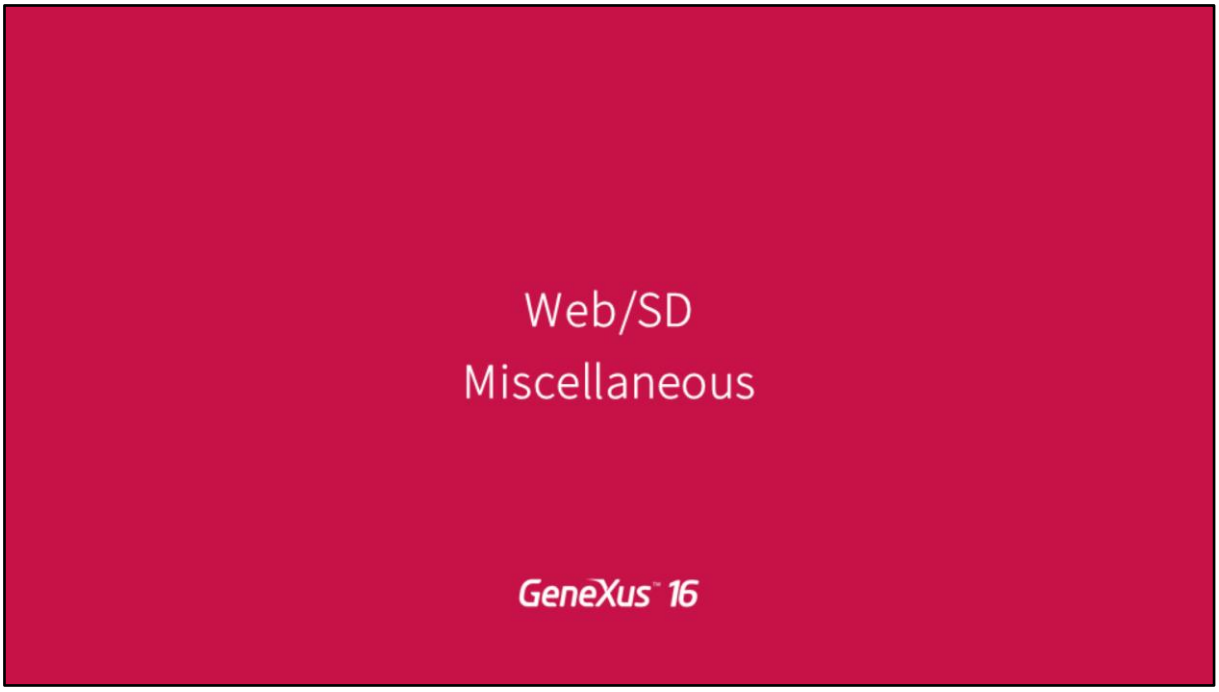
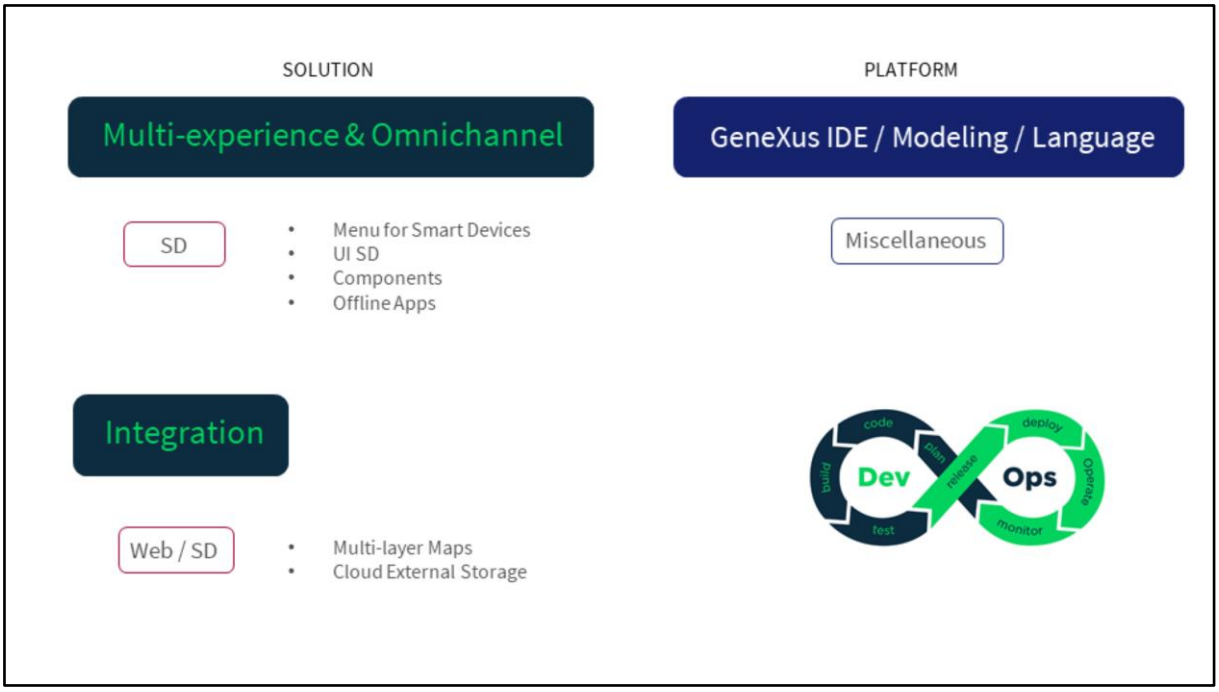
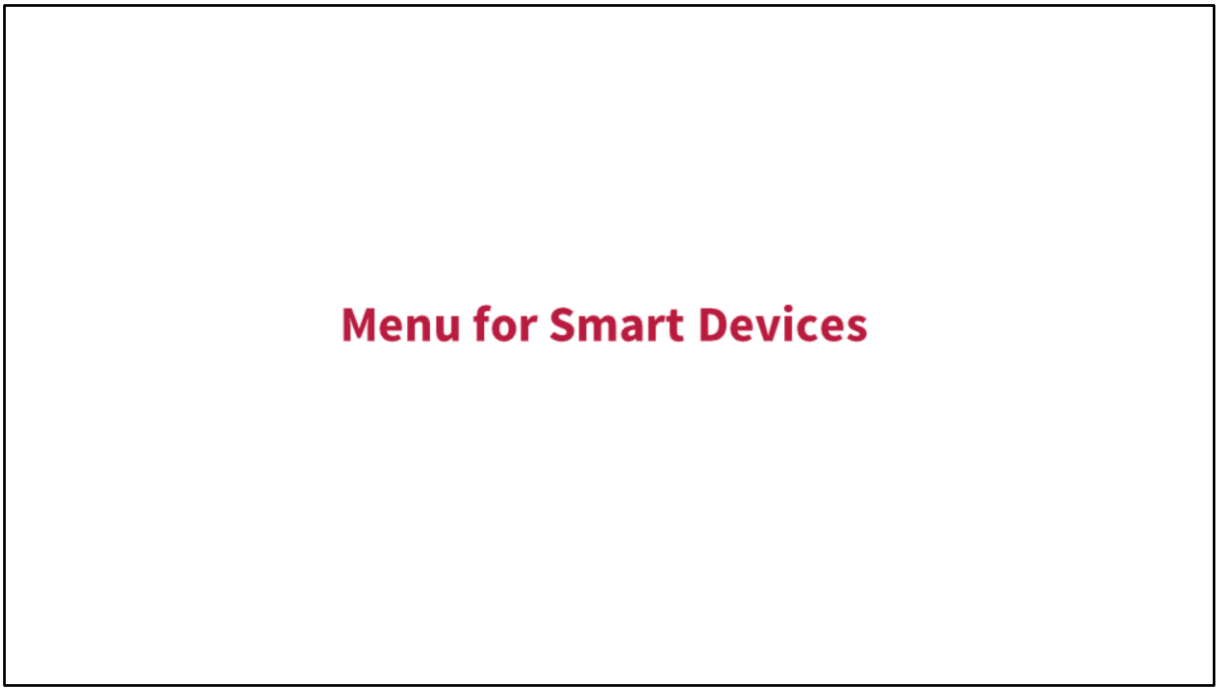

GeneXus™
The power of doing



Hola a todos, mi nombre es Pablo Mazzilli, ahora vamos a continuar, yo voy a profundizar con Smart Devices, en realidad cosas de UI.
En este flujo que estábamos marcando para el curso, voy a hablar de algunos otros recursos que tenemos para IU, son cosas simples pero interesantes que incorporamos en la versión 16 y en los últimos Upgrade de la versión 15.

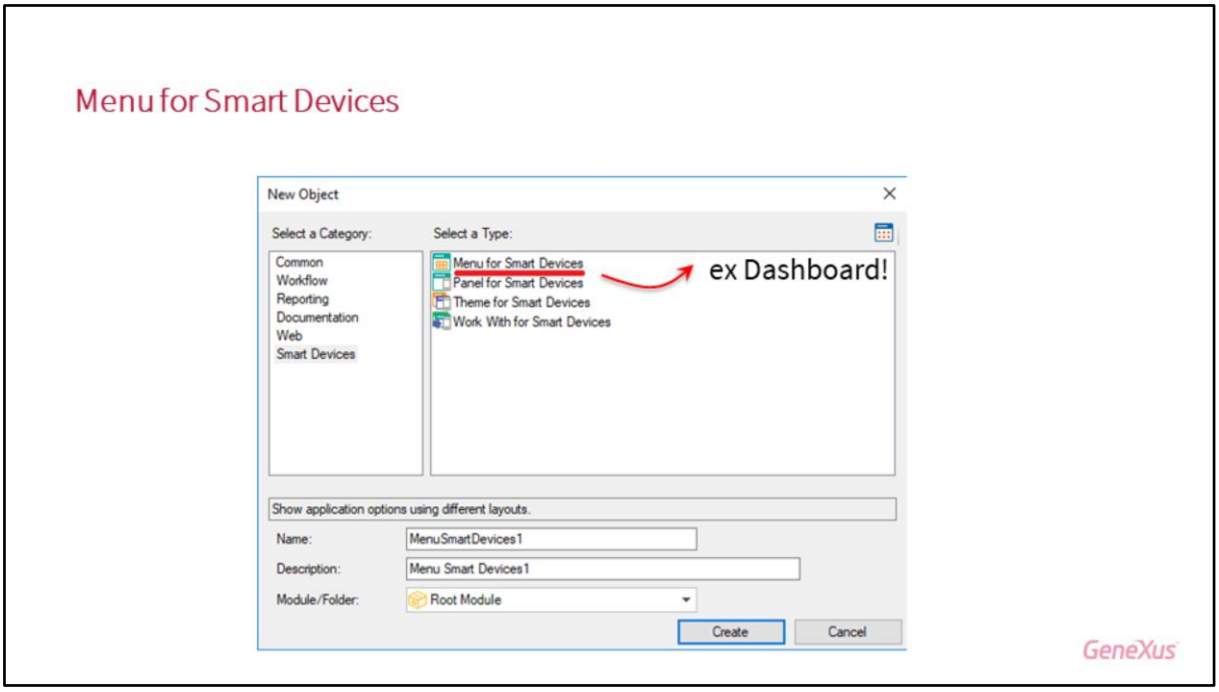


Después sí, voy a profundizar un poquito más comentándoles las novedades y haciendo un repaso un poco más general del tema y después Javier sí va a continuar con la parte de mapas, el External storage y algunos otros recursos aspectos del IDE.



Bien, vamos entonces a repasar con la parte de Smart Devices.

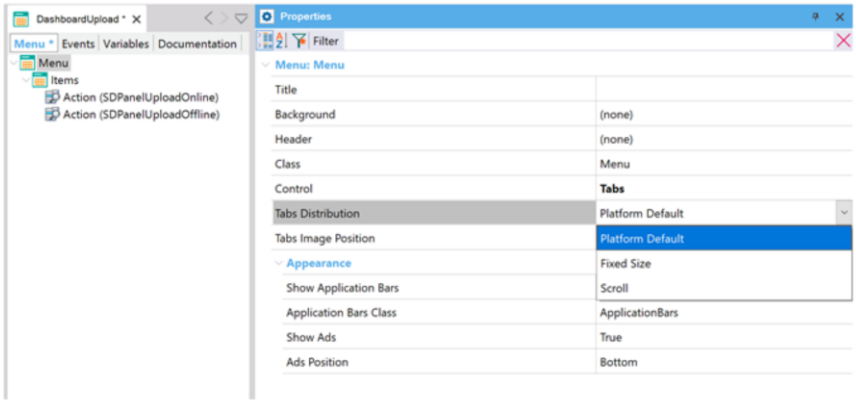
Habrán notado los que ya se han actualizado a los últimos Upgrade de la versión 15 o los que van a pasar ahora a la versión 16 que cambiamos el nombre del objeto, lo que se llamaba “Dashboard” ahora se llama “Menu for Smart Devices”.



Van a notar esa diferencia, el cambio es un cambio simple de nombre para no confundir con el otro objeto de Dashboard que está incluido para toda la parte de query y de definir Dashboard asociados a consultas en la parte Web.

Menu for Smart Devices / Tabs Control

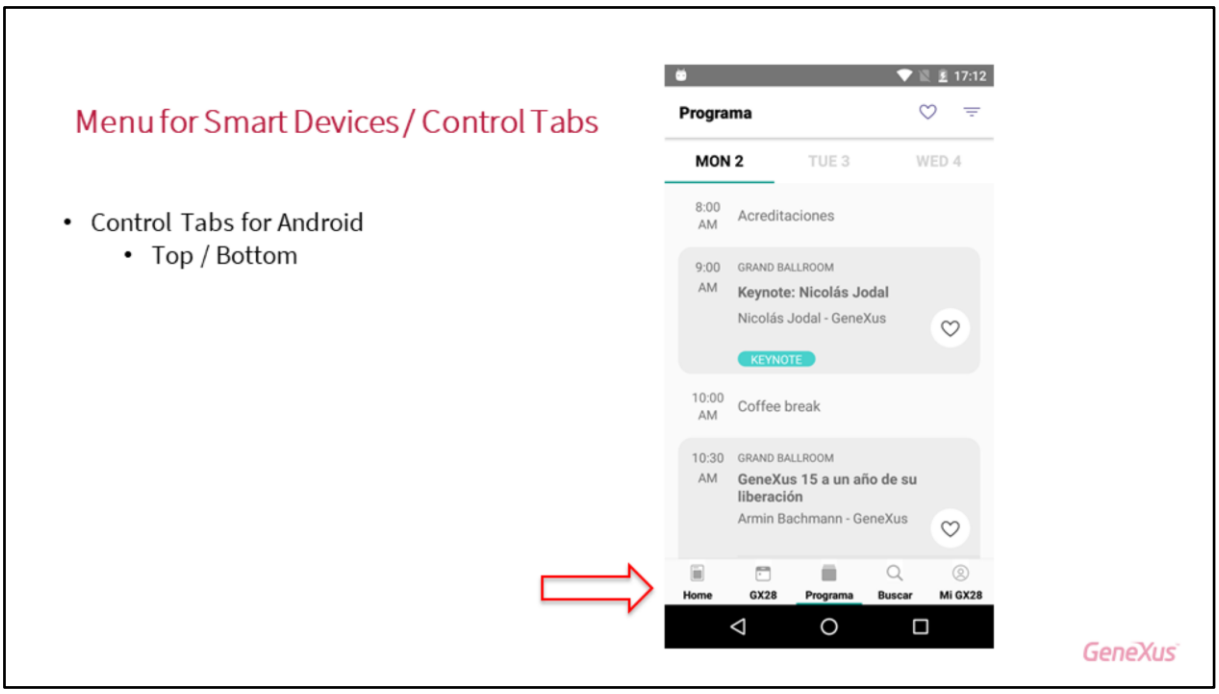
- Tabs Distributions property



GeneXus

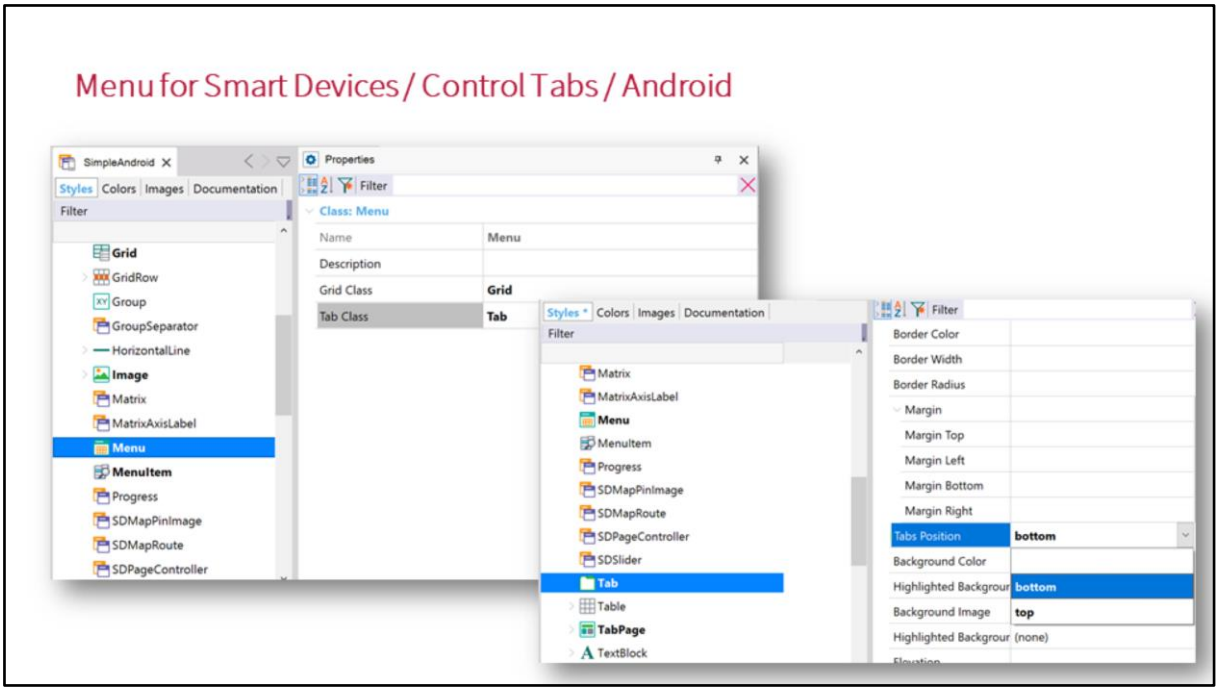
Hemos agregado en ese objeto, (el objeto menu) algunas propiedades nuevas, en particular quiero hablar de esta propiedad, la “Tabs Distributions” que es una propiedad que nos permite que cuando estamos trabajando con un objeto menu, donde los objetos se van a visualizar como “Tabs” podeos indicarle como va a ser la distribución de los items del Tabs, hay dos opciones y ahí incluso basado en cada una de las plataformas, hay un comportamiento predeterminado, para iOS el comportamiento predeterminado es “fixed-side” que esto quiere decir que se distribuyen uniformemente cada uno de los items en la barra inferior.

Y para el caso de Android, depende un poco de la cantidad, si la cantidad de items supera los 5 elementos, automáticamente se usa el Scroll y sino, si son menos de 5, se distribuyen de forma uniforme que es el “fixed-side”.



Otra cosa relacionada con el tema del menu Tabs: son cosas que le están dando más potencia al objeto menú, que en muchos casos dejaban de usarlo por algunas limitaciones, entonces ahora quisimos darle un poco de potencia.

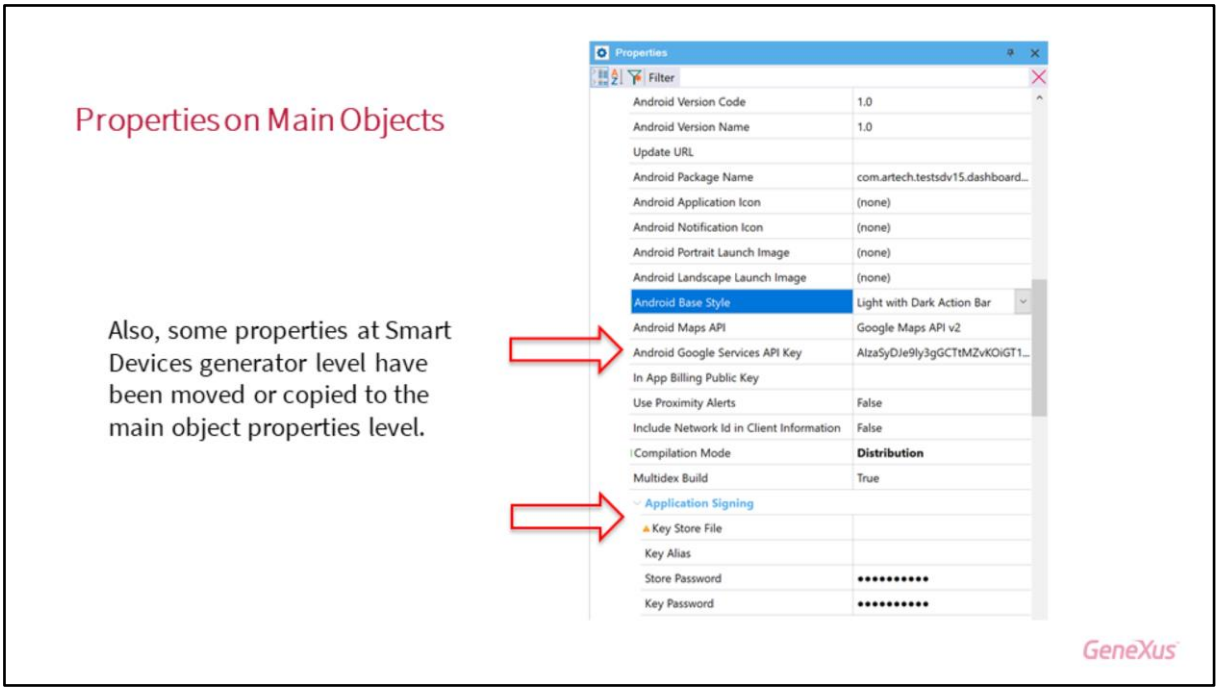
Es el soporte en Android del menú como Tabs inferior, o sea, podemos mostrar un Tab en la parte inferior de la pantalla.
Antes, el comportamiento predeterminado que se ajustaba a las reglas de diseño de Android era mostrar siempre la parte superior, para iOS se mostraba siempre en la parte inferior.
Android ahora tiene, (con el surgimiento del material design), un concepto nuevo que es el navigation button y bueno, con eso nosotros lo que estamos haciendo es guardando la posibilidad de que cuando tenemos un menú como Tabs, poder mostrarlo tanto arriba como abajo.



¿Cómo configuramos eso? (o sea, la posición del Tabs en Android)

A través del tema, vamos a ver que en el objeto menú, tiene una clase menú (ya estaba pre-configurada), entonces vamos a ir al objeto tema> vamos a ir a la clase menú> en la clase menú le decimos cuál es la clase asociado al Tab class, o sea, cuando el menú se muestra como Tab ¿cuál es la clase asociada? El Tab> vamos a la clase Tab y ahí le podemos decir la posición del Tab, si lo queremos en la parte inferior o superior.

Esa es la forma entonces que en Android les decimos dónde mostrar el Tab.



Con respecto a los objetos Main en general, no solo al objeto Menú, sino a cualquier objeto que podamos definir como Main, van a ver que aparecen nuevas propiedades y algunas cambiaron sus valores, por ejemplo:

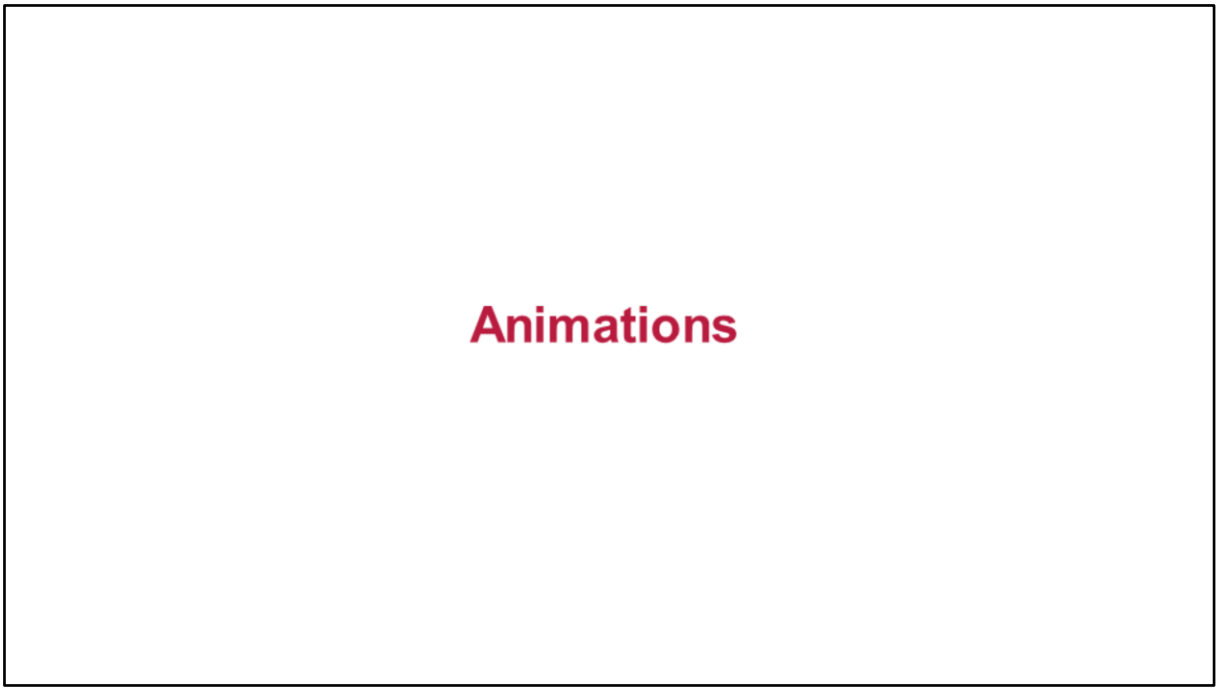
“Android Base Style”, que antes el valor pre-determinado era “Dark”, ahora cambió a “Light with Dark Action Bar”, que lo que permite es ajustarse al nuevo diseño del tema Main.

Por otro lado, van a ver que aparecen estas propiedades nuevas, algunas estaban en otras partes, por ejemplo lo que es la configuración del Android Maps API , las claves para cuando utilizamos los mapas antes estaban a nivel de generador, se configuraban en el generador Android, aparecían esas propiedades.

Lo mismo que las credenciales para firmar la aplicación, que eran todas propiedades que estaban a nivel del generador, o sea, eran genéricas para ese environment, ahora las podemos configurar a nivel de menú, eso nos da la posibilidad de tener diferentes Main en ese mismo environment para Android, con diferentes propiedades, era un requerimiento que había surgido de algunos escenarios.

En particular la “Application Signing”, que era una propiedad que como les decía, estaba configurada a nivel del generador, ahora va a aparecer a nivel de Main y únicamente cuando la Compliation Mode está en modo “distribution”, cuando está en modo “Develompent” (que es la otra opción que tiene esa propiedad) estas propiedades no son necesarias configurarlas, se puede probar una aplicación con mapa, en ese caso se está firmando la aplicación, se están usando credenciales que el propio GeneXus disponibiliza, pero a la hora de poner sus aplicaciones en producción, tienen que necesariamente cambiar la Compliation mode para Distribution, sino en Google no van a poder publicar su aplicación al Google Play y ahí sí les va a pedir que coloquen sus propias credenciales, se registran.
Están todos documentados en el wiki como son los pasos para obtener esa información y ponen las credenciales propias de ustedes, porque antes lo que nos pasaba era que utilizaban credenciales de GeneXus y terminaban publicando con credenciales que eran de GeneXus, que no es lo recomendable.

Bien, eso es lo que respecta al ´Menú y a los Main en general.



Otro recurso interesante que hemos incorporado en los últimos Upgrade de la 15 y en la versión 16, es el concepto de Animations.

Las animaciones son un recurso que realmente nos va a mejorar notablemente lo que es la experiencia usuario y la interfaz de usuario en nuestras aplicaciones.

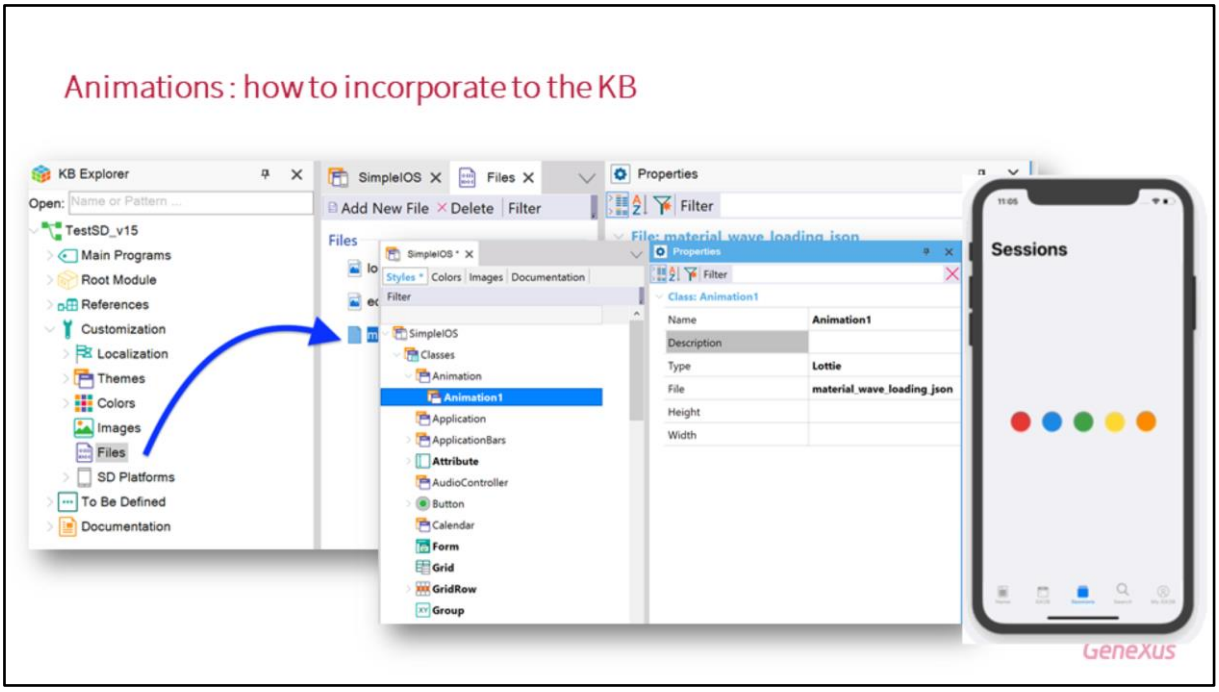
Las animaciones, específicamente las que llaman “Lottie” Animation, es una biblioteca creada por Airbnb y ellos crearon también una API (Json), donde nosotros podemos descargar esos recursos y utilizarlos en nuestras aplicaciones.

Déjenme mostrarle de qué estamos hablando cuando hablamos de Animaciones.

Estas son algunas animaciones:
Lo interesante es que las animaciones Lottie o la mayoría de las herramientas de diseño permiten exportar estos archivos, o sea, permiten creare esas animaciones y exportarlos con ese formato, ese Json que es el formato Lottie.

Pero también tenemos sitios como este: lottifiles.com, donde nosotros podemos descargar animaciones que están ahí, disponibles para ser utilizadas.
Van a ver que hay muchas animaciones, aquí la más populares por ejemplo.
Esos recursos, cuando nosotros queremos por ejemplo mostrar que se confirmó una acción, en lugar de crear una imagen estática o colocar un Gif, (que no es lo recomendable), bajar este tipo de animaciones.

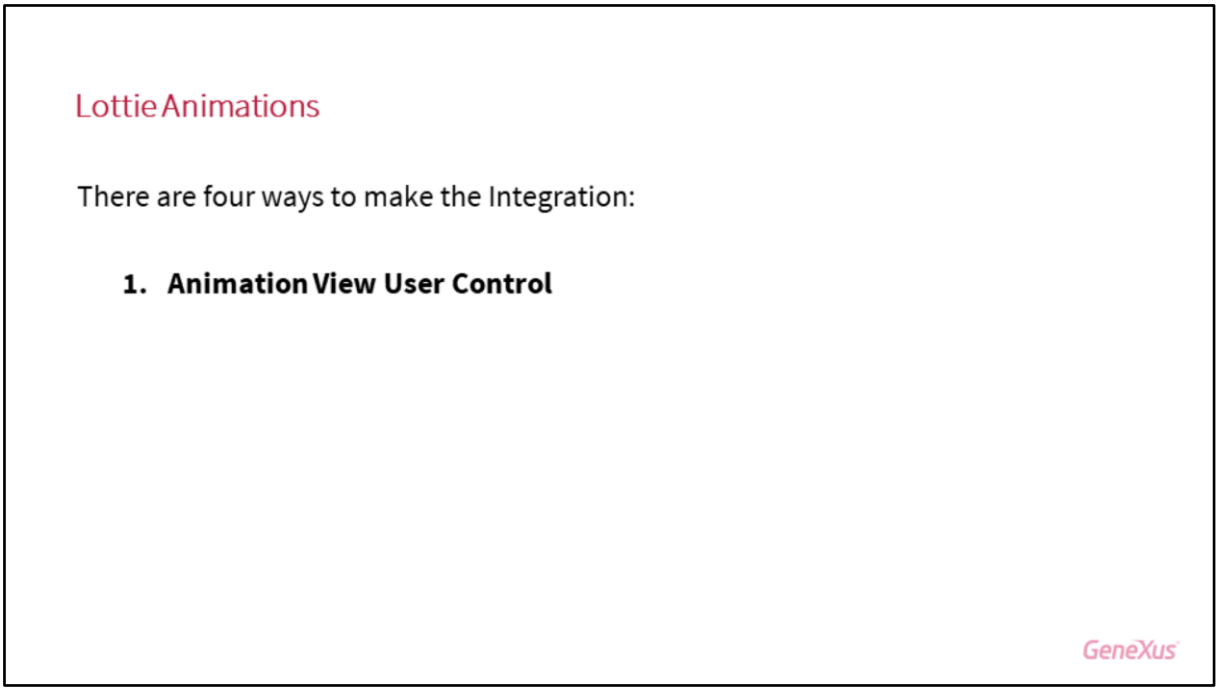
Como les decía, es un archivo Json que los diseñadores saben cómo crearlos, sino pueden descargarlos de acá.
Incluso algunos de este sitio, permiten personalizarlos, si elegimos este de aquí, podemos cambiarle los colores que queremos que utilicen, por ejemplo en nuestro diseño “no quiero este anaranjado, quiero un color más azul” y bueno, va cambiando ese diseño y luego sí, con este botoncito que aparece ahí abajo podemos descargar el archivo y lo vamos a incorporar a nuestra KB.



¿Cómo incorporamos esos archivos?

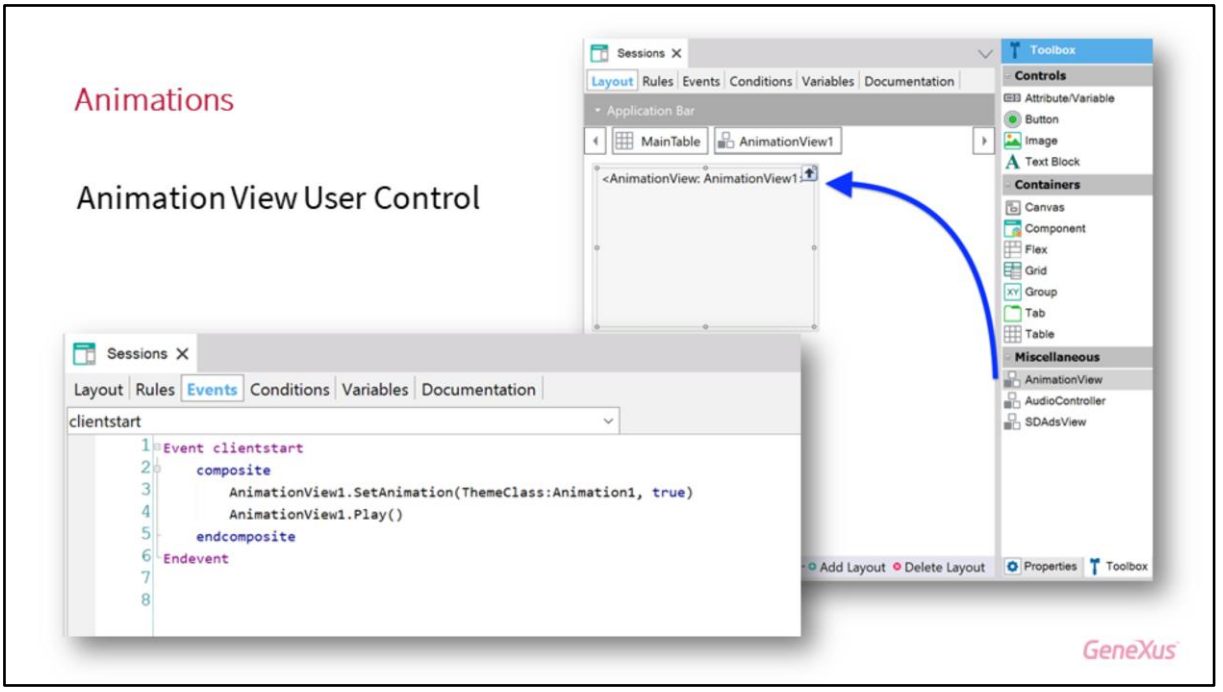
Una vez que tenemos el Json que nos pasó el diseñador o lo descargamos nosotros a través de uno de estos sitios que nos brindan los recursos, vamos al “files” en la KB (tenemos ahí la opción de los archivos que están en la KB) > importamos ese Json (ahí queda disponible el archivo) > y después tenemos que ir al Objeto Tema, vamos a ir a crear una Sub-Class dentro de la clase Animation, donde vamos a decir que esa clase Animation es de tipo Lottie y al archivo le vamos a indicar el archivo que descargamos previamente, que está incorporado en la KB.

Y una vez que tengamos eso, entonces podemos usar esa imagen en nuestras aplicaciones.



¿Cómo la podemos usar?

Hoy en día podemos usarlo de esta forma:
Una es a través del Animation View User Control (es un User control que aparece en la paleta de controles, donde nosotros podemos arrastrar ese control en un SD Panel)

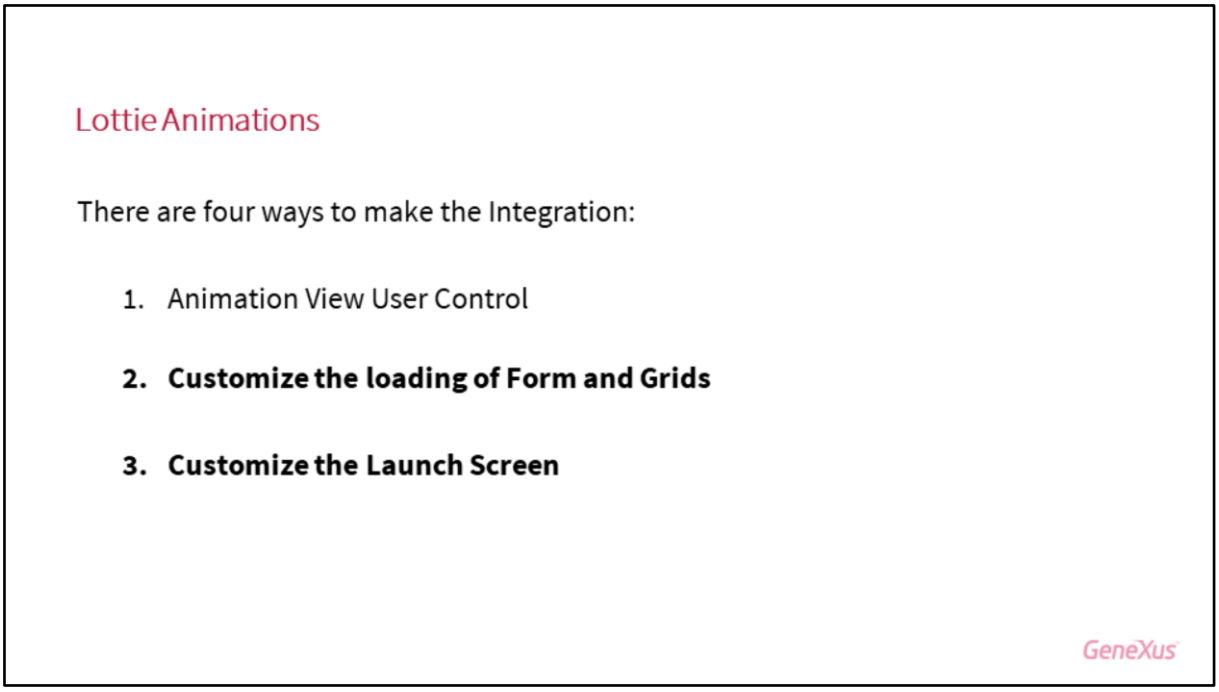


A través de las propiedades de ese control, como vemos acá es un Animation View, lo colocamos en nuestro panel y después tenemos algunas propiedades asociadas a ese control, como es por ejemplo el "SetAnimation" para indicar cuál es la animación que vamos a utilizar, acá le estamos indicando que es la ThemeClass Animation 1, que es la que había creado previamente y el play es para que comience a ejecutar esa animación.

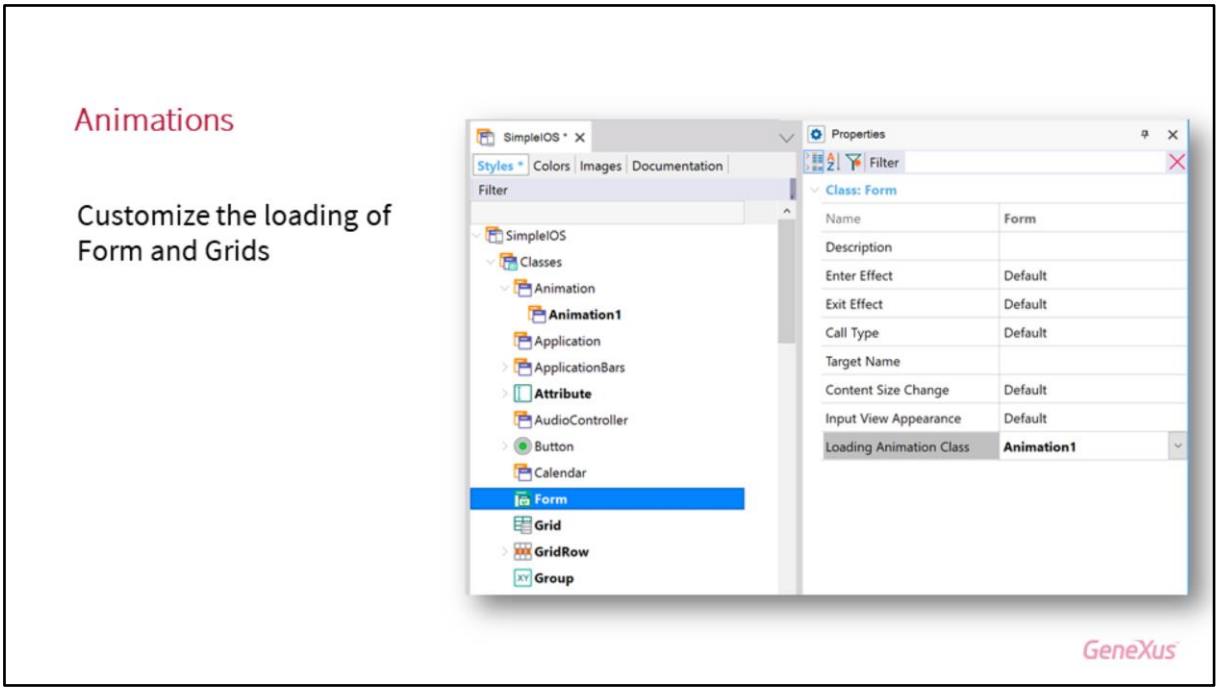
Existen otros métodos que van a estar asociados al control Animation View, como son por ejemplo:
El pausar la animación
O inclusive el Play tiene un par de parámetros adicionales que son opcionales, para indicar desde dónde queremos que comience la animación y cuanto tiempo queremos que dure, porque existen animaciones que muestran el "Play" y el "Pausa" por ejemplo y queremos que mientras estamos reproduciendo un audio, que se muestre el "Play" y que cuando comience a reproducirse que se muestre la imagen de "pausa" pero cuando le doy pausar, que se muestre el "play" para volver a reproducirlo… lo podemos todo eso, con un mismo archivo, gobernar en que posición queremos que comience y finalice, basado en esos métodos adicionales.

Van a ver que está disponible ahí cuando utilicen el método, podemos también decirles si queremos que quede en loop siempre ejecutando o que ejecute una vez y que termine. Imagínense que es como un Gif que tenemos más propiedades y una potencia mucho mayor.

Eso es lo que tiene que ver con la parte de Animation View User Control.



También lo podemos usar para personalizar la carga del Form o de los Grids, vieron que hoy por defecto, para Android se muestra un circulito donde esta cargando un Form o un Grid con muchos datos, hoy tenemos una animación que viene pre-determinada en nuestras aplicaciones, podemos cambiar esa imagen por cualquiera de estos archivos Lottie o también la parte cuando levanta nuestra aplicación, la “launch screen”



Por ejemplo, para Customizar la carga del Form o del Grid, hay propiedades nuevas en la clase asociada al Form, donde le podemos indicar cual es la clase Animation y lo mismo para el Grid.

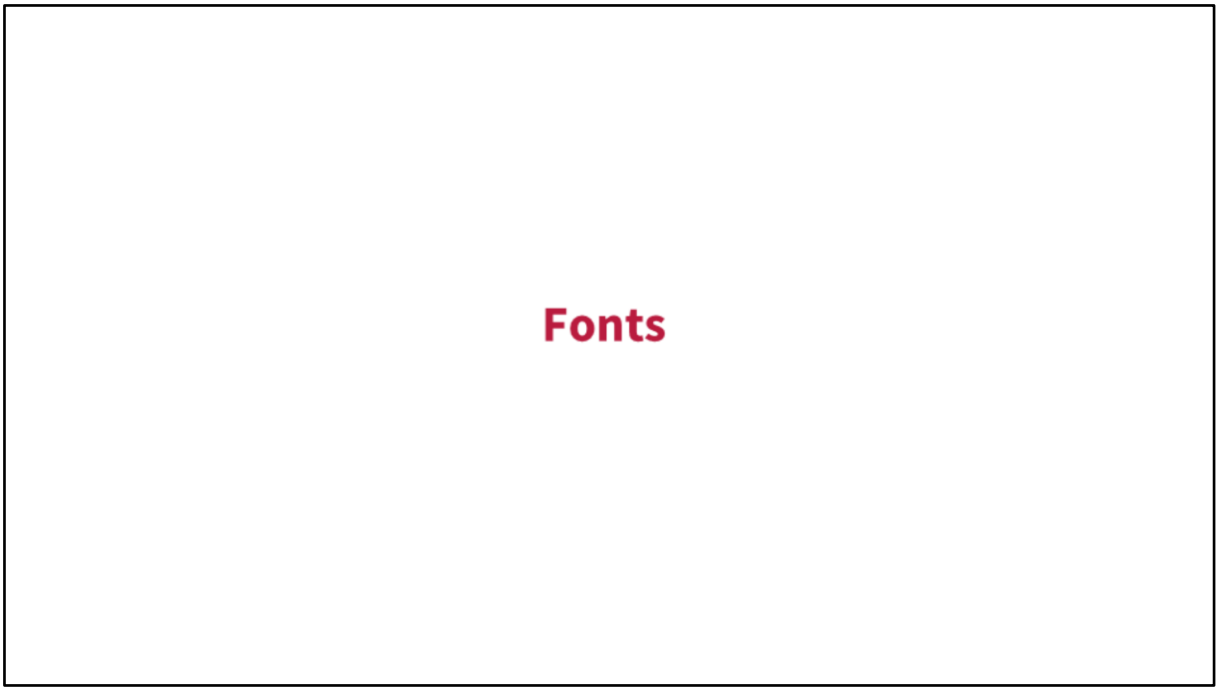
Lottie Animations

There are four ways to make the Integration:

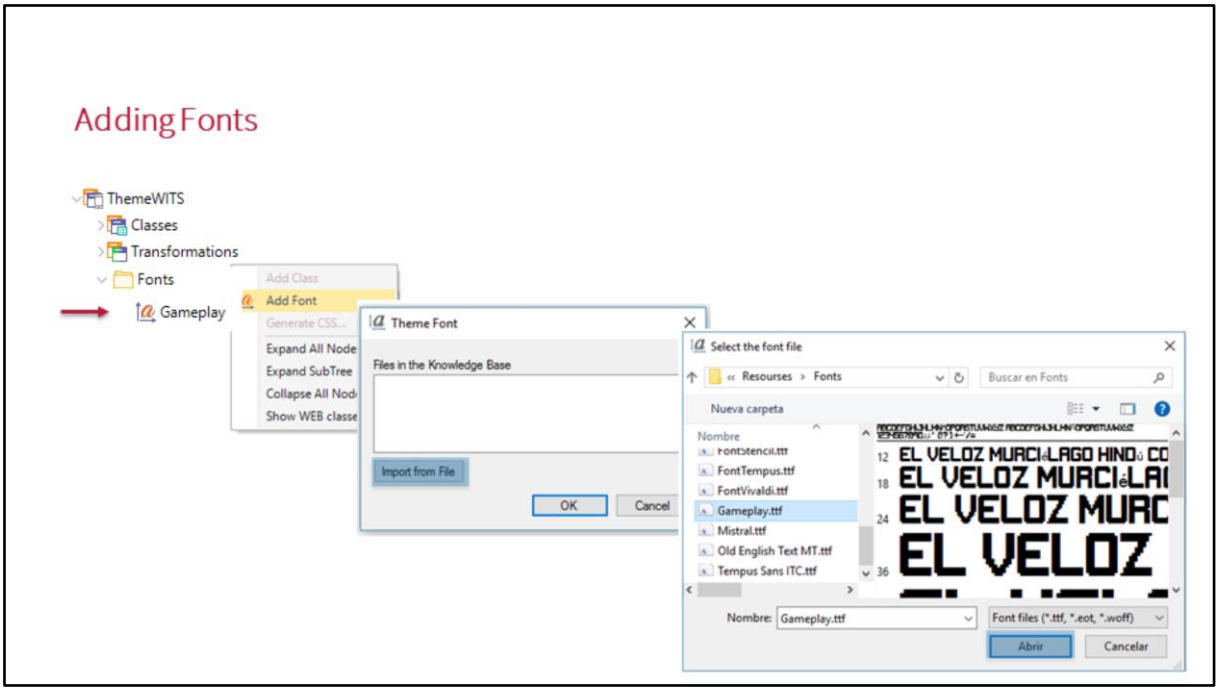
1. Animation View User Control
2. Customize the loading of Form and Grids
3. Customize the Launch Screen
4. **Customize the Progress Indicator control**

GeneXus

Y por último, también el control “Progress Bar Indicator”, que hoy tiene una imagen, una animación pre-determinada, podemos nosotros asociarle una animación diferente a través de estos archivos.
Todo eso a través de propiedades en las clases correspondientes.



Fuentes

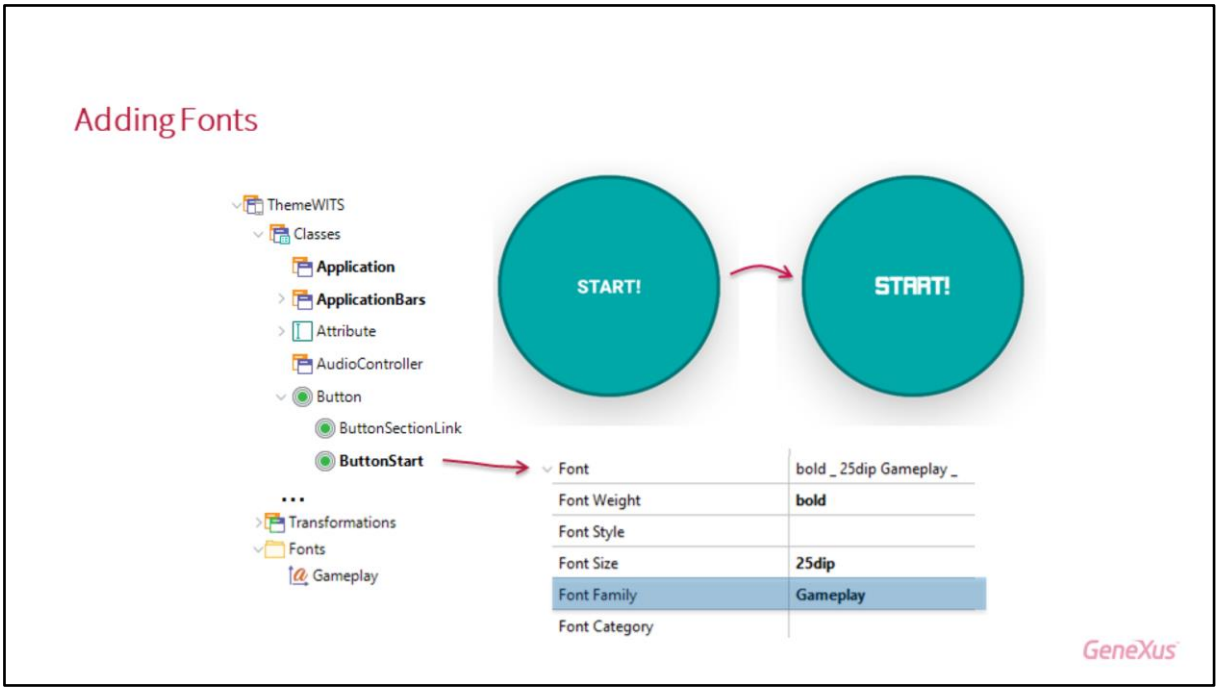


Otra cosa interesante, es poder incorporar fuentes.

La idea con esto de las animaciones y ahora con las Fonts, es: Todos los recursos que ustedes precisen para sus aplicaciones, que estén incorporados dentro de la propia KB.

La Fonts también, antes si queríamos utilizar alguna Fonts que no estaba pre-determinada en nuestros dispositivos teníamos que incorporar esos archivos en los proyectos, referenciarlos de una forma “por fuera”, hacer algunos “truquitos” para que eso funcione, copiarlos ahí en las carpetas de los proyectos, hoy la idea es ponerlo todo dentro de GeneXus.

Si precisamos alguna Fonts adicional, alguna Fonts específica, vamos devuelta al Tema en la opción Fonts que aparece en nuestro objeto Tema, tenemos la opción “Add Font”, allí nos va a pedir para agregar cual es la Font para importarla, vamos a seleccionar, por ejemplo aquí “Gameplay”, la abrimos y entonces queda como una Font dentro de nuestro Objeto Tema…

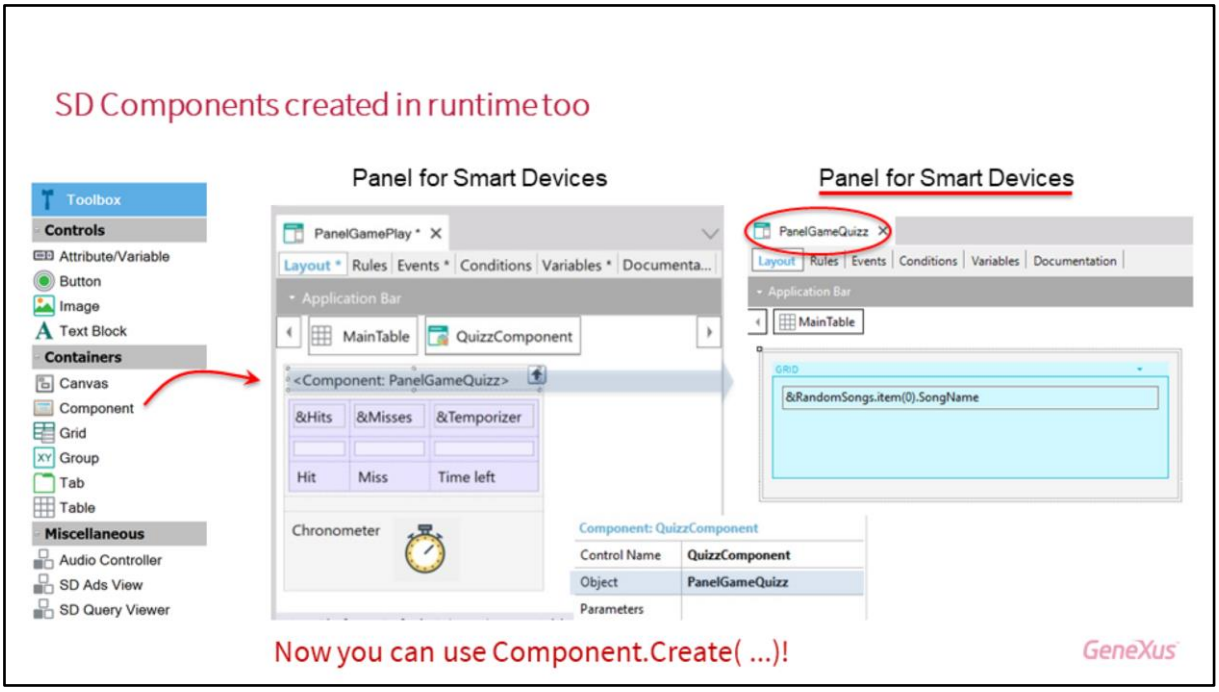


y después por ejemplo en un botón, está la clase asociada a un botón, podemos decirle que la Fonts familiy que va a usar la clase de este botón sea la que incorporamos recién que es el Gameplay.

Entonces queda todo referenciado, queda todo dentro de la KB, eso da facilidades, porque si tenemos que llevar esa KB de GXServer tenemos todos los archivos necesarios como los lottie, las Fonts, está todo completo, no tenemos que estar agregando recursos adicionales por fuera de la KB.

Así es como se vería después de aplicar esta Font.

Components

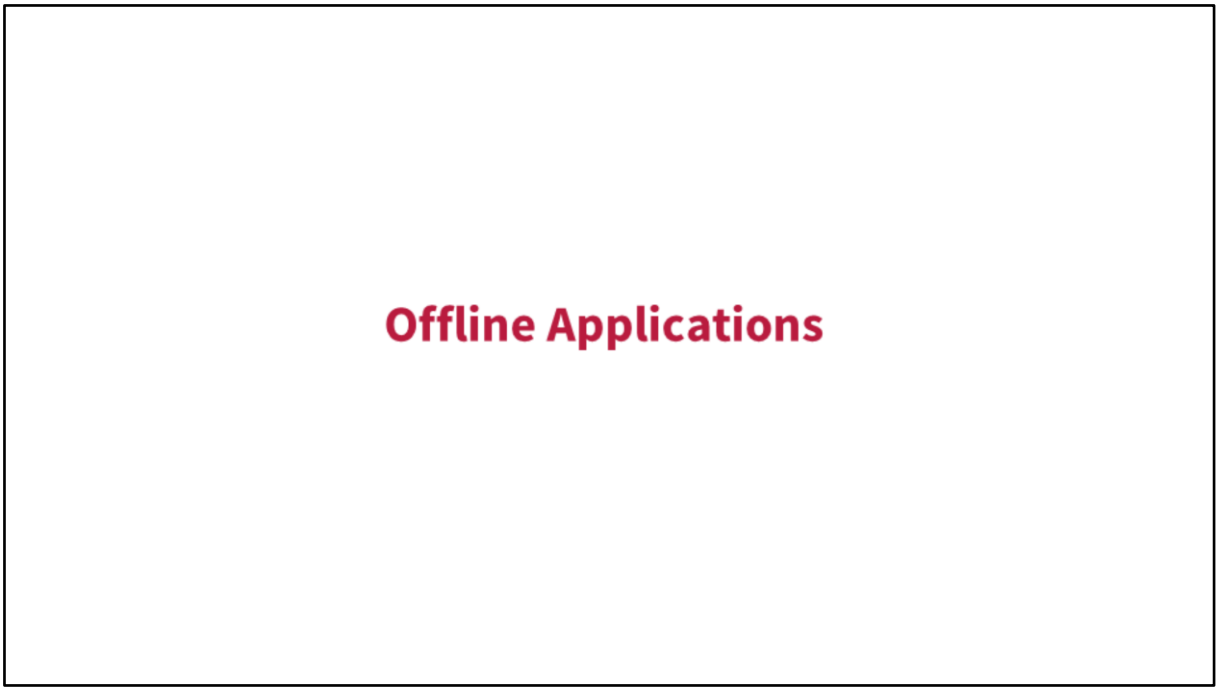


Otra cosa interesante, para nivelar un poco la parte SD con lo que ya teníamos para Web, es la posibilidad de crear componentes dinámicamente en nuestras aplicaciones en Smart Devices, hasta ahora podíamos en un Layout colocar un Objeto Component, “SD Components”, pero teníamos que referenciarlo directamente en las propiedades, es decir, teníamos otro panel aquí, por ejemplo este “PanelGameQuizz” donde muestra un Grid, podíamos poner entonces un component en este layout y en las propiedades le decíamos cual era ese objeto, bueno, ahora podemos utilizar el “create” en los eventos para crearlo dinámicamente.

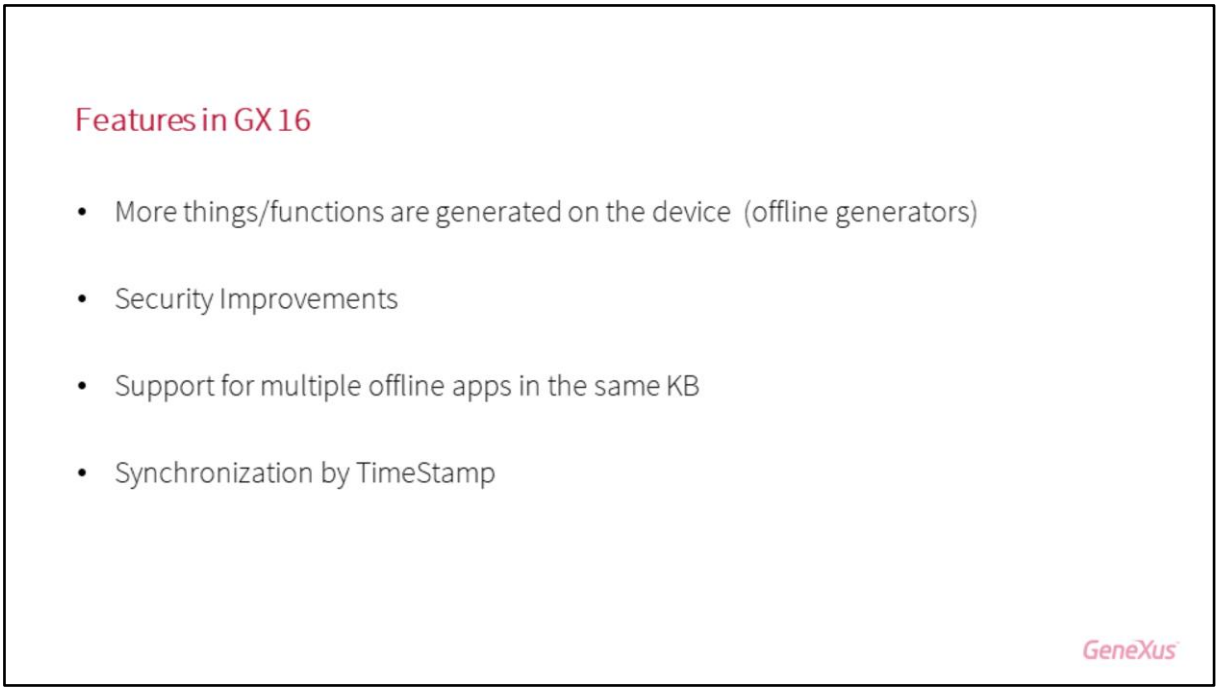
Eso te da la posibilidad entonces en runtime poder elegir que panel queremos mostrar en una determinada sección de nuestra pantalla.

La aplicación del evento, la aplicación GX28 utiliza en una de las pantallas bastante este concepto, porque en una pantalla nosotros queríamos que dinámicamente según el día se mostrara diferente tipo de información, por ejemplo la lista preliminar de charlas, pero después queríamos que se mostraran de nuevo las charlas, cuando comenzó el evento, empezar a mostrar las charlas diferentes, con los Tabs. Entonces si ven después cuando publiquemos esa base de conocimiento , van a ver que hay algunos layout que utilizan fuertemente este concepto, para dinámicamente dependiendo de … En este caso “el día” , cambiar que información le estamos mostrando al usuario en esa pantalla.

Bien, eso es lo que respecta a recursos de UI que fueron incorporados, complementando un poco lo que hablaron hoy para Smart Devices.



Lo otro que quería comentarles, era sobre el tema de las Aplicaciones Offline.



Aquí hubo una serie de mejoras, lo primero es:

Que se han agregado más funcionalidades, más funciones y cosas que antes no estaban soportadas por los generadores Offline.
Ustedes tengan en cuanta que una aplicación cuando es Online, gran parte del procesamiento se ejecuta en el servidor, es decir, si yo tengo un panel que estoy mostrando una lista de productos, lo que llamé es un Data Provider, un procedimiento o un servicio que fue a buscar e hizo todo ese procesamiento para traer lo que quiero mostrar en ese panel, lo trajo al dispositivo y el dispositivo a través de la metadata muestra esa información. Pero la llamada a todos los procedimientos ocurre en el servidor, era generado con los generadores Java o .Net.

Cuando la aplicación es Offline, todos esos proc, Data Provider, Bussines Components que está llamando la aplicación, se ejecutan todos en el propio dispositivo, eso es transparente para ustedes porque simplemente cambiando las propiedad es GeneXus quien determina “si esta aplicación es Oflline tengo que generar todo en el propio dispositivo“ y entonces ahí todos esos procedimientos, Bussines Components, Data Provider, se están generando con los generadores de los dispositivos, sea Swift para iOS o Java para Android.
Entonces teníamos algunas limitaciones en cuanto a funciones, métodos, recursos que antes podíamos generar perfectamente en Web pero no en el dispositivo, entonces empezamos a nivelar aquellas cosas que sí son necesarias para Smart Devices, por ejemplo, el tipo data expression no estaba soportado en los generadores Offline, ahora está soportado.
La posibilidad de llamar a a Web Servides “suop” no podíamos invocarlo desde una aplicación Offline, ahora sí podemos y así un montón de otras funcionalidades y funciones.

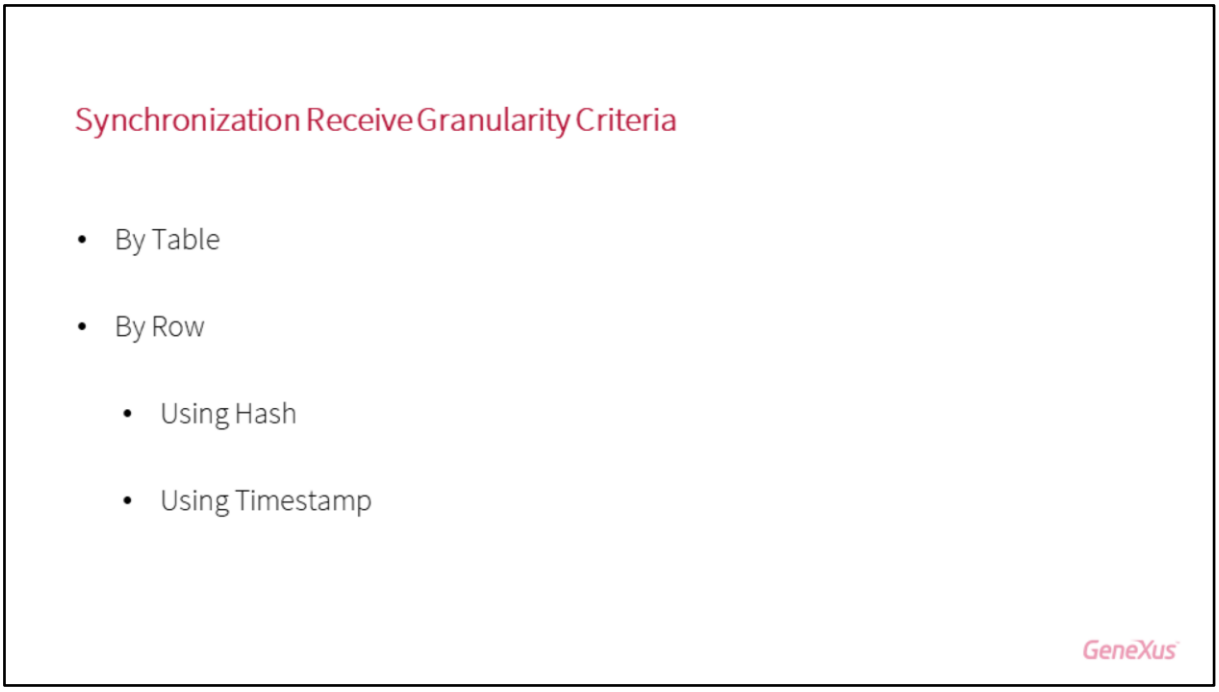
Por otro lado, se ha mejorado en la parte de seguridad
Antes con la aplicación Offline teníamos muchas comunicaciones con el servidor, cada servicio, cada panel esta ejecutando una invocación al servidor para traer los datos, para traer la información, para realizar cálculos, ahora todo es en el dispositivo y la interacción que hay con el servidor es a través del servicio de sincronización, básicamente es un único punto de entrada con el servidor, que es el que envía y el que recibe los datos del servidor.

Bueno, mejoramos toda la parte de seguridad de esos servicios cuando se utiliza GAM y por otro lado se implementa una nueva propiedad para encriptar la base de datos Offline. En algunos escenarios, sobre

todo de aplicaciones relacionadas con la banca, era necesario que la base de datos esté encriptada, que no se pueda acceder a la misma, solamente a través de la propia aplicación, eso es una propiedad que tienen adicional para poder encriptar entonces toda la base de datos Offline.

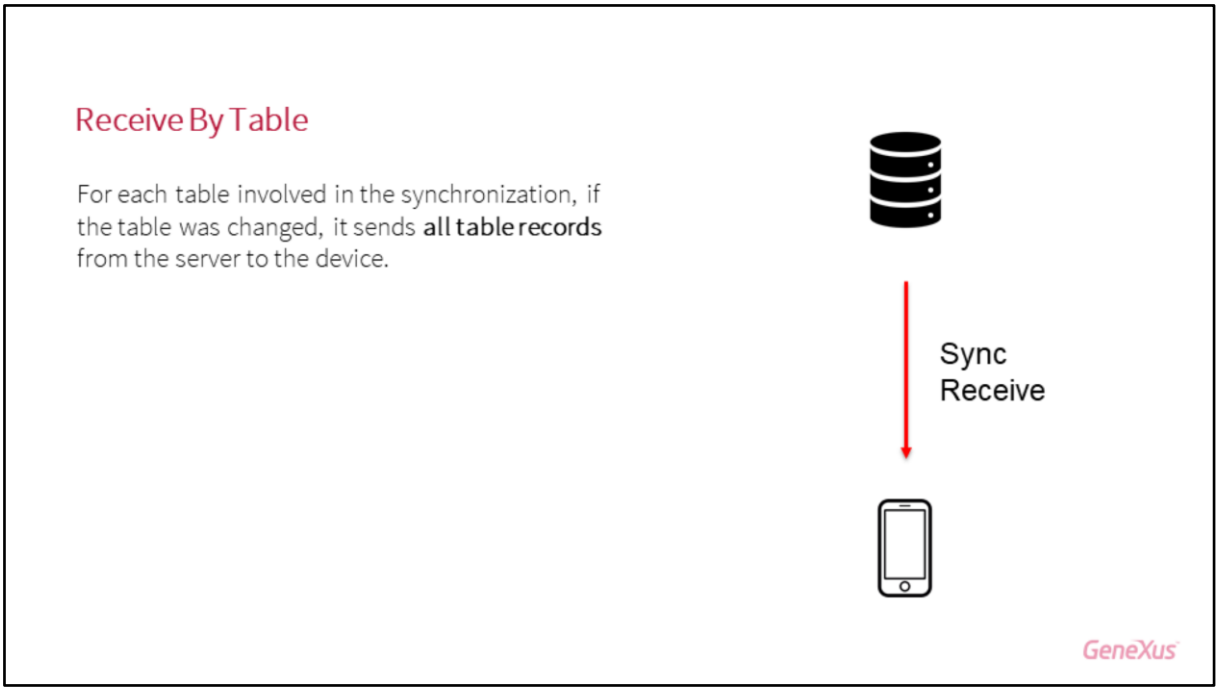
Lo otro es el soporte de múltiples aplicaciones Offline dentro de una misma base de conocimiento, eso era una limitación que teníamos simplemente de alcance, antes podíamos tener solo una aplicación Offline, ahora podemos tener más de un main Offline dentro de una misma base de conocimiento.

Y el último punto que quiero mencionar, es el tema de la sincronización por TimStamp, pero para entender el porqué de esta última “feature” me gustaría hacer un repaso general de este tema de la sincronización, sobre todo la parte del receive, para mostrarles un poco.
Si bien lo nuevo es la sincronización por TimeStamp, quiero repasar un poco todo para nivelar el conocimiento y entender un poquito el porqué de esta feature.



Cuando nosotros tenemos una aplicación Offline, hay una propiedad que se llama “Synchronization Receive Granularity Criteria” que es decir: “cuando se sincroniza la aplicación, cuando trae los datos del servidor al dispositivo ¿cuál es la granulería de eso?
La propiedad tiene dos opciones, una que es por Tabla o por Registro.

Dentro de registro, la sincronización que hasta ahora hacíamos era por Hash (ahora voy a profundizar en cada uno de esos puntos rápidamente) y lo que agregamos acá es la propiedad nueva, o sea, cuando usamos la Sincronización por registro agregamos esta opción por TimeStamp.



¿Qué quiere decir que la sincronización sea por Tabla? ¿Qué es lo que ocurre en ese caso?

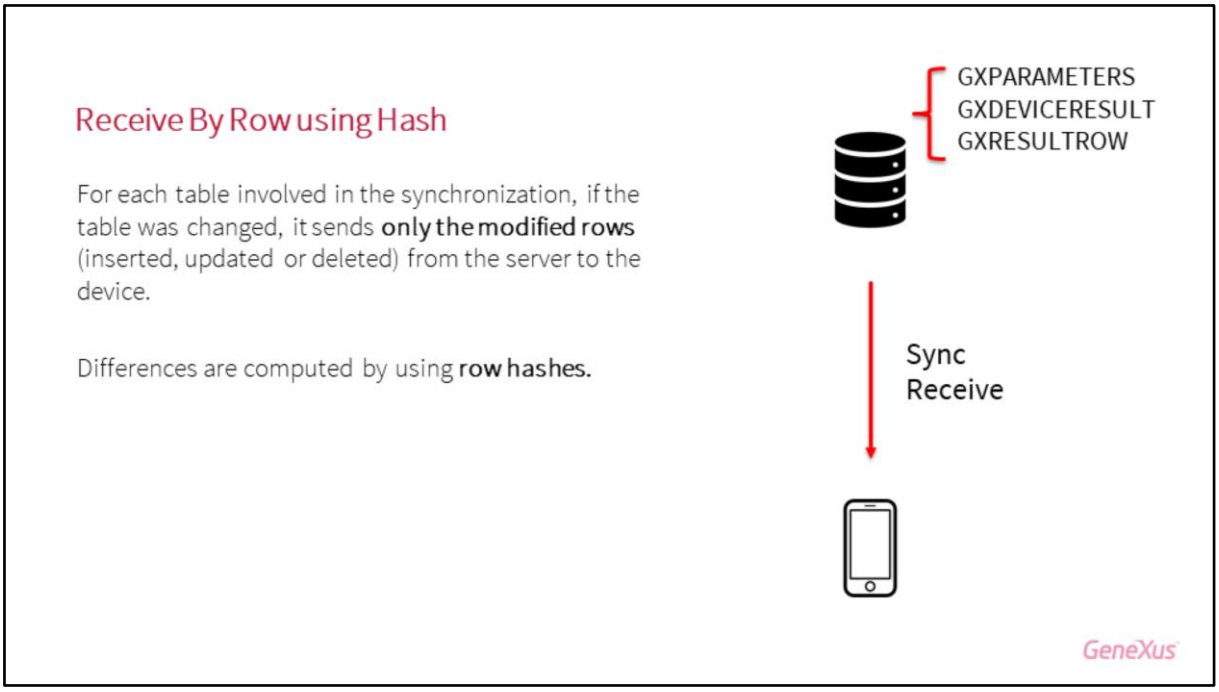
Cuando la sincronización es por tabla, lo que sucede es lo siguiente:
Nosotros creamos una aplicación que es Offline, entonces GeneXus determina a partir de ese main Offline todas las tablas que precisa para esa aplicación funcionar de esa forma desconectada, entonces va a crear una base de datos en ese QLite con las tablas necesarias para esa aplicación.

Cuando la opción es esa sincronización del Receive por Tabla, cualquier cambio que se haga en el servidor en algunas de esas tablas que participan de esa aplicación, GeneXus lo que va a hacer cuando va a recibir los datos es: Si hubo cambios, borro el contenido de esa Tabla local y traigo todos los datos nuevamente.

Si se quiere, es el algoritmo más simple, no hay tanta complejidad atrás, lo que se detecta es si hubo cambios en alguna de esas tablas con respecto a la última sincronización de ese device (o sea, la información es por device), entonces borro todo y traigo todos los datos nuevamente.
Esto puede ser útil si las tablas son pequeñas o la mayoría de los datos de esas tablas cambian, porque si igual tengo que traer casi toda la tabla, que la borre toda y la cargue toda de nuevo no va a ser un problema.

Por ejemplo, podría ser una aplicación para uso interno de alguna empresa donde los vendedores tienen la lista de clientes a visitar, quieren esa lista de clientes tenerla Offline porque quieren tener acceso a esa información independientemente de la conectividad; Y esa lista cambia todos los días, hoy tengo esta lista de clientes a visitar, mañana es una lista completamente diferente y entonces puede ser una opción, porque tiene que borrar todos los datos y traer todos los datos nuevamente.

¿Cuál es el problema de esta opción?
Cuando la tabla es muy grande, si la tabla o la cantidad de registros que voy a utilizar que preciso de esa aplicación Offline es muy grande, tenemos un tráfico de datos importante.
También cuando la aplicación quizás no es para uso interno y es más para una aplicación que vamos a poner a público general que va a estar en las Store, entonces hay muchos usuarios que pueden estar utilizando esa aplicación, cada vez que va a sincronizar un usuario, descargar todos esos datos puede ser que la experiencia no sea tan buena, entonces quizás no sea la mejor opción.



… Ahí es donde surge la segunda opción que es la “Sincronización por registro”, que hasta ahora usábamos el concepto de Hash que es lo que voy a explicar.

¿Qué es lo que hace la sincronización por registro? Lo que hace es:
En lugar de estar enviando cada vez que hay un cambio en el servidor, todo el contenido de esa tabla, borrar acá y cargar todo nuevamente, voy enviar solamente los cambios, es decir aquellos “insert”, “updates” o “deletes” que sucedieron en el servidor.
Entonces eso va a mejorar notablemente lo que es el tráfico de los datos, porque ahora sí ya tengo la tabla de clientes y aquí en el servidor cambió el nombre, el teléfono o la dirección de uno de los clientes, en lugar de enviar todos los clientes, simplemente va a enviar esa actualización, ese único registro, entonces ahí vamos a ver una mejora notable en lo que es el tráfico de datos, que de nuevo hay que considerar los escenarios, pero si la conectividad no es tan buena o si es una aplicación para el público general, puede ser mejor esta opción que la anterior, de hecho ésta es la opción por defecto cuando creamos una aplicación Offline.

Ganamos entonces en el tema de tráfico de datos pero obviamente que tenemos un mayor procesamiento del lado del servidor para detectar qué es lo que tenemos que enviar a cada uno de los dispositivos, eso GeneXus lo resuelve automáticamente, es decir, GeneXus va a crear un conjunto de tablas en la base de datos, en el Data Store principal, es decir en el servidor, va a crear tres tablas adicionales, creo que ahora son cuatro si mal no recuerdo, porque por el tema de usar más de una aplicación, de tener más de un main Offline ahora creamos una cuarta, pero básicamente son estas tablas las relevantes.
En esas tablas lo que se va a hacer es mantener el estado de la sincronización de cada uno de los dispositivos ¿qué quiere decir? Que este dispositivo, cuando estas pidiendo un recibo quiere actualizar los datos, ese dispositivo le va a enviar un Hash, cuando se cargó la primera vez, se ejecutó la “sentencia”, vamos a suponer que acá tengo la tabla de clientes y en realidad lo que estoy cargando son los clientes activos (mi proceso de sincronización lo que trae son los clientes activos o se adapta a un conjunto de clientes), cuando se cargó la primera vez, se guardó esa información y se crea un Hash con ese “select”, (el que trajo todos los clientes activos, el resultado de ese select), cuando va a pedir la próxima sincronización, le va a mandar ese Hash en el servidor, se va a volver a revisar la consulta para crear esa tabla que es la de los clientes activos y va a crear un Hash con el resultado de esa consulta , va a comparar con lo que tenía el dispositivo.
¿Son diferentes? si no hay diferencia quiere decir que la tabla no cambió, no preciso enviarle nada al

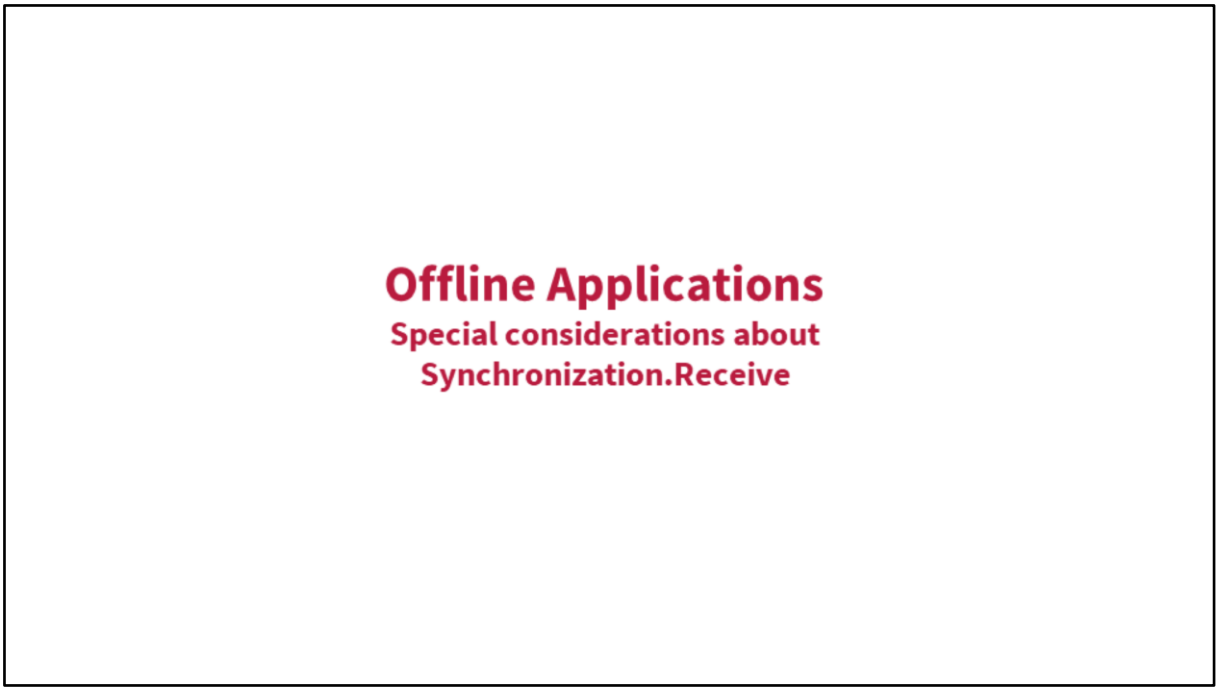
dispositivo.
Si hay diferencias, entonces ahí lo que va a hacer es comparar el Hash de cada uno de los registros, que también se guardaron esas tabas auxiliares para ese dispositivo para ver si hubo un cambio para saber cuál registro tengo que enviarle, cuál diferencia tengo que enviarle, le decís: si hay un Hash para un registro que hay diferente, quiere decir que ese registro fue actualizado y tengo que enviárselo al dispositivo.
Si hay un Hash nuevo, quiere decir que hay un registro nuevo en esa tabla que cumple las condiciones de carga de ese dispositivo, tengo que enviárselo a ese dispositivo, si hay un Hash que ya no está quiere decir que hubo un delete.

Entonces hay un procesamiento adicional del server, voy a precisar seguramente más storage para mantener toda esa información en el servidor del estado de sincronización de cada device, pero después con la ventaja de enviar menos información al dispositivo.

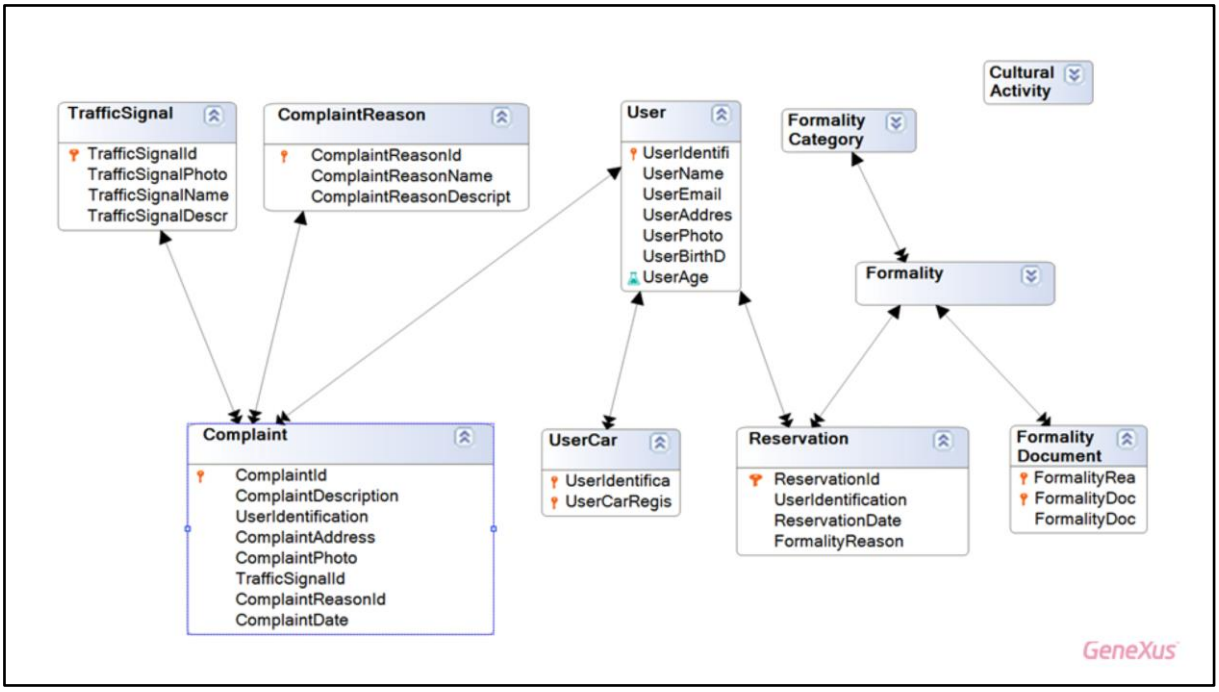
¿Se entendió un poco como es que funciona? Está todo documentado en el wiki pero quería que tengan un panel general de cómo es el tema.

¿Dónde podría estar el problema de este algoritmo, de este método de sincronización?

Cuando el dispositivo precisa muchos datos, o sea, tengo una tabla grande que necesariamente quiero que esté todo en el dispositivo, mismo aplicándole condiciones de carga para esa tabla Offline, es una aplicación que necesito que esté toda la tabla de productos completa en el dispositivo y tengo miles o decenas de miles de productos y además eventualmente muchos usuarios utilizando la aplicación o algunos usuarios, pero que requiere sincronización cada cierto tiempo, no es que sincronizo una vez por semana sino que preciso que se vaya actualizando la información en el dispositivo. ¿Por qué? Porque si tengo una tabla con muchos registros en el dispositivo, la primera opción de sincronización por tabla obviamente que no es la mejor porque va a borrar toda esa tabla enorme y traer todos esos datos cada vez, puede ser muy malo en cuanto al tráfico de datos y la sincronización por Hash (por registro usando Hash) si bien va a traer solamente los registros que modificaron… Imagínense una tabla de 8mil productos y se actualizaron 2 productos en el servidor que fueron actualizados por otro dispositivo o desde el backend se actualizó el precio de 2 productos, para sincronizar van a viajar solamente los 2 productos que fueron actualizados, pero para cuando detectó el algoritmo de Hash que esa tabla cambió, tuvo que barrer toda la tabla, es decir todos los registros que cumplen con las condiciones de carga para saber cuales fueron los registros que se actualizaron para enviar esos dos, entonces si son muchos, si la tabla es muy grande y tengo muchos dispositivos que están sincronizando, eso puede tener una carga importante del lado del servidor y ahí es que surge la propiedad que está en la versión nueva, (todo esto ya estaba desde antes) que es la sincronización por TimeStamp.



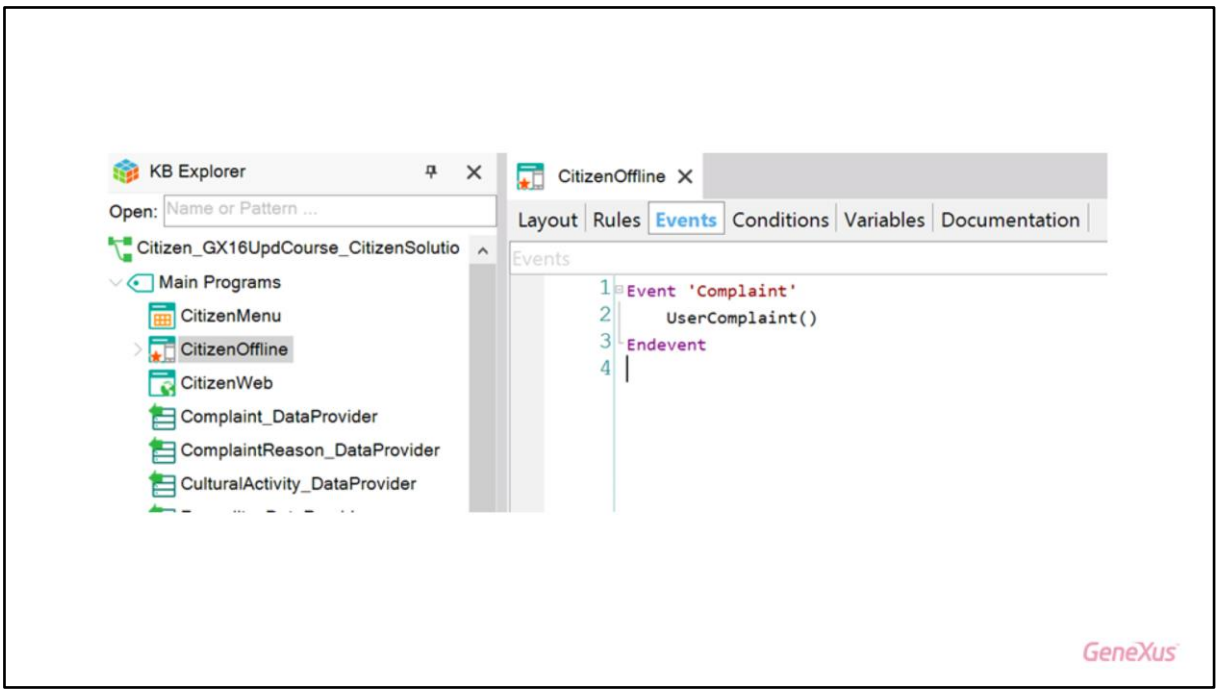
Algunas consideraciones con el tema, específicamente estoy hablando siempre del receive, la parte del Send no hablé porque me parece que ésta es la parte que está bueno entender un poco más cómo funciona…



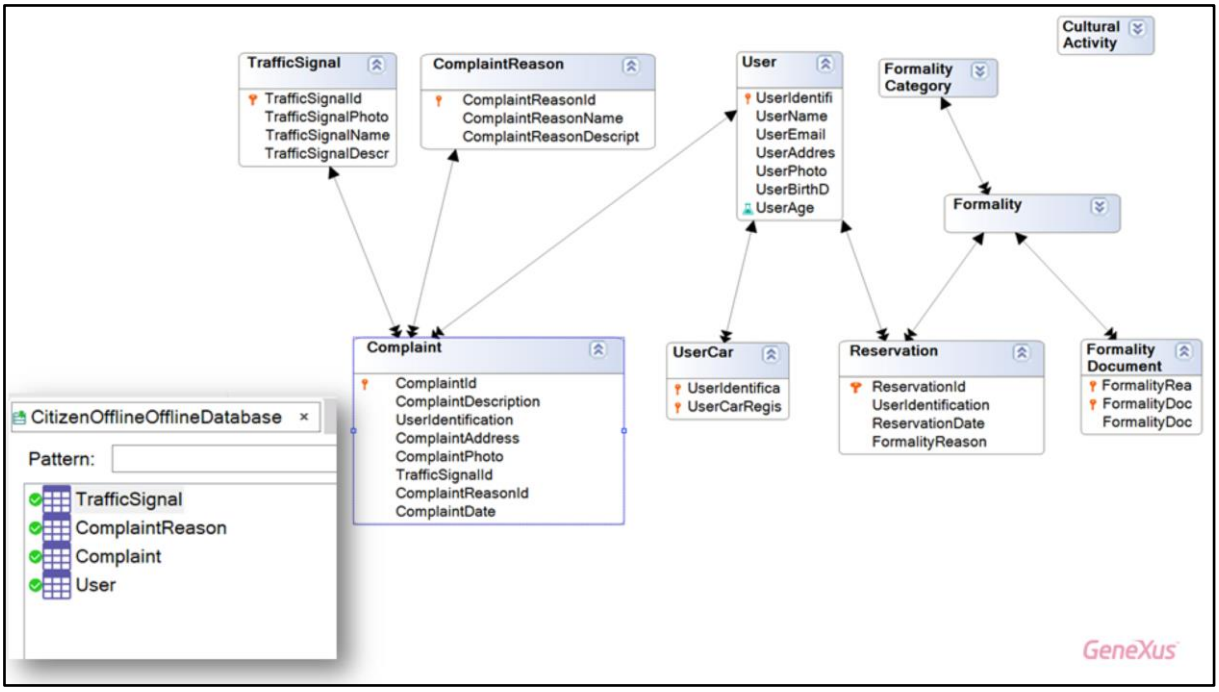
Lo que les decía recién, es que es importante que en nuestras aplicaciones Offline solo incrustemos los datos mínimos necesarios, es decir, el conjunto mínimo de tablas y además el registro de esas tablas.

Para entender un poco, traje el ejemplo que ustedes están utilizando en el práctico, que es la base de conocimiento del gobierno local, intendencia o prefectura.
Estas son las entidades, aquí está bien simplificado, 10 entidades donde tenemos: El usuario, los reclamos que realiza el usuario, los tipos de reclamo, tipos de señales de tráfico, las reservas, etc.
Ustedes ya seguramente se familiarizaron con el tema en el práctico.

Vamos a suponer que queremos hacer una aplicación Offline, donde lo único que queremos que esa aplicación haga (y lo estoy simplificando para entender bien como es que funciona esto) es que el usuario pueda reportar un reclamo, hacer un reclamo, nada más, es decir… va en el tránsito, ve un problema, algo que está roto en la calle, un árbol caído o una señal del tráfico que está mal… reportar ese reclamo y que sea Offline porque eventualmente el lugar donde el usuario está no hay conectividad, entonces quiero que quede Offline y cuando pueda se envíe al servidor, simplemente eso y es Offline!

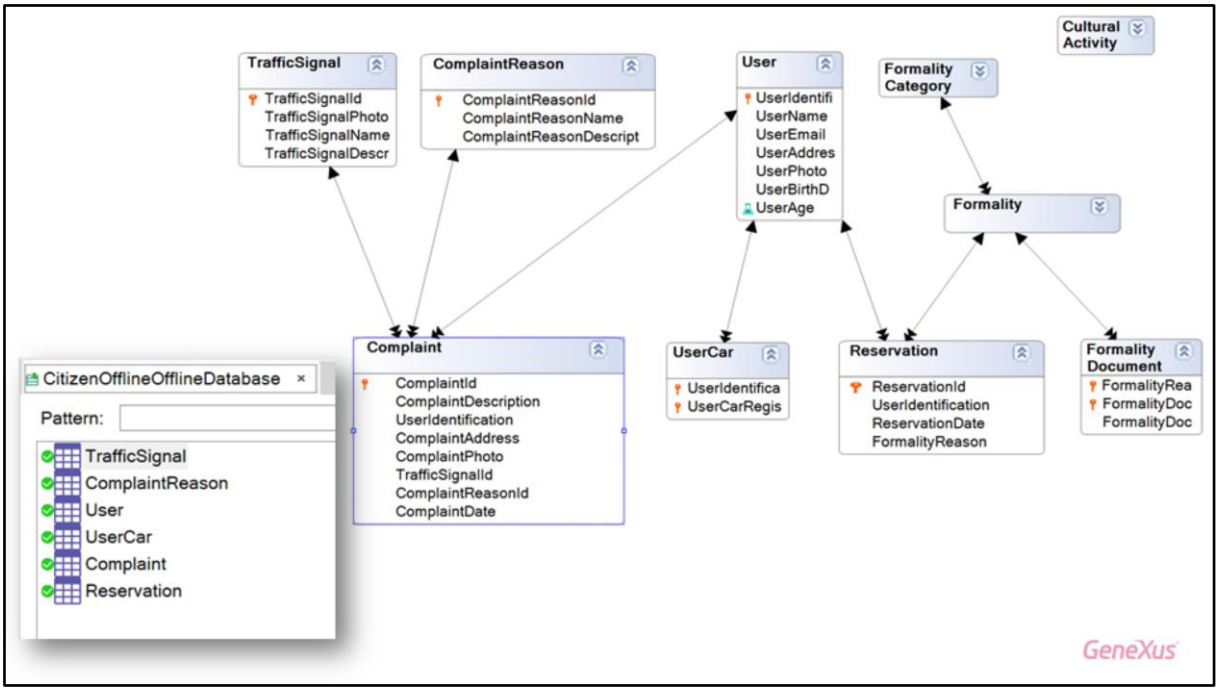


Lo que haríamos entonces sería:
Un main en Smart Devices, le decimos que es Offline y lo único que llama es este objeto (que ustedes tienen y van a ver en el práctico) que es el User Complanit, que es para insertar un reclamo, lo único que hace es insertar un registro en la tabla de reclamos que es esta de aquí.



… Si yo creo ese Main, ese objeto Main que solo hace un insert en la tabla reclamo por medio de un Work With, Smart Devices o a través de un Bussines Component, GeneXus va a determinar que para esa aplicación Offline se precisan estas tablas (no se si se llega a ver desde ahí), esas cuatro tablas:
La tabla de reclamo obviamente, porque voy a insertar un registro en esa tabla Offline y después preciso los datos de estas tablas, porque si voy a insertar un reclamo preciso los tipos de señales de tráfico, los motivos del reclamo y el usuario, son las foreing Keys de esta tabla, la extendida, entonces GeneXus determina.
Bien, para ésta aplicación Offline, sólo preciso de éstas tablas, no preciso de todas estas.

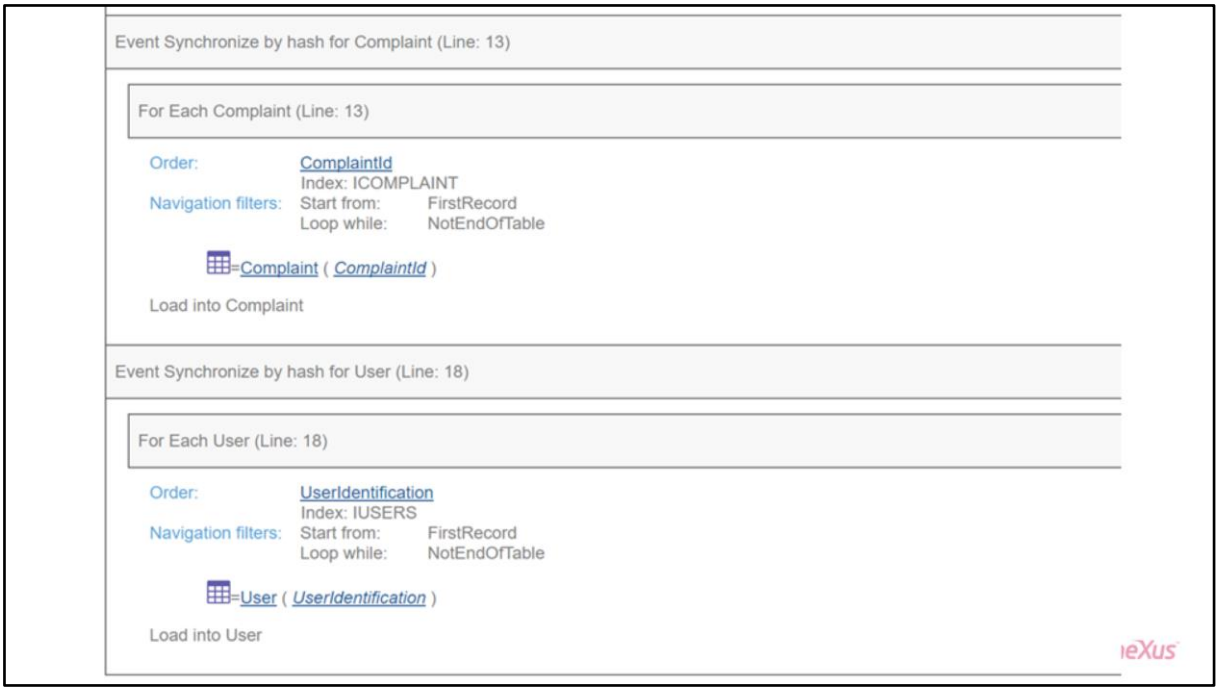
Es importante que analicemos, en análisis de impacto de esa aplicación Offline, vieron que cuando hacemos un Build de un main que es Offline, GeneXus muestra un listado de las tablas que van a participar en esa aplicación, de las tablas que van a ser creadas en el SQLite.



¿Por qué les digo que es importante?
Porque si se agrega alguna tabla, ya sea que GeneXus interpretó que hay otra tabla más que es necesaria o ustedes tenían alguna llamada de algún objeto que no sabían y que por lo tanto está yendo a otra tabla que no es necesario para la realidad de esa aplicación, por ejemplo, la tabla de reservas, todos los algoritmos de sincronización que yo les mencionaba antes (cuando les decía que comparan si hubo cambios o no) van a estar analizando otras tablas que capaz no sea necesario, que no tengan ustedes interés en que se esté analizando, por eso es importante que vayan solo a las tablas necesarias.

De hecho cuando yo hice el ejemplo este en la KB del curso, de definir este objeto simple que lo único que llama a este User Complaint y miré el análisis de impacto, no me creó eso, me creó éste análisis de impacto (señala), me agregó dos tablas más además de las que ya estaban recién, que es la tabla de reservas y la de UserCar, ¿Por qué? Porque en el panel para insertar un reclamo había definido una variable basada en un BC User, simplemente para traer el nombre el usuario.

Como estaba ese Bussines Component, GeneXus en la clausura del árbol de llamadas de ese main dice “ah, acá preciso el Bussines Component del usuario”, ese Bussinesa Component del usuario llama también al Work With para insertar, el Work With User y el Work With User tenía una llamada a estas dos tablas para mostrar los autos del usuario y las reservas del usuario, entonces en realidad me trajo todo esto, eso quiere decir que la aplicación Offline va a crear estas tablas y va a cargar los algoritmos de sincronización, van a cargar los datos de todas estas tablas cuando yo no preciso para nada de estas tablas, entonces por eso es importante analizar… porque nos ha pasado de reportarnos problemas de TimeOut en la sincronización o errores de sincronización o demoras importantes en la sincronización y es por causa de estar utilizando tablas que realmente no eran necesarias en esa aplicación.

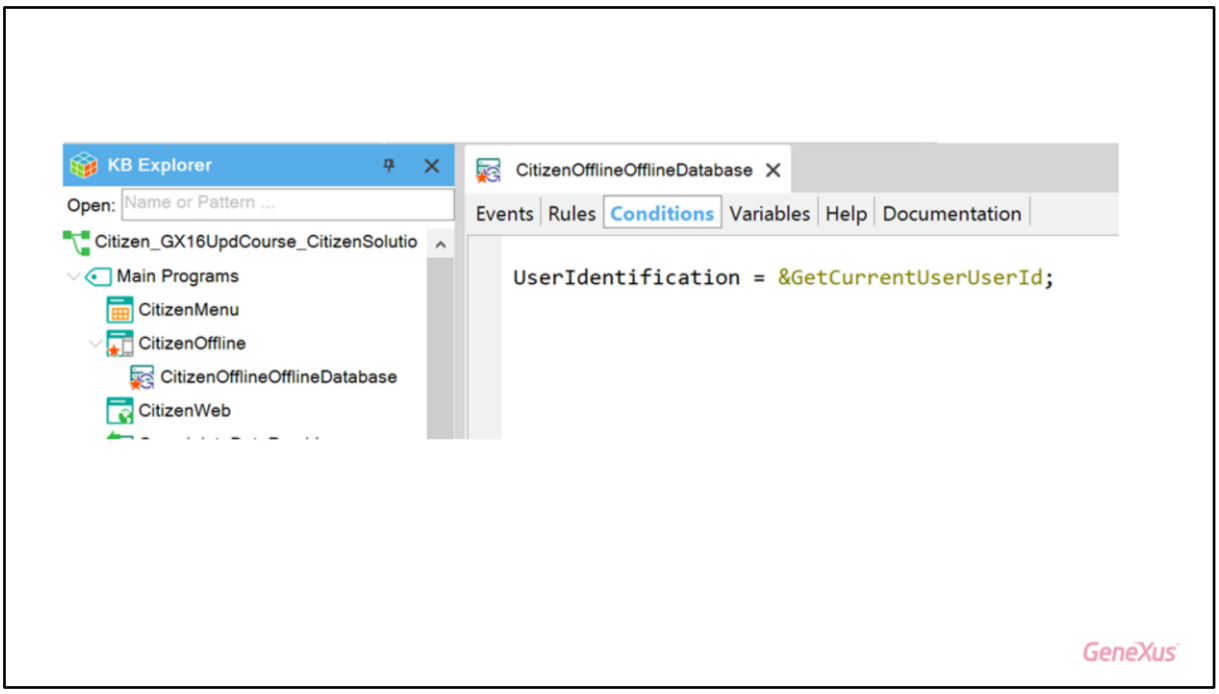


…eso es lo primero, lo segundo, no solo las tablas, sino que también los registros que participan y que voy a precisar en la aplicación Offline, porque no los voy a precisar todos.

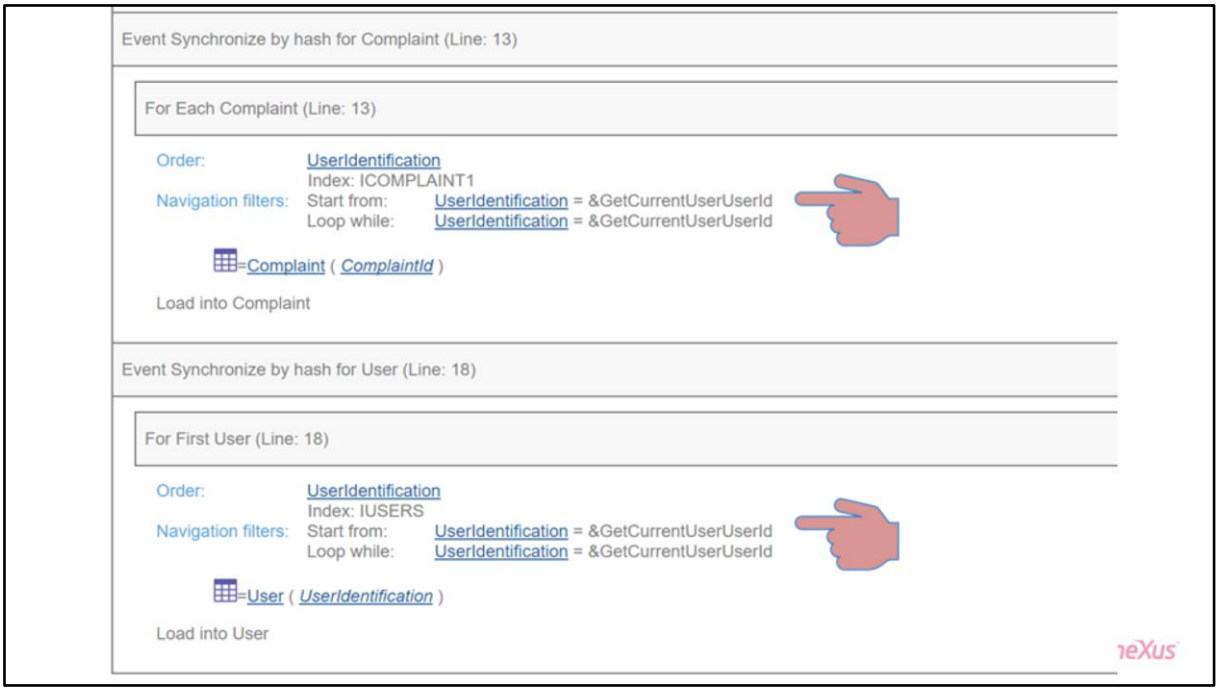
Cuando hacemos el Build del main Offline, nos muestra el análisis de impacto (lo que les mostré recién) y además nos muestra la navegación del Objeto Offline.
La navegación del objeto Offline, lo que muestra es cómo va a ser el procedimiento de carga, cuáles son los datos que va a traer del servidor al dispositivo, va a decir que va a recorrer toda la tabla ComplaintId, para traer todos los datos, para usar la sincronización de esa tabla y para la tabla de usuario, lo mismo.

¿No notan algo raro ahí? El objetivo de la aplicación, recuerden que era insertar un reclamo, el usuario nada más…

Está trayendo al dispositivo a todos los usuarios, todos los usuarios que están en el servidor están viajando al dispositivo, no los preciso para nada… está cargando al dispositivo todos los reclamos que hicieron todos los usuarios, no los preciso para nada… si lo único que quiero es insertar un nuevo reclamo de ese usuario.
Entonces, de nuevo… ¿Por qué eso es importante? ¿Cómo resolvemos esto antes?



Bueno, en el objeto Offline tenemos el Tab Condition donde podemos agregar condiciones, en este ejemplo, claramente lo que tengo que agregar es la condición “UserIdentification =“ obtener el usuario que va a estar en la sesión, el usuario que se logueó o el ID del dispositivo, de alguna forma podemos determinar si el usuario tiene un log-in o la identificación del dispositivo (que eso lo podemos tener en la sesión) agregamos esta condición, eso va a hacer que el procedure de carga ahora agregue estas condiciones…



… es decir, va a cargar solamente mi usuario, va a traer un registro y no los 10 mil usuarios que puedan utilizar la aplicación y va a traer solamente los reclamos hechos por mi, de hecho podría poner una condición que incluso no traiga ningún reclamo porque no quiero visualizar reclamos que yo hice antes (si voy a solamente insertar) pero como mínimo decir “bueno, traer los reclamos por si quiero visualizar algún reclamo que hice antes, solo los reclamos que yo hice”.

Agregué solo una condición, GeneXus ahí cuando hace la navegación del objeto Offline, mira en todos los lugares, en todas esas navegaciones donde puedo aplicar la condición esa, acá obviamente la tabla de usuario la va a aplicar porque es la calve primaria, en la tabla de Complaint está como Foreign Key, entonces también la voy a aplicar pero si uso Tabla extendida, también GeneXus la va a aplicar, entonces lo que va ahora a traer son menos datos.

… y eso es muy importante, ¿por qué? Porque cuando hoy les decía que la sincronización mira todos los cambios que hubo, la sincronización por Hash que hace un barrido de la tabla de los cambios, no es de todos los registros que están en el servidor, sino de los que fueron sincronizados, es decir, si cambié algún dato del reclamo de otro usuario, el algoritmo de sincronización no le interesa, ni siquiera se va a fijar si hubo cambio… cuando la tabla cambió, va a mirar si hubo cambios en los reclamos de éste usuario pero no de otros.

- En el caso anterior, cuanto tenías esas dos tablas demás ¿Cómo hago para eliminarlas de la sincronización?
- Bueno, el caso que yo les mencionaba hoy, en realidad lo que hice fue sacar esa variable Bussines Component porque no la precisaba, era para ir a buscar el nombre (usaban un Bussines component) lo resolví con un Foreach y ahí ya al no llamarle Bussines Component ya no se llama a la transacción.
- ¿Eso dentro del procedimiento que utilizaste ahí?
- El problema estaba en el panel, en el SD Panel que mostraba para hacer el insert del reclamo (que me pedía cuál es el reclamo, cuál es el usuario, cuál es el motivo) había en ese panel una variable basada en el user, ese panel lo identifiqué porque vi estas tablas que estaban demás, ahí me puse a buscar con el Cross reference (con el Cross reference de ese objeto) dije “ ¿a ver, como llego acá por esta transacción?” “¿y ésta transacción por qué esta agregando esto?” bueno, ahí mirando todo el árbol de llamadas ves y ahí podes hacer el corte.
- ¿De acá no es editable? Digamos, ¿no podría sacarlo?
- No, no, eventualmente si hay algún caso donde no pueda sacar… porque GeneXus está agregando

una relación y no la podes sacar, siempre lo que podes hacer en la parte de condiciones es, por ejemplo… la tabla de reserva me la está agregando y no la puedo sacar porque justo hay un árbol de llamadas que cortar eso me resulta difícil… pongo una condición, la clave primaria de reserva, “reserva ID menor que cero” y la va a crear pero se que nunca la va a sincronizar y nunca van a participar para nada del proceso de sincronización…

Synchronization Receive Summary

- Check involved tables and their load conditions
  - <Main>OfflineDatabase database report
  - <Main>OfflineDatabase navigation report
- Minimum time between receives
- Data Receive granularity

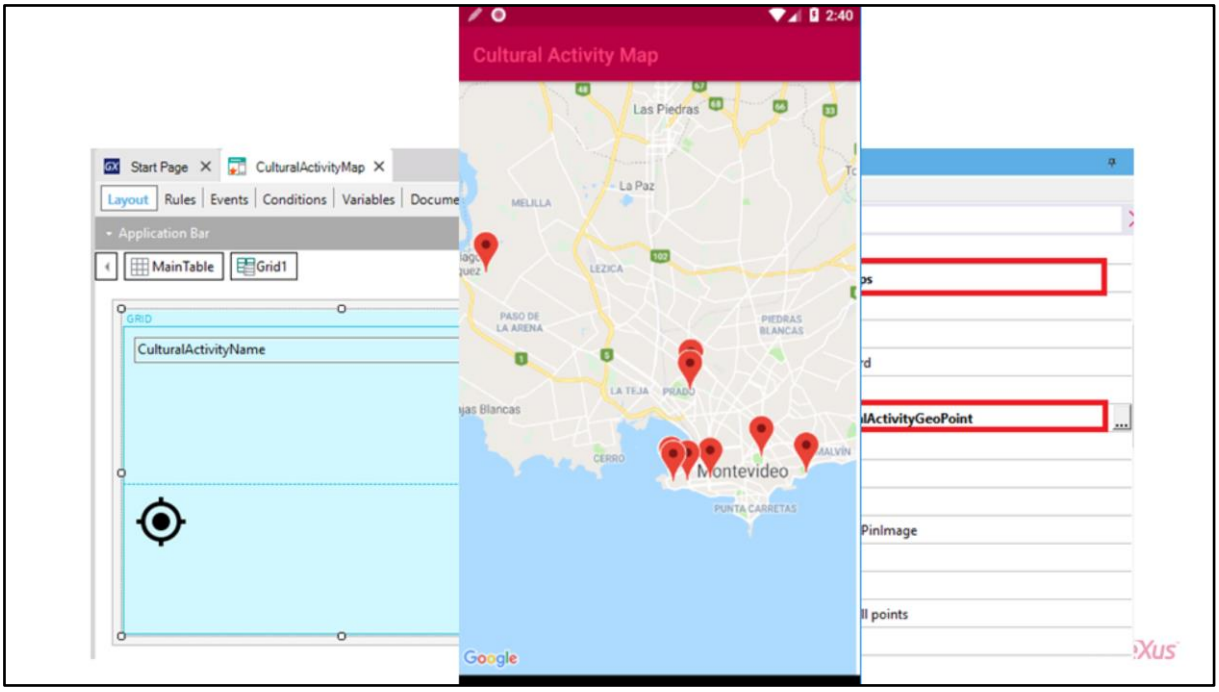
GeneXus

Bien, entonces.. Para terminar,

Revisar, es importante ver el reporte de creación de las tablas Offline.
Lo mismo la navegación del objeto para ver la extensión de registros solamente necesarios
La granuleidad, ahí definen si es por registro, por Hash o por TimeStamp, dependiendo de las cosas que vimos recién.
Y después el tiempo mínimo entre receives, es otro dato importante que vean si son muchos los dispositivos que están sincronizando y la cantidad de datos de ver si necesariamente tienen que hacerlos muy seguidos o no, porque eso puede llegar a sobrecargar el servidor.

Multi Layers - SDMaps

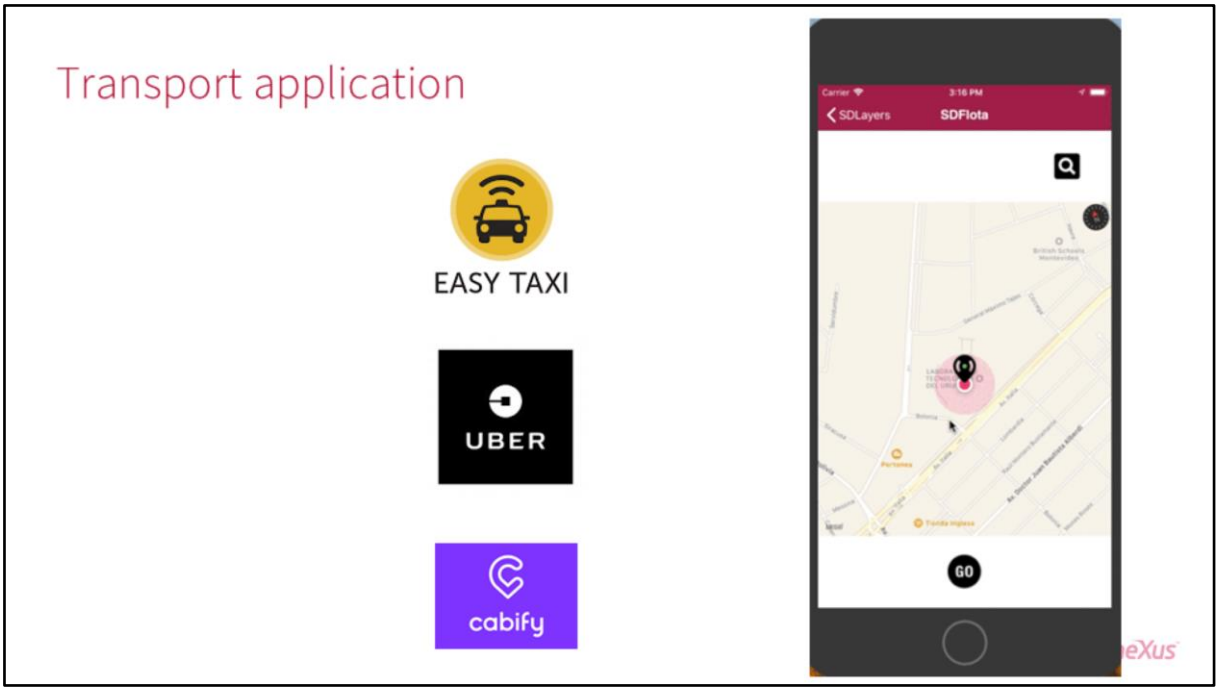
Yo les voy a contar un poco lo que estamos haciendo en el área de geografía, lo que llamamos Multi Layers, o SD Layers o SD Maps



Primero los voy a poner un poco a tono de qué es lo que tenemos desde hace tiempo, esto lo tenemos desde aplicaciones SD.
Cuando queremos mostrar un mapa de alguna forma y poder mostrar allí geometrías sencillas, o sea puntos, lo que tenemos es la posibilidad con una propiedad que nos las da la propiedad Control type SDMaps, a un grid yo le digo que es “control type SDMaps” y me habilita una cantidad de propiedades, entre ellas la propiedad “Location Attribute”, este Location Attribute es una coordenada en coordenada geodésicas, es una latitud/longitud.

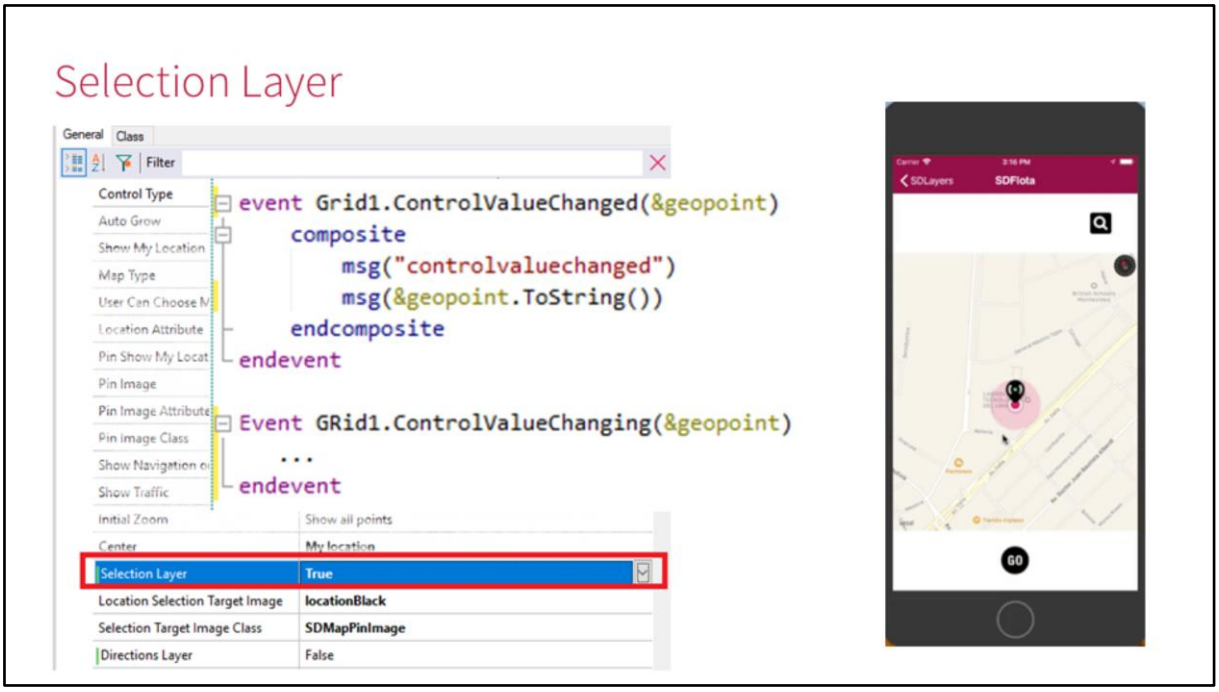
Tenemos dos formas de especificarlo, a partir de la versión 15 de GeneXus este atributo puede ser lo que nosotros llamamos al inicio de todo esto “el dominio GeoLocation”, es decir, un string que tiene concatenada la latitud y la longitud o puede ser un atributo GeoPoint o una variable GeoPoint, GeoPoint es un tipo de datos específicos que la base de datos entiende, en donde almacena también información geográfica.

Entonces, si yo en un grid que vería una lista (este es el ejemplo que también están viendo en el laboratorio, están viendo las actividades culturales) si yo le prendo esa propiedad lo que voy a ver es esto:
Es un mapa con distintos puntos, los valores de esos puntos son los que tiene el atributo GeoLocation o el atributo GeoPoint.



Lo que nos empezó a pasar es que muchos clientes nos estaban pidiendo aplicaciones de este estilo, tanto cualquier aplicación de transporte, ya sea Easy Taxi, Uber, Cabify o muchas otras, lo que tiene es algo así:
La aplicación principal es un mapa y el mapa tiene un ícono que está fijo en el centro y el mapa es lo que se va moviendo, cuando yo voy moviendo, a medida que suelto me va dando de alguna forma la posición donde estoy, obviamente también estas aplicaciones me permiten tener una búsqueda, yo lo puedo buscar por el ícono o puedo hacer una búsqueda de places o geo-codificar de alguna forma y después cuando yo presiono un botón o cualquier acción, lo que hace es mostrarme el camino entre ese punto que ingresé y mi posición o el origen que yo le di.

Ese es un poco un diálogo, una forma de navegar las aplicaciones que es bastante común.
Nosotros tenemos algo parecido a esto con lo que denominamos “Pick Location”, el pick location vendría a ser nuestro prom de mapas.
Si yo quiero capturar una posición, en realidad tengo una navegación similar, pero el objetivo de ese pick location es devolver un valor y acá el valor lo tengo dentro de la aplicación y con ese valor empiezo a hacer cosas.

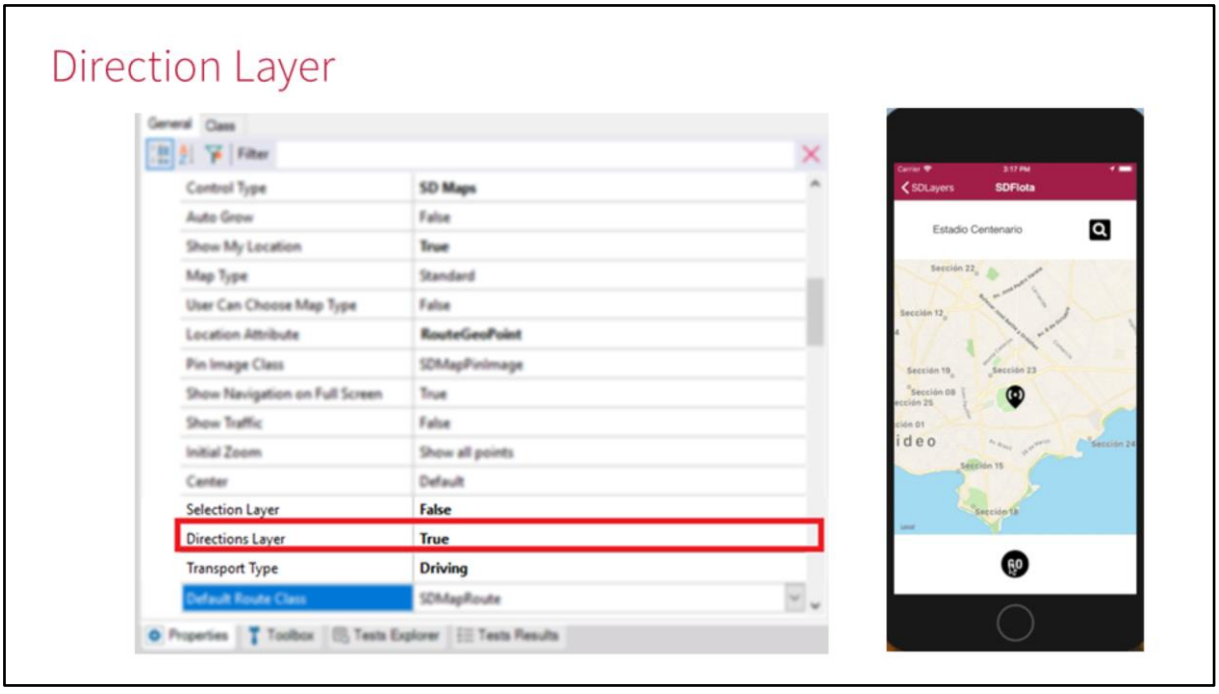


Bueno, lo que hicimos de alguna forma es que, intentamos abstraer esos comportamientos y simplificarlos un poco (lo que intentamos hacer en GeneXus, simplificar y abstraer la forma de definir eso)

Entonces, definimos dos propiedades:
Cuando yo tengo el Control Types SDMaps, además del Location Attribute y otra cantidad de propiedades que tengo disponibles, ahora tengo una propiedad nueva que es “Selection Layers”

La Selection Layer hace esto que les decía al inicio, me permite definir un ícono en el centro, para también se va a habilitar una propiedad que me va a decir ese ícono qué imagen le voy a dar y también me permite definir una clase que es la que va a dibujar esa imagen.
Entonces cuando yo tengo el control type prendido y la propiedad Selection Layer, en mi grid voy a ver los puntos que tenga cargados igual que en cualquier caso, pero voy a tener este tipo de navegación.

Y lo otro que voy a tener son dos eventos, un evento que se dispara cuando yo solté el mapa, (quedó quieto y el ícono quedo posicionado) ahí me va a devolver la posición donde quedó.
Y otro que se va disparando a medida que yo me voy moviendo.
Con estos dos eventos, lo básico que tengo es que esto es una variable de entrada para nosotros, que me dice la posición donde está.
Yo podría, por ejemplo, dibujar la ruta ahí o guardar este valor en la base de datos o lo que fuera.

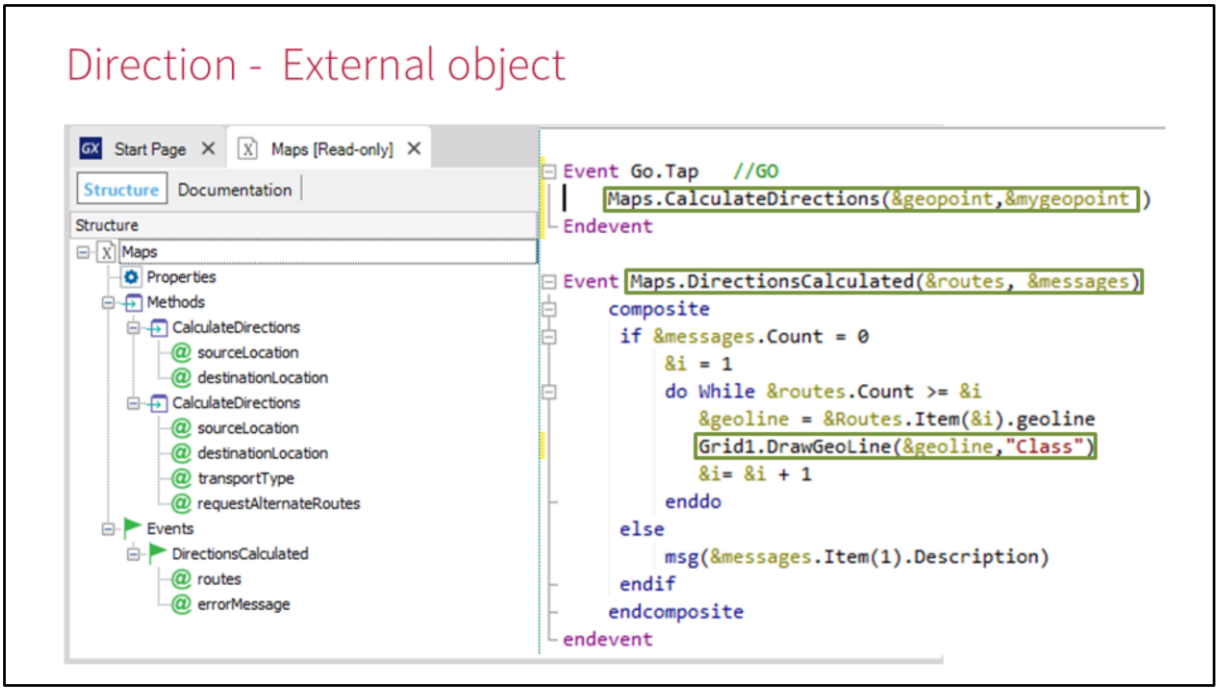


La otra propiedad que definimos para abstraer este comportamiento es la “Directions Layer”

La Directions Layers lo que hace es, (así como cuando yo tenía un control type SDMaps, me mostraba por cada GeoPoint que tenía ingresado en ese grid la posición, me dibujaba, me mostraba por ejemplo, los lugares de las atracciones culturales) bueno, ahora en lugar de mostrarme los lugares, me va a hacer un camino entre esos lugares,, me va a mostrar la ruta que une a esos lugares.

Me va a habilitar también dos propiedades más:
La primera es el Transportation Type, o sea, ese camino va a ver si lo voy a hacer manejando (en un auto) o si va a ser caminando, o si va a ser en bicicleta, (o tengo un par de parámetros allí) y después otro es si es la clase que me dibuja esa línea, yo a esta clase le puse color rojo y bastante gruesa.

Básicamente con estas dos propiedades fue que yo hice el ejemplo de Transportation, que ahora les voy a contar un poco cómo lo hice, pero además de la Direction Layers, de la posibilidad de mostrar el camino que hay entre dos puntos o más, creamos un External Objet nuevo…



Este external Object que se llama “Maps” es muy parecido al External Object GeoLocation que tenemos desde hace varias versiones en GeneXus.

Cuando salieron los generadores Smart Devices, para Samart Devices creamos estas propiedad Control Type SDMaps para los grids y además creamos un External Object que se llama GeoLocation, ese External Object es el que me permite obtener la posición de mi Device, iniciar un tracking, calcular la distancia entre dos puntos, tiene una cantidad de funcionalidades.
La entrada de ese External Object siempre es lo que nosotros llamábamos un “Dominio GeoLocation”, un string, ahora estamos pasando a emigrar todas esas funcionalidades también al tipo de dato Geographic, GeoPoint y GeoLine que son los nuevos tipos de datos que liberamos en la versión 15. (solo eso para ver un poco de contexto)

Bueno, éste External Object, entre otras propiedades que va a tener, tiene la propiedad CaculateDirections.

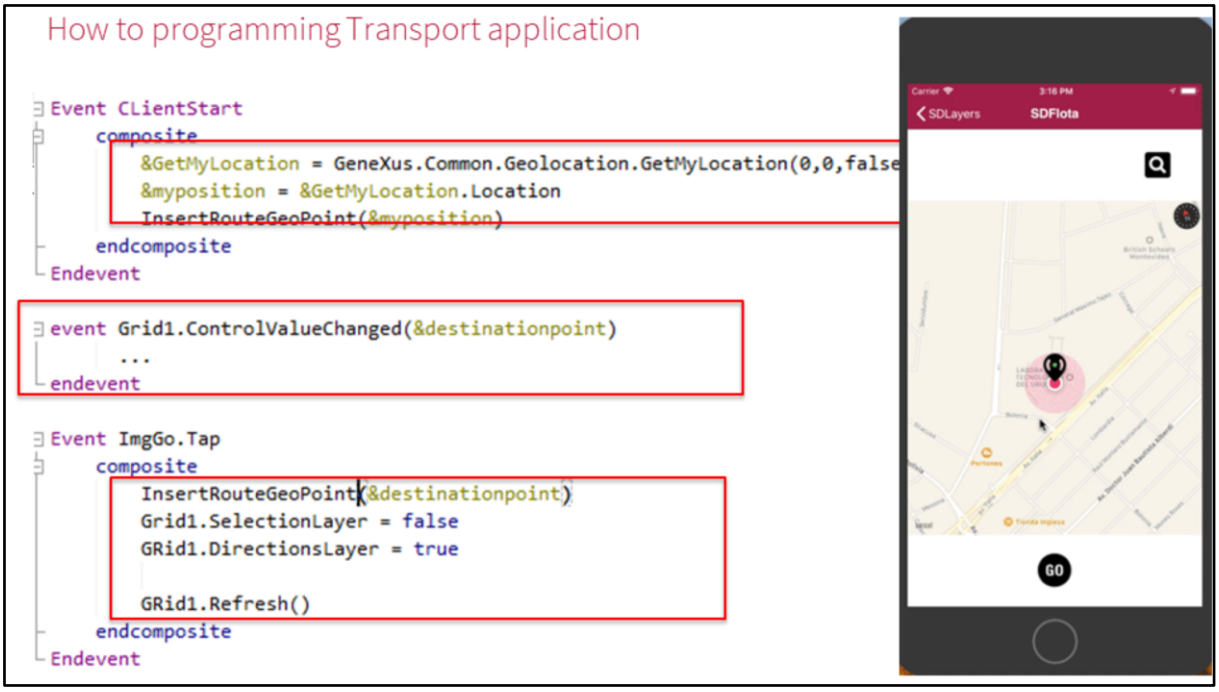
Esta tiene sobrecarga de métodos, puedo solo ponerle un origen y un destino o puedo ponerle origen/destino/TransportationType.

Y ésta que no la pusimos a nivel de grid pero también podría estar, que me devuelve más de una ruta, me devuelve rutas alternativas entre esos dos o “n” puntos que yo tengo.
Cuando yo llamo a ese método, lo que se me va a habilitar es un evento que se llama “DirectionsCalculated”, en códigos sería algo así,
Yo en ese botón que calculaba el camino, lo que podría hacer en lugar de hacerlo sólo con la propiedad de la grilla, podría hacerlo con el External Object, hacer el CalculateDirecions entre dos puntos y en este evento (DirectionCalculated) voy a obtener una lista de las rutas, una lista de GeoLines y algunos otros datos que me devuelve.
Esa ruta yo me la podría guardar en la base de datos por ejemplo, o se la podría compartir a alguien en un archivo.

La otra cosa que agregamos en la 16 es una propiedad al grid que se llama “DrawGeoLine”, es decir que, en ese Grid si yo le paso una GeoLine y le digo con que clase la dibuja, me dibuja una línea.

Esta funcionalidad, tanto de Selection como DirectionLayers, están disponibles en la 16 liberada para el

generador iOS, en Android en los primeros Upgrade va a estar, ya tenemos los prototipos armados, simplemente por un tema de tiempos no lo pudimos incluir.



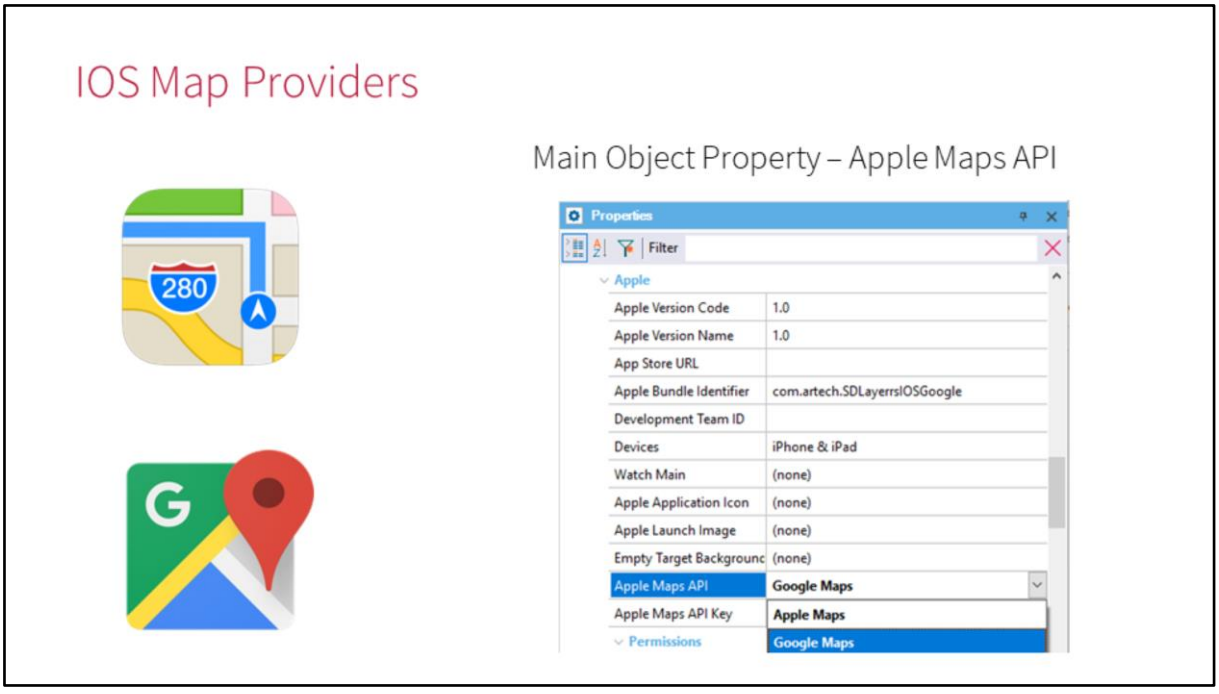
Para fijar ideas, no es que haya una forma de hacerlo, yo les quería contar ese ejemplo del principio que les mostré de una Transport Application, ¿cómo lo hice…

Bueno, en mi caso lo que hice para armar esto, no usé el External Object, usé directamente las propiedades del grid nada más.
Si bien tengo una tabla con todos los lugares de interés, me creé una tabla nueva que va a ser solo la que va a mostrar el ruteo, la que va a mostrar la dirección entre los puntos y a esa tabla le puse un grid en un SD panel, acá tengo la búsqueda que es una Geo Codificacion (no me voy a meter en este punto, es algo que tenemos hace tiempo) y a ese grid le puse la propiedad Selection Layer prendida, con eso lo que va a pasar es que cuando levante el panel, voy a tener el ícono del centro y me voy a poder mover.

¿Qué hago en mi panel?
Lo primero que hago cuando entro al panel, doy de alta en este caso a mi posición que es lo que a mi me interesa, porque yo quiero en este ejemplo mostrar desde donde yo estoy, hasta algún lugar de interés que yo encontré, cual es la ruta.
Entonces me guardo en mi tabla… calculo (como yo les decía) con External Object Geolocation que me devuelve cual es mi posición, me guardo esa posición en mi tabla, luego con el Evento ControlValueChanged voy a estar teniendo en esta variable “Destinationpoint” el lugar donde va a seleccionar el usuario y en el evento de usuario ingresé el destination en la tabla, apago la SelecionLayer y prendo la DirectionsLayer y hago un refresh.

Lo podría hacer de otras formas, es simplemente para fijar ideas para que vean cómo funcionan de alguna forma estas dos propiedades nuevas que tenemos.

Les cuento que también estamos trabajando (que va a salir en los primeros upgrades también) en otra capa que es la “AnimationLayer” que me va a permitir entre dos puntos o entre una línea, animar un ícono, el típico ejemplo en las Transport Application es cuando me muestra la flota de Taxis que tengo cerca o de Deliverys que tengo cerca, o me hace una animación desde donde yo estoy hasta donde voy (con un autito), son efectos y cosas de interfaz de usuario que son necesarias y las estamos atrayendo y generándolas en estas propiedades del grid.



Bueno ¿que otras cosas estuvimos haciendo a nivel de geografía?

Seguimos trabajando, desde hace tiempo estamos trabajando en tener más y mejores proveedores, los proveedores de cartografía antes para iOS teníamos Apple Maps, para Android tenemos Google, también Auto Navy y Baidu que es de un proveedor de China.

Lo que hicimos ahora es reescribir todas las funcionalidades en iOS para Google, es decir, en cualquier objeto main voy a tener una propiedad Apple Maps appi, que ahora voy a poder elegirle que dibuje Google o siga usando los mapas de Apple como hacía al inicio.

Nosotros vamos a seguir trabajando y sobre todo en área Web, (no lo puse acá) en el área Web nosotros la forma de dibujar, de tener cartografía Web es a través de un Map Control, es un control que ya tiene bastante tiempo, ahí estamos manejando algunos proveedores, pero estamos con el User Control Object que vieron hoy temprano, estamos haciendo un User Control que van a utilizar ustedes donde estamos de alguna forma reescribiendo el Map Control, básicamente con las mismas funcionalidades que tiene pero estamos intentando agregar nuevos proveedores, básicamente por el momento vamos a agregar los mapas de Open Street Map y también estamos trabajando con los mapas de ESRI.

- Una pregunta ¿Transportar, se puede cambiar en tiempo de ejecución?
- Sí, de hecho en el ejemplo este, en tiempo de ejecución yo cambiaba…
- Y otra cosa… ¿se puede usar la misma llave de Google Maps para Android y para iOS?
- A ver perdón, ¿la Key pero de qué, para qué? Porque acá hay dos Keys…
- Cuando habilito el main program me abre posibilidades con los porveedores de Mapas para Android y para iOS…
- No, es a nivel de objeto en realidad, viste que la tenes a nivel de main…
- Un objeto como main, puedo compliar para Android y para iOS.
- Tenés razón! No me acuerdo, pero supongo que sí porque no la tomamos de otro lado. Hay un detalle ahora que dijiste esto… hay un tema interesante, sobre todo con Google, Google ahora cambió las APIS, tiene APIS para dibujar mapas, tiene APIS para Directions que es lo que estamos usando acá y después tiene otra APIS de places, o sea, se paró en tres APIS, uno puede a su API darle permiso para cada una de esas cosas o podes tener tres APIS, nosotros hoy tenemos una sola, es decir que si vos queres mapas y además Directions, en tu API Key tenés que configurar habilitarle las dos cosas
- Sí, pero es una única llave?

- Es una única llave! Podríamos tener dos, porque vos podrías tener separado por un tema de costos, porque ahora hubo un cambio de política bastante grande en Junio de Google Maps, que pasó de tener 25 mil request por día a 28 mil por mes, cambió bastante la política y está haciendo bastante ruido, hay una movida grande con otros proveedores, pero bueno, hoy tenemos un API, por un tema de simplicidad en realidad, podríamos hacerlo pero … sí, a tu pregunta sí, se puede usar las dos
- Gracias!

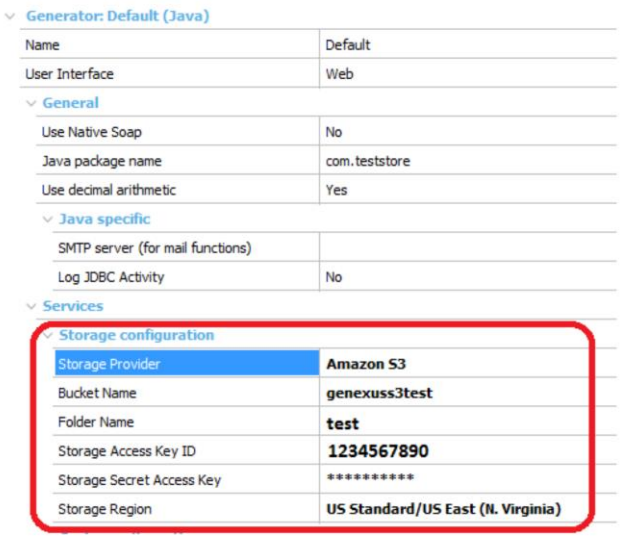
Bueno, a nivel de Geografía no mucho mas.

Cloud External Storage…

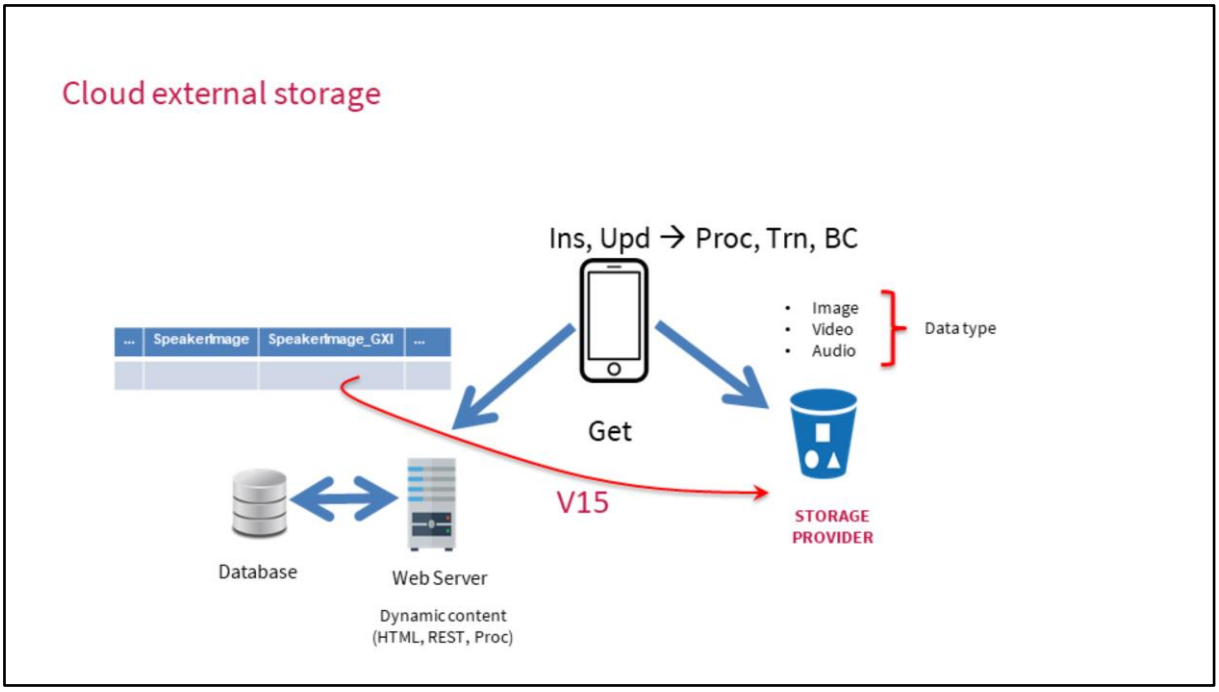
Cloud external storage

Storage Provider Property

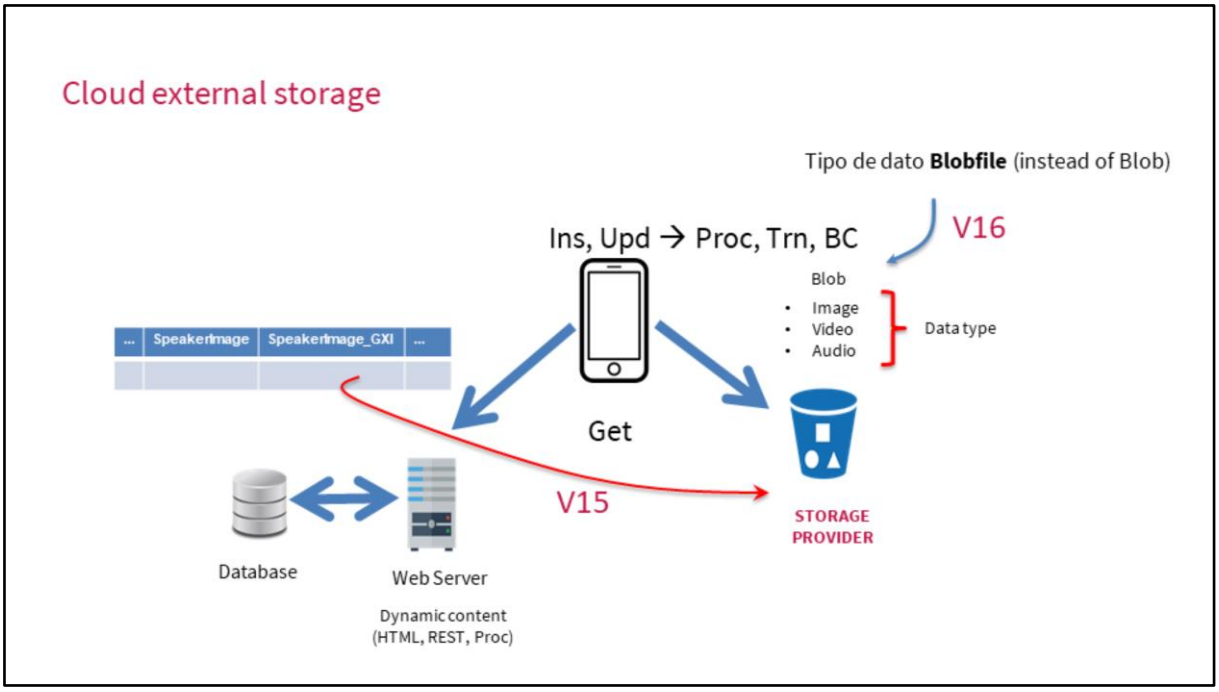
- AmazonS3
- Micrsoft Azure
- Ibm Bluemix
- Google Cloud Storage
- Local (default)
- OpenStack



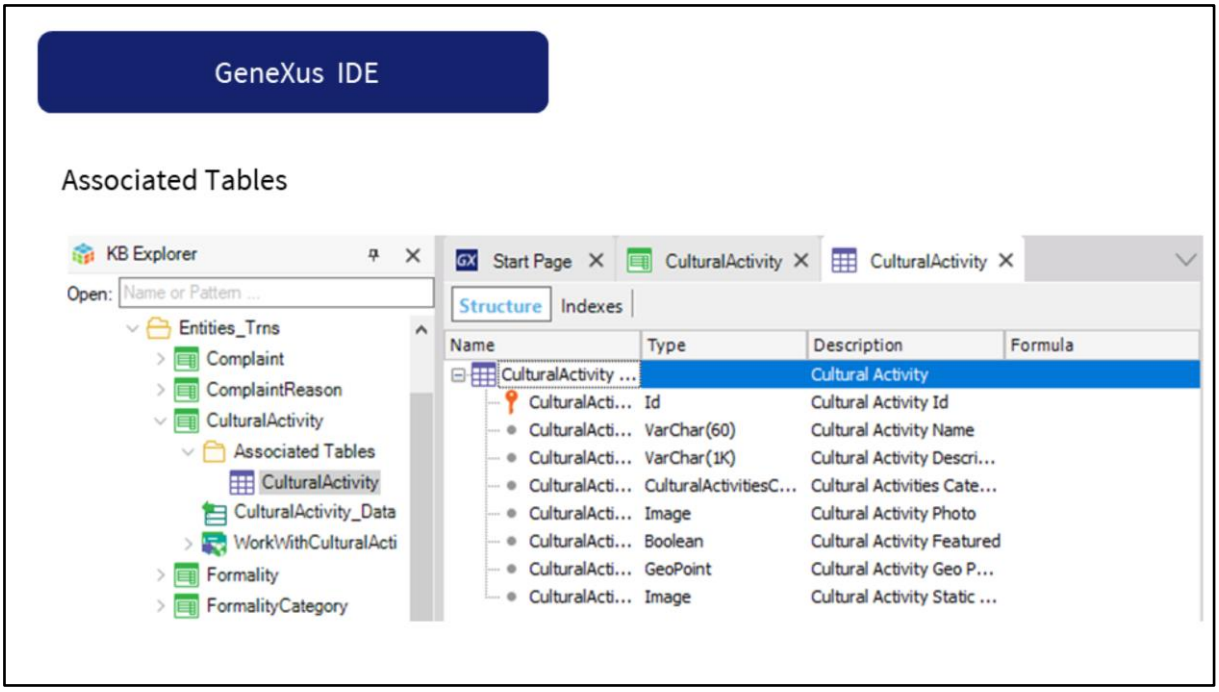
Esto es para almacenar externamente, es algo que liberamos en la versión 15, tenemos distintos proveedores de Amazon, Microsoft Azure, Bluemix, que me permiten almacenar archivos en un repositorio externo, en mi aplicación.
Y básicamente si queremos usar esto, lo que tenemos es una propiedad que es el Storage Provider Property, le digo qué proveedor de estos voy a usar, por defecto tenemos “Local”, es decir, almaceno todo lo que sean videos, audio, imágenes, los almaceno locales pero si cambio alguno de estos, tengo que darle las credenciales de la cuenta que tenga y almacenarlos en la nube.



Entonces, en la V 15 lo que teníamos era esto, teníamos el almacenamiento local y ahora tenemos la posibilidad de guardar en el Storage Provider, ya sea Image, Video o Audio.

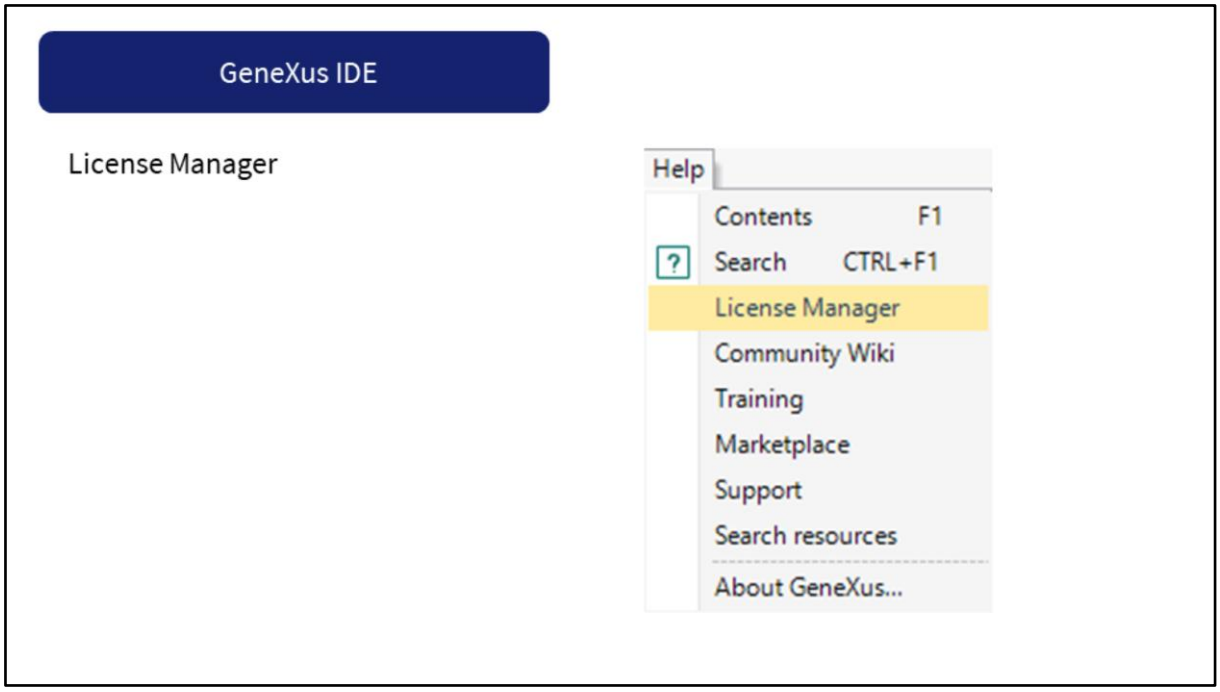


A partir de la V 16 lo que tenemos es que los Blob también los podemos almacenar ahí.
Lo que tuvimos que hacer para esto es algo parecido a lo que hicimos con las Image, creamos un tipo de datos nuevo que se llama “Blobfile” que es muy parecido (para fijar idea) al Blob, la única diferencia es que se almacena distinto, entonces si uno va a usar un External Storage, lo tiene que pasar al tipo de datos Blobfile.

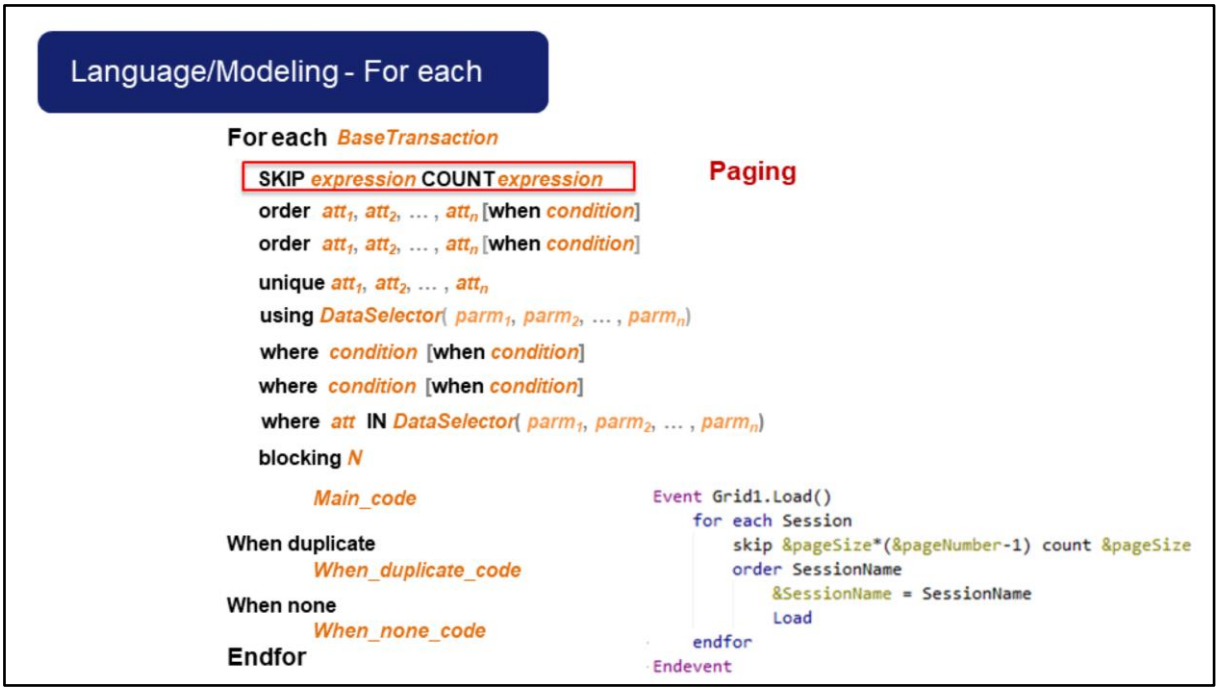


Algunas cosas más que nada de visualización del ID de GeneXus, hay unos miscellaneous que queríamos contarles, una por ejemplo, es que ¿cómo se ve ahora una tabla asociada a una transacción?

Yo creo la transacción y tengo como un folder “Associated Tables” y ahí es donde realmente veo la tabla, no es un objeto suelto como teníamos antes.



Tenemos acceso al License Manager desde la opción de “help” del IDE.



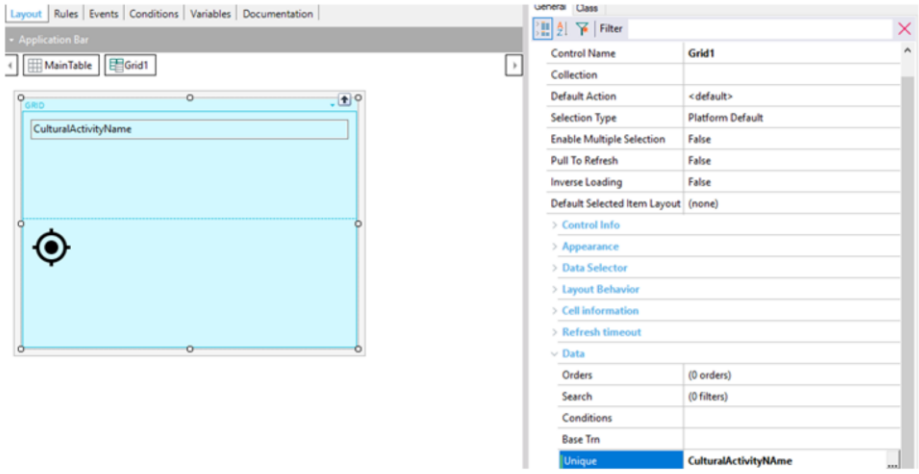
Esta es interesante, que en el comando “For each” tenemos dos clausulas nuevas, la “SKIP” y la “COUNT”, están pensadas para hacer un paginado.

Entonces el “SKIP” me dice cuántos registros quiero yo saltearme y después cuántos voy a mostrar se los digo con el COUNT… (La próxima página cuántos va a tener)
Le digo “en este for each salteame “x” registros, lo pongo “SKIP X” y quiero mostrar 10 registros, pongo “COUNT 10”.

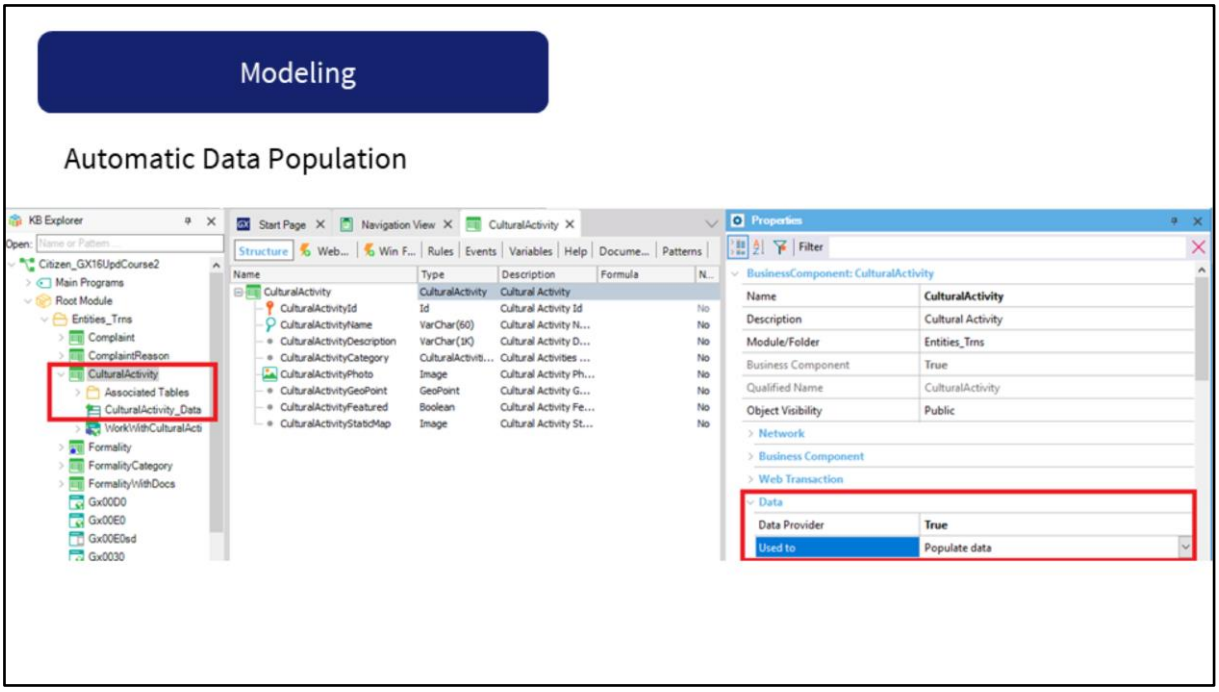
- Pregunta, ¿Cómo se implementa eso en los motores de base de datos?
- Yo he visto los Select, hay cláusulas y te dice cómo hacer un paginado…
- Sí, cada base de datos tiene sus cláusulas… vas a ver la sentencia SQL que se genera diferente para justamente lo que va a recuperar es solamente ese grupo de registro de la base de datos, o sea, lo que te trae de la base de datos es esa página…
- Eso ya lo ves en el código generado o prendiendo un log se ve la sentencia tal cual…
- Ya estaba esto, es algo que ya estaba en el Data Provider y ahora se agregó al For each.

Modeling

Unique Grid Property



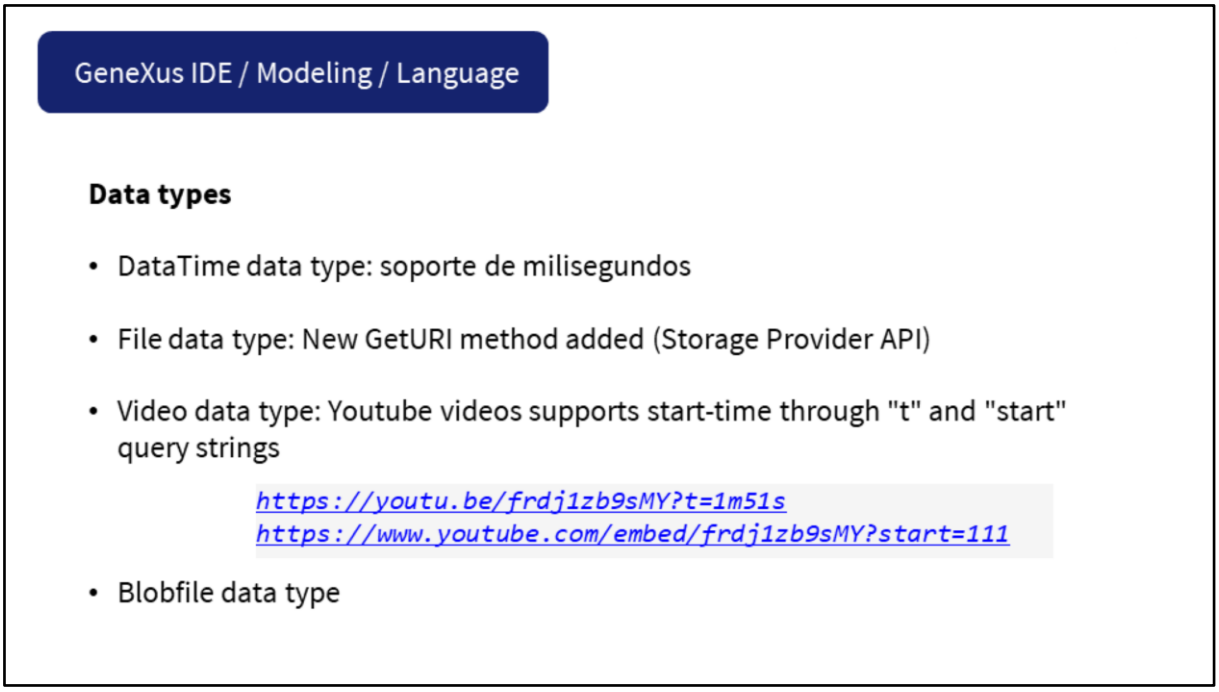
Esto también es algo que teníamos en el For each, un tema de completitud básicamente, lo teníamos en el For each, la clausula “Unique”, ahora la tenemos a nivel de Grid, o sea que tengo la propiedad y le puedo decir que atributo quiero que sea único.



Y esto también es algo a nivel de transacciones, cuando nosotros seteamos la propiedad Data Provider in true y tenemos para popular datos, automáticamente se nos crea un Data Provider que nosotros podemos programar y este se llama automáticamente cuando se ejecuta, cuando una da F5, (cuando uno hace la creación), se carga en la KB de ejemplo que están viendo, ya usamos esa característica, lo pueden ver ahí cualquier duda.

¿El llamado es automático o no?
-Lo que pasa que éste curso arranca desde GeneXus 15 U0 hasta GeneXus 16, con esto de que cambia cada 2 meses, tenemos un upgrade, por eso es que pueden ver cosas repetidas que ya habían visto en GeneXus 15.

La mayoría de estas cosas son del upgrade de la 15, que en realidad no se liberaron con la 15.



El tipo de datos Date Time ahora tiene soporte de mili segundos, eso lo hicimos en el U11 (acá le pusimos los Upgrade)

El tipo de datos File data type, para todo esto que estábamos hablando del External Storage tenemos una “Storage Provider API” que me permite hacer el manejo de los archivos que están en el repositorio, en la nube y para eso teníamos la necesidad de crear un nuevo método que es el GetURI.

Y después esto que en realidad es para Android, que también se creó y me permite a un video, si yo en la URL uso éstas variables que me permiten decir dónde “inicia”, las toma en cuenta.
Es decir, puedo reproducir un video, no desde el inicio sino desde una posición específica.

El Blobfile data type que les decía, es igual al Blob pero en la forma que se almacenan los datos es distinta y es lo que precisan los External Storage.


GeneXus™
The power of doing