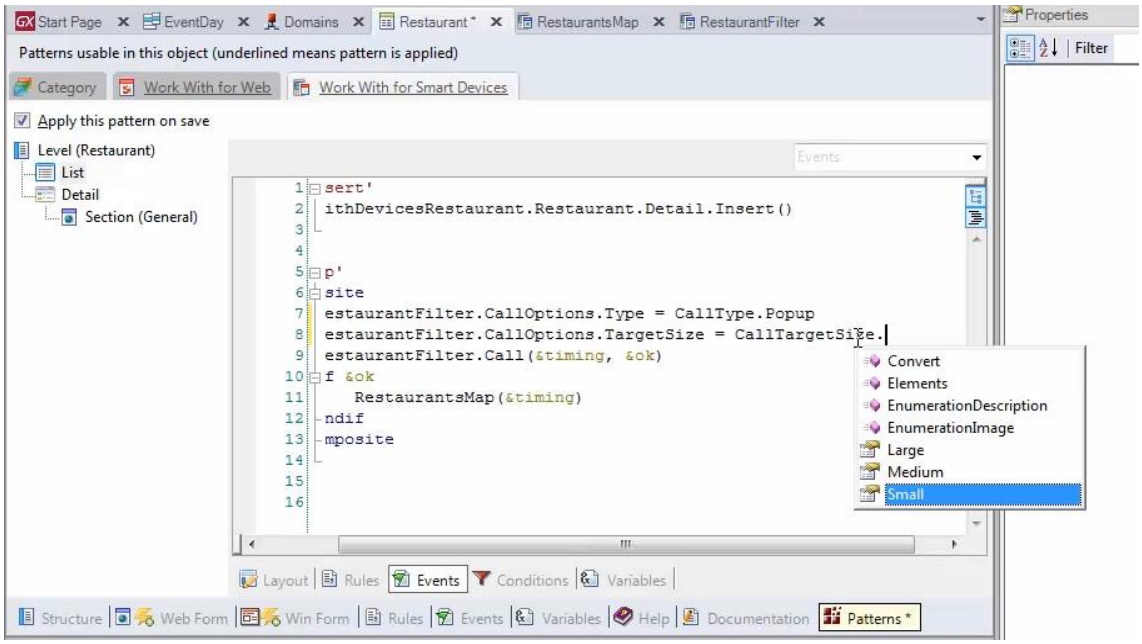
Y para esta opción también existe un dominio : CallTargetSize



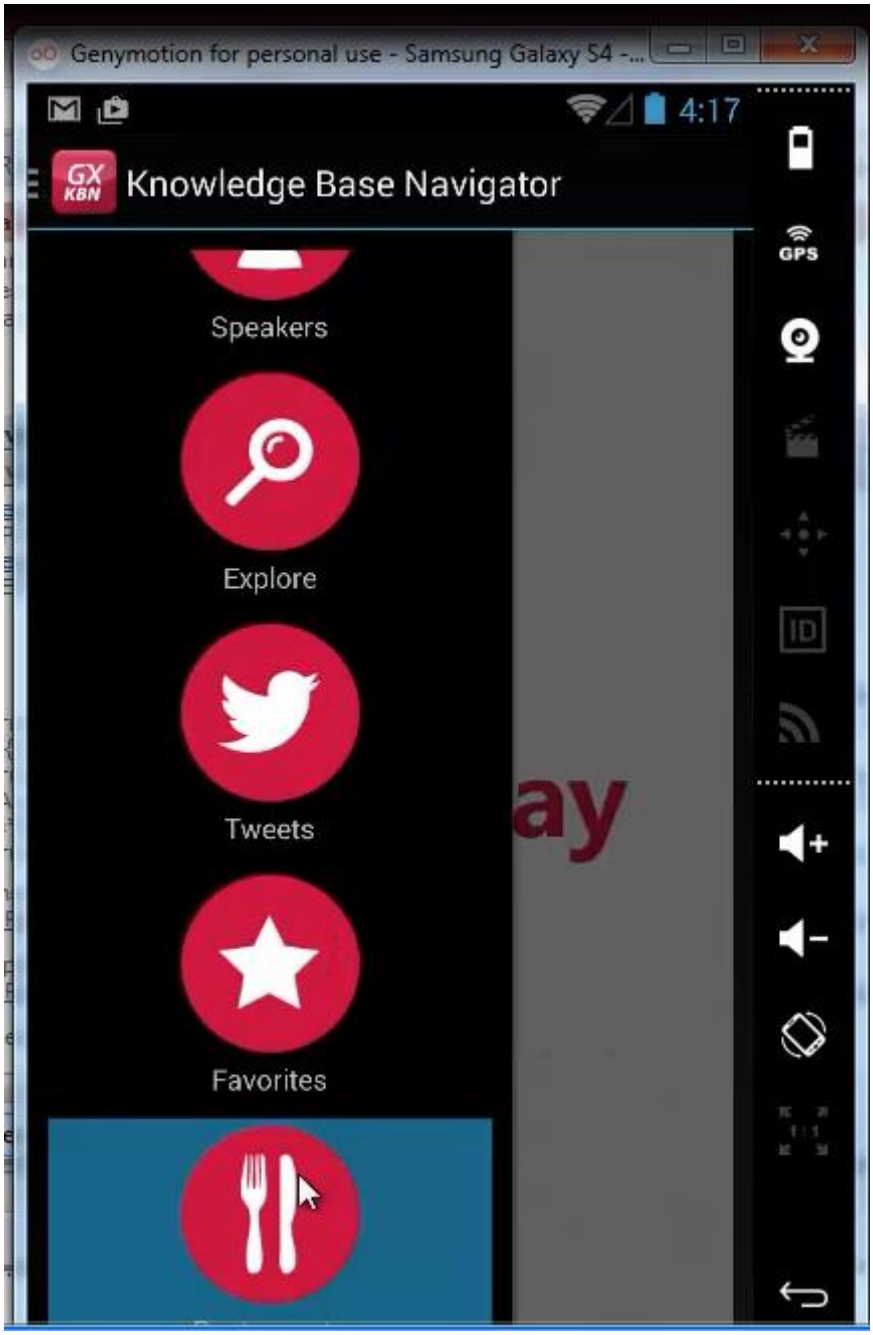
que vemos asume estos 3 valores:

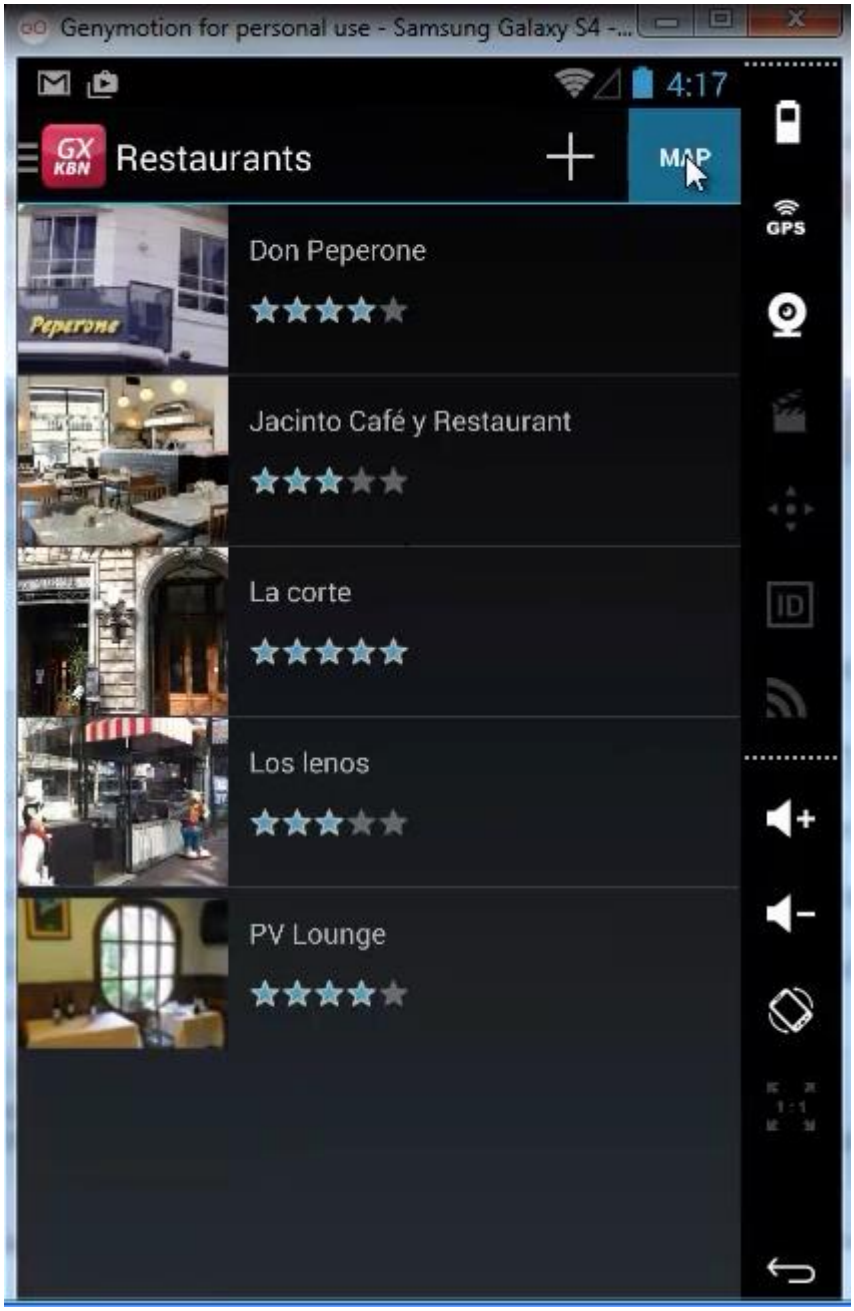


Vamos a definir: CallTargetSize.Small

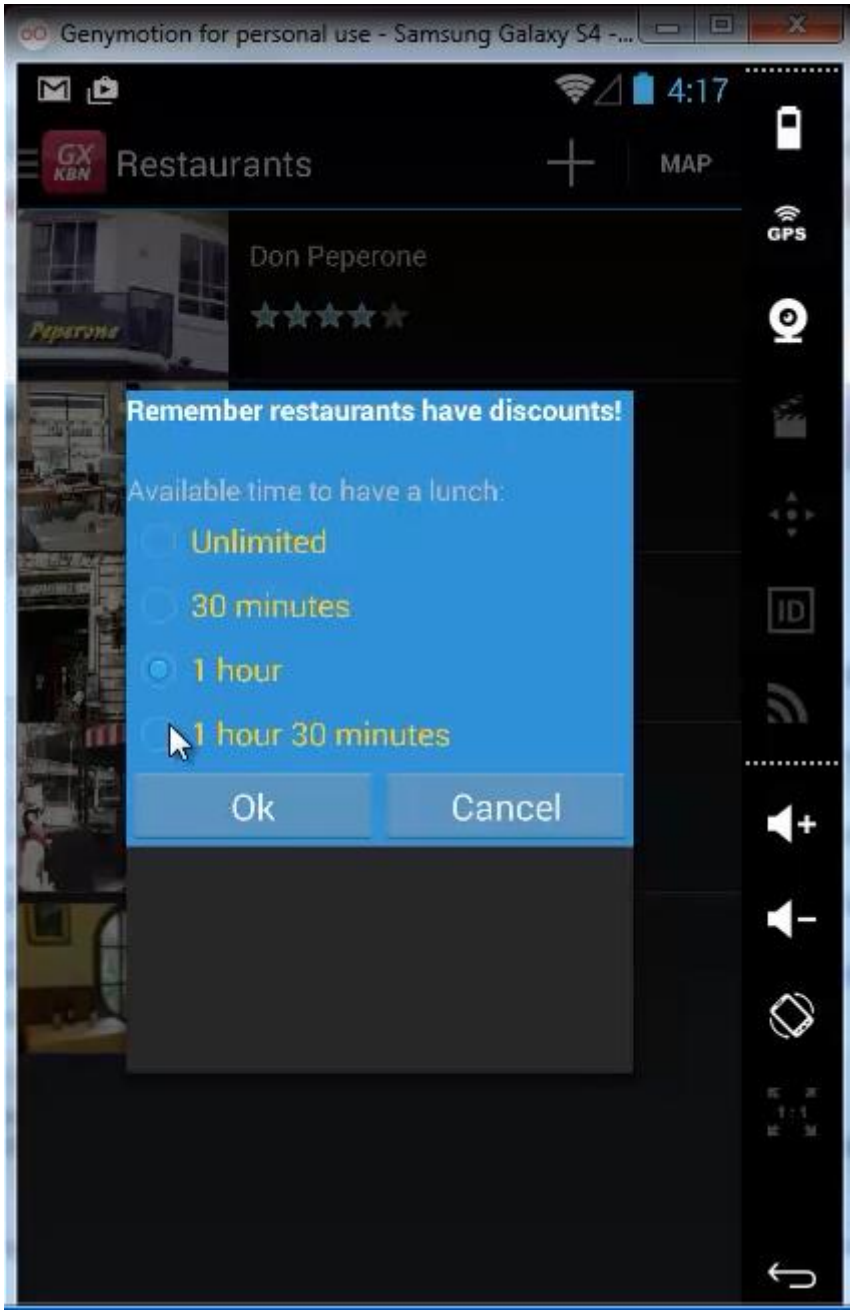


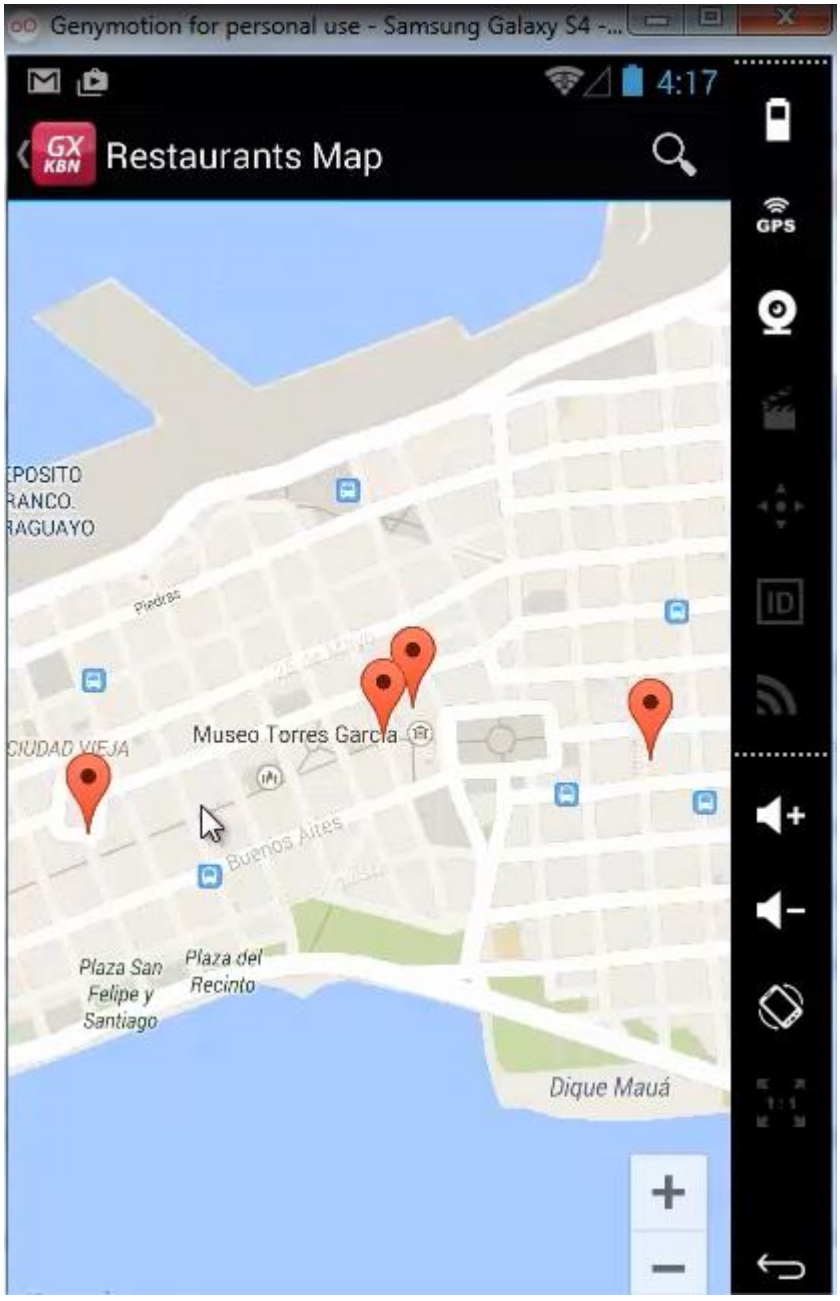
Hagamos F5 para probar.


Genymotion for personal use - Samsung Galaxy S4 -...
4:17
Knowledge Base Navigator
Speakers
Explore
Tweets
Favorites
GPS



Y aquí lo vemos


Genymotion for personal use - Samsung Galaxy S4 -...
4:17
Restaurants
MAP
Don Peperone
Remember restaurants have discounts!
Available time to have a lunch:
Unlimited
30 minutes
1 hour
1 hour 30 minutes
Ok
Cancel
GPS



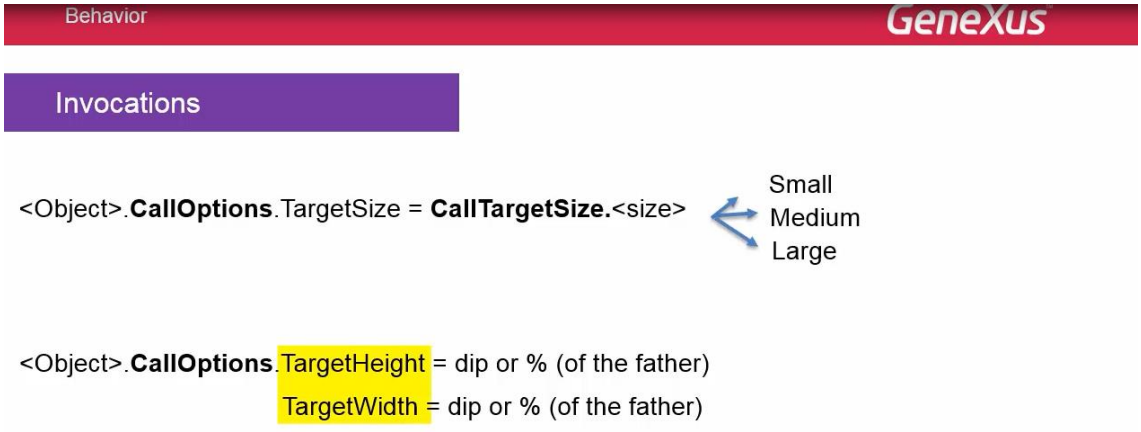
Invocations

<Object>.**CallOptions**.TargetSize = **CallTargetSize.**<size> ⇄ Small, Medium, Large

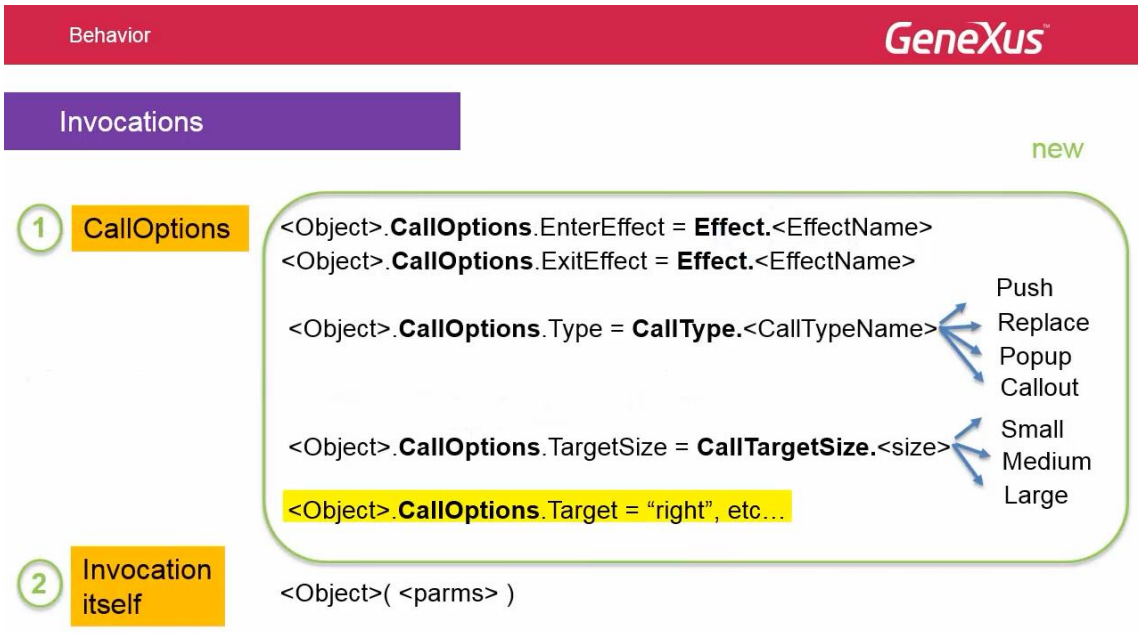
<Object>.**CallOptions**.TargetHeight = dip or % (of the father)
TargetWidth = dip or % (of the father)

También podemos definir el tamaño de la ventana llamada, en dips o porcentajes relativos al padre.

Esto es a través de las propiedades: TargetHeight y TargetWidth

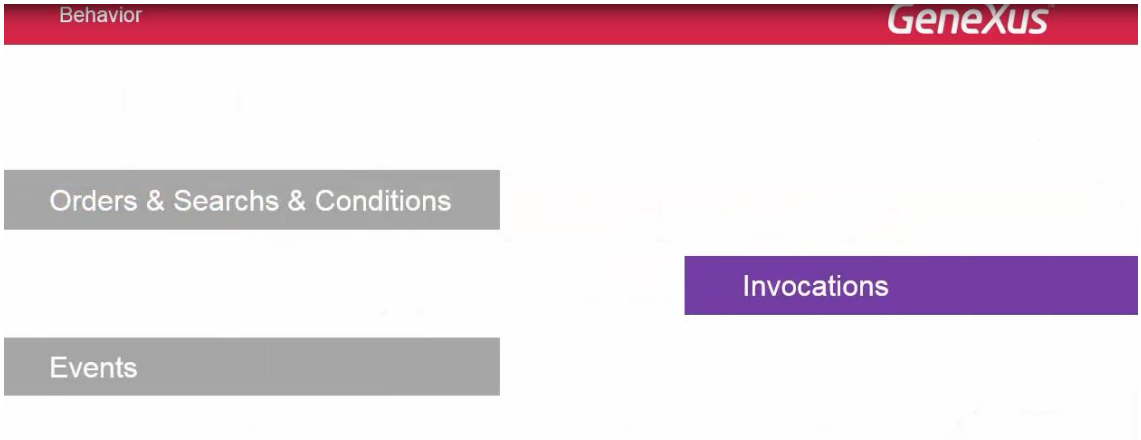


Sólo nos resta mencionar que las CallOptions Target



utilizadas cuando utilizadas cuando se tiene pantalla partida (no tiene sentido en teléfonos) para indicar en cuál de los targets posibles se desea cargar la pantalla llamada, no puede utilizarse con Callout o Popup.

Puede ver más de todo esto en nuestro wiki.



Con esto terminamos de ver lo más relevante en cuanto a la especificación del comportamiento de las aplicaciones móviles online.

Lo invitamos a adentrarse ahora en las aplicaciones total o parcialmente desconectadas: offline.