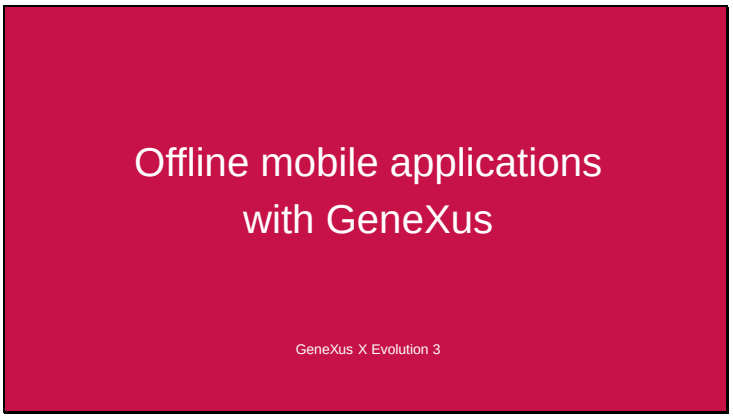
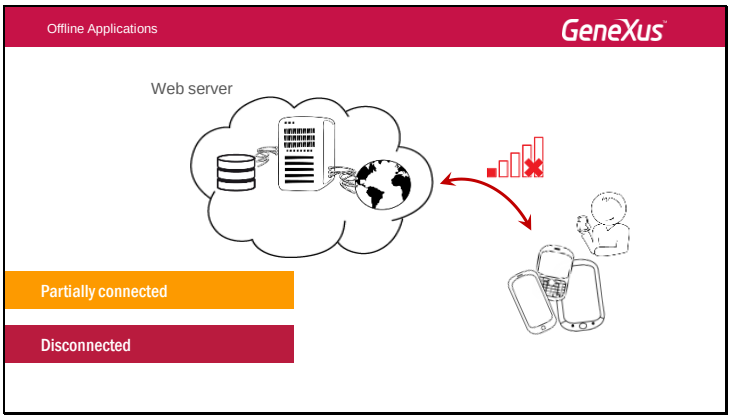
Aplicaciones Offline con GeneXus



Hasta ahora hemos desarrollado aplicaciones para dispositivos inteligentes, que estaban siempre conectadas a través de Internet al servidor web, que almacenaba los servicios y los datos necesarias para la aplicación.

Un objetivo importante para los dispositivos inteligentes es permitir que la aplicación, o parte de ella, se siga ejecutando cuando se encuentra desconectada de internet.



Hay tareas que debido a su sensibilidad, deben ser validadas en la base de datos centralizada. También las tareas en las que sus datos cambian muy frecuentemente, requerirán acceso continuo al servidor web.

En nuestro ejemplo, el login deberá ser con conexión, y el panel que muestra los tweets es deseable que también lo sea.

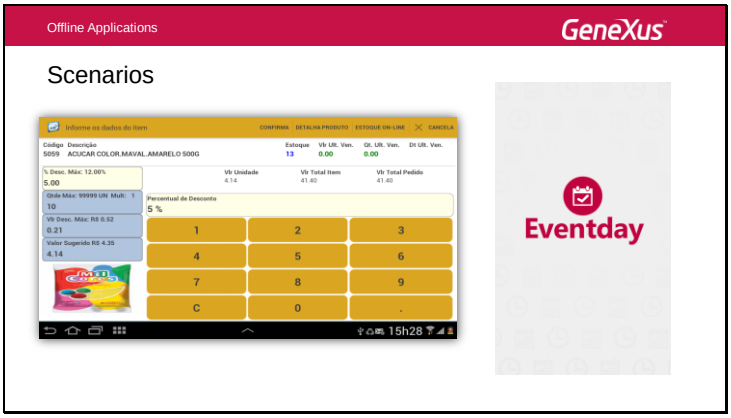
Otras tareas sin embargo, pueden ejecutarse sin conexión y sincronizar sus datos después.

Por tanto, debemos poder elegir qué objetos de la aplicación pueden ejecutarse offline y cuáles no.

A estas aplicaciones las llamamos PARCIALMENTE CONECTADAS ya que tienen acceso a datos locales y la posibilidad de ejecutar lógica compleja en el dispositivo.

A las aplicaciones que una vez instaladas no tienen necesidad de conectarse al servidor, las denominamos DESCONECTADAS. Es el caso de aplicaciones que manejan datos personales y que no hay interés de enviar esos datos a ningún servidor.

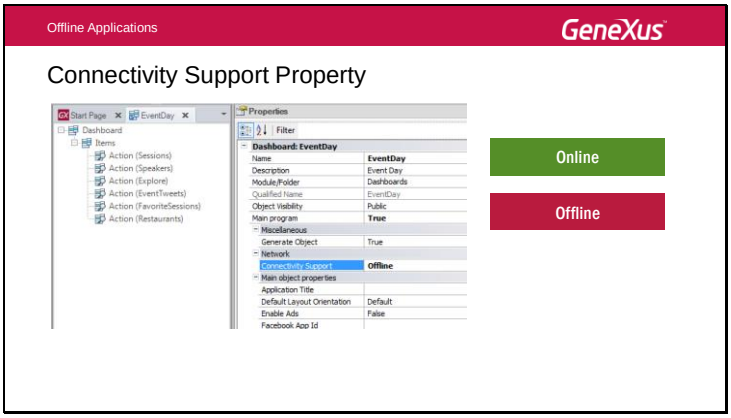
Las aplicaciones parcialmente conectadas son las más frecuentes. Como ejemplos de estas aplicaciones podemos citar a los Puntos de Venta o las aplicaciones para Eventos.



En el caso de los Puntos de Venta, se habilita a que vendedores o distribuidores que recorren las calles, puedan emitir órdenes de compra y facturar, estén o no estén conectados. Luego al obtener conexión, sincronizan sus datos con los de la empresa.

O como en el caso que estamos viendo, aplicaciones para eventos, donde los datos no cambian muy frecuentemente, se carga la base de datos al instalar la aplicación y luego se puede seguir usando la aplicación en forma desconectada. Y por otro lado hay información que requiere una conexión frecuente, como poder leer los tweets publicados.

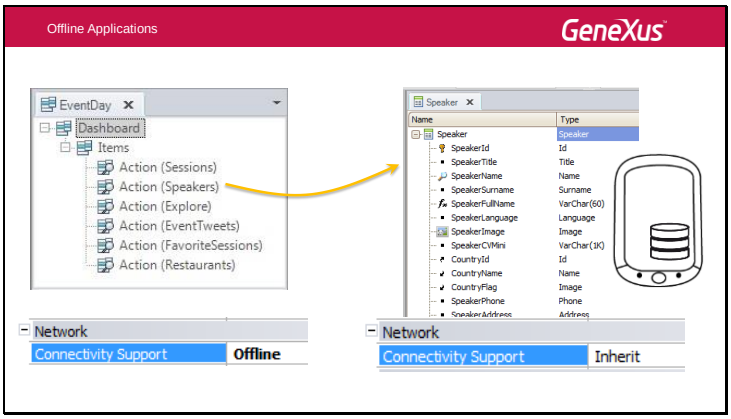
Ahora bien: ¿cómo se indica en GeneXus que se va a usar una aplicación offline?
Con la propiedad Connectivity Support, a nivel de los objetos SD que son Main.
En nuestro ejemplo este objeto es el dashboard.



Esta propiedad puede tener el valor **Online** u **Offline**, indicando cómo va a ser el funcionamiento del objeto main.

Además, todos los objetos que no son main y son invocados por el objeto main, tienen la propiedad **Inherit**, para indicar si heredan el comportamiento del objeto main que los invoca. Esto es válido para las transacciones que tienen Business Components asociados, los objetos WorkWithDevices, los paneles para SD, los Data Providers y los procedimientos que estén en el árbol de invocaciones del objeto main.
Veamos esto con un ejemplo.

Supongamos que se tiene una aplicación para un evento.
¿Qué sucede cuando se configura la propiedad **Connectivity Support** del Dashboard EventDay con el valor Offline?

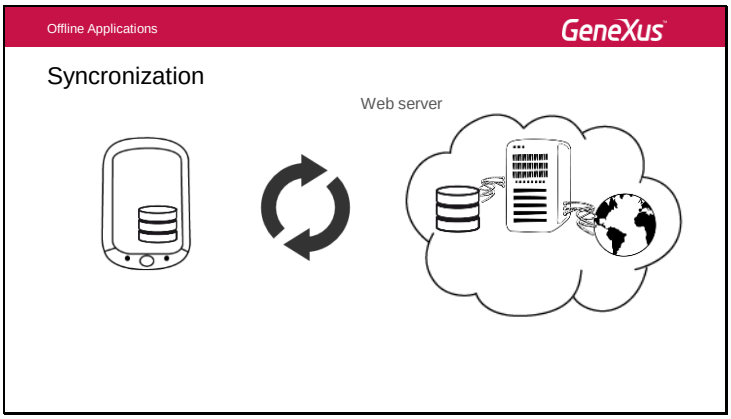


Las transacciones asociadas a business components utilizados en objetos WorkWithDevices invocados desde el Main (directa o indirectamente) y que tengan la propiedad Connectivity support con el valor **Inherit**, generarán tablas que serán creadas en la base de datos local del dispositivo. Además se crearán todas las tablas que sean necesarias de acuerdo a la integridad referencial o por atributos mencionados en

paneles o prompts, para asegurar que la aplicación offline funcione igual que cuando estaba con conexión.

En aquellos objetos que queremos que siempre estén con conexión, configuramos la propiedas Connectivity support con el valor **Online**.

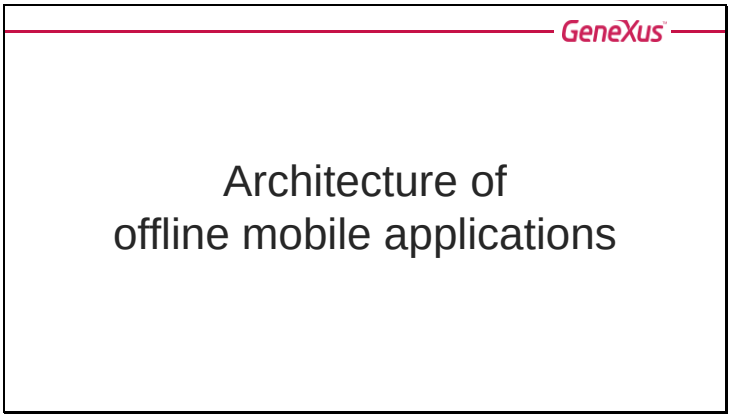
Cuando se instala la aplicación en el dispositivo, se creará la base de datos en el propio dispositivo y también las tablas que correspondan. También se puede lograr que la primera vez que se establezca la conexión, se traigan los datos del servidor al dispositivo, para poblar las tablas.

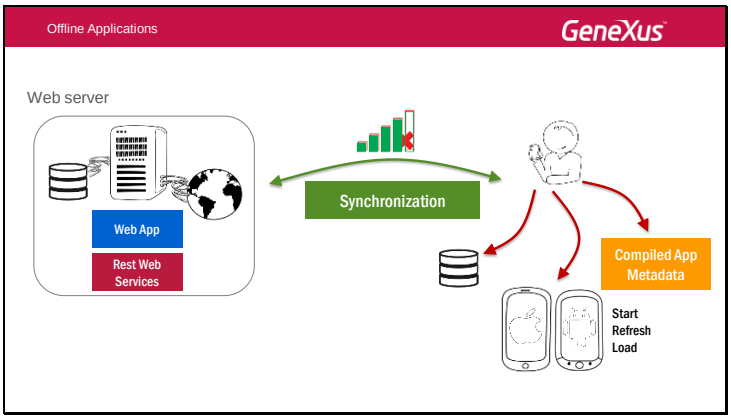


Una vez hecho esto, si el dispositivo pierde la conexión, la aplicación va a seguir funcionando con los datos almacenados localmente. Una vez que el dispositivo obtiene conexión, la información almacenada localmente se sincroniza con los datos que se encuentran en el servidor. También los datos del servidor que cambiaron se envían al dispositivo para ser actualizados, cada cierto tiempo o a demanda.

Más adelante veremos que es posible cambiar este comportamiento, por ejemplo se puede querer que nunca se sincronice o sincronizar a pedido.

Veamos ahora cuál es la arquitectura de las aplicaciones offline.





Una aplicación offline se puede dividir en dos componentes:

- Un componente local que incluye una base de datos local (normalizada) con un subconjunto de las tablas del servidor y toda la lógica necesaria para ser ejecutada en forma local, y...

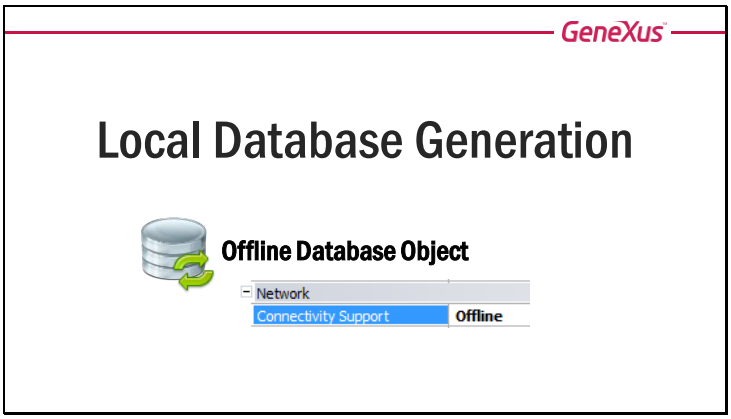
- Un componente del lado del servidor, donde se encuentra el backend de la aplicación, la base de datos y la capa de servicios para la sincronización con el dispositivo.

Ambos componentes se comunican via web services REST para la sincronización y para cargar la base de datos local y luego enviar las modificaciones hechas en forma local al server.

Esa comunicación es conocida como Sincronización.

Para poder instalarse el componente local en el dispositivo, la aplicación deberá estar compilada, con lo que incluirá toda la información para acceder a los servicios REST.

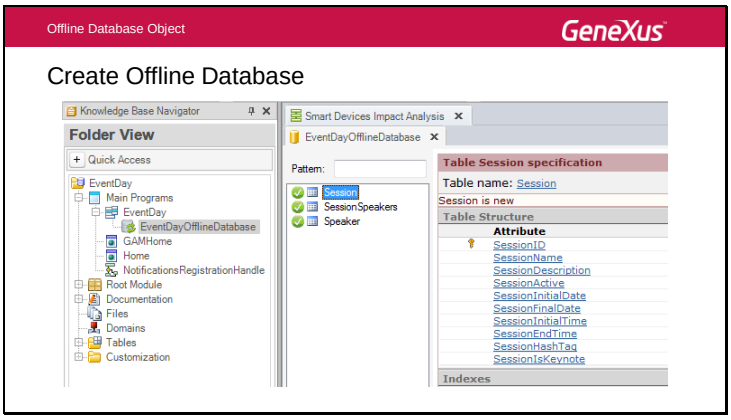
Para cada objeto main que tenga la propiedad Connectivity Support en **Offline**, se crea un objeto llamado **Offline Database**.



Este objeto es el encargado de determinar cuáles son las tablas que van a la BD local y también cuales son los datos que se llevan cuando se sincroniza, porque obviamente la BD local no será la misma que la BD del servidor, sino una versión reducida.
La base de datos que se creará en el dispositivo, es una base de datos SQLite.

Luego de cambiar la propiedad Connectivity support del objeto main de nuestra aplicación al valor offline, se debe ejecutar la acción Rebuild All.

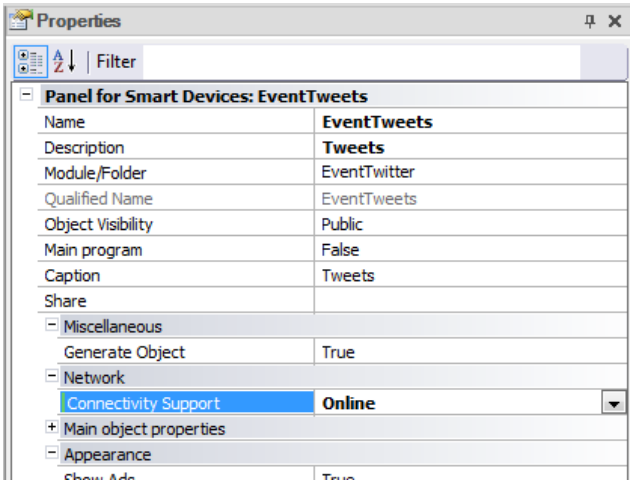
Cuando se hace Build por primera vez del main object con la propiedad Connectivity support en el valor **offline**, se crea el objeto Offline Database y se realiza un análisis de impacto de qué tablas serán creadas en el dispositivo, asociadas al objeto offline database.



Vamos a hacer esto en GeneXus.

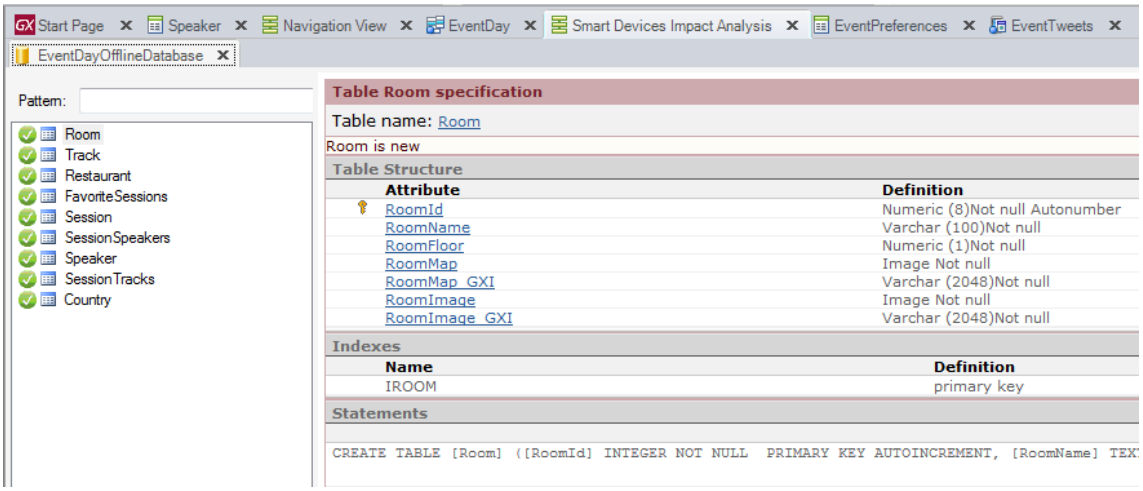
Abrimos el objeto dashboard EventDay, y cambiamos su propiedad Connectivity support del valor por defecto Online a **Offline.** Nuestra intención es que toda la aplicación pueda funcionar sin conexión a internet, salvo la pantalla que muestra los tweets, que queremos que funcione online, debido a la velocidad con la que cambia esa información.

Así que abrimos el objeto EventTweets y modificamos la propiedad Connectivity Support asignándole el valor **Online**.

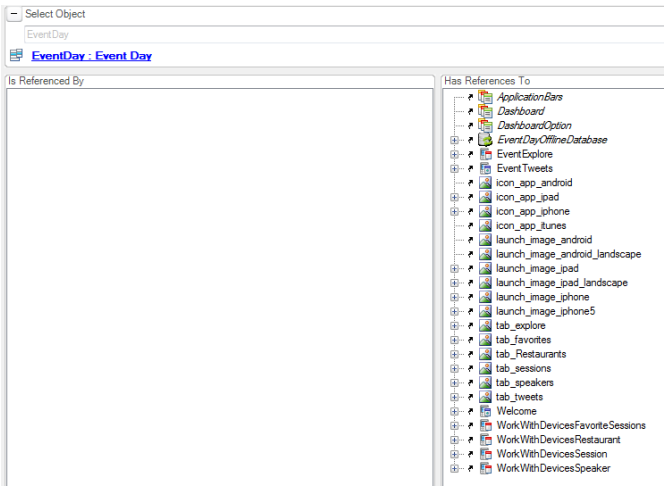


Ahora presionamos el botón de Rebuild All.

Vemos que se nos muestra el análisis de impacto con la creación del objeto EventDayOfflineDatabase y las tablas que se crearán en el dispositivo, que en este caso serán todas ya que no modificamos el valor por defecto de la propiedad Connectivity Support de ninguna transacción y dejamos a todas el valor **Inherit**.



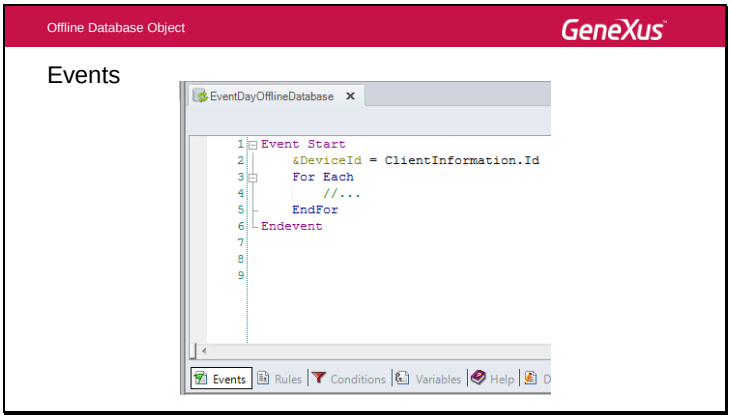
Observemos que no se incluye la tabla EventPreferences.
Si hacemos click derecho sobre el objeto EventDay y seleccionamos la opción References, vemos a qué objetos invoca el dashboard y no encontramos que se invoque al business component EventPreferences ni directa ni indirectamente.



Cuando se instala la aplicación en el device se hace el create de las tablas y se copian los datos desde el server, si hay conexión.

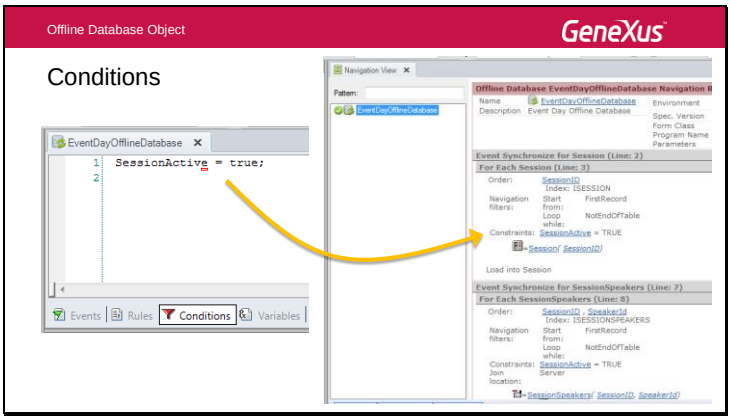
Volvamos al objeto Offline Database

Se trata de un objeto simple que tiene solamente eventos, condiciones y propiedades.

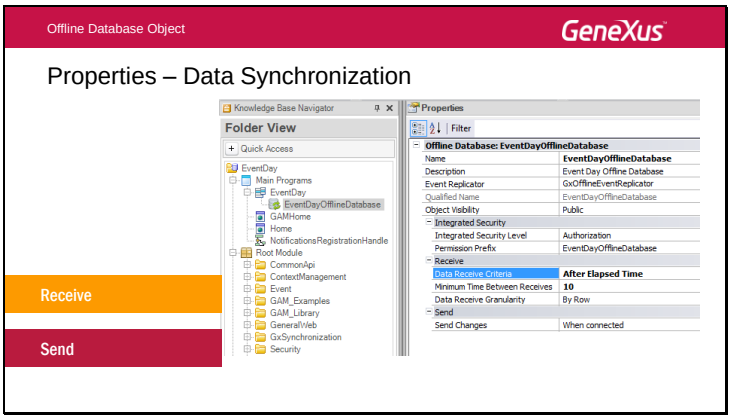


El único evento que se puede programar es el evento Start. Este evento se ejecuta en el servidor, antes de cada envío de datos al cliente para sincronización.

El evento Start está pensado para la inicialización de variables y algún otro procedimiento que se debe hacer antes de la sincronización de tablas.



Los filtros que incluyamos en la solapa Conditions, se utilizan como en cualquier otro objeto GeneXus. Son filtros que aplican a las tablas para saber qué datos se llevan al dispositivo. Son globales, por lo que se aplican a todas las tablas que corresponda.
En las conditions se pueden utilizar variables que pueden cargarse en el evento Start.

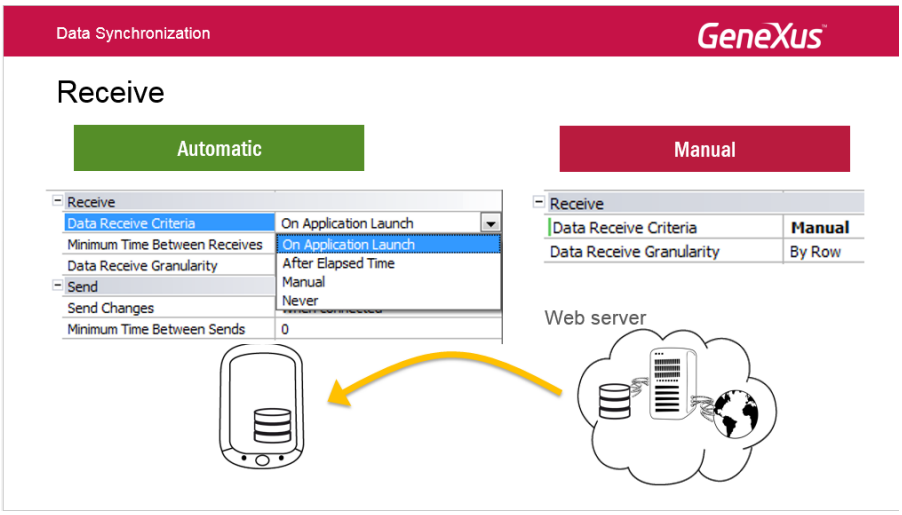


El objeto OfflineDatabase incluye propiedades que podemos utilizar para definir cuándo enviar o recibir datos hacia y desde el servidor respectivamente.

- Receive, que definen cómo y cuándo recibir un subconjunto de datos desde el servidor.

Y en

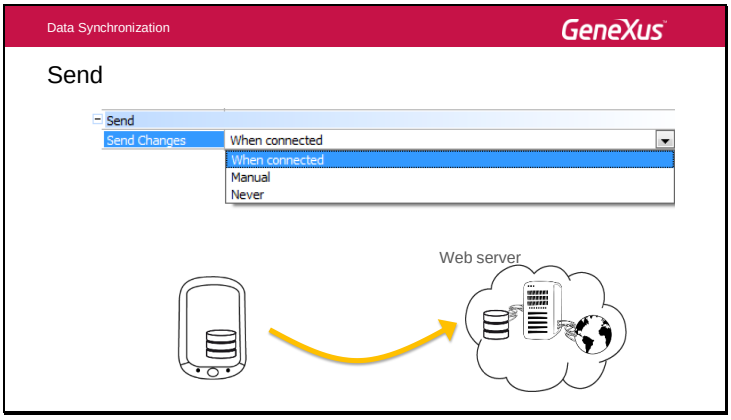
- Send, las que definen cómo y cuándo se envía datos desde el dispositivo al server.



Al recibir datos desde el servidor, existen dos formas de realizar la sincronización:

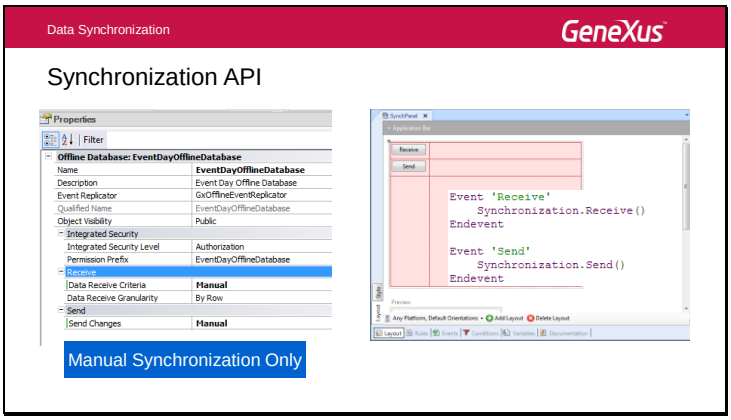
- En forma **automática**, mediante la cual se puede configurar en qué momento realizar la sincronización, si al abrir la aplicación por primera vez y habiendo pasado cierto tiempo desde la anterior, o siempre, o cada cierto tiempo, o nunca.

- O en forma **manual**, en la que se debe programar por código la forma de recibir los datos desde el servidor.



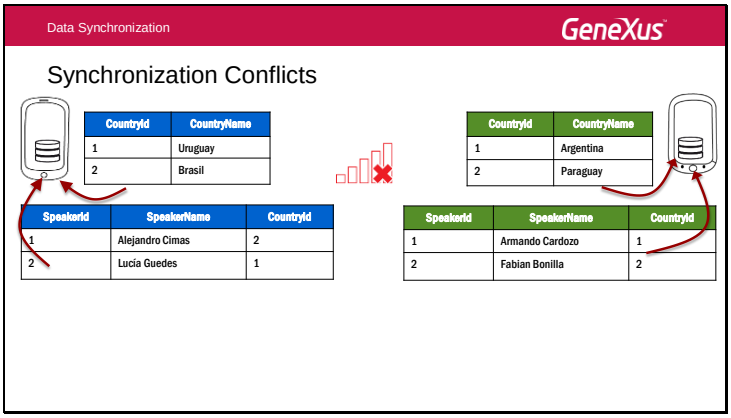
El envío de datos al servidor también puede configurarse si será automático, cuando el dispositivo obtiene la conexión, o manual que dependerá de lo que programemos para hacerlo, o nunca, si no se desea llevar datos del dispositivo a la base de datos centralizada.

En el caso en el que tanto el Receive como el Send se hagan en forma manual, se debe usar la Synchronization API para realizarlos. Esta API no se encuentra como objetos dentro del folder SmartDevicesAPI, sino que es parte de la gramática.



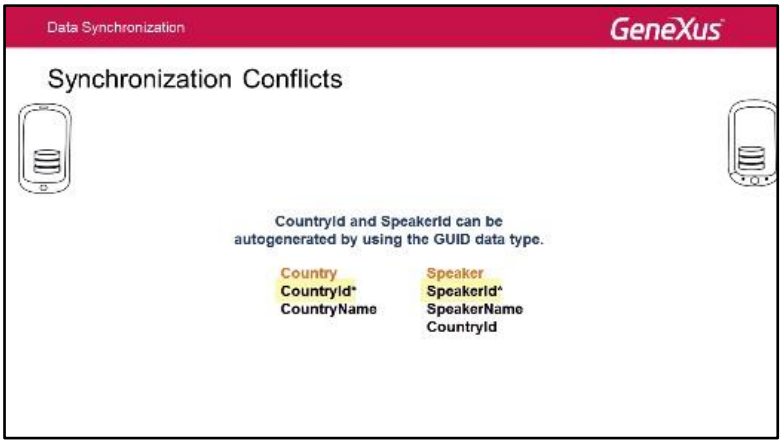
Como ejemplo, se puede hacer un SD panel que se ejecuta desde el device, donde se programa el send y el receive, como vemos en la imagen.

Cuando se envían o reciben datos desde el dispositivo, puede haber conflictos de sincronización ya que la aplicación puede estar instalada en varios dispositivos y cada uno de ellos puede insertar datos con los mismos identificadores pero para datos diferentes.

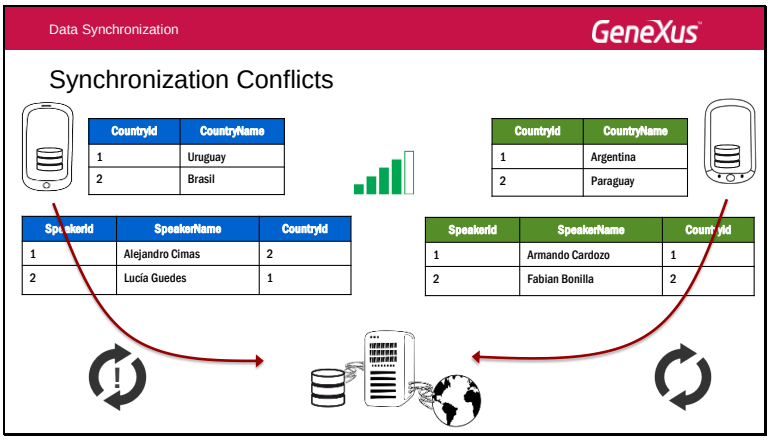


Veamos esto en nuestro ejemplo.

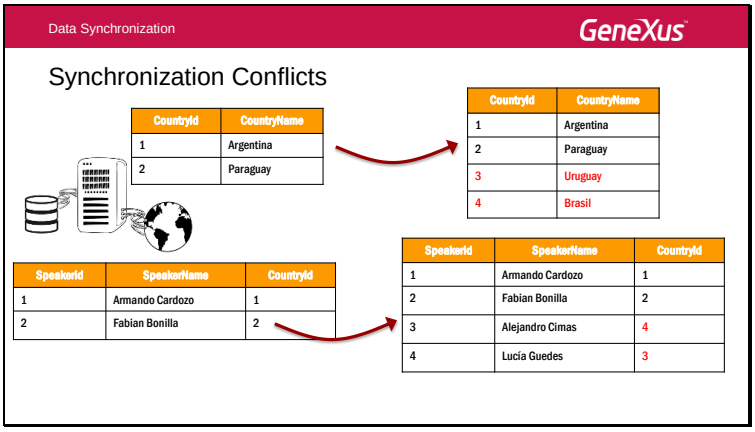
En la KB de EventDay, tenemos la transacción Country con los atributos CountryId y la transacción Speaker, que tiene al atributo CountryId como clave foránea.



Por ejemplo, aquí vemos que un dispositivo creó los países Uruguay con el Id=1 y Brasil con el Id=2, mientras que en un segundo dispositivo se insertan los países Argentina con el Id=1 y Paraguay con el Id=2. Ambos dispositivos están trabajando en forma offline, sin conexión, con lo que todos los datos quedan guardados en la base de datos local de cada dispositivo.



Una vez que se obtiene la conexión, uno de los dispositivos se sincroniza con el server, enviando toda las operaciones que realizó a través de Business Components sobre la base de datos local.
Cuando el segundo dispositivo intenta sincronizar da conflicto porque se repite la clave del país.
Como se usan claves autonumeradas, GeneXus resuelve el conflicto.

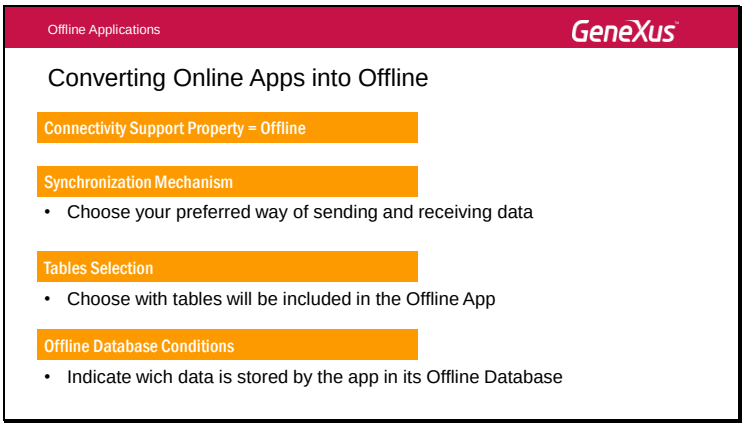


¿Cómo lo hace? Resolviendo el conflicto en el servidor.

GeneXus genera una nueva clave para los países que se repiten, y actualiza los datos de la tabla del server con ese Id generado nuevamente.

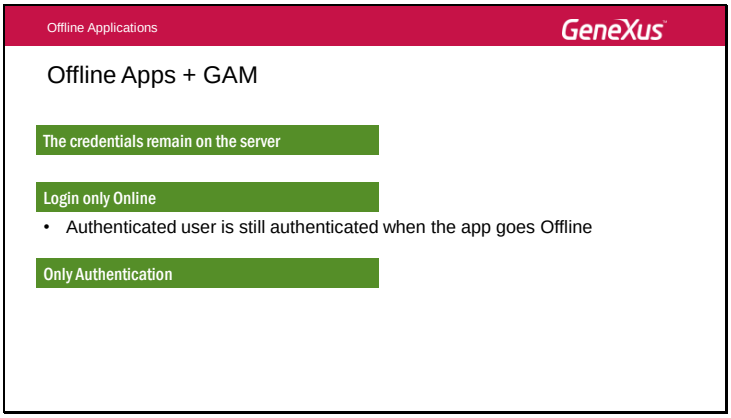
Luego actualiza la tabla de oradores para que las claves foráneas se correspondan.
Por último, actualiza en el dispositivo los datos, para que en el device, queden las claves correctas también.

A modo de resumen de lo que hemos visto, para convertir una aplicación online, en una aplicación offline, debemos seguir los siguientes pasos:



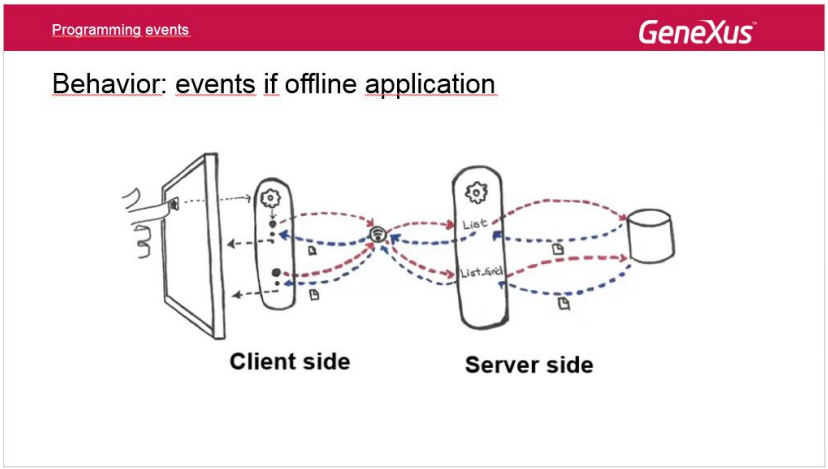
1. Configurar la propiedad Connectivity Support en el valor Offline
2. Seleccionar el mecanismo de sincronización
3. Indicar cuáles tablas se van a crear en el dispositivo
4. Y agregar condiciones para filtrar que registros van al dispositivo

Si se tiene una aplicación Offline que usa GAM, hay que tener en cuenta que las credenciales siempre estarán en el server, por lo tanto el login sólo se puede hacer estando Online.



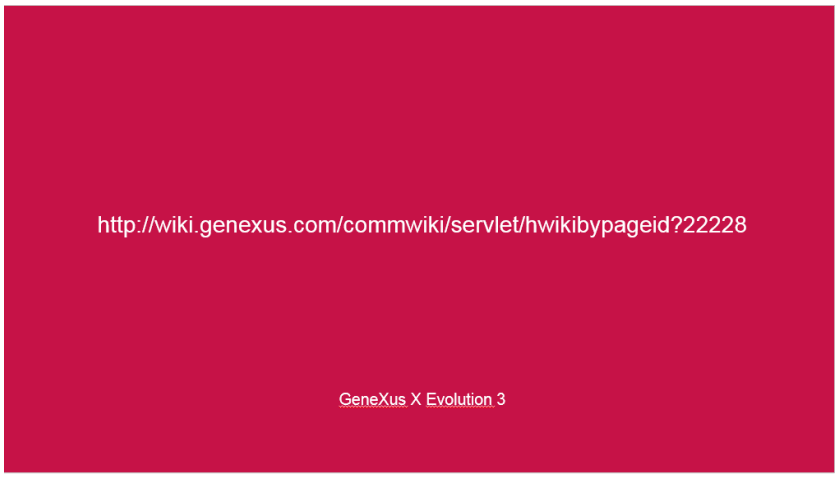
Una vez que se loguea, sí se puede trabajar en forma Offline.

La programación de eventos en una aplicación offline, tiene ciertas consideraciones que la diferencian de las aplicaciones online.



Veremos los eventos de una aplicación offline en un próximo video.

Podemos obtener más información sobre el tema Offline, en la dirección que se muestra en pantalla:
http://wiki.genexus.com/commwiki/servlet/hwikibypageid?22228



Video filmado con GeneXus X Evolution 3