Modalidad Offline

Bienvenidos

Mi nombre es Martin Torrado, trabajo en el equipo de soporte de Artech y en este webinar vamos a estar hablando de aplicaciones SD offline para la nueva versión de X Evolution 3.

Entonces vamos a comenzar por lo básico, es decir a ¿qué le llamamos aplicaciones offline?

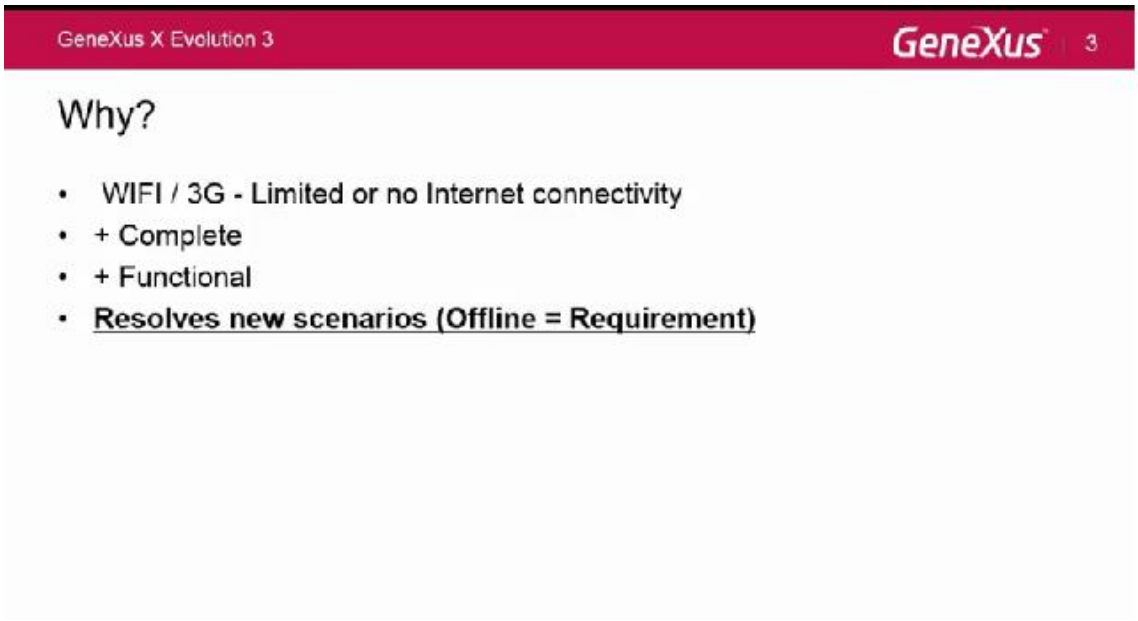
Las aplicaciones offline son aplicaciones parcialmente conectadas, son aplicaciones autónomas.

¿Qué quiere decir esto?

Quiere decir que, si yo estoy utilizando mi aplicación, en mi devices conectado a internet - y yo pierdo esa conexión a internet - la aplicación va a seguir funcionando y eventualmente cuando yo retome la conexión a internet se va a generar un proceso de sincronización con la base de datos del servidor centralizado.

Entonces decimos que son aplicaciones que funcionan en todo momento, ya sea que tengan o no tengan conexión, y que acceden a datos locales con la posibilidad de ejecutar lógica compleja del lado del dispositivo.

¿Por qué hacemos este tipo de aplicaciones?



Las hacemos precisamente por lo que decíamos antes, porque no siempre vamos a contar con escenarios en los cuales dispongamos de conexión a internet. Por lo tanto yo voy a poder estar en un lugar en el cual no consiga conexión a internet y la aplicación va a tener que seguir funcionando.

Entonces decimos que hacemos aplicaciones más completas y aplicaciones más funcionales. Y sobre todo - yo en las ppts lo subrayé y lo puse en negrita porque me parece muy importante – es que resolvemos un abanico muy grande de nuevos escenarios en los cuales, que las aplicaciones sean offline, ya deja de ser algo deseable y pasa a ser un requerimiento del sistema.

Ejemplos tenemos miles. Por ejemplo: aplicaciones de fuerza de venta en las cuales mi venta no puede depender si mi dispositivo está conectado o no está conectado a Internet.

Entonces ya sabemos que son las aplicaciones offline y porque las hacemos

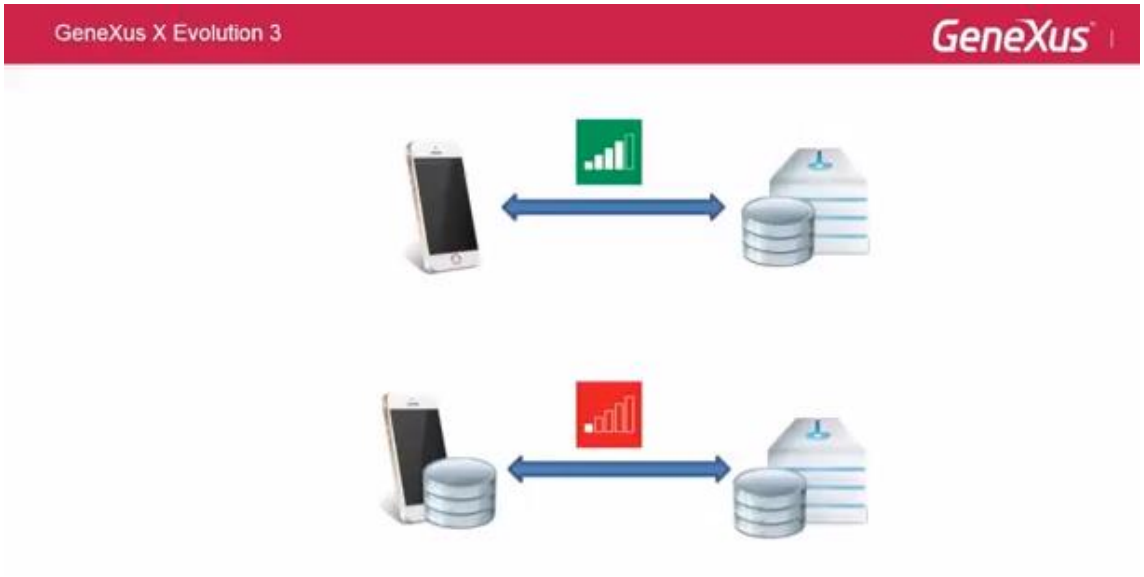
Ahora lo que les quería comentar es un poco sobre la Arquitectura…

Y que mejor para hablar de arquitectura, que comprar con un modelo que nosotros ya conocemos que es el modelo de la arquitectura aplicaciones de SD en la XEv2.

Si recuerdan, lo que nosotros teníamos hasta en la XEv2 es nuestro Devices con la aplicación instalada que se conecta a través de una capa de servicios rest a nuestra base de datos de servidor centralizado, y se conecta para nutrirse de datos y para ejecutar “Insert, update, deleted” del lado del servidor.

Básicamente esa era la arquitectura que nosotros veníamos trabajando hasta la XEv2.

Y si bien hay muchas cosas que se mantienen en la Arquitectura de la XEv3, nosotros ahora vamos a contar con una base de datos del lado del cliente.



Una base de datos sqlite que ya viene pre instalada tanto en Android como en IOS, es un motor de base de datos que viene pre instalado en Android y en IOS, entonces nosotros tenemos esta base de datos del lado del cliente porque tenemos más requerimientos de procesamientos de datos del lado del cliente.

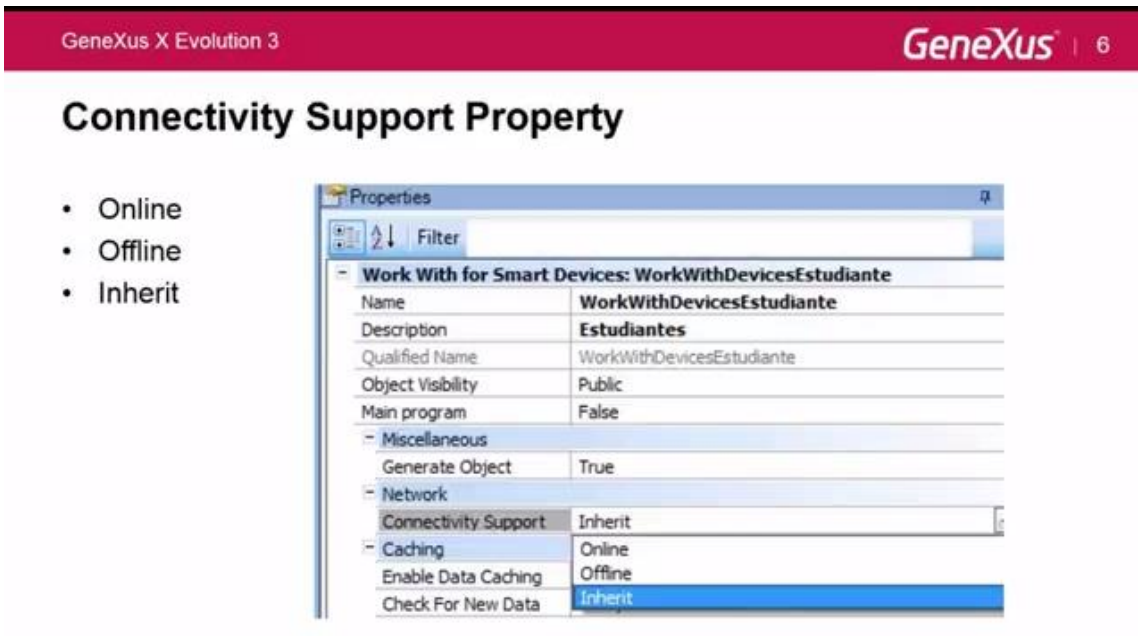
Entonces se va a ejecutar lógica mucho más compleja del lado del cliente, y vamos a necesitar entonces generar código nativo que va será JAVA en Android y Objective-C en IOS

Lo que tenemos también es hablar un poco de los eventos Start / Refresh / Loand se que ejecutaban antes del lado del servidor en la arquitectura online, y ahora pasan a ejecutarse del lado del cliente accediendo a datos locales.

Los eventos de usuario siguen siendo interpretados solo que las aplicaciones offline, si llaman a un objeto offline, lo hacen de forma local y no precisan ir al servidor para ejecutarlo.

También una cosa destacable en esta arquitectura es que tanto la base de datos del cliente como la base de datos del servidor se van a tener que mantener consistentes, se van a tener que mantener coherentes una con la otra, y eso lo hacen a través de un proceso que nosotros llamamos “proceso de sincronización” que lo vamos a ver ahora, más adelante en esta charla.

Pasemos a ver cómo es que hacemos esto nosotros en GeneXus.



Nosotros contamos con una propiedad que se llama Connectivity Support que está a nivel de todos los objetos SD. Tiene 3 opciones:

- Online
- Offline

- Inherit (o heredar)

Y ¿qué es lo que hacen estas tres opciones?

La online es la default en un objeto SD main, y es como nosotros veníamos trabajando hasta en la XEv2, con nuestra aplicación totalmente conectada. La propiedad Inherit viene por default en todos los objetos SD que no sean main, es decir que van a heredar el comportamiento del padre que lo llama. Y la propiedad offline es la que hoy más nos interesa.

¿Qué es lo que pasa cuando yo a un objeto pongo la propiedad con connectivity support offline?

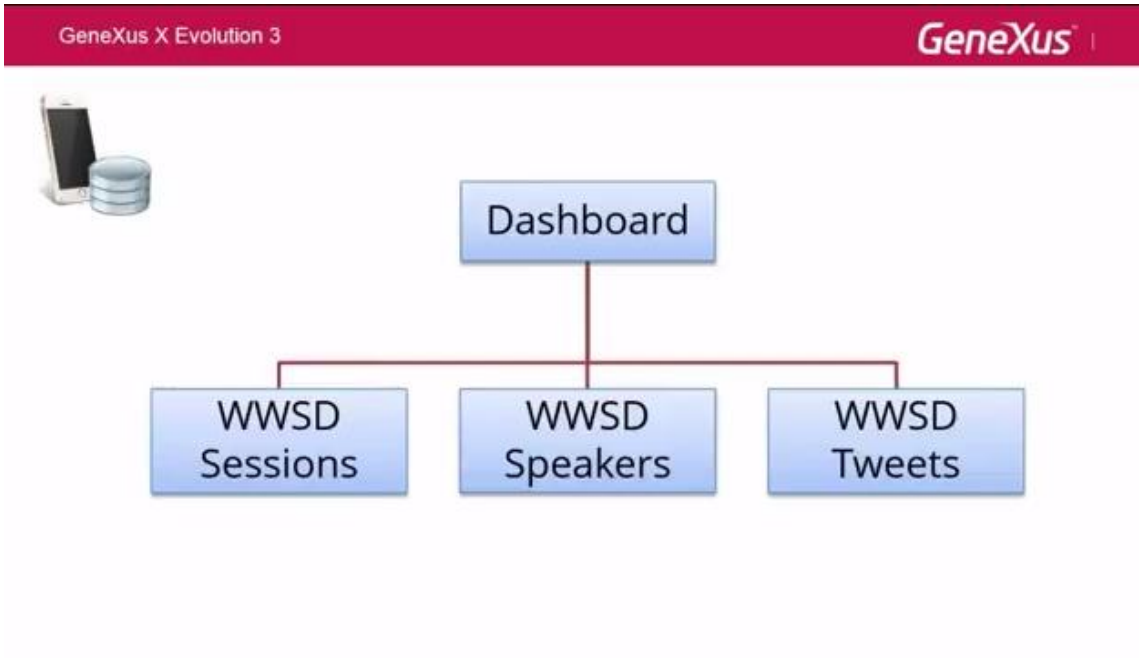
Supongamos que tenemos una aplicación para un evento por ejemplo, y tengo un dashboard (que lo tengo seteado como main) al cual le pongo la propiedad connectivity suppprt offline.

¿Qué es lo que pasa inmediatamente después? La próxima vez que yo ejecute o compile se va a crear un nuevo objeto que se llama Offline DataBase Object.

Es el encargado de determinar cuáles son las tablas que van a la base de datos del cliente y también cuales son los datos que se llevan cuando se sincroniza. Porque obviamente la base de datos local no va a ser la misma que la base de datos del servidor, va a ser una versión reducida.

Entonces ¿cómo determinamos cuales son las tablas que se van a llevar la base de datos de un cliente?

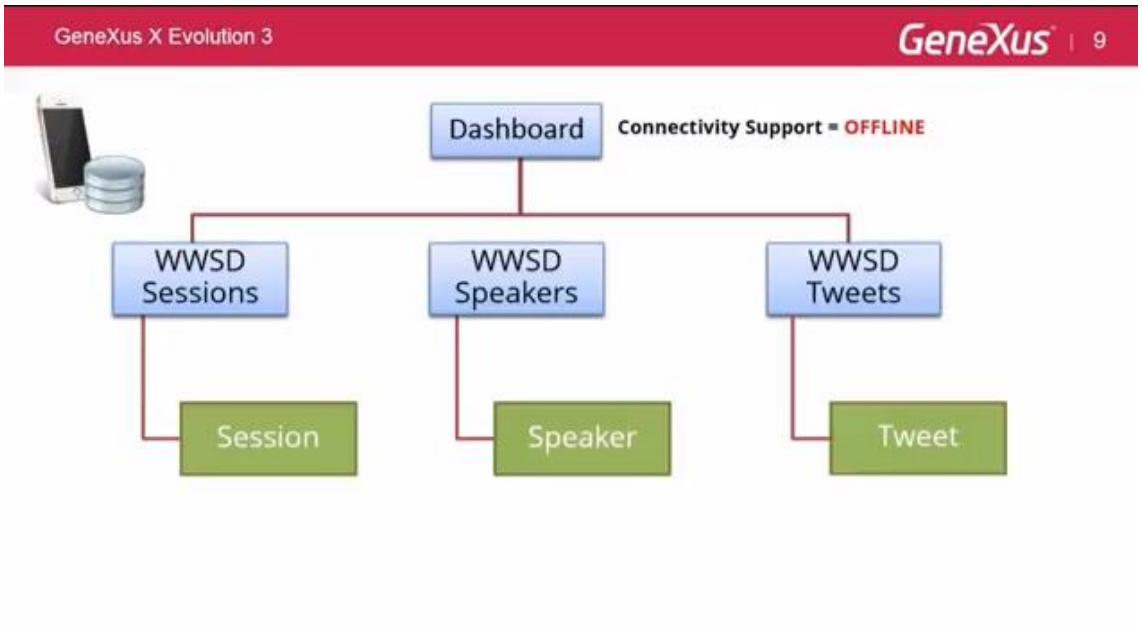
Les traje este ejemplo para que este concepto quede bien claro, y es siguiendo el hilo de lo que veníamos hablando, por ejemplo una aplicación para un evento.



Tengo un dashboard que llama a un work with sesiones, un work with speakers y a un work with tweets.

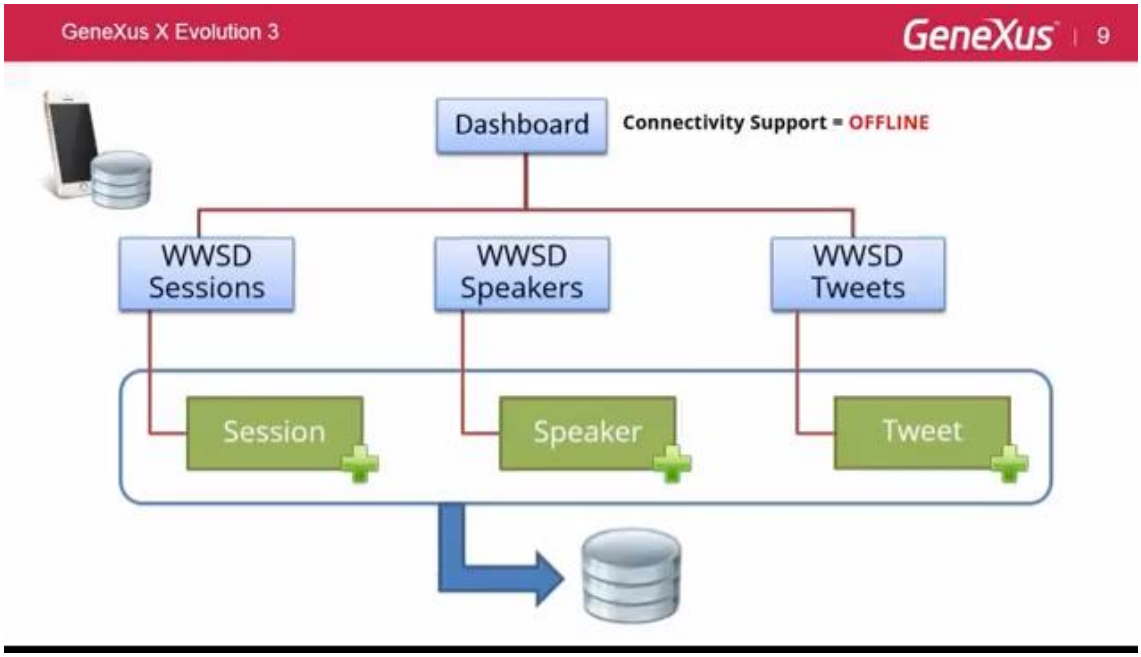
Entonces, ¿qué tablas van a ir a parar a la base de datos interna del Devices?

Yo a mi dashboard le pongo Connectivity Support offline, que mi dashboard como deciamos es un objeto SD main que por default va a venir con la propiedad connectivity support online, entonces yo se la cambio y se la pongo a offline. Le digo que la aplicación a partir de ahora yo quiero que funcione sin conectividad.



Entonces si recordamos, tenemos el objeto work with Sessiones, speaker y tweets. La propiedad que van a traer esos tres paneles por default va a ser inherit, es decir heredar el comportamiento del padre que lo llama en este caso es el dashboard que tiene la propiedad offline.

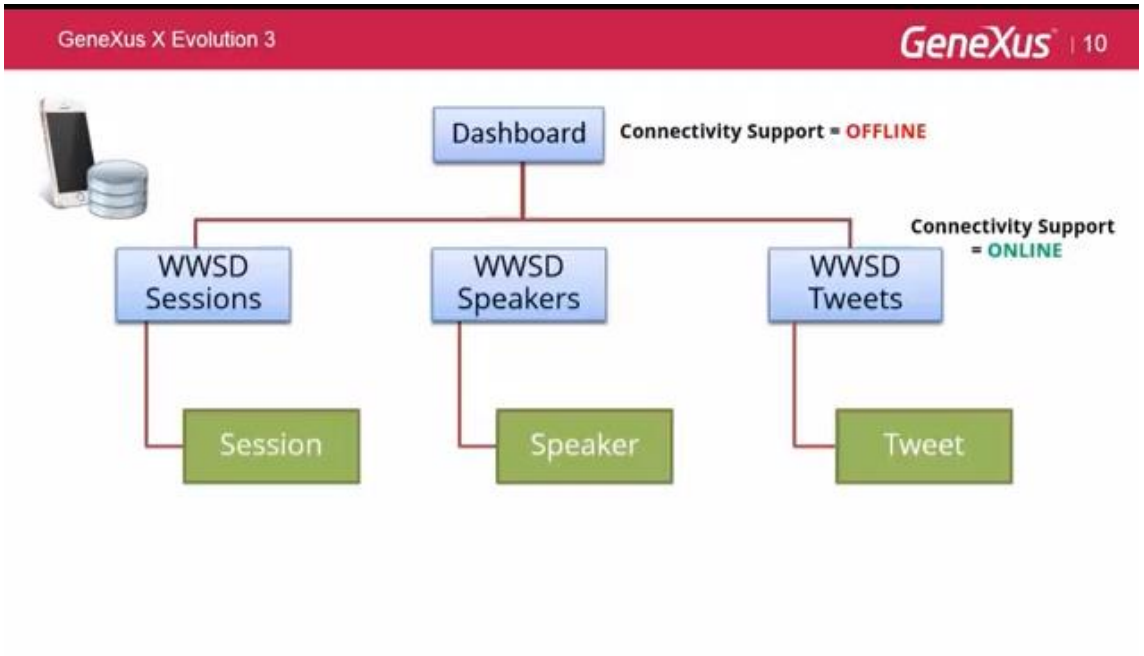
Por lo tanto ellos van a heredar ese comportamiento offline…



y se van agregar a las tablas que acceden, se van agregar a la base de datos del device; es decir se van agregar la tabla sesiones, la tabla speaker y la tabla tweets.

Y que pasa si yo quiero que los tweets se vean solamente con conectividad a internet porque quiero que siempre el usuario lo vea actualizados, que tenga conexión a internet cuando quiera ver los últimos tweets.

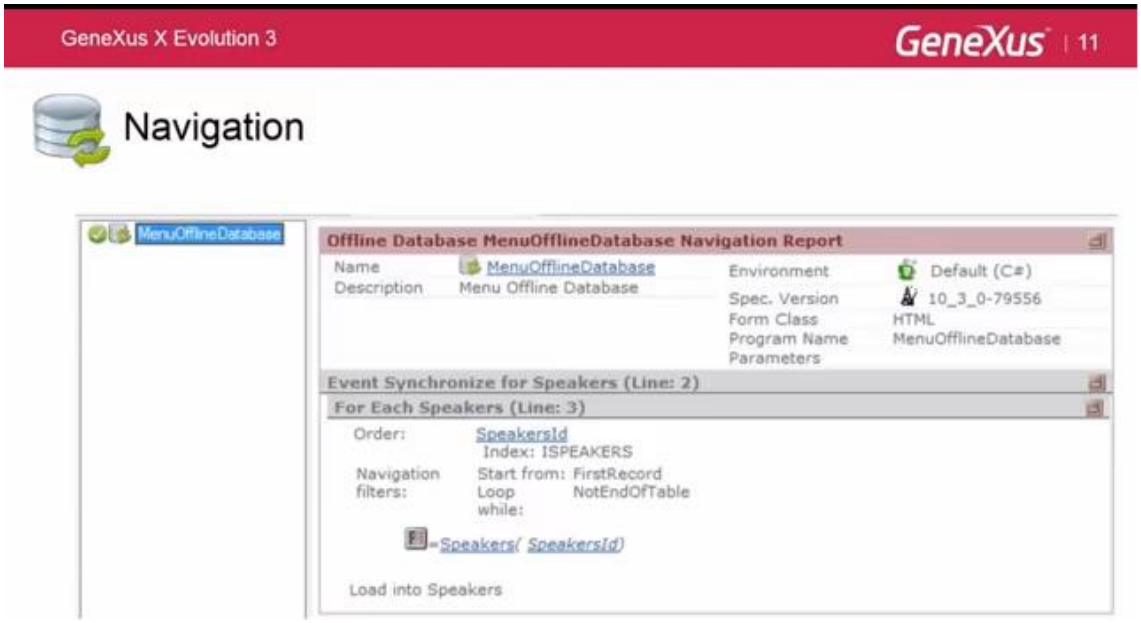
Entonces lo que hago es jugar con la propiedad connectivity support y al panel work with SD le pongo connectivity support online



Entonces ahí yo estoy obligando que los tweets se vean solamente cuando hay conectividad. Por lo tanto las tablas ahora se van a llevar al devices van a ser solamente sesión y speaker.

Cabe aclarar también que si una tabla tiene una Foreing key a otra tabla, la tabla referenciada también se va a llevar al dispositivo.

Entonces vamos a ver ahora la navegación de este objeto offline Database.



Como pueden ver tiene un evento Synchronize para cada una de las tablas involucradas y en cada uno de esos eventos, que es un for each básicamente a la tabla, vamos a ver cuáles son las condiciones que aplica.

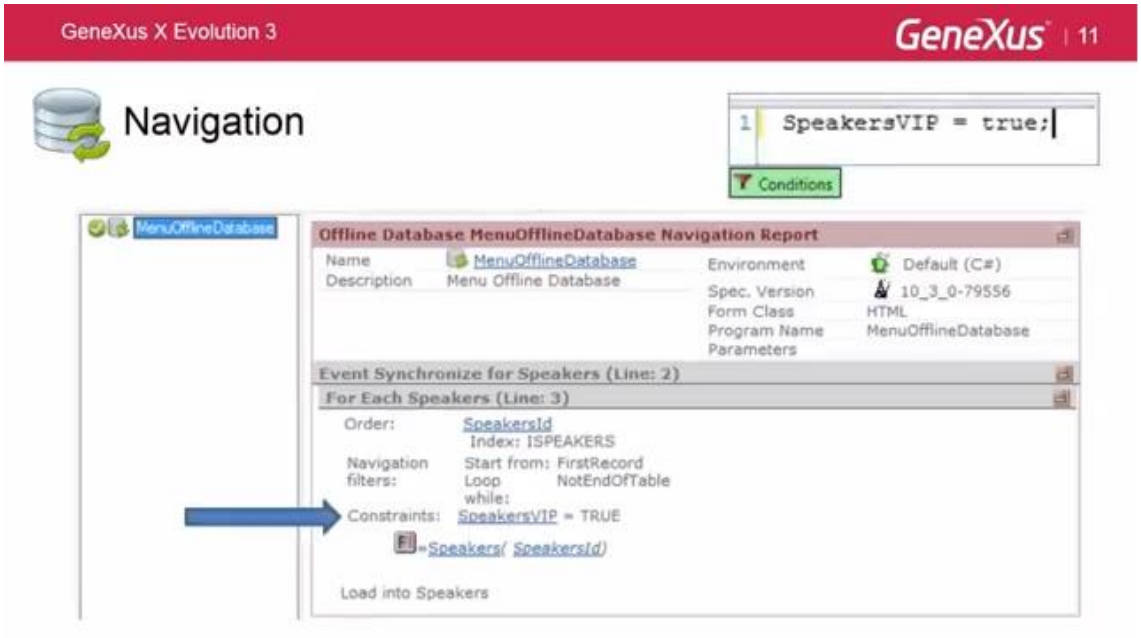
Las condiciones se utilizan como cualquier otro objeto GeneXus. Se aplican a las tablas para saber qué datos se llevan al dispositivo.

¿Por qué se aplican estas condiciones?

Bueno, puede ser por temas de seguridad que yo no quiero que mi usuario vea todos los registros, o por tema de capacidad, de memoria del device. Yo no puedo llevar todos los registros de la base de datos del servidor a mi devices porque quizás son miles de registros y supera la capacidad de mi devices.

Por esas dos razones nosotros vamos a poder aplicar filtros. Como vimos en las ppts anteriores, nosotros vamos a poder filtrar por tablas, es decir podemos diferenciar o discriminar por tablas cuales son las tablas que yo quiero que se lleven de la base de datos del servidor a mi devices pero también, como vamos a ver ahora podemos filtrar por registros.

¿Cómo hacemos esto?



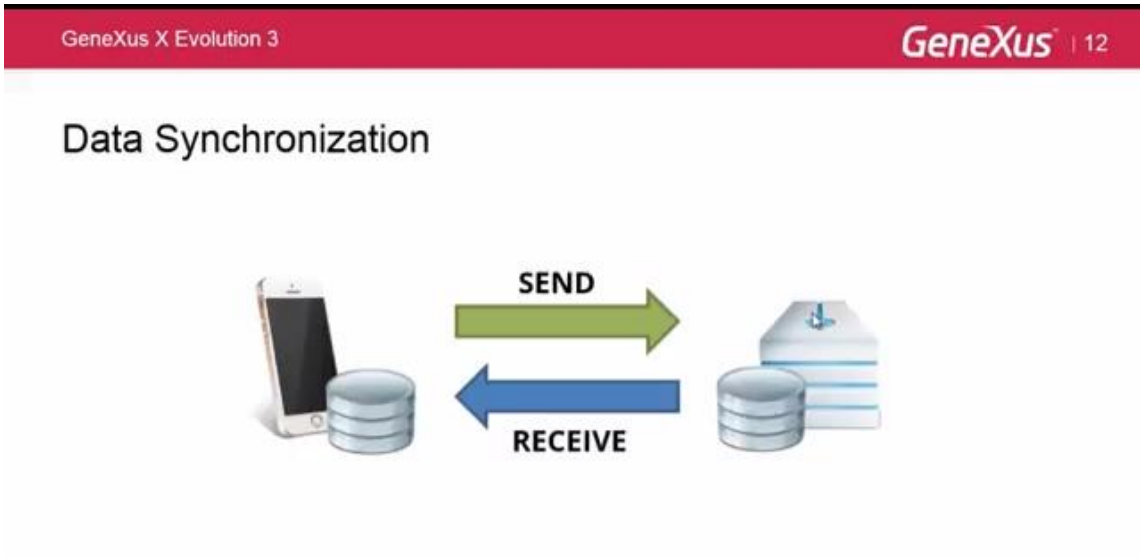
Agregamos una condition - supongamos con el mismo ejemplo que venimos trabajando- que yo quiero que los speakers de mi evento solo quiero que me traiga al device los speakers que para mi son importantes.

Es decir los speaker vips = true

Entonces yo agrego en la solapa CONDITIONS esa condición y como pueden ver se me agrega una Constraints a la navegación del objeto que va a seguir haciendo ese for each a la tabla de speakers pero a su vez le agrego una constraints en donde los speakers vips igual true… quiere decir los speakers importantes.

Una nota importante que vale la pena aclarar es que no tenemos reorganización, es decir la base de datos hace un create cada vez que modificamos el modelo de datos.

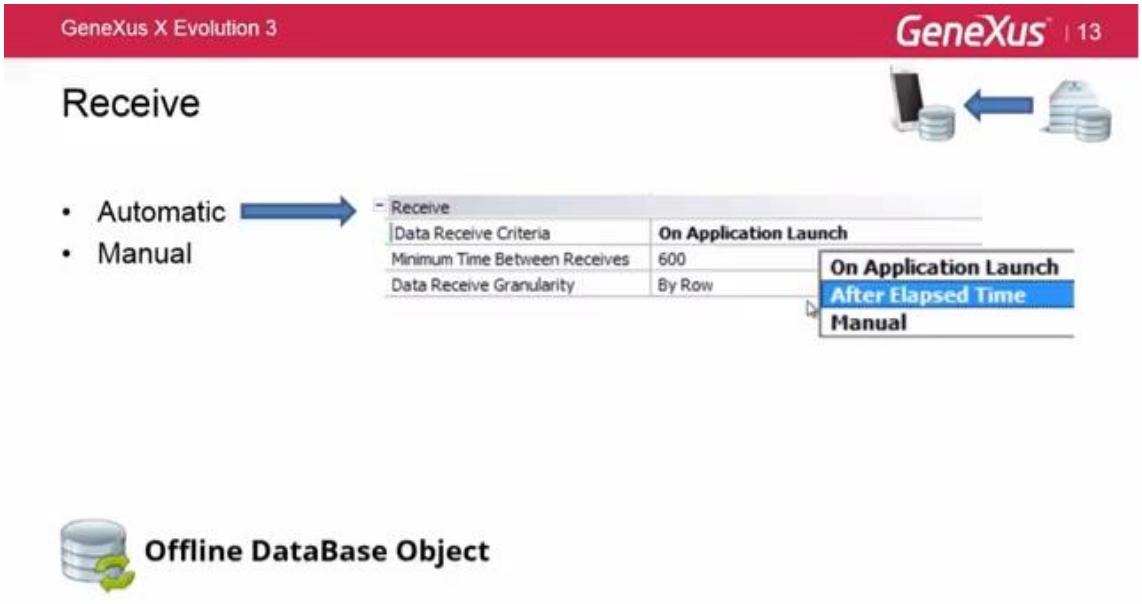
Ahora vamos hablar un poco sobre la sincronización de datos.



Es una sincronización que va a tener un proceso de enviar datos desde la base de datos de nuestro devices al servidor y un proceso que recibe datos del servidor a la base de datos del devices, es decir a la base de datos del cliente.

Entonces empezando por el proceso de recibir datos, hay dos formas de sincronización: Automática y Manual.

De la forma automática tenemos las siguientes propiedades que están a nivel del objeto offline Database.



Tenemos la propiedad Data Receive Criteria que por default va a venir en una on application launch pero también tiene la propiedad After Elapsed Time y manual, que la vamos a ver más adelante.

¿Qué pasa cuando yo seteo Data Receive Criteria con la propiedad On Application Launch?

Lo que va a pasar es que yo voy a recibir datos cada vez que la aplicación hace un launch, es decir cada vez que yo levante la aplicación y además haya pasado el minimum time between receives, que es la segunda propiedad que actualmente esta seteada en 600 segundos, es decir van a pasar 10 minutos. Cuando yo abra la aplicación y a su vez hayan pasado 10 minutos la aplicación va a ser Receive de datos.

Que pasa cuando yo seteo la propiedad Data Receive Criteria en After Elapsed Time, esta propiedad es complementaria a la anterior quiere decir, se va a efectuar un receive de datos cada vez que hayan pasado 600 segundos desde la última vez que recibi datos por lo tanto es complementaria a la propiedad anterior.

Y este recibe datos se puede dar de dos formas, que es la última propiedad que Uds. Ven en pantalla, que es el Data Receive Granularity. Se puede dar by Row o la otra opción que es by table.

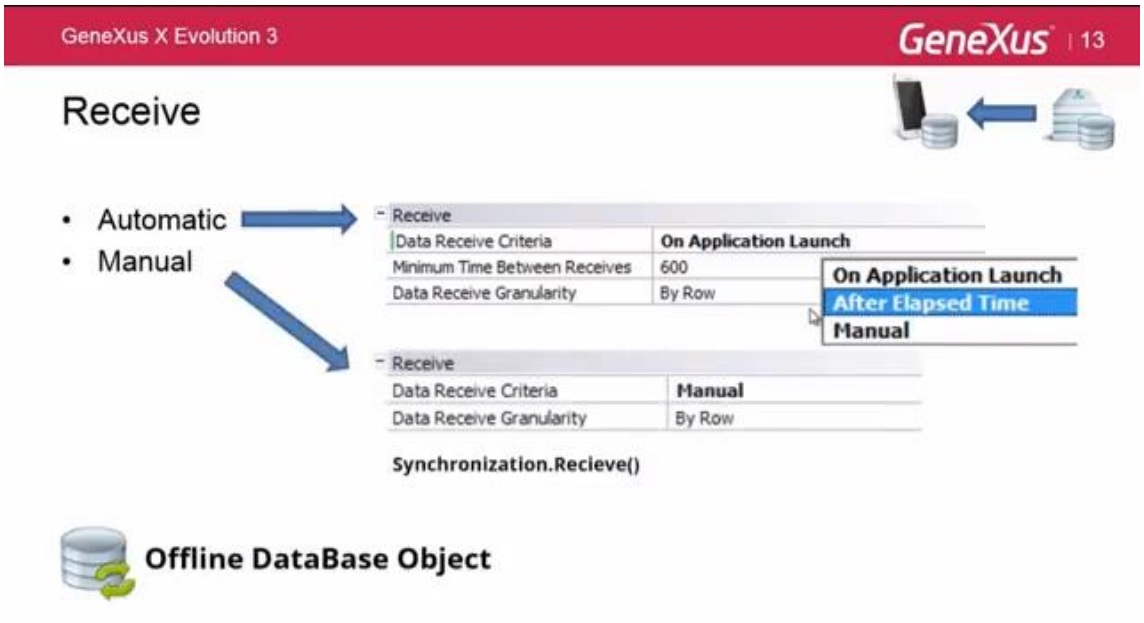
¿Cuál es la diferencia entre estas dos?

By Row lo que tiene es, me va a enviar los datos del servidor a la base de datos del cliente solamente los datos modificados, es decir solamente las rows modificadas. Y by table lo que va a hacer es cada vez que haya alguna modificación me va a enviar toda la tabla. Me va a enviar la tabla íntegra, la va a borrar del lado del cliente y me va a traer de vuelta toda la tabla.

Esto tiene ventajas y desventajas. El by Row tiene la ventaja de que la transferencia de datos desde el servidor al cliente es poca, porque solo me va a estar enviando los datos modificados. Pero la desventaja que tiene es que hay mucho procesamiento del lado del servidor porque tiene que buscar cuales son esos datos que fueron modificados.

En cambio en by table es al revés. Tiene poco procesamiento del lado del servidor porque manda toda la tabla cada vez que hay una modificación pero hay muchos datos en la transferencia porque precisamente está enviando toda la tabla del servidor al cliente.

Y luego teníamos como decíamos la propiedad manual.



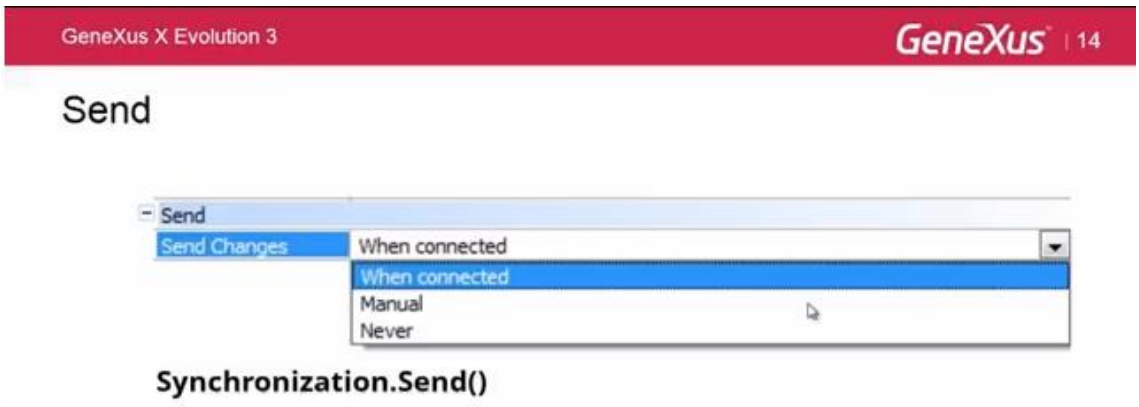
Que es que nosotros vamos a decidir cuándo se hace Receive de Datos.

Como ven cuando seteamos la propiedad en Manual, la segunda propiedad que teníamos antes que se llamaba mínimum time between receive desaparece porque ya no tiene sentido el tiempo de espera entre los receives, porque los voy a ejecutar yo manualmente.

Y ¿cómo lo ejecuto manualmente? Como muestra en pantalla, lo ejecutamos con Synchronization.Receive.

Vale destacar que yo si yo seteo la propiedad en automático, es decir con Application Launch o After Elapsed Time, eso no quita que también pueda utilizar la sincronización manual, yo también voy a poder utilizar, a parte, el Synchronization.Receice en eventos de usuario, en algún botón, etc.

Esto en cuanto a recibir datos.



Ahora vamos hablar un poco sobre el Send de Datos, es decir como yo enviado datos desde la Base de datos local desde el Devices a la Base de datos del servidor. Para eso contamos con la

propiedad a nivel del objeto offline data base object que se llama Send Changes y tiene como pueden ver en pantalla estas tres opciones:

La opción por default que es When Connected que lo va a hacer es enviar datos a penas yo consigo conexión internet en mi Devices. Yo estoy trabajando de forma offline y cuando yo consiga conexión a internet automáticamente se envían esos datos al servidor.

Luego tenemos la opción Manual que lo que hace es homologa al receive de datos. Utilizando el método Synchronization.Send envio, cuando a mí me parezca a nivel de programación, envio los datos del cliente al servidor. Puedo tener la opción When Connected seteada y utilizar igualmente la forma manual de mi programación, utilizar este método Synchronization.Send

Y por último la opción Never, que es que nunca mandamos datos del cliente al servidor. Una utilización de esta propiedad puede ser por ejemplo en una aplicación de finanzas personales en la cual la sensibilidad de los datos es tal que yo no quiero que nunca se envíen esos datos del cliente al servidor.

Antes de terminar con esta ppt quería hacer dos aclaraciones. La primera es que cuando nosotros creamos una aplicación offline en GeneXus la instalamos en el Device, se va a crear una tabla que se llama GXPendentEvent en la cual se van a guardar todas las modificaciones que yo hago de manera desconectada, es decir cuando yo estoy desconectado en mi aplicación en internet, se van a guardar todas las modificaciones que yo voy haciendo. Y en el mismo orden que yo las voy haciendo para yo luego poder – cuando consiga conexión a internet – replicarlas en el mismo orden en la base de datos del servidor para que justamente esas bases de datos queden consistentes entre sí.

Además esa tabla- la GXPendentEvent - vamos a tener asociado a cada modificación un estado al cual vamos a poder acceder a través de la Synchronization Event Api, que es una api que nosotros disponibilizamos para ver si hubo errores en el envio de datos y tomar decisiones pertinentes en cada caso.

Se podría decir que con esta api podemos tomar decisiones o hacer un manejo de errores en la sincronización.

Otra de las aclaraciones que quería hacer es que todo lo que yo quiero que se sincronice de mis aplicaciones offline lo tengo que hacer con business component, no se toma en cuenta ni los news, ni los for each.

Hecha estas dos aclaraciones pasamos a ver ahora los conflictos en las sincronizaciones.

Es un caso específico en el cual nosotros tenemos dos Devices con la misma aplicación instalada, y los Devices están por el momento desconectados a internet.

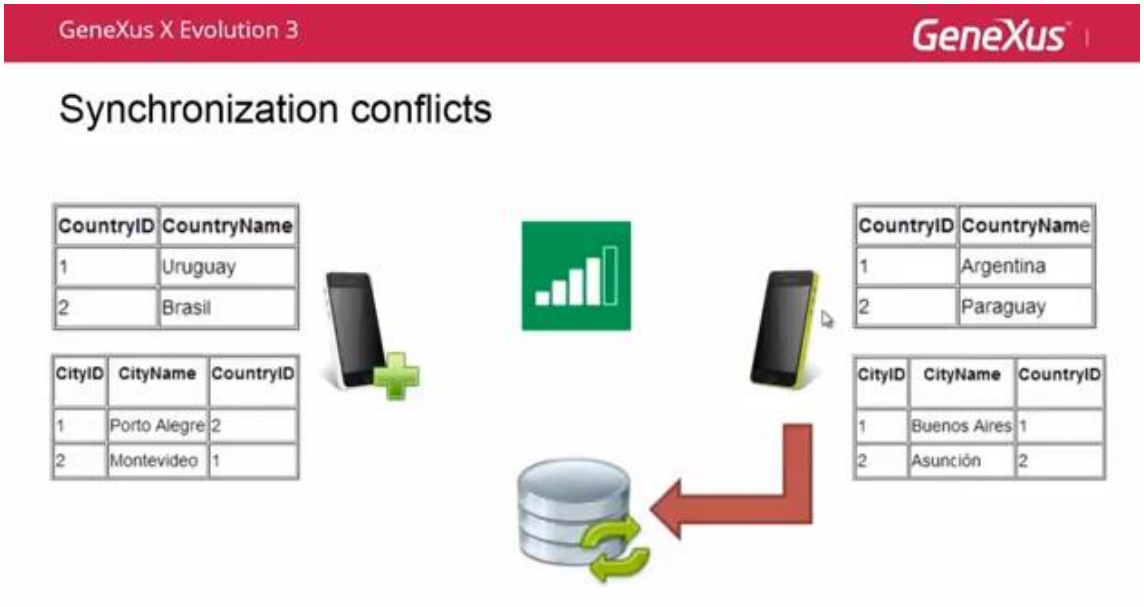
También tenemos una base de datos centralizada en la nube que en una etapa inicial no tiene datos.

Entonces el Device1 hace un insert del país Uruguay y del país Brasil con sus respectivas ciudades - Porto Alegre y Montevideo - en las tablas de country y en las tablas de city.

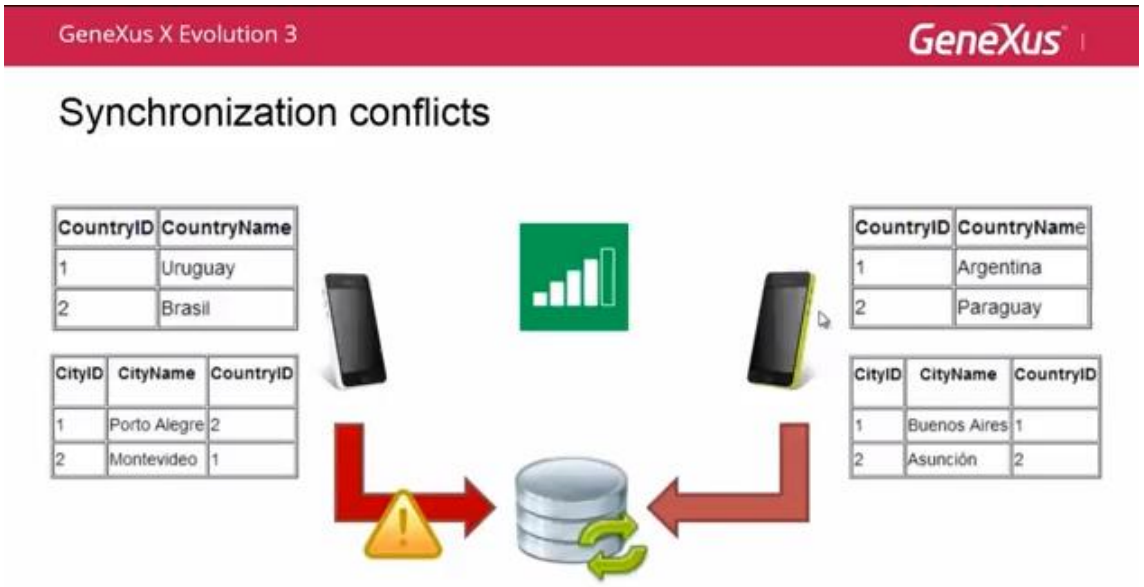
Hay que destacar que es muy importante que tanto las claves de CountryID como la de CityId con autonumeradas. Esto es muy importante para este caso específico para el manejo de conflictos.

Entonces como decíamos, el Device1 hace ese insert de Uruguay-Brasil y el Device2 hace un insert de Argentina y Paraguay en las mismas tablas, también con las claves CountryId y CityId autonumeradas.

¿Qué pasa cuando esos dos Devices consiguen conexion a internet y quieren sincronizar con el servidor de base de datos centralizado?

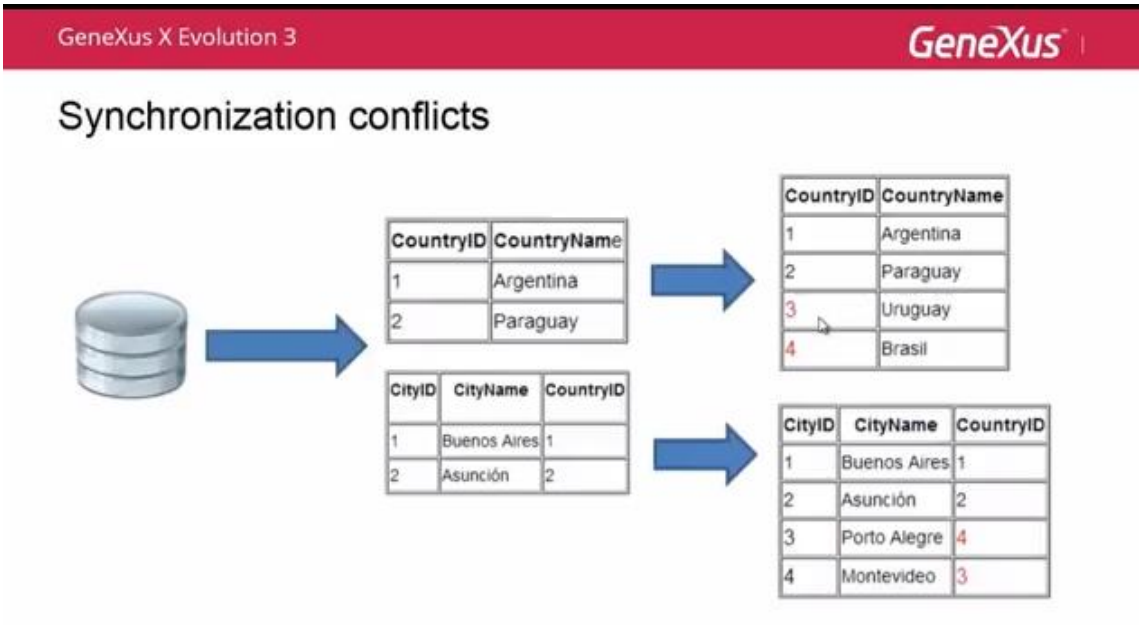


Supongamos que el Device2 es el primero en conseguir la conexión y sincroniza los datos con el servidor. Entonces va a enviar “Argentina-Paraguay” “Buenos Aires-Asuncion”, los va a enviar a las tablas del servidor.



Y cuando el Devices1 quiere hacer la sincronización se va a generar un conflicto. ¿Por qué´? Porque está tratando de hacer un insert en las mismas tablas con los mismos ID. Es decir, cuando va a querer Uruguay con el CountryID1 se va encontrar que ya existe un CountryID1 que es Argentina que fue el ingresado por el Device2.

Entonces… ¿cómo se soluciona ese conflicto?



Lo soluciona automáticamente el servidor, como son claves autonumeradas, lo que hace es seguir sumando esos números en la numeración. Es decir toma a Uruguay y a Brasil pero le asigna el valor siguiente al último valor que ya existe. Es decir, como ya existía Uruguay 1 y Paraguay 2, a Uruguay le cambia el ID y le pone a Uruguay 3 y a Paraguay 4. MIN 19 aprox

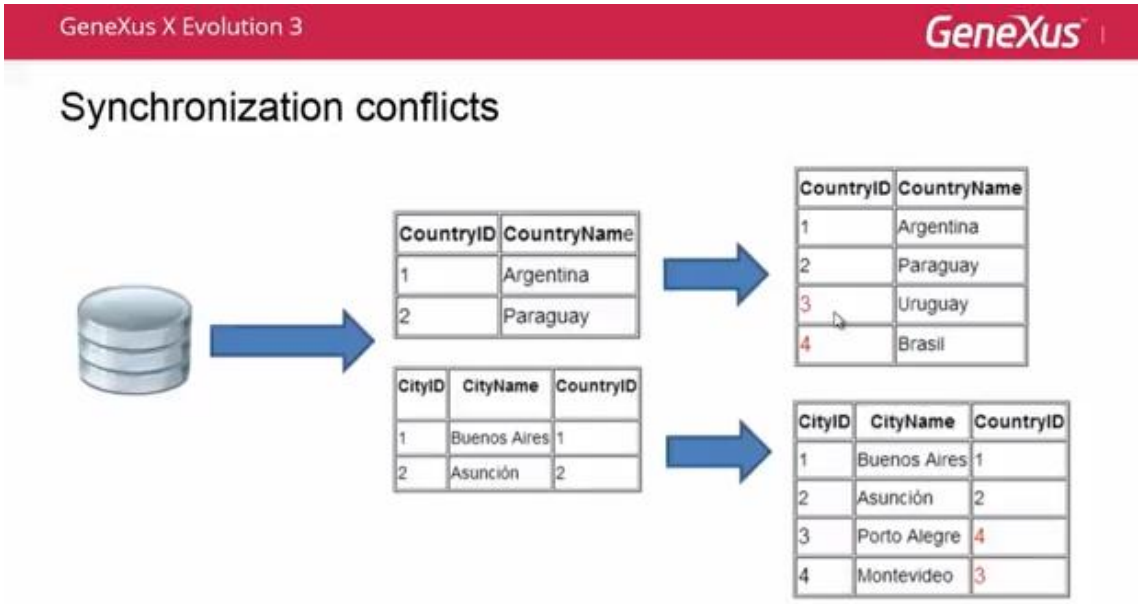
Lo interesante de todo esto es que las modificaciones también se hacen en cascada. Es decir, Uruguay con el ID3 y Brasil con el ID4 también los modifica en la tabla City como tiene clave foránea va y los modifica ahí para que quede todo consistente.

Y que pasa con los datos que yo tenía en mi Device? Que tenía Uruguay1 y Brasil2?

Lo que hace es modificar hacia atrás. Una vez que modifico en el servidor y quedo todo consistente va y modifica en el Devices. Entonces queda todo el ambiente consistente y con los valores finales que Uds. están viendo ahora en pantalla.

De esta manera es que se resuelven los conflictos de claves autonumeradas.

A continuación les voy hablar es un poco de Prototipacion y GAM



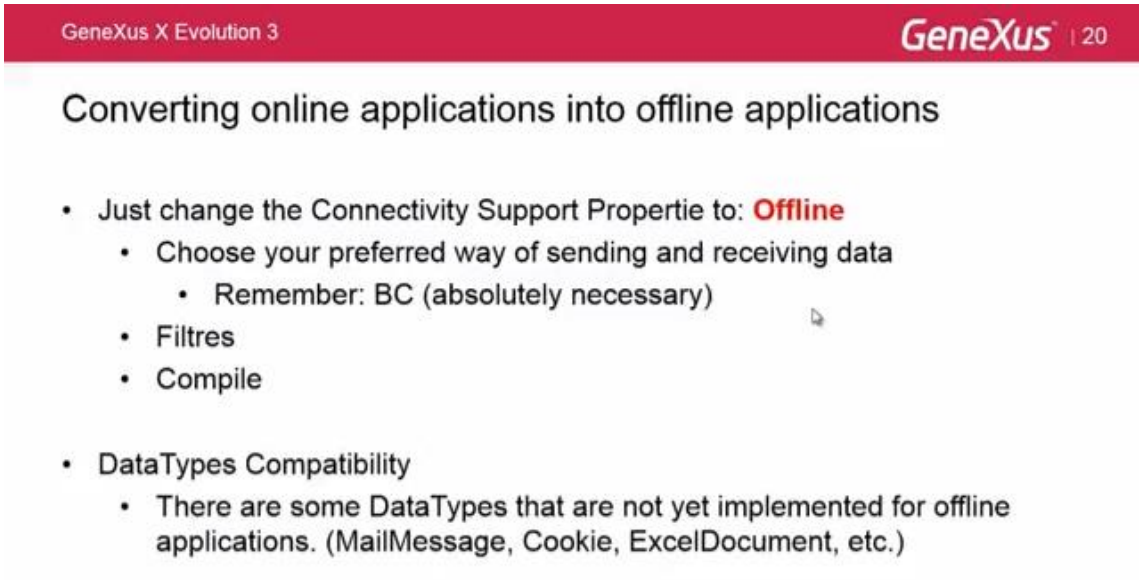
En el tema de prototipación, lo que les quería comentar es que el KBN no va a estar disponible para testear aplicaciones offline.

¿Qué quiere decir esto? Nosotros como les contaba vamos a estar generando nuestras aplicaciones offline, lógica compleja, que va requerir la generación de programas en código nativo - será JAVA en Android y Objective-C en IOS – y el KBN solo interpreta metadata, por lo tanto no va a poder ejecutar este tipo de lógica por lo tanto no nos va a servir para el testeo de aplicaciones offline.

Nosotros lo que vamos a tener que hacer es compilar y ejecutar esas aplicaciones en los Devices

Y luego sobre el GAM, lo que quería comentarles es que; punto número uno las credenciales se mantienen en el servidor por un tema de seguridad yo no voy a llevar las credenciales de los usuarios a la base de datos de los dispositivos. Las credenciales se mantienen en el servidor y de esto se desprende el punto numero dos que el loguin será solamente online. Yo solo voy a poder hacer un loguin en mi aplicación cuando tenga conexión a internet. Entonces si miramos el punto número tres podemos decir que la autentificación de usuarios se va a poder hacer solamente en forma online pero si yo luego de hacer la autentificación pierdo la conexión voy a poder seguir accediendo a las pantallas offline de mi aplicación.

Por ultimo comentarles como convertimos nosotros las aplicaciones que veníamos trabajando en la XEv2, aplicaciones online; como las pasamos a aplicaciones offline en la XEv3

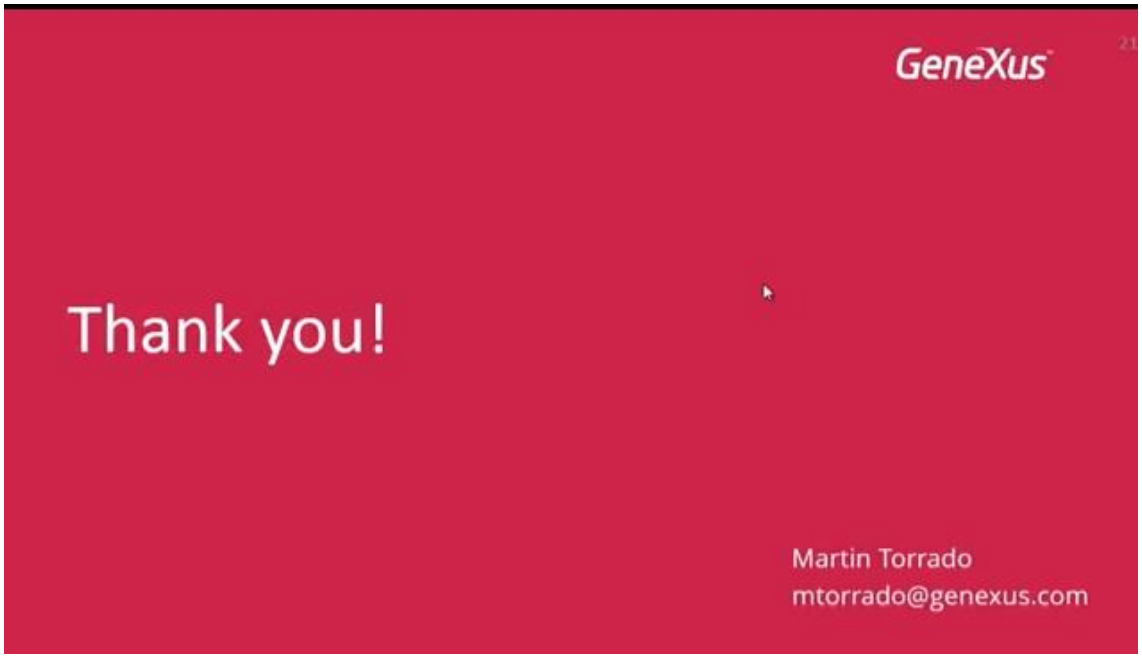


Lo que nosotros tenemos que hacer es lo que comentábamos antes en esta charla, es decir cambiar la propiedad connectivity support a offline, por default nosotros vamos a tener la propiedad seteada en online y se la vamos a cambiar en offline. Luego vamos a elegir qué forma queremos que se envíen y reciban datos, como vimos en las pantallas anteriores, si queremos que sea automático, manual, si va hacer by row, by table, esas opciones tenemos que setearlas, y muy importante recordar que es absolutamente necesario que los cambios que nosotros queramos que se envíen a la base de datos del servidor tiene que ser mediante Business Component.

Teniendo en cuenta eso también aplicar filtros, si se acuerdan nosotros podemos decir que tablas queremos que se envíen a la base de datos del Device. De esas tablas cuales son los registros que nosotros queremos que nos envíen, entonces aplicar esos filtros y luego hacer un Rebuild all del objeto main y recordar también que hay algunos DataTypes que no son compatibles.

Tenemos algunos ejemplos como el MailMessenger, Cookie, ExcelDocumento, etc… para más información pueden verlo en el wiki pero dejar como conclusión que es muy fácil hacer el cambio de nuestras aplicaciones online a offline.

Esto es todo lo que quería comentarles por hoy sobre las aplicaciones SD offline. Los invito entonces a descargarse la Evolution 3, a probarla y a utilizar todos los beneficios que ésta brinda, todo lo que estuvimos conversando en esta charla y cualquier consulta les dejo mi correo.


GeneXus
Thank you!
Martin Torrado
mtorrado@genexus.com