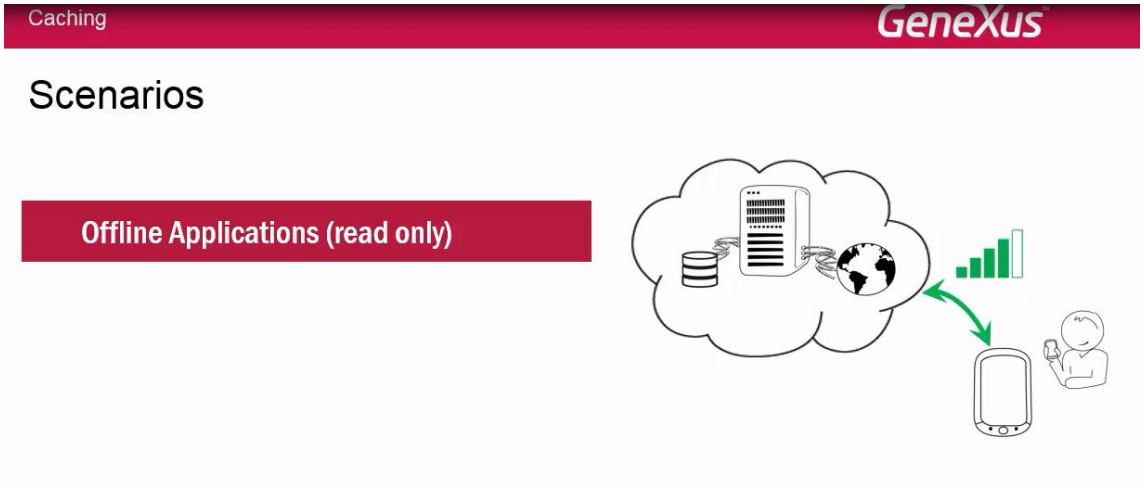
Behavior: Caching

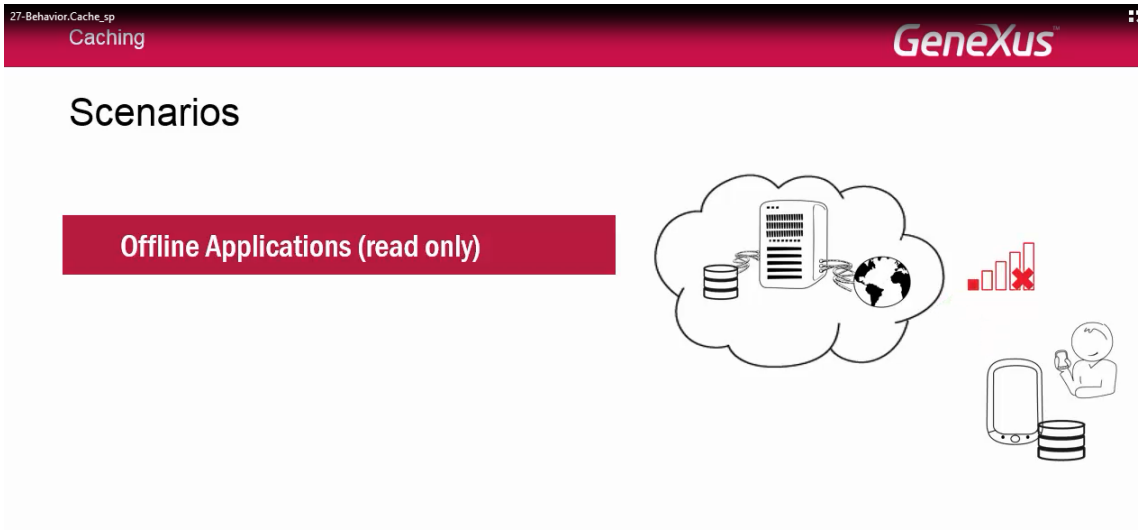
En este video veremos cómo habilitar o deshabilitar el caché de los datos navegados en el dispositivo, teniendo en cuenta aplicaciones de arquitectura online.



Existen varios escenarios en los que se necesita caching para el almacenamiento de datos en una aplicación generada para smart devices.

Por un lado, teniendo una aplicación de arquitectura online, es decir, que requiere conectividad para poder tomar acciones sobre la información, como leerla y modificarla, vamos a querer que los datos que ya fueron leidos del server, permanezcan en la memoria del dispositivo.

De modo que si se pierde la conectividad



se pueda seguir mostrando esa información en las pantallas de la aplicación. Es decir, que las aplicaciones puedan ser **Offline al leer**.

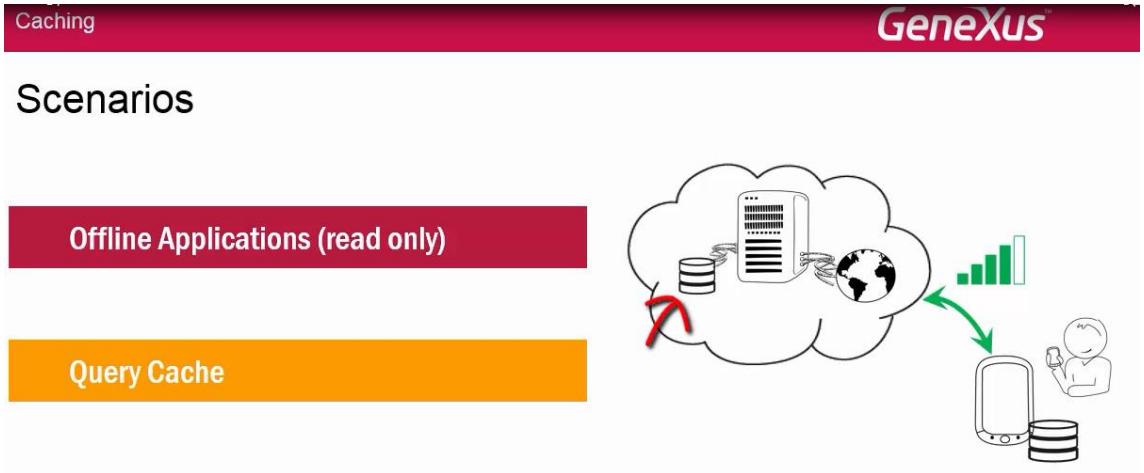
Las pantallas en las que no se haya ingresado previamente, es decir, aquellas para las que nunca se obtuvieron datos, estarán inaccesibles.

Este comportamiento es implementado automɑticamente por GeneXus en toda aplicación, almacenando en una base de datos SQLight que viene con todos los devices, la información devuelta por los objetos REST

junto con la fecha en la que esa información fue recuperada.

Por otro lado, para reducir el tráfico entre el cliente y el servidor

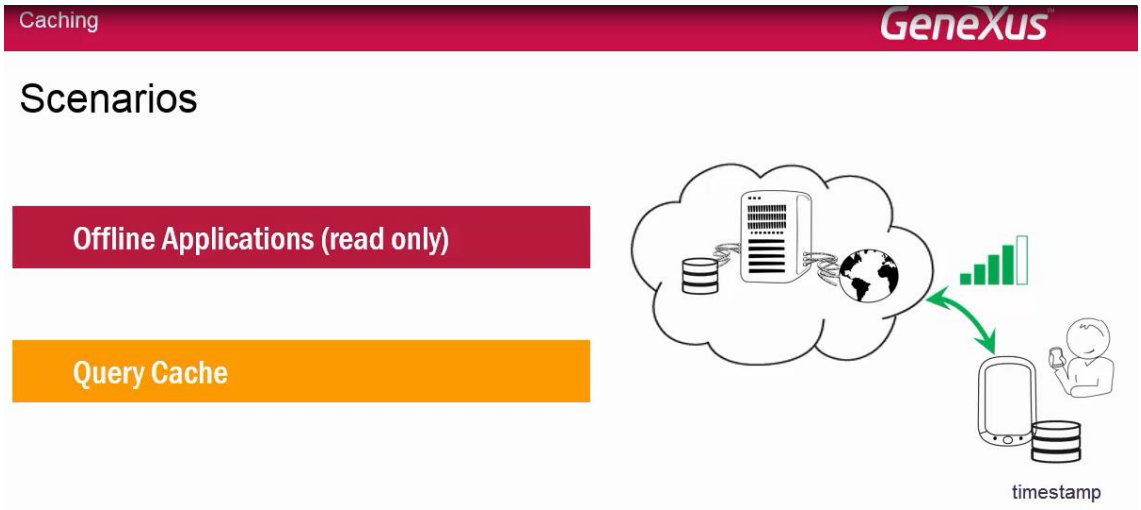
Podemos aprovechar este almacenamiento en caché de los datos ya consultados para evitar cuando tengamos conectividad, ir a buscarlos nuevamente



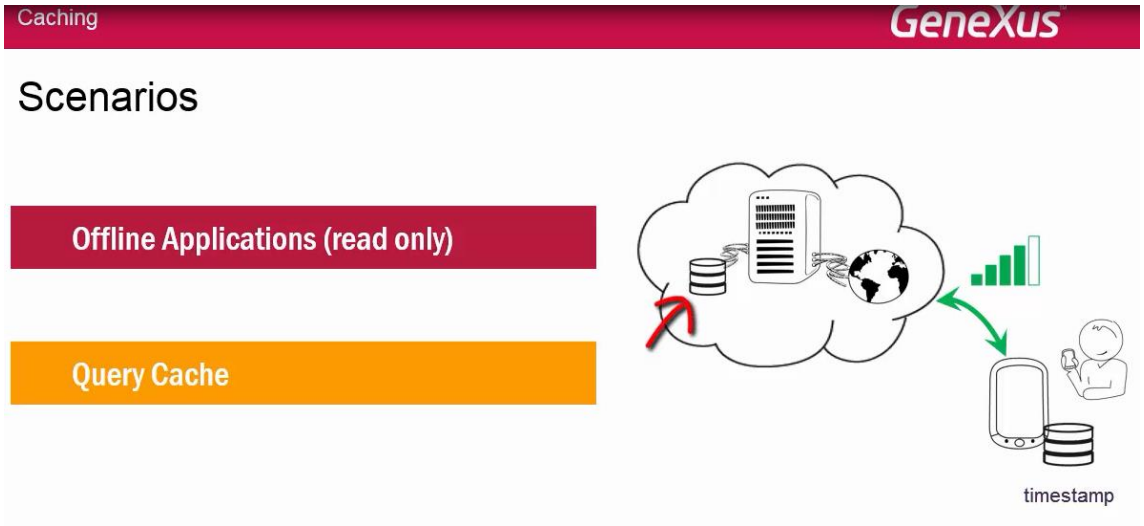
cuando estos no han cambiado.

Para resolver esto, es necesario saber para una consulta realizada desde el dispositivo, si los datos involucrados en ella, fueron cambiados desde la última vez que fue realizada la consulta.

Es para eso que ante cada consulta, se graba en el dispositivo, la fecha de la misma: timestamp

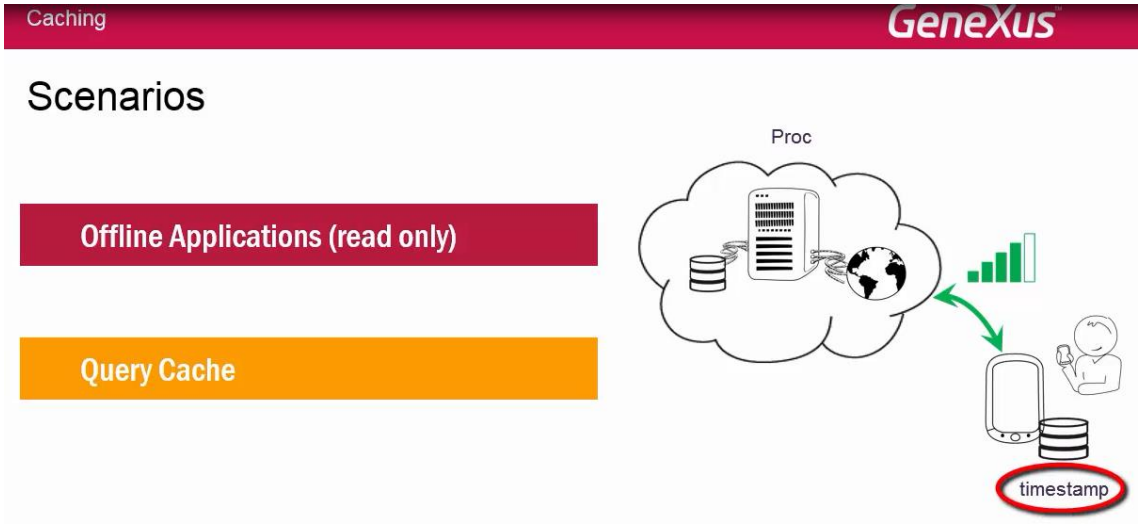


Y por otro lado en el server, se graba también

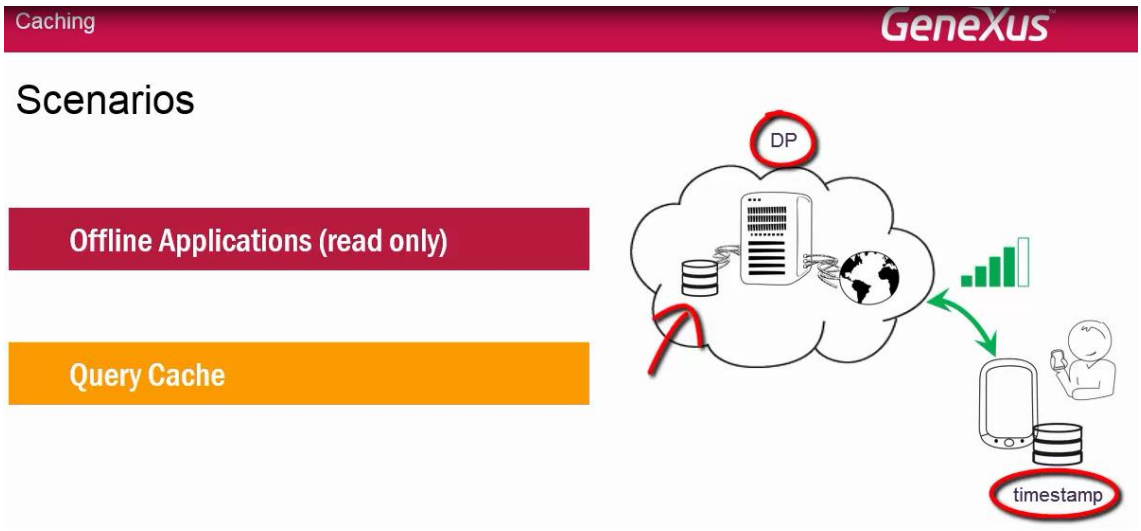


la fecha de última modificación que algún objeto GeneXus hizo, de cada tabla y registro.

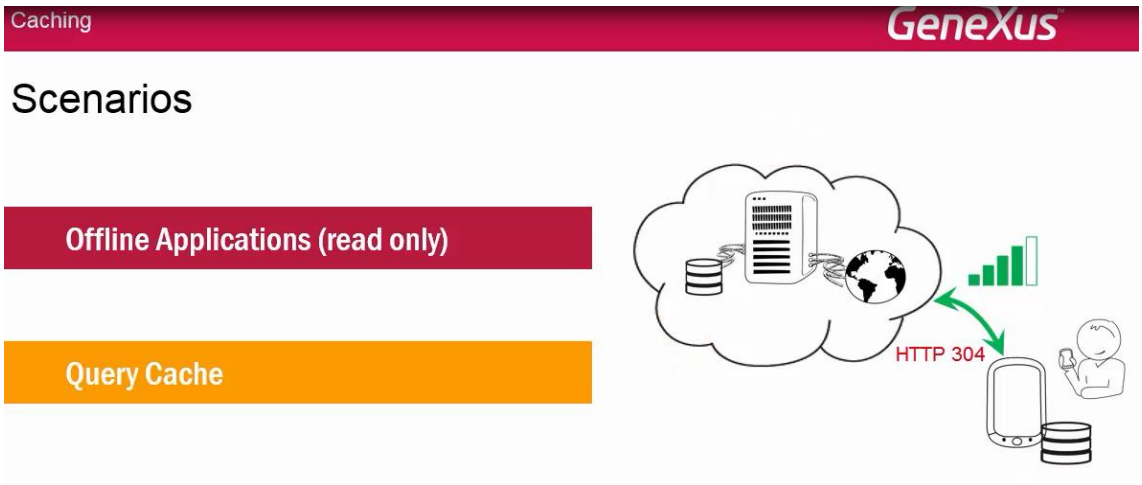
Así, cuando un dispositivo tiene que cargar la información de una pantalla, al invocar al Data Provider, Business component o procedimiento REST correspondiente, envía ese timestamp



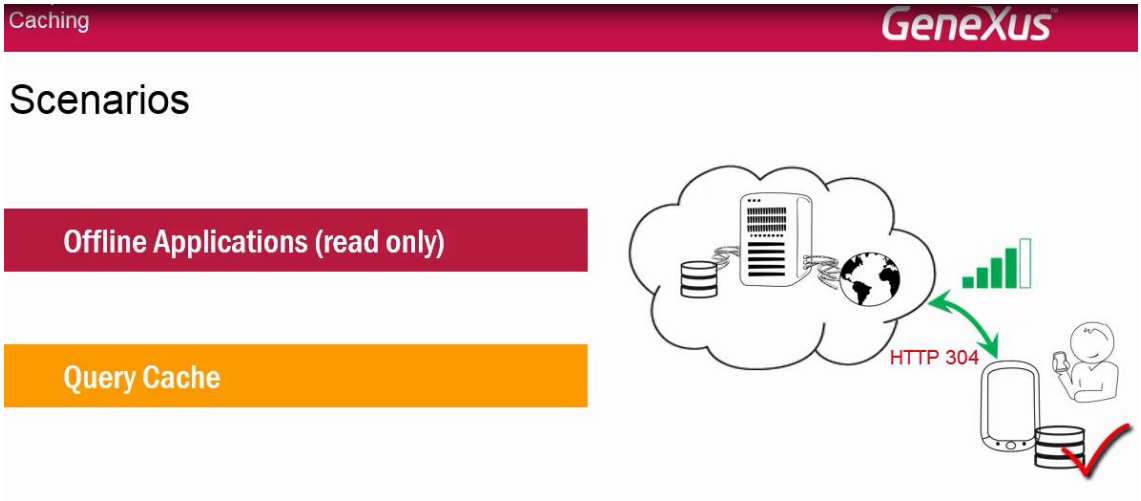
Antes de ejecutar el servicio REST, el server deberá comparar la fecha recibida, con las fechas de última modificación de los datos de las tablas que deben accederse, para saber si ejecutar la consulta nuevamente



Y con ellos los eventos del server: Start (si es la primera vez), y Refresh y Load.. y enviar los datos… o enviar una respuesta al cliente HTTP304

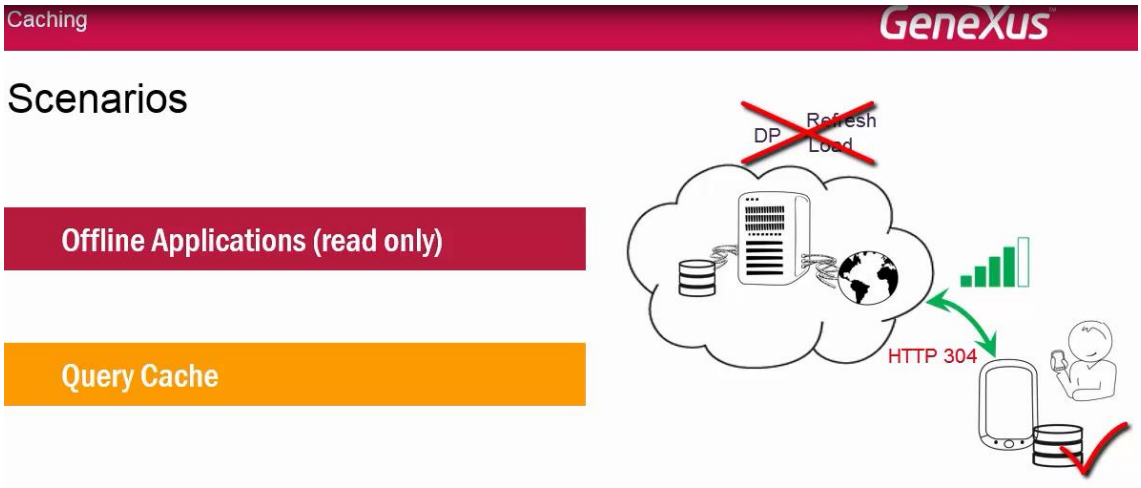


que le informará a este que los datos que ya tiene en caché, son los últimos

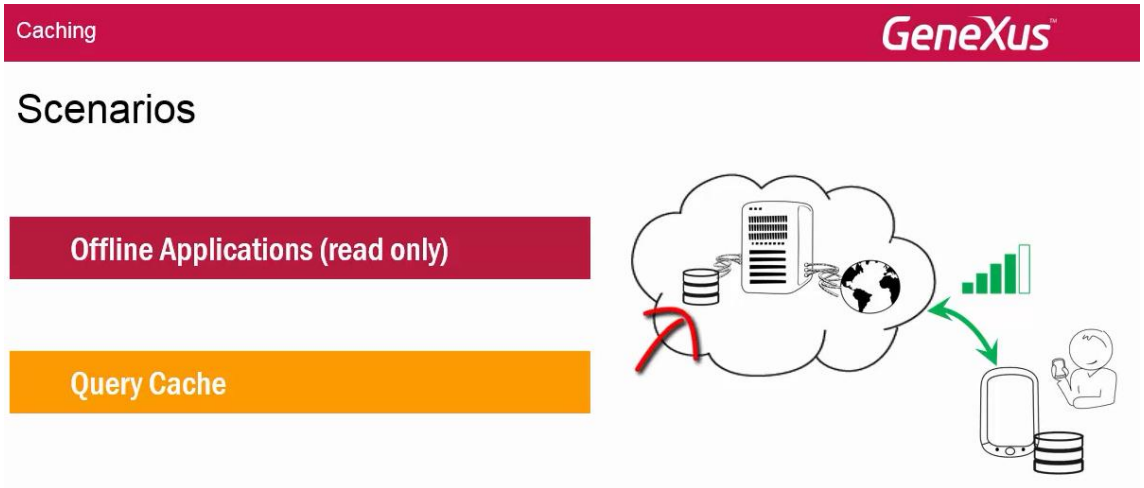


Así que debe utilizar esos.

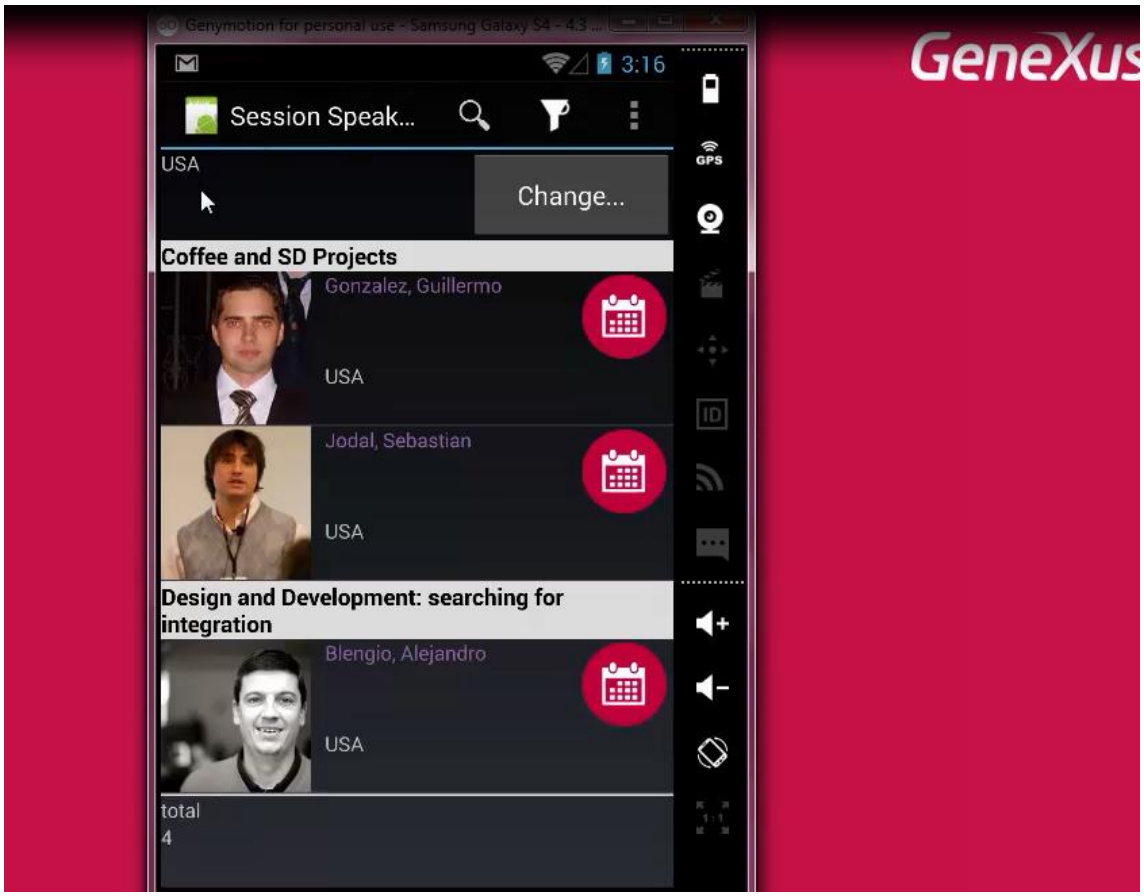
Observemos que en este caso, no se dispararán los eventos Refresh y Load



Pero… qué pasa si los datos son modificados a través de una aplicación “no GeneXus”? Es decir que no mantiene los registros de última modificación en la base de datos central?

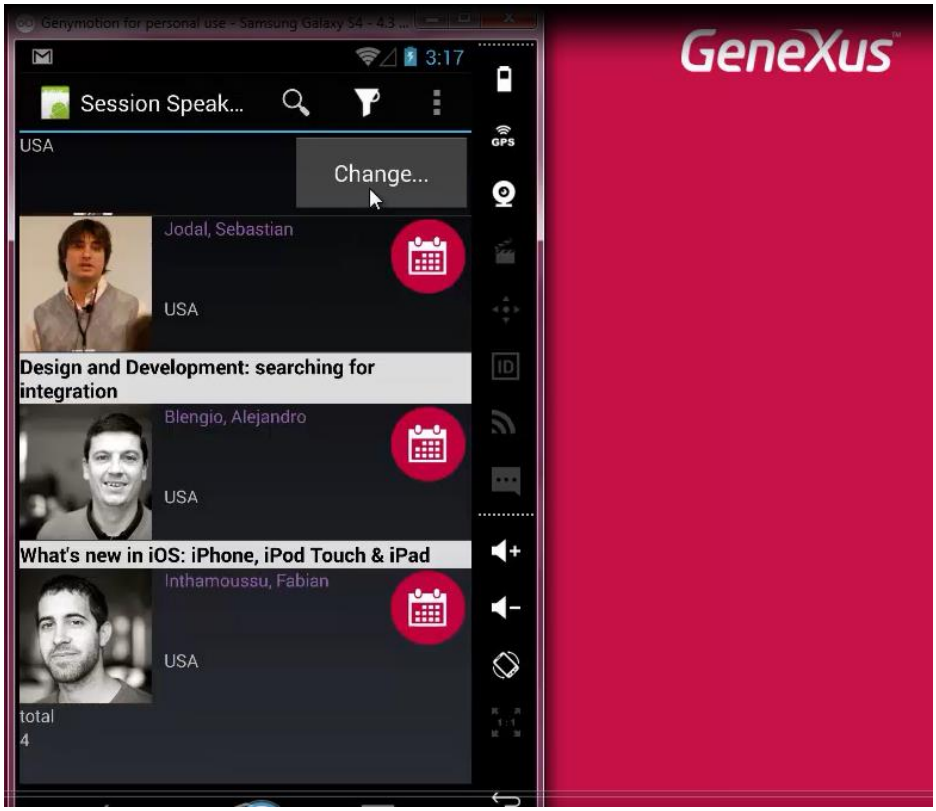


O incluso…. ¿qué sucede si necesitamos que se ejecute el evento Refresh para refrescar la pantalla, por ejemplo para que se vuelvan a cargar las variables aún cuando los datos de la base de datos no se han modificado?

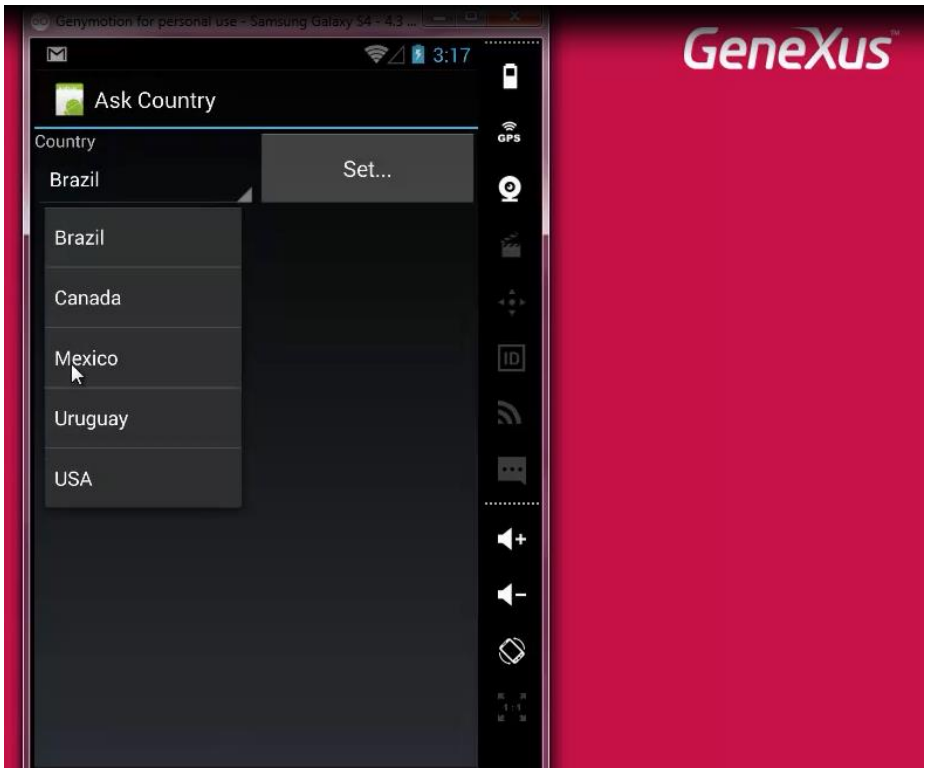


Por ejemplo, en el caso de una pantalla que muestre el país obtenido de una websession y las conferencias con oradores de ese país… y la cantidad total de oradores cargados.

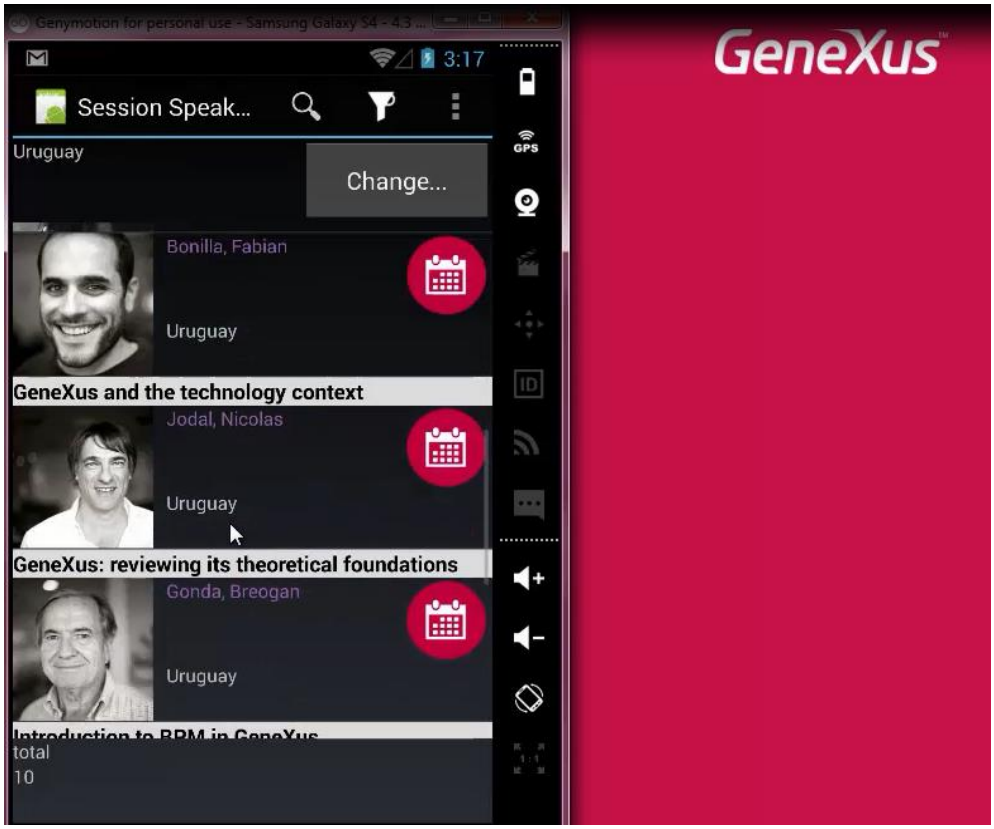
Tenemos asimismo un botón



para cambiar el país

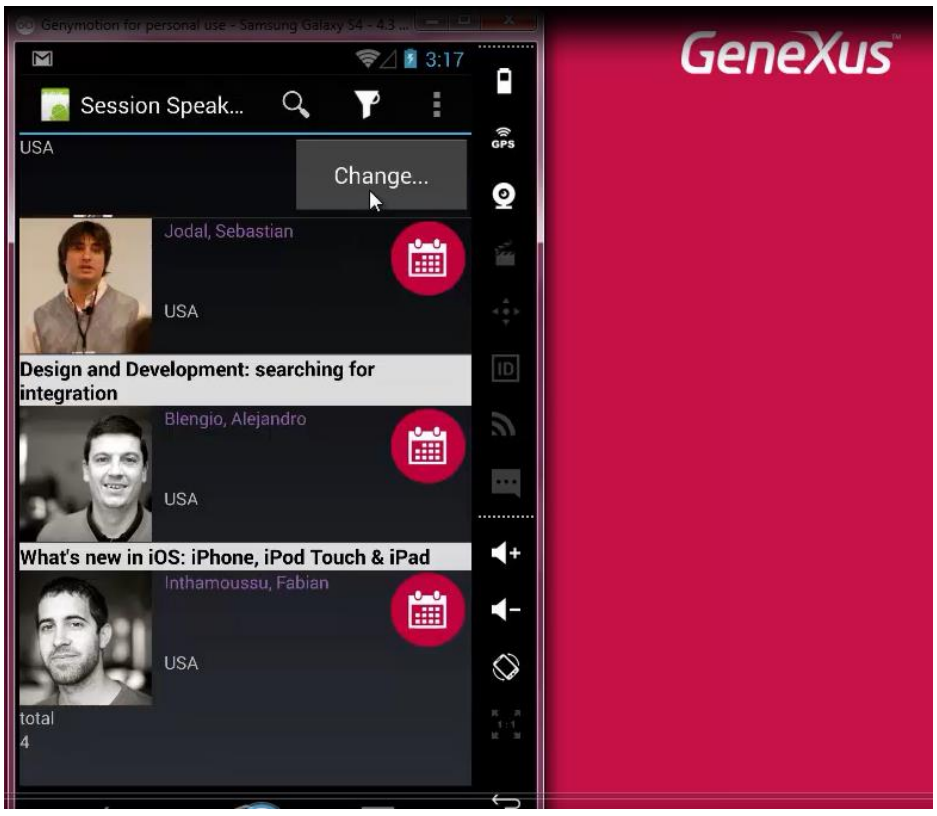


En la websession… y volver a cargar este panel



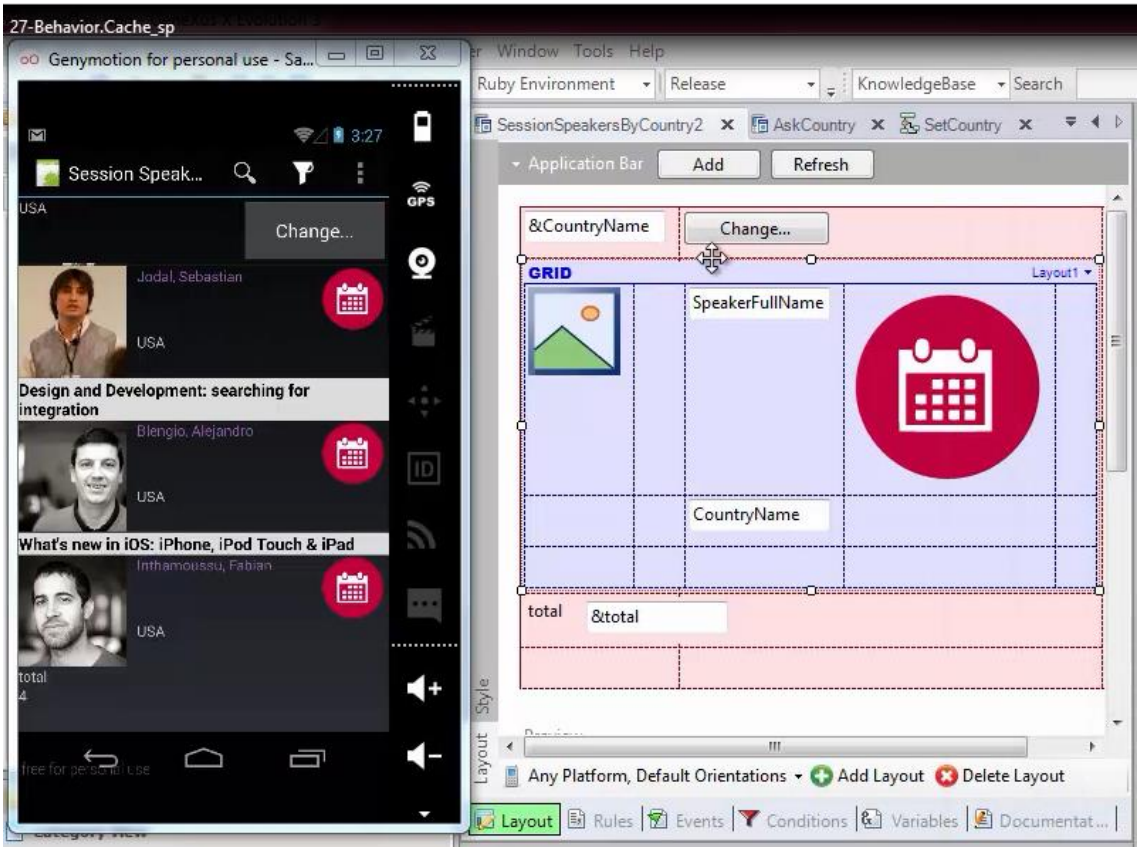
que está mostrando ahora las conferencias con oradores de Uruguay.. y el total.

Si volvemos a cambiar,

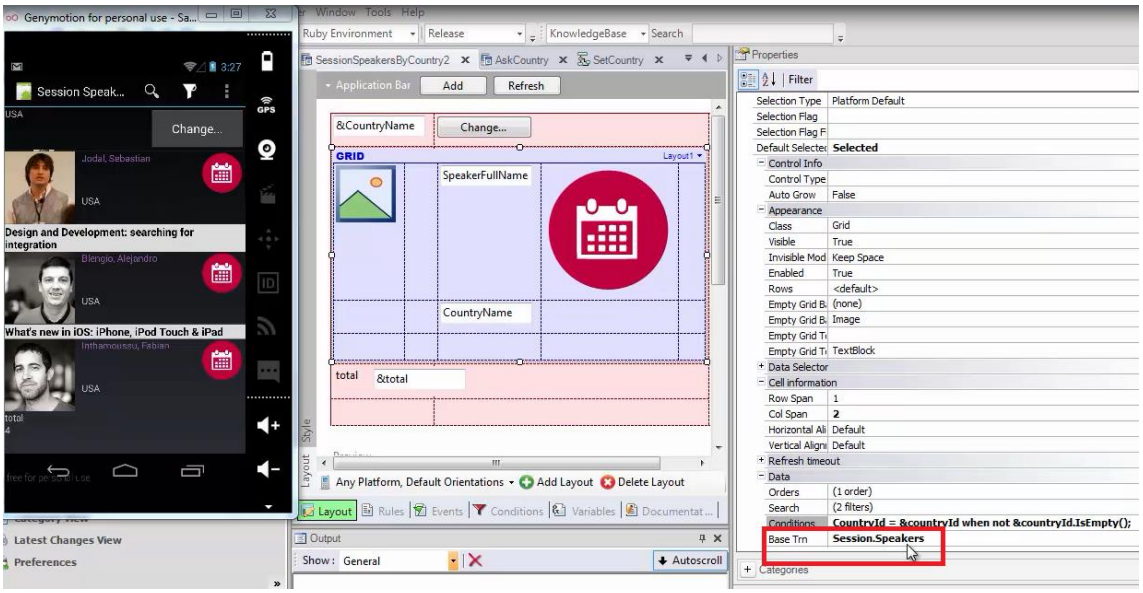


este será el comportamiento esperado.

Veamos qué tenemos que programar en GX para conseguirlo.



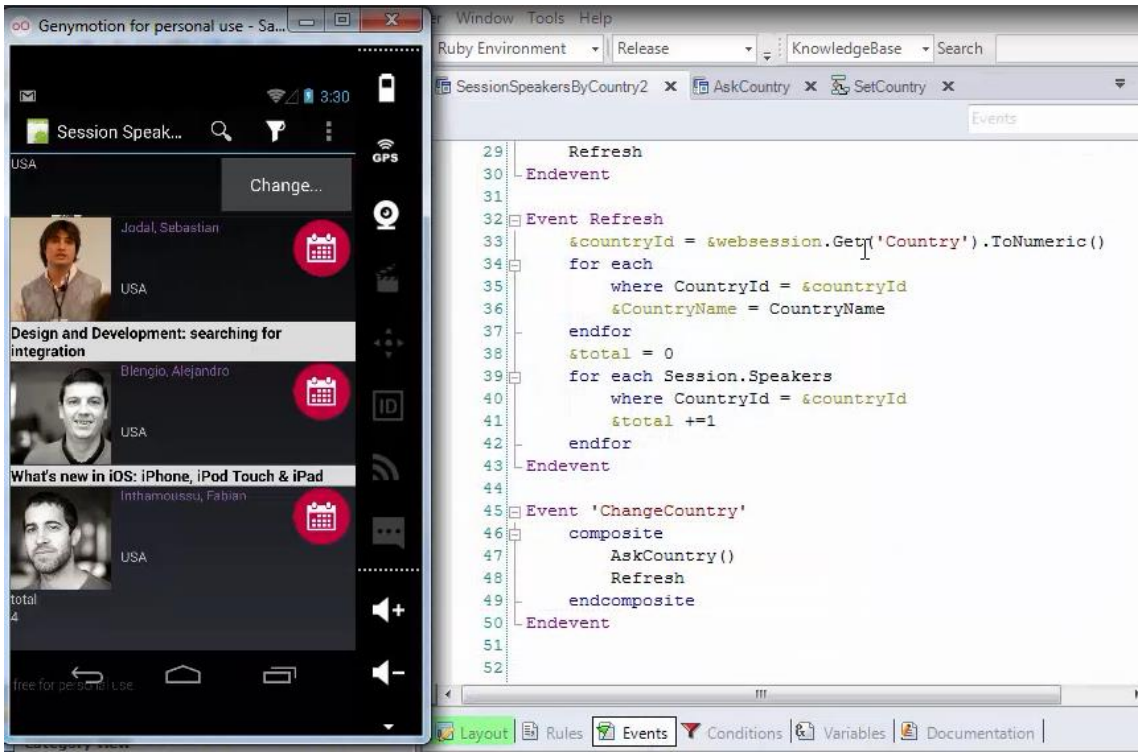
Aquí tenemos el panel con una variable CountryName, el botón Change, la variable &total, y un grid cuya tabla base corresponde a la de oradores de las sesiones (de las conferencias)



Vemos que el grid se está filtrando a su vez, por esta condición de aquí, donde la variable &countryId deberá ser recuperada a partir de una websession.

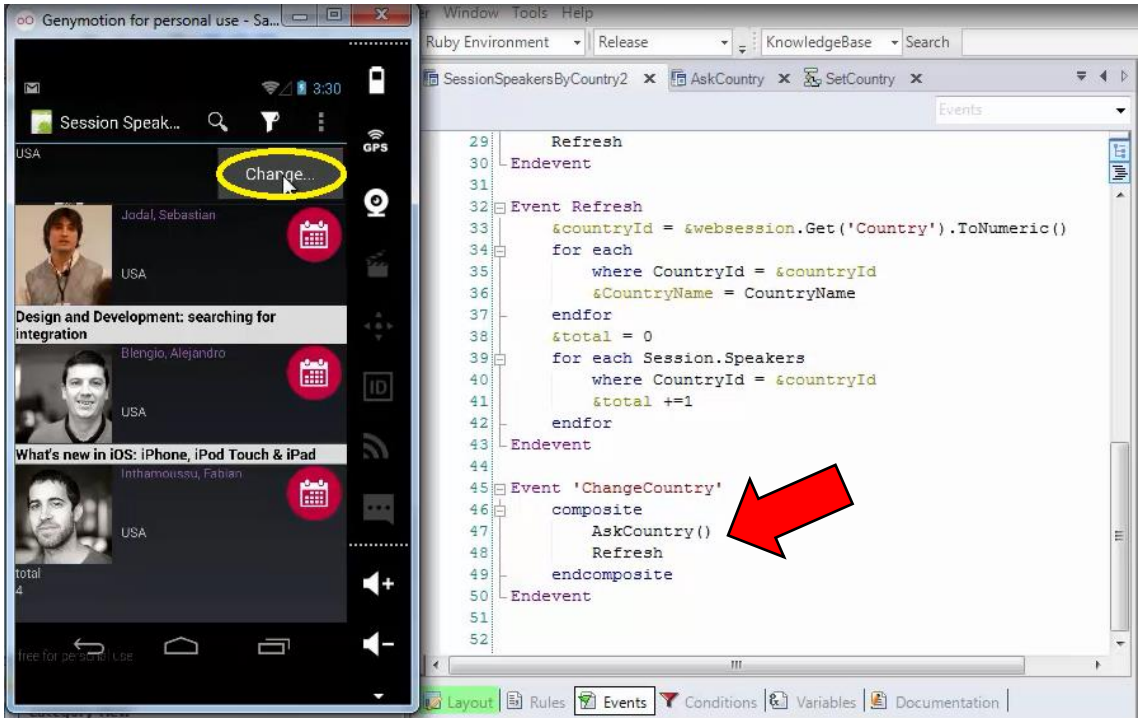
Será por tanto en el evento Refresh, donde vamos a tener que recuperar ese valor de la websession, buscar el &CountryName correspondiente a ese &CountryId y a su vez totalizar los oradores correspondientes, los que serán cargados luego en el Load en el grid.

Si vamos a ver los eventos,



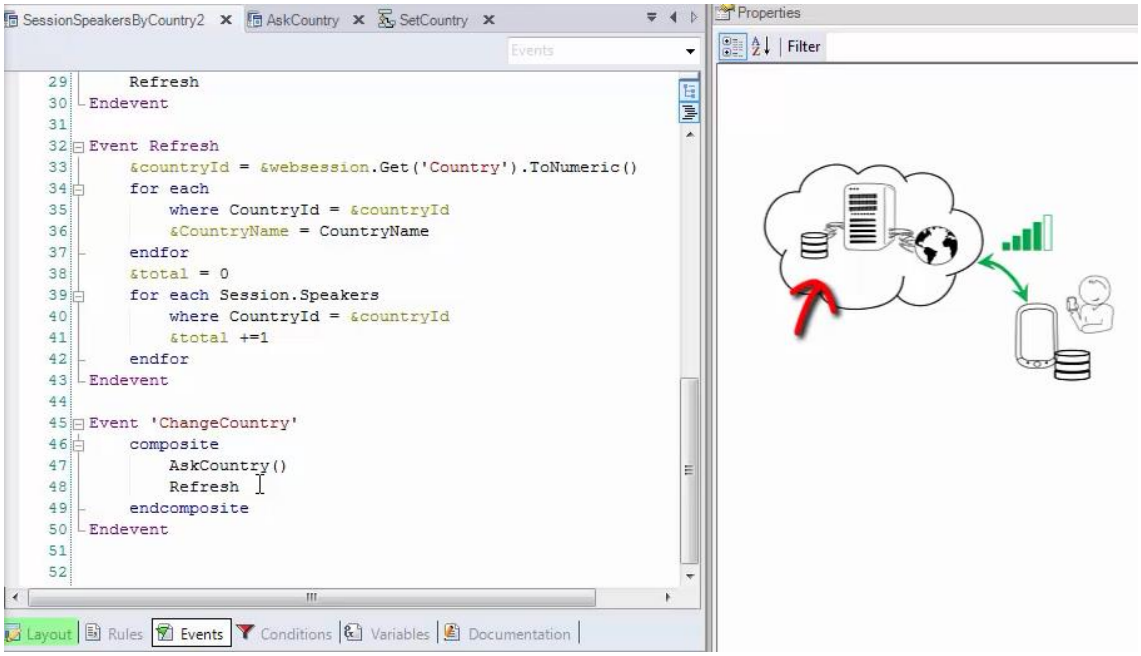
aquí tenemos la invocación al método Get de la websession para recuperar el &countryId allí grabado… aquí recorremos la tabla de países para poder recuperar el nombre del país (el que irá cargado en la variable que se mostrará en pantalla)… y por otro lado aquí, estamos contando la cantidad de registros recuperados en este otro For each.

Cuando queremos cambiar el país, presionamos este botón que estará asociado a este evento: 'ChangeCountry'



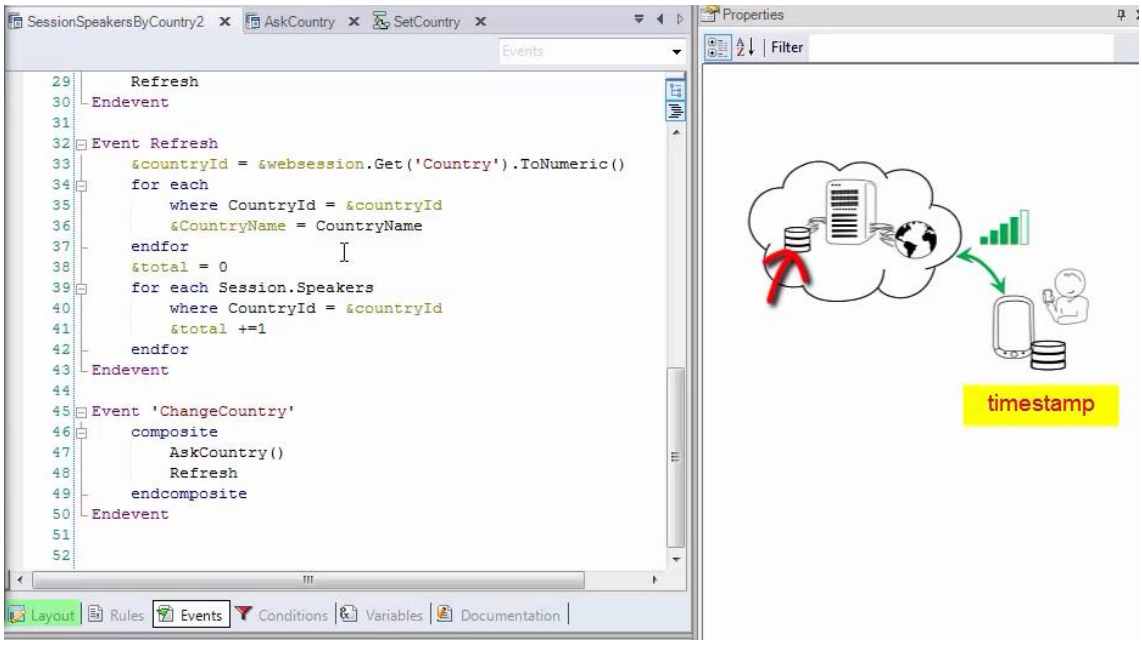
que lo que hará será llamar a otro panel 'AskCountry' que es el que efectivamente modificará el país y lo hará en la variable de sesión 'Country' y luego, al volver de ese panel realizará un Refresh, en el que necesitaremos que se vuelva a ejecutar el evento Refresh, para obtener el nuevo valor de la variable de sesión.. asignárselo a CountryId.. a partir de ese CountryId cargar el &CountryName correspondiente y mostrarlo al usuario en la pantalla… y totalizar la cantidad de registros que se cargarán en el grid… así como cargar ese grid con esa nueva condición.

Pero… si tenemos el caching habilitado, al realizar este Refresh, se va al servidor web

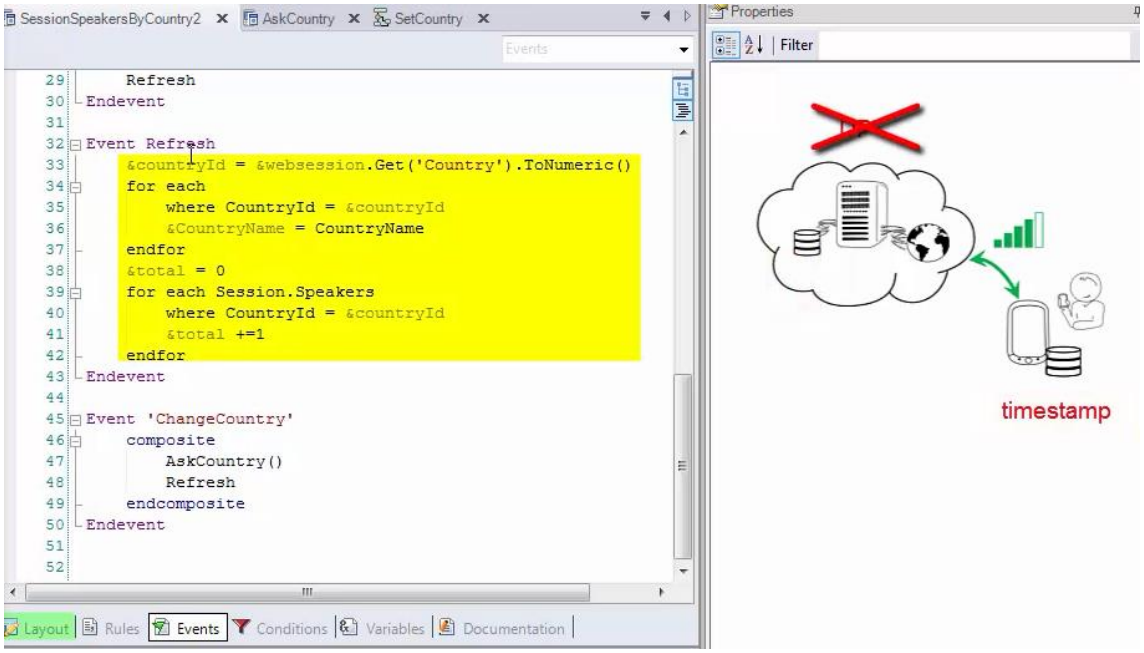


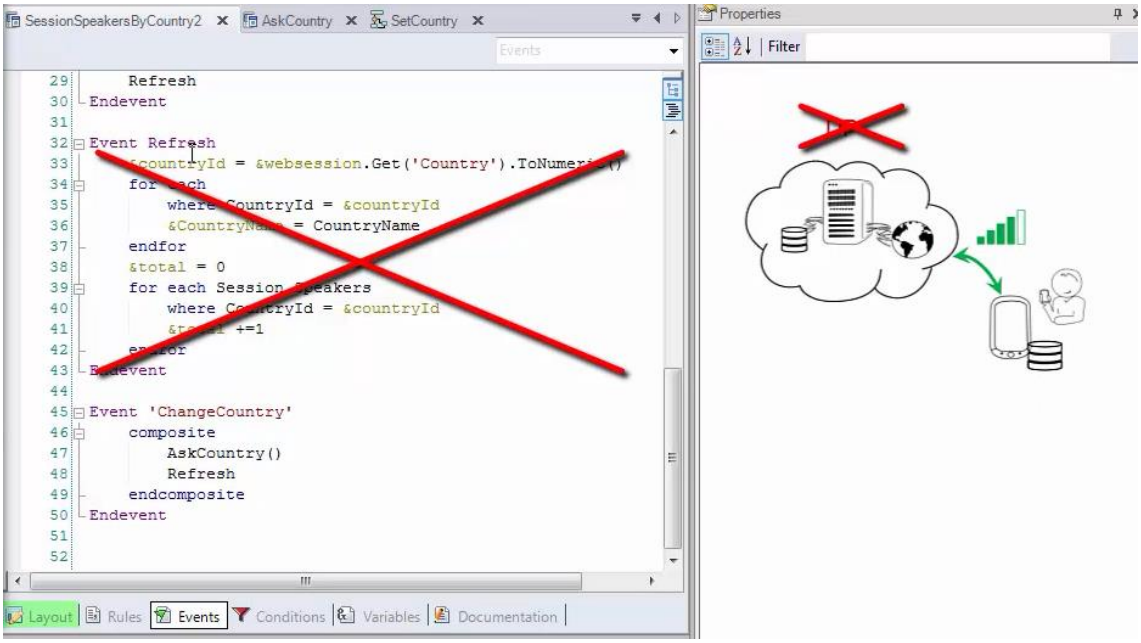
enviando la fecha de la última consulta realizada a la base de datos.

Y si en la base de datos centralizada, ningún otro objeto de la aplicación, modificó los valores de country o de los oradores de las conferencias, entonces el servidor encontrará que la fecha de la última consulta efectuada por este dispositivo,

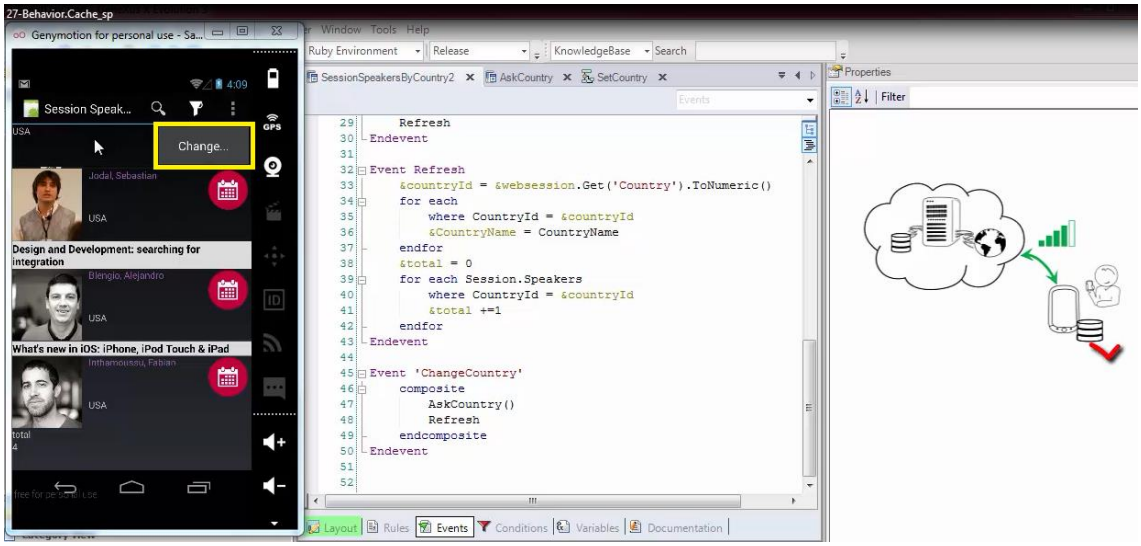


es posterior a la fecha de la última actualización en la base de datos, por lo cual el data provider correspondiente en el que se encuentra este Refresh, no será ejecutado

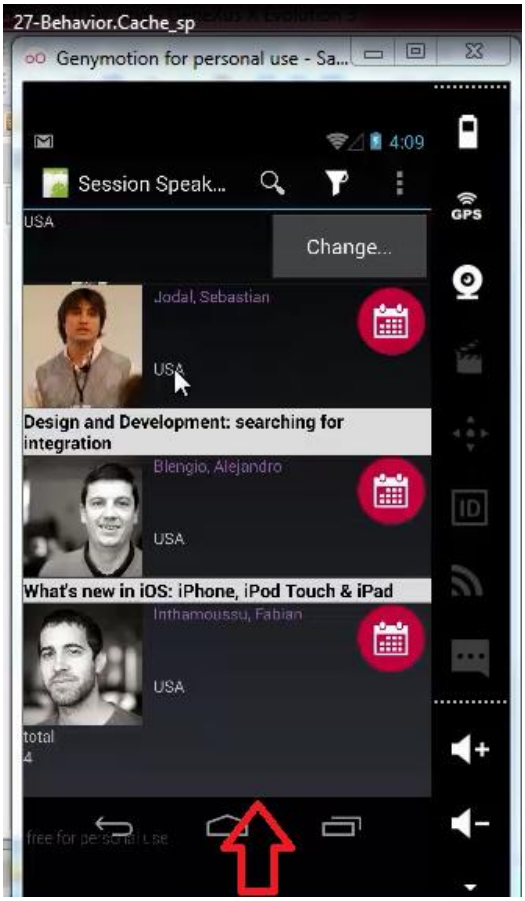




así como tampoco el data provider que corresponde al grid… por lo cual, lo que deberíamos ver en ejecución, al presionar este botón y cambiar el país

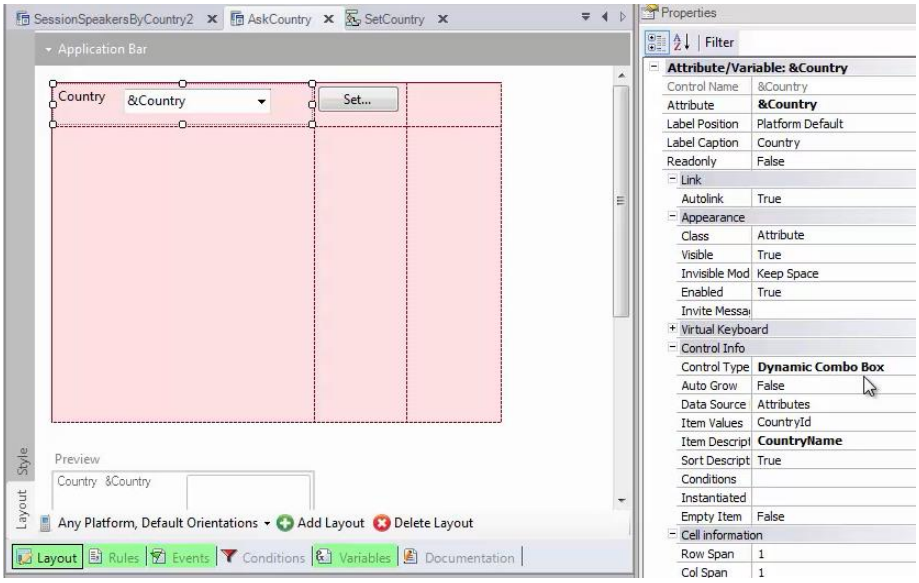


es esta misma pantalla inmodificada

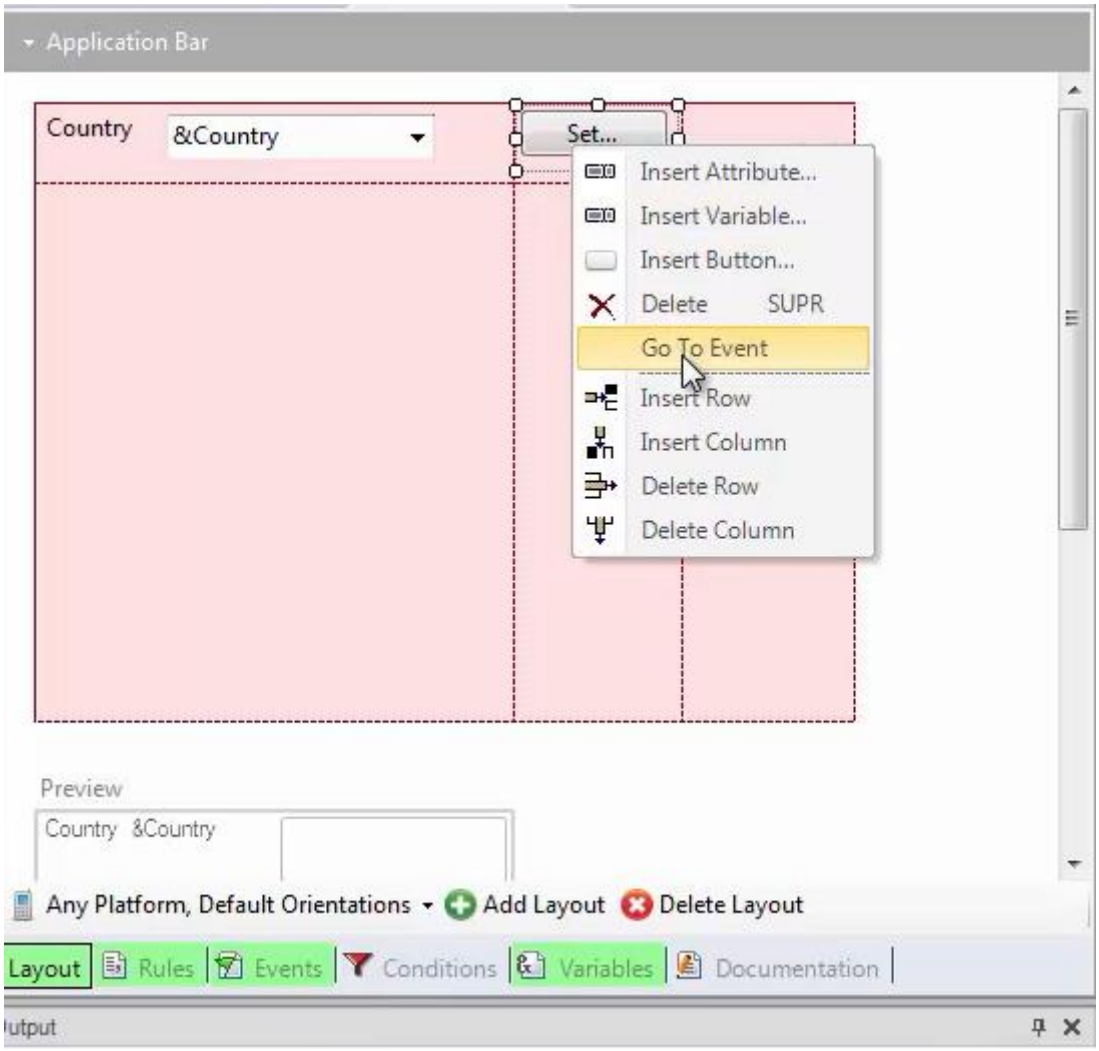


¿Por qué esto no sucedió en nuestra ejecución?

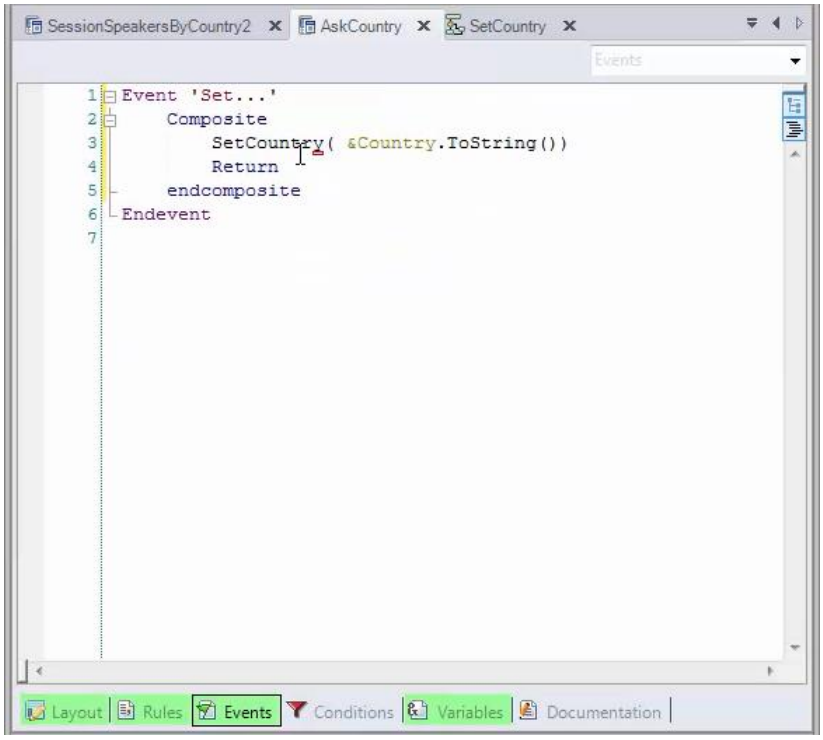
Porque desde el panel AskCountry



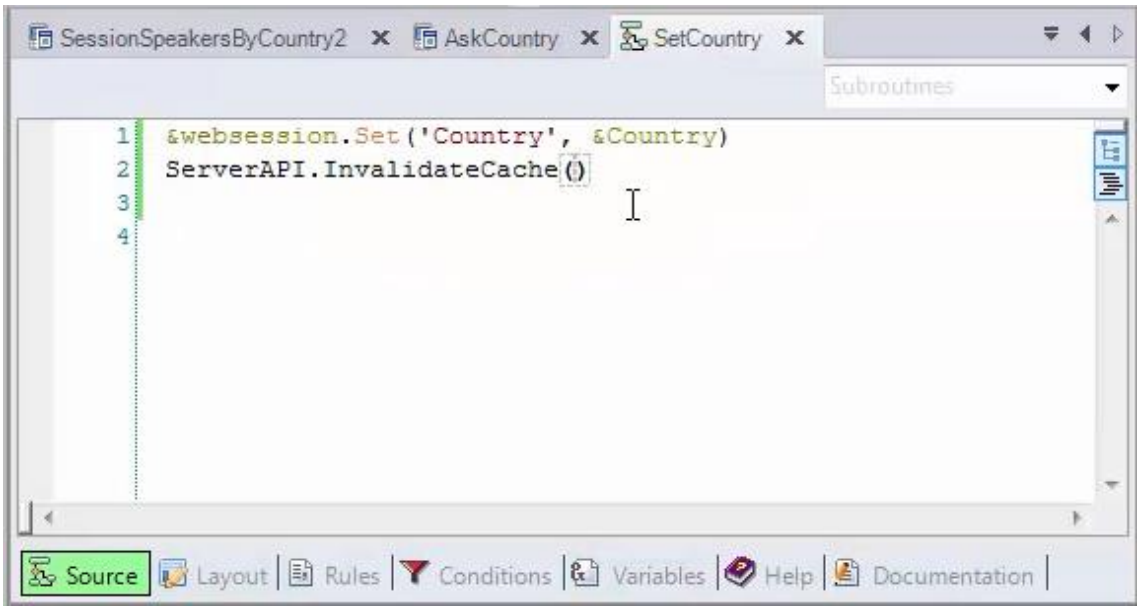
que pide al usuario a través de esta variable Dynamic Combo Box que elija el país,



se está invocando

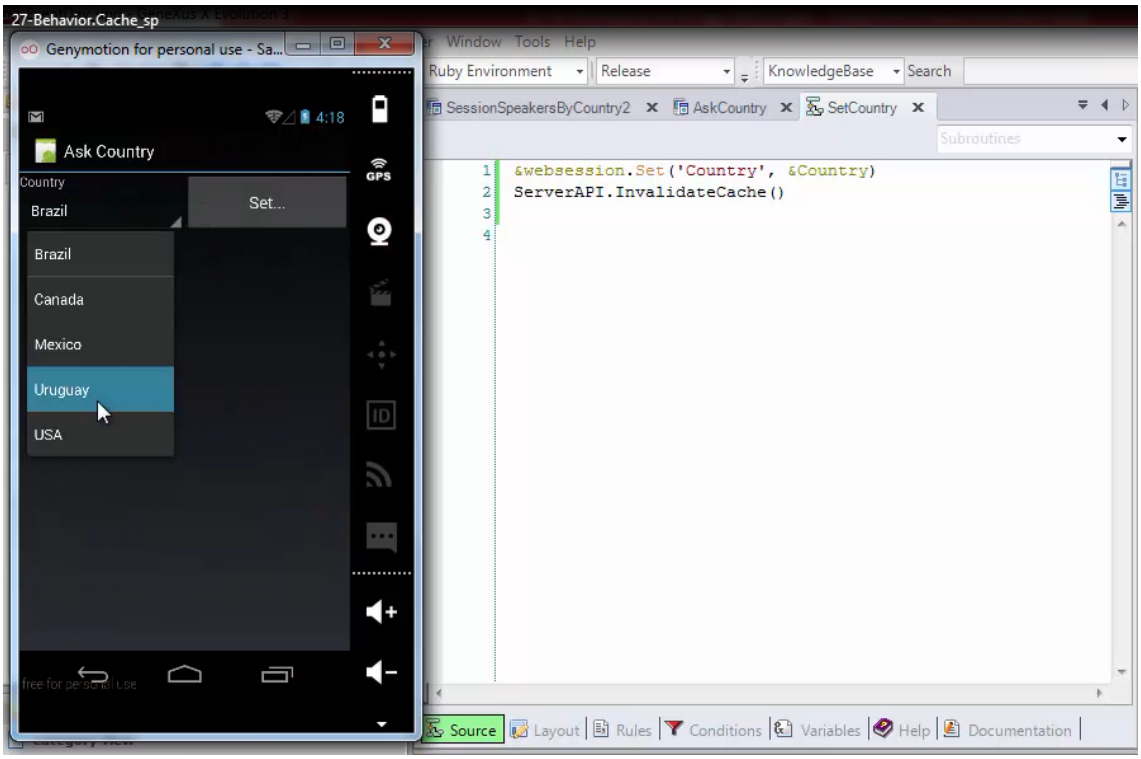


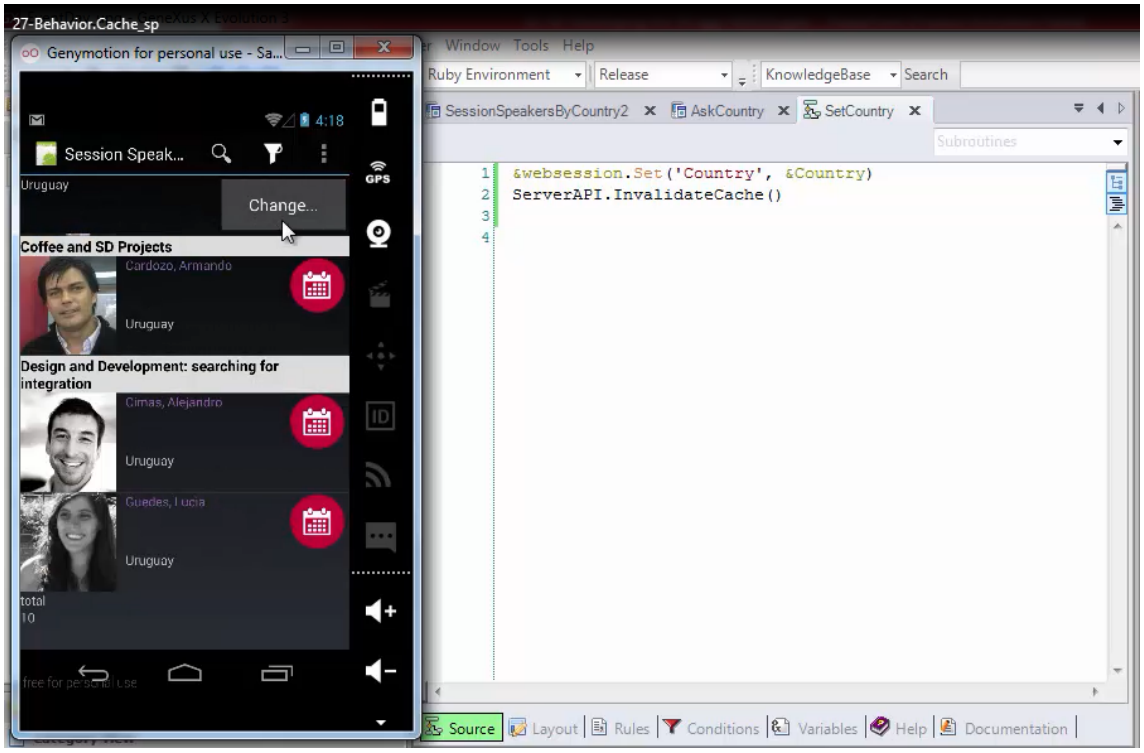
al procedimiento SetCountry, que está enviando ese identificador de país (el elegido)



y este procedimiento ejecutado en el servidor, es el que cambia la variable de sesión con ese país, pero además, observemos que está invalidando el caché., a través de la api ServerAPI (método InvalidateCache).

Es esta invalidación del caché, la que está consiguiendo que al modificar el país

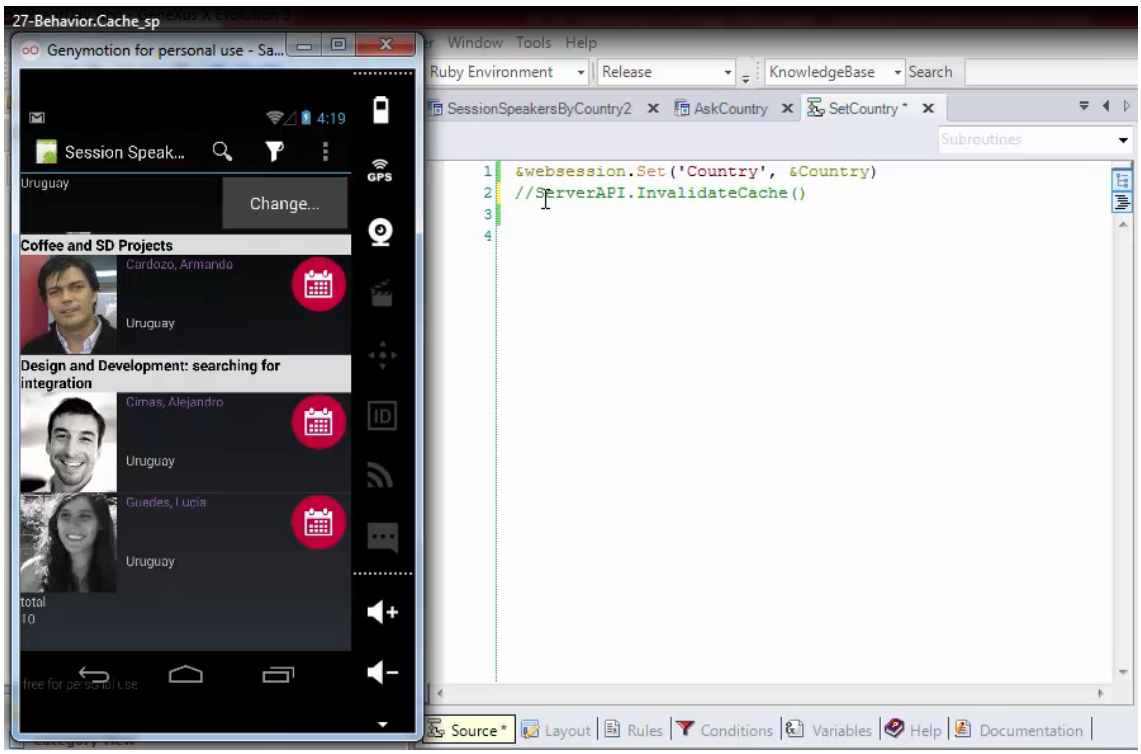


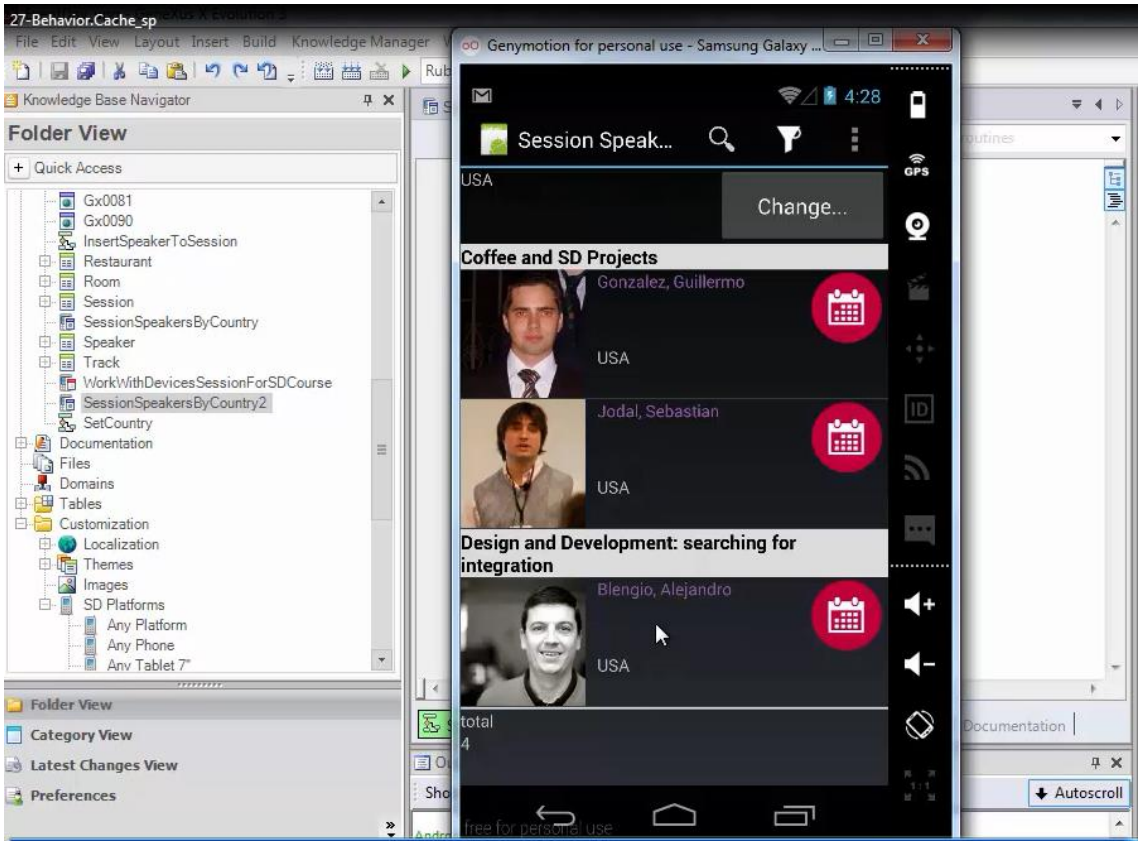


se vuelvan a cargar los data providers correspondientes a la parte fija y al grid.

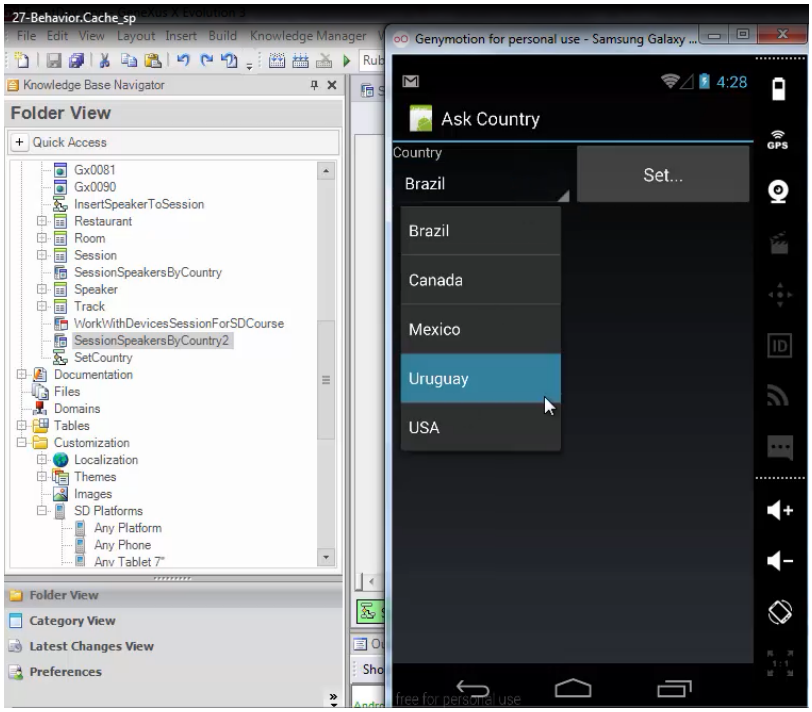
Observemos qué sucede si la comentamos..

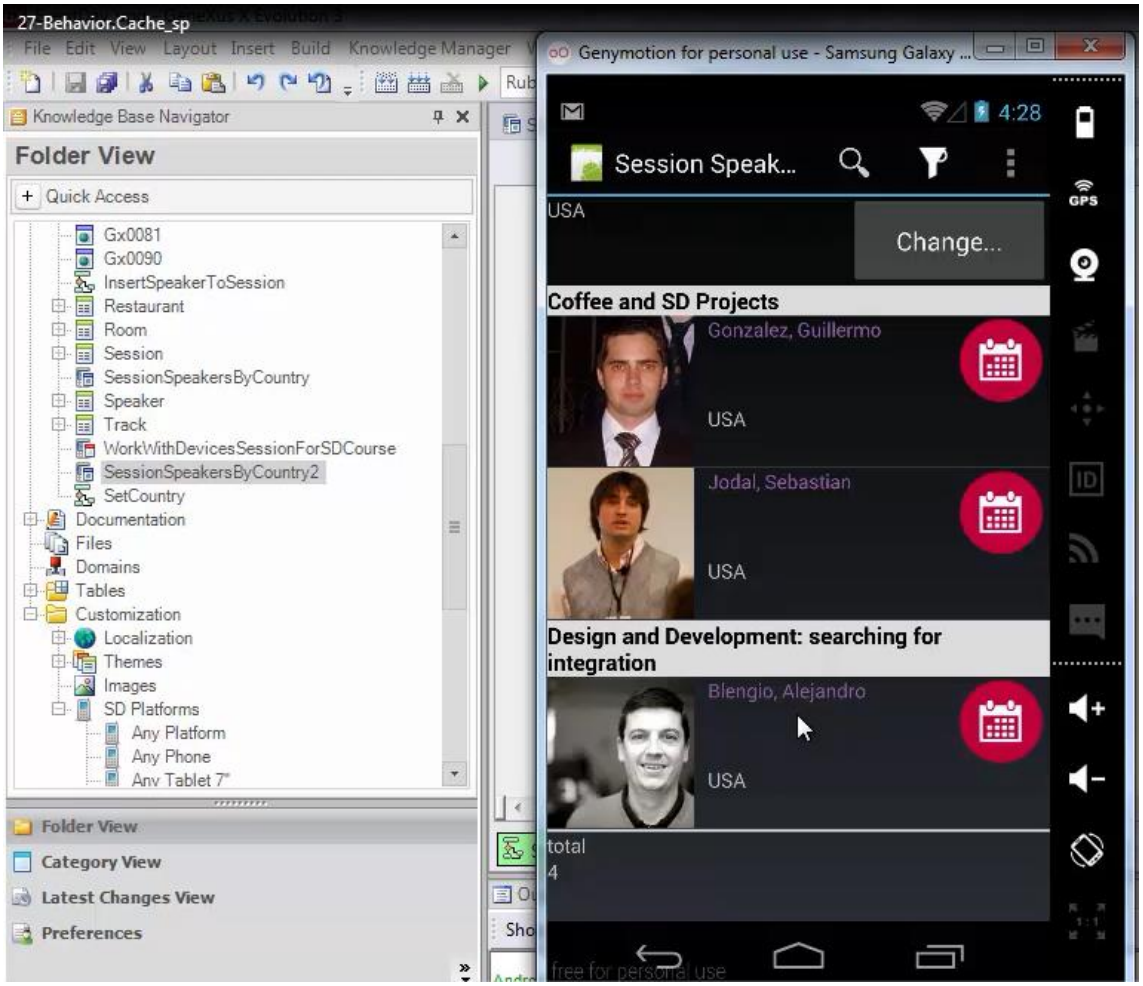
Presionemos F5



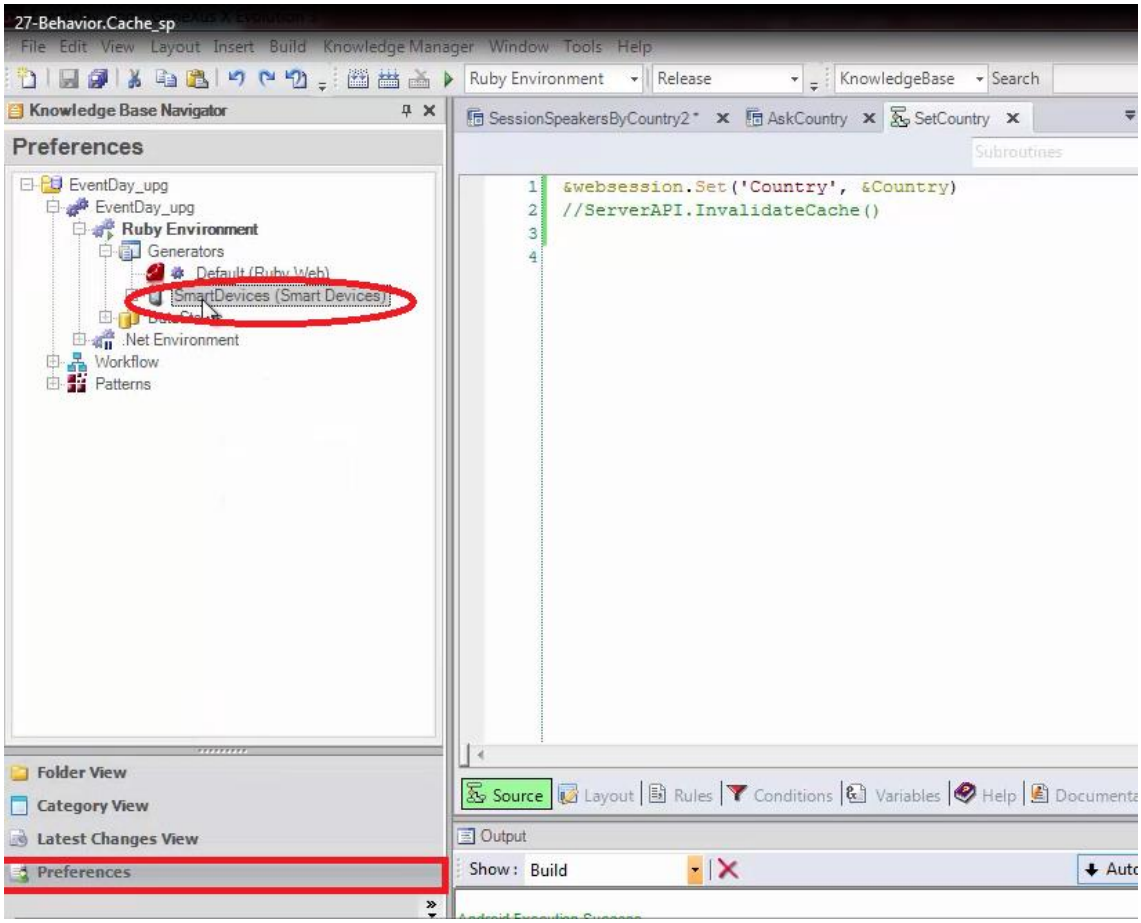


Aquí tenemos Estados Unidos… vamos a cambiar por Uruguay

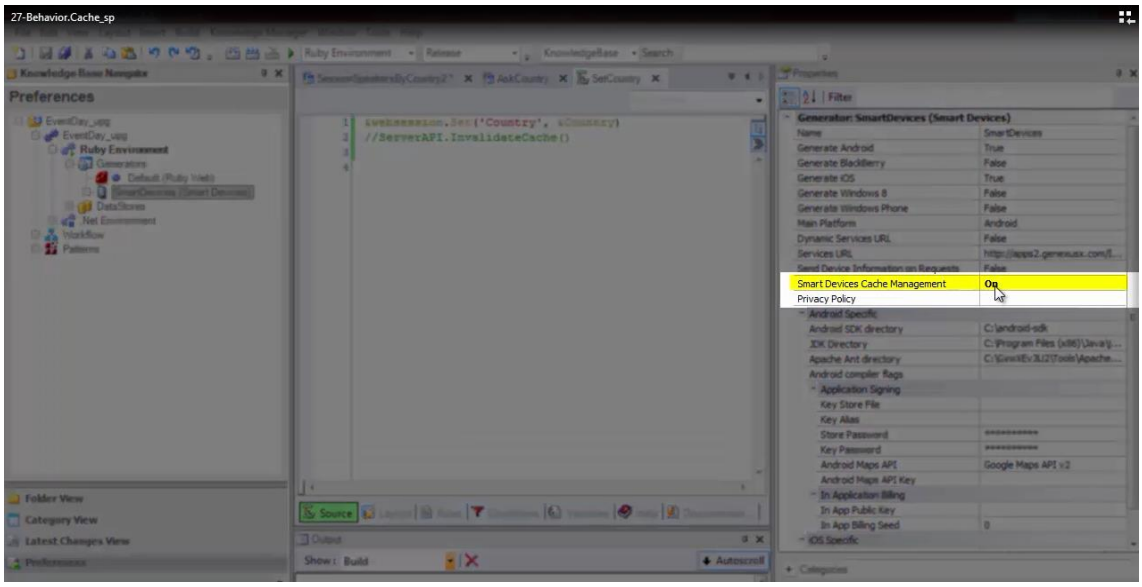




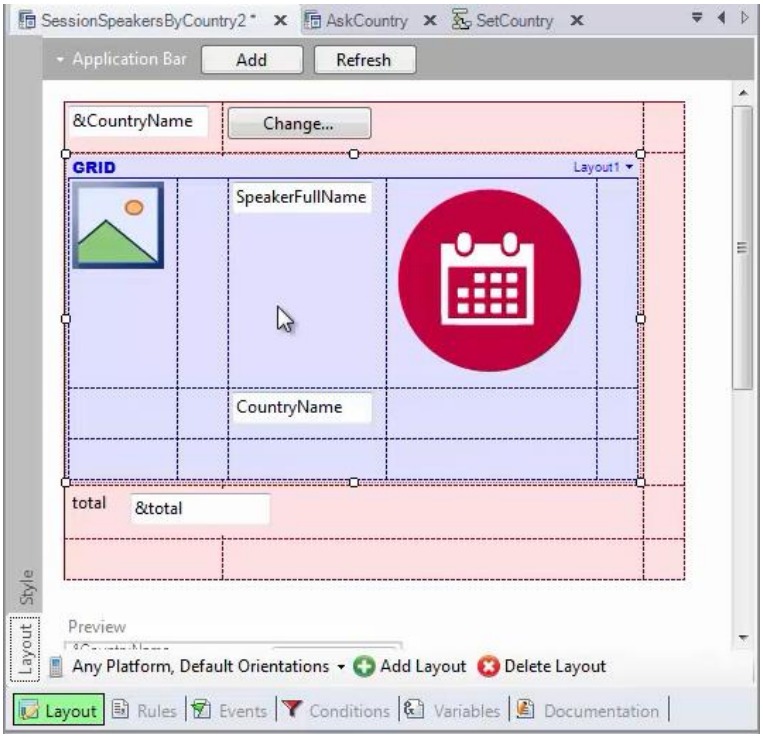
y vemos que efectivamente no ha cambiado (no se ha refrescado) la pantalla.



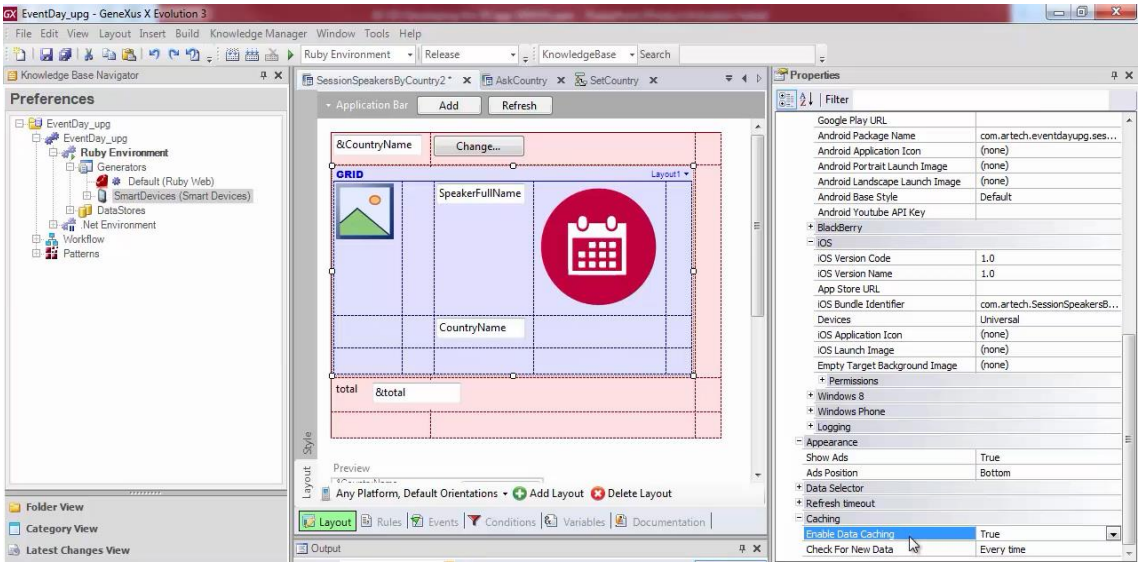
Por defecto, toda aplicación para Smart Devices, tendrá el caché habilitado



Esto significará que ante toda consulta, por ejemplo la de este panel:



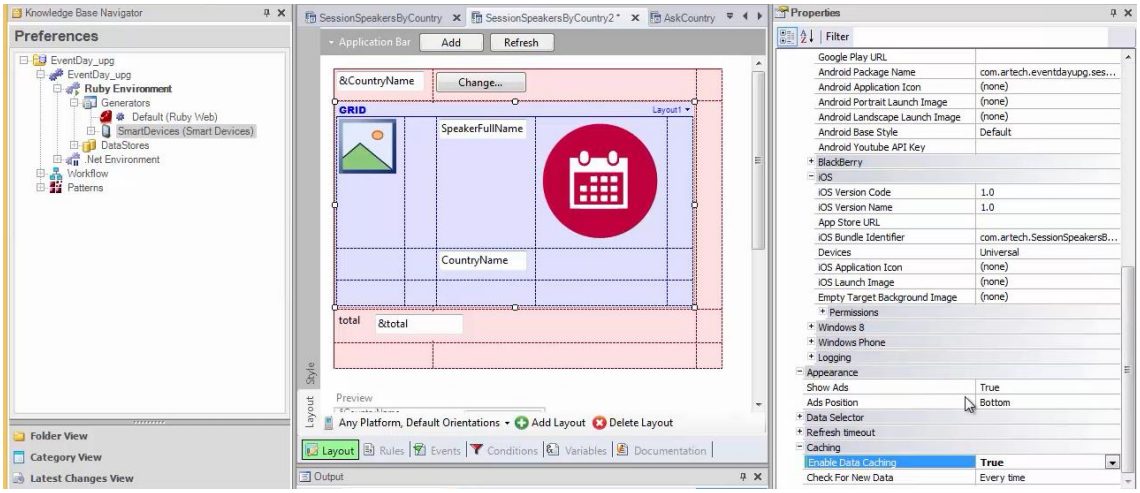
Si no se ha cambiado su valor por defecto para la propiedad **Enable Data Caching**



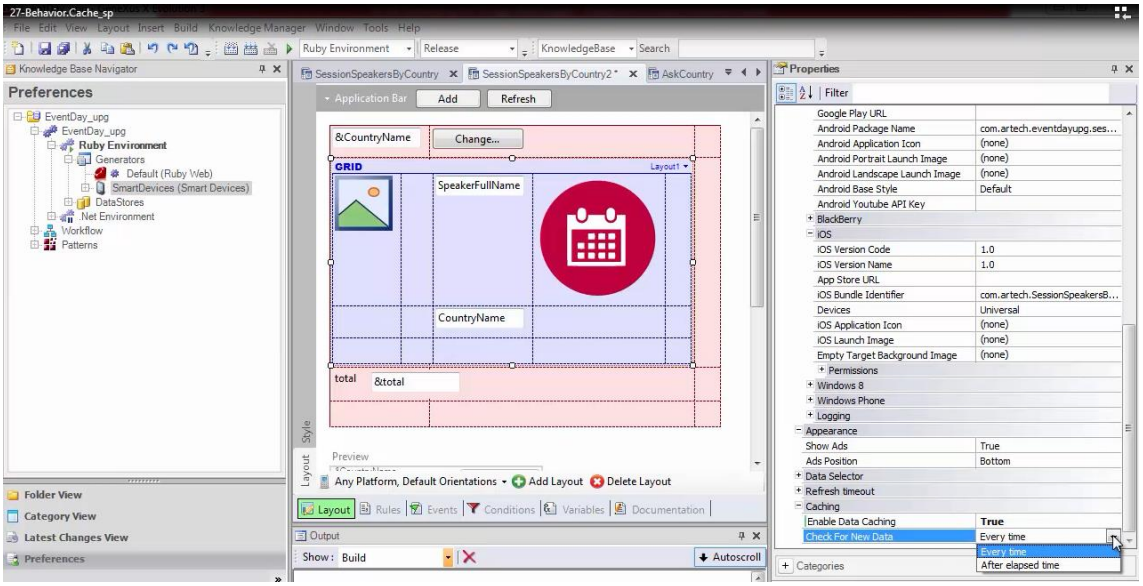
Cada vez que se realice un Refresh, y debido al valor Every time de esta propiedad, el servidor, en base a los valores de última actualización, devolverá los datos ejecutando los data providers correspondientes a la parte fija y al grid, o devolverá un mensaje http 304 indicando que los datos no han cambiado, de manera que el panel trabajará con los datos cacheados.

Si modificamos el valor de esta propiedad pasándola a False, siempre que se haga un Refresh, se traerán nuevamente los datos del servidor, independientemente de la información cacheada.

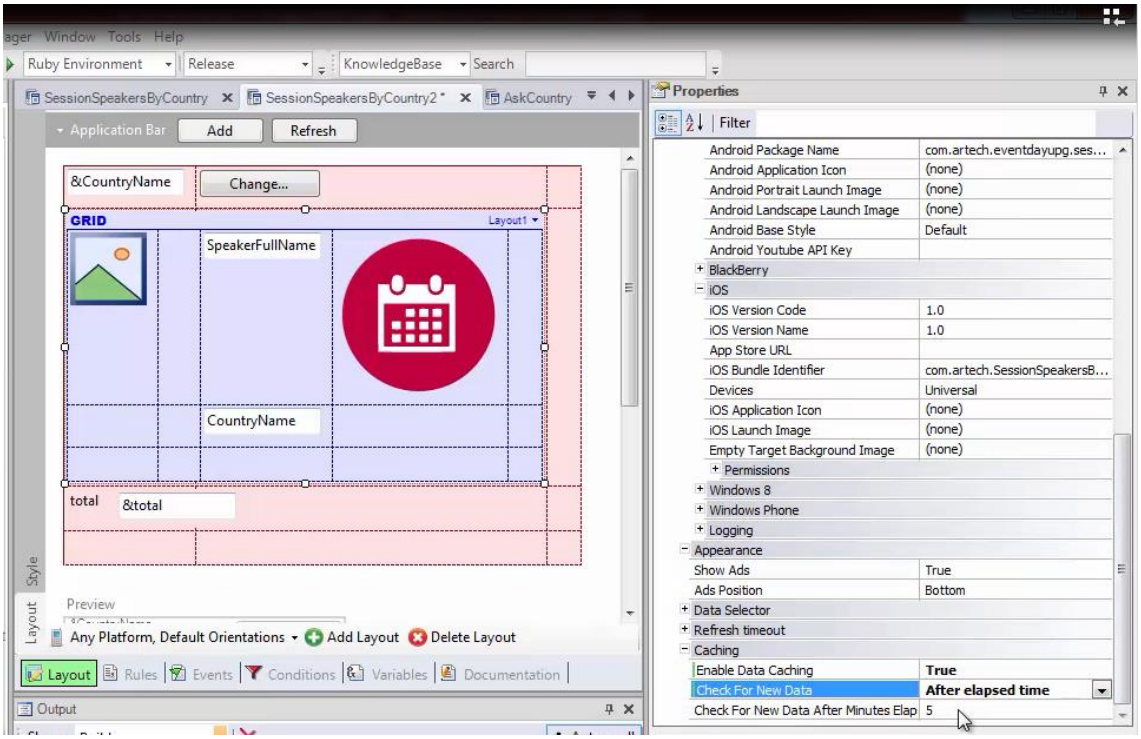
Por otro lado, si dejamos en True la propiedad:



podemos modificar también la propiedad que indica cada cuánto queremos realizar el chequeo de nueva información:



Si cada vez que se hace la consulta… o: Luego de un determinado tiempo

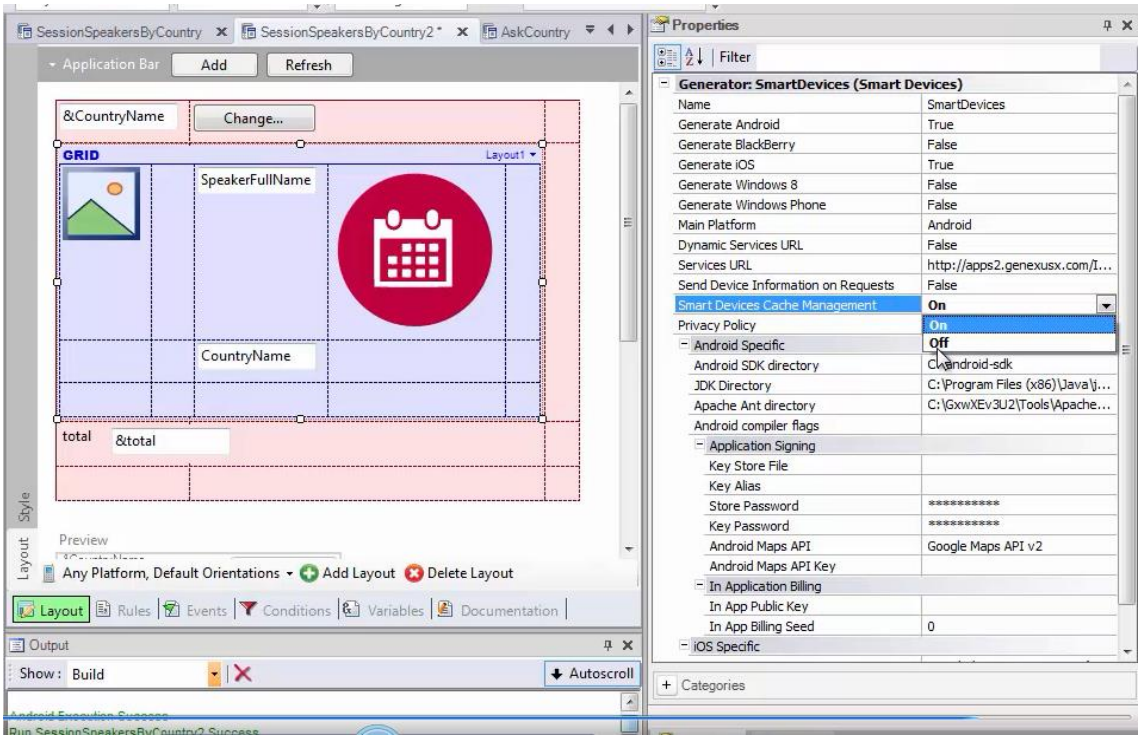


En este caso: 5 minutos.

Aquí lo que estamos indicando es que si vamos a Realizar un Refresh y el anterior se realizó hace menos de 5 minutos, entonces se va a trabajar con la información cacheada. En caso contrario, es decir si pasaron más de 5 minutos, se va a chequear en el servidor si se tienen que mandar los nuevos datos o si estos no se modificaron.

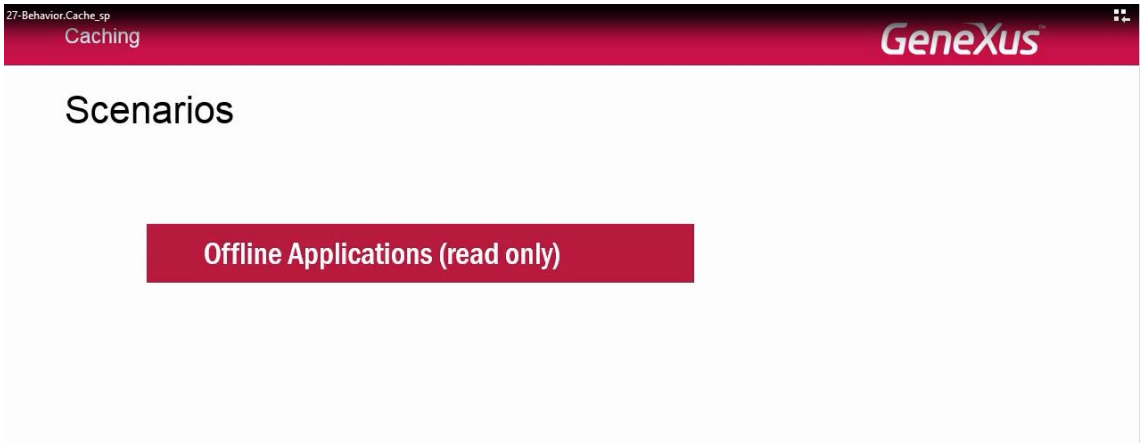
Esto será útil en aquellos casos en los que la información consultada, presumiblemente no va a cambiar en ese espacio de tiempo. Y de esta manera, evitamos tráfico innecesario entre el cliente y el servidor.

Para finalizar, modificar el valor de la propiedad a nivel del generador de Smart Devices, pasándola a Off:

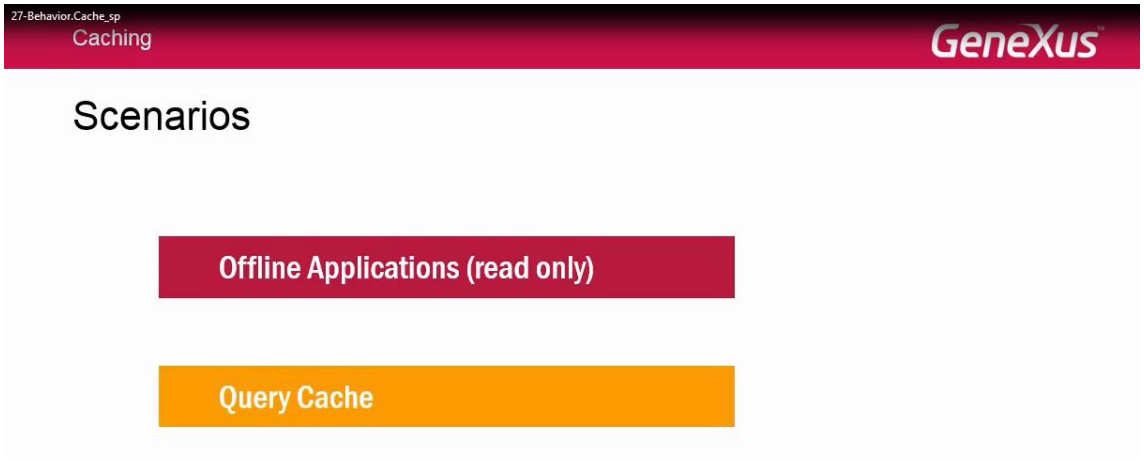


solamente tendrá consecuencias, cuando la aplicación tenga conectividad, pues en ese caso siempre se va a hacer la consulta sobre la base de datos centralizada.

Pero si la aplicación está desconectada, en aquellas pantallas donde tengamos datos cacheados, se va a trabajar con esos datos.

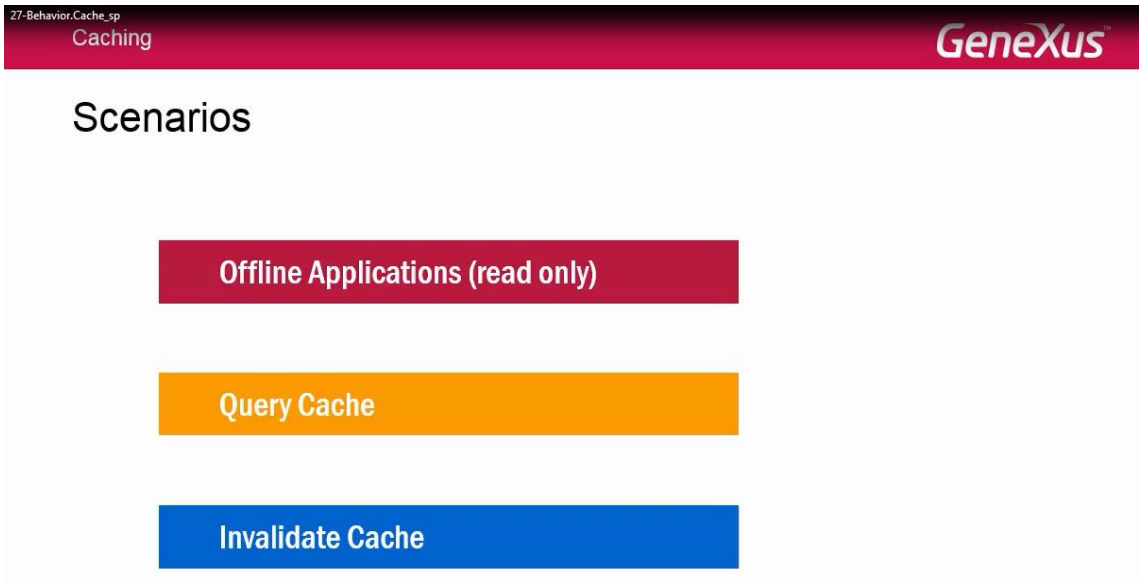


En resumen, hemos estudiado cómo las aplicaciones de arquitectura online, manejan caché para leer datos previamente consultados cuando están desconectadas,



Y para evitar intercambios de datos inecesarios con el server cuando están conectadas.

Para aquellos casos en los que se necesita que aunque los datos del server no hayan cambiado, se vuelvan a ejecutar los eventos Refresh y Load,



se ofrece una api para invalidar el caché.

Las aplicaciones de arquitectura offline, como se verá, trabajan siempre sobre una base de datos local en el dispositivo… y los intercambios de información con el server, se realizan a través de programas de sincronización.


27-Behavior.Cache_sp
LIVE
GeneXus™
TRAINING PROGRAM