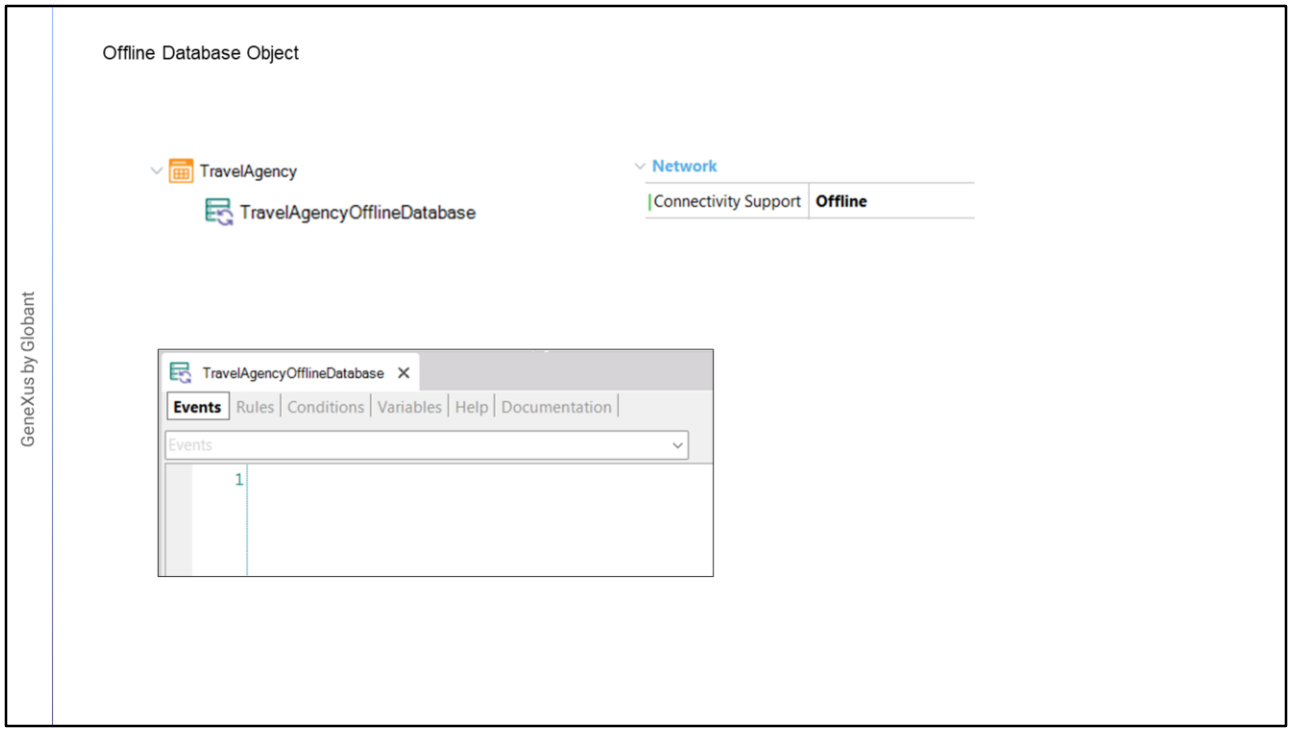

GeneXus by Globant
GeneXus™
by Globant
training.genexus.com

GeneXus by Globant

Offline Database Synchronization

Diego Marranghello

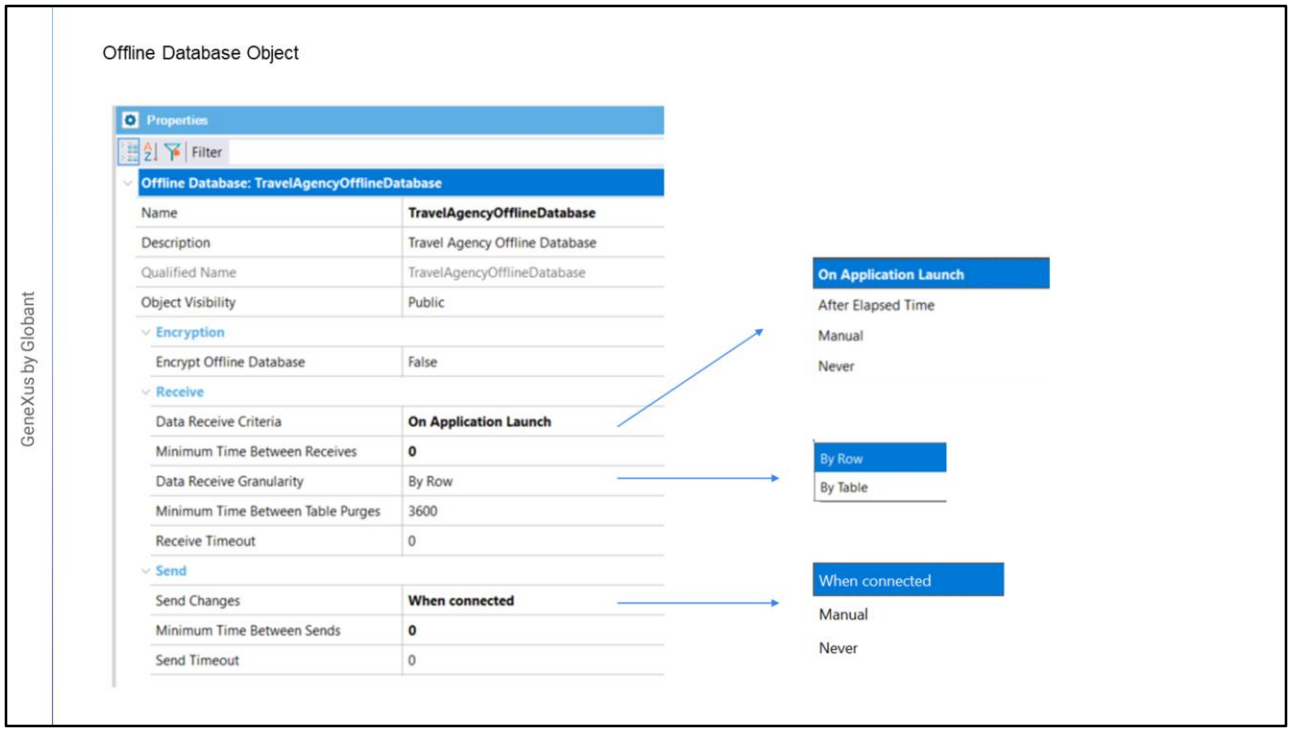


Para cada objeto main que tenga la propiedad Connectivity Support = Offline, se crea un objeto llamado Offline Database.

Al crearse, podremos ver cuáles son las tablas que se crearán en la BD local y se crean también, en el lenguaje nativo del dispositivo, los programas para crear la base de datos local.

Este objeto es el encargado de determinar cuándo se produce la sincronización y cuáles son los datos que interesan cuando se sincronizan con las tablas del server.

El objeto Offline Database dispone además de eventos y condiciones que permiten determinar su comportamiento.



Vamos a ver algunas de las propiedades mas importantes del objeto Offline Database.

En el grupo Receive tenemos la propiedad “Data Receive Criteria” que se usara para la recepción de datos, los valores que podemos utilizar son:

-Cuando se lance la aplicación, que es el valor por default, indica que la aplicación iniciara la recepción de datos cuando esta se inicie.

-Con el valor After Elapsed Time, la recepción se hará cuando se cumpla el tiempo indicado en la propiedad Minimum Time Between Receives, expresado en segundos, además también se realizara cuando la aplicación se inicialice.

-El valor Manual indica que la recepción se efectuara solo en forma manual, aquí se deberá desarrollar una acción para lanzar la recepción.

-Por ultimo, el valor Never indica que nunca se llevara a cabo la recepción de datos por defecto y GeneXus tampoco generara los programas de sincronización necesarios, los cuales deberán ser implementados manualmente en caso necesario.

Como ya habíamos visto, tenemos la propiedad “Data Receive Granularity”, que nos permite establecer cual será el mecanismo de recepción a implementar, puede ser by Row que es el valor por defecto o by Table.

Luego tenemos la propiedad Send Changes que nos permitirá establecer cuando

deseamos realizar el envío de los datos desde el dispositivo hacia el server, los valores que disponemos son:

-When Connected, que es el valor default, indica que el envío se realizara en forma inmediata cuando se detecte que hay conexión.

-El valor Manual, indica que el envío se efectuara solo en forma manual, aquí se deberá desarrollar una acción para lanzar la recepción.

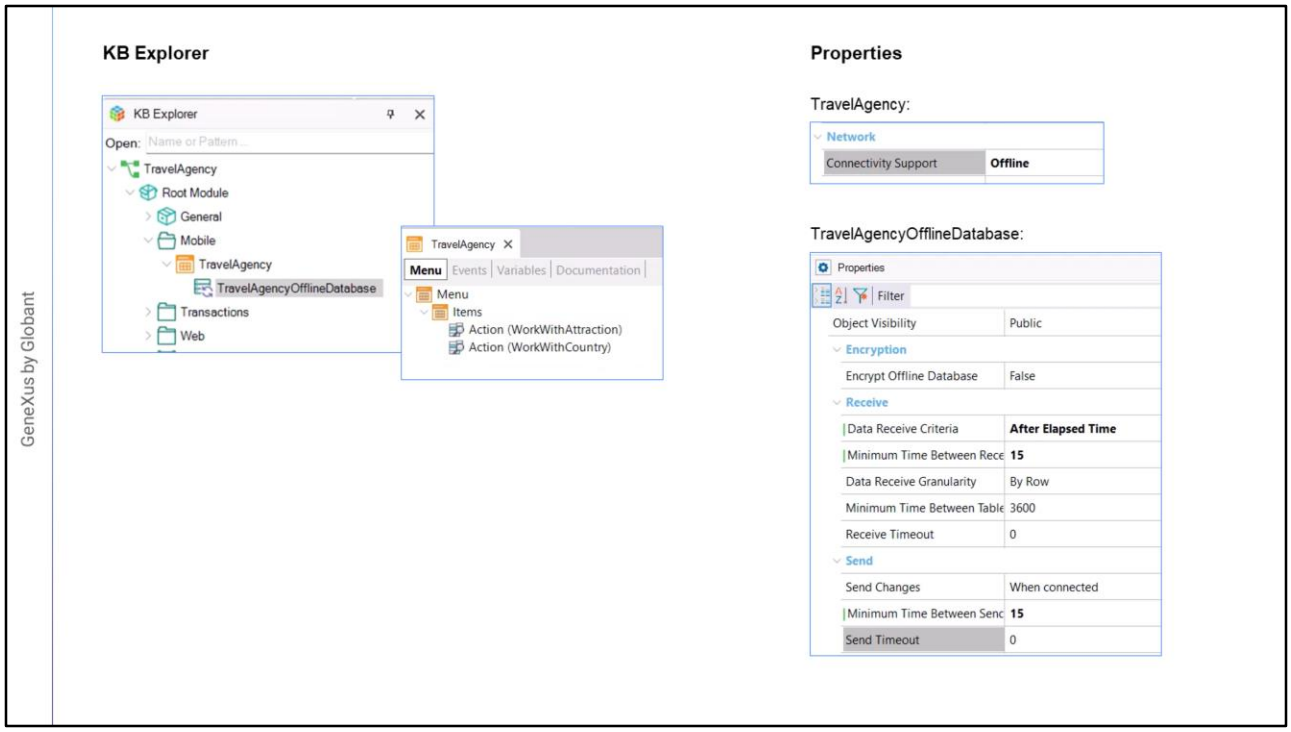
-Y el valor Never, indica que no se enviaran datos en forma automática por GeneXus, los registros modificados no se almacenaran en la tabla auxiliar GXPendingEvent, por lo que queda la responsabilidad del lado del desarrollador en caso de necesitar el envío, de programar toda la lógica necesaria.

Ya sea en el caso de usar para el envío o la recepción el valor Manual o para darle mayor funcionalidad al usuario, necesitamos poder desarrollar acciones que lleven a cabo la sincronización.
Para ellos contamos con la API Synchronization.

Esta API no se encuentra en las References como el resto de las APIs sino que es parte de la gramática.
Cuenta con los métodos Send y Receive para la sincronización, ServerStatus para determinar el estado del server, y ResetOfflineDatabase que retorna la base de datos local a su estado inicial, ya sea haciendo un Create Database para vaciar las tablas o cargando una base datos precargada.

Con el uso de esta API podríamos por ejemplo desarrollar en un panel opciones para que el usuario dispare la sincronización cuando él lo desee.
Aquí podemos ver un ejemplo muy simple donde se utilizan los dos métodos en un mismo evento.

Veamos un ejemplo en GeneXus.



Tenemos creada parte de una aplicación para una agencia de viajes, donde tenemos creado un objeto Menu y como ítems los objetos WorkWith de Atracción y de País.

Primero vamos a configurar el objeto Offline Database para que la sincronización se realice cada 15 segundos.

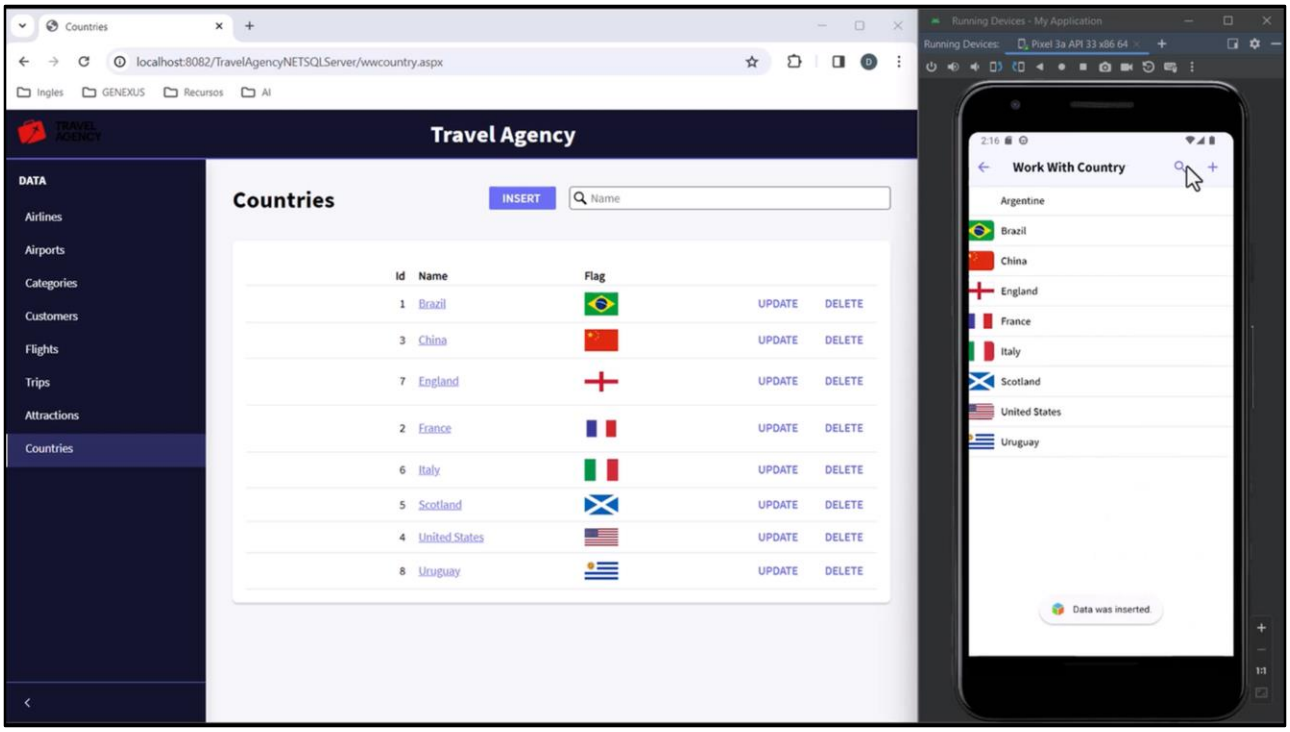
Abrimos el objeto TravelAgencyOfflineDatabase, que fue creado a partir de configurar la propiedad Connectivity Support en Offline.

Vamos a las propiedades y en la propiedad Data Receive Criteria vamos a poner el valor After Elapsed Time y en Minimum Time Between Receives vamos a poner el valor 15.

Ahora en el grupo Send, en la propiedad Minimum Time Between Sends también vamos a poner el valor 15.

Con esta configuración, el tiempo mínimo que habrá entre un envío/recepción de datos en nuestra aplicación será de 15 segundos.

Vamos a probar esto, grabamos y hacemos un rebuild all de la aplicación.



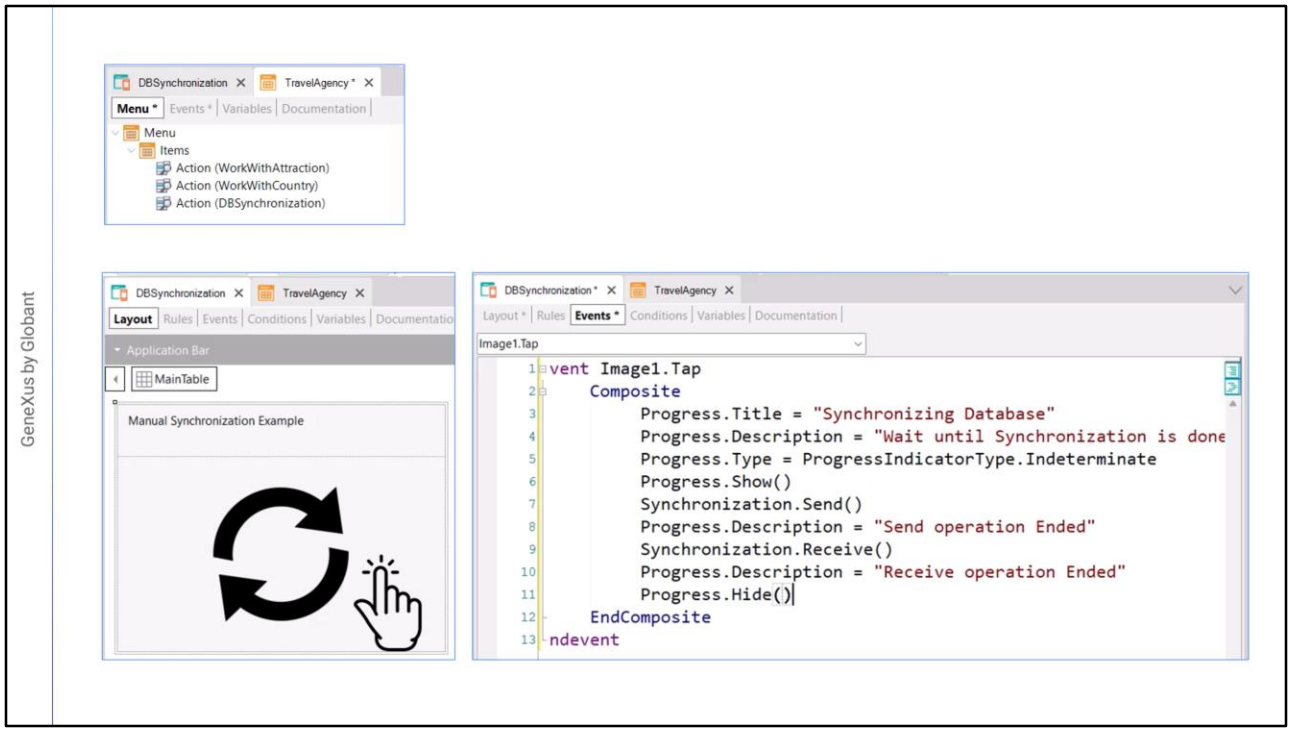
Bien, ya tenemos los cambios.

Vamos a ejecutar en mobile y en web, ingresando en ambos casos al Work With de Country.
Vamos a crear un nuevo país desde mobile.

Al refrescar en web, vemos que inmediatamente nos actualiza la información con el país que se acaba de ingresar, esto porque el primer send es automático. Pero si antes de los 15 segundos ingresamos otro registro, al refrescar en la pantalla web todavía no aparece. Esto es porque no pasaron los 15 segundos que configuramos, entre un ingreso y otro. Al no haber pasado ese período, vamos a tener que esperar ese tiempo para la sincronización.

Vamos a ir refrescando el panel, y ahí aparece el país registrado.

Ahora ingresamos al primer país creado y vamos a borrarlo. Confirmamos.
Vamos a web y refrescamos. Como entre el último ingreso que hicimos y la eliminación pasaron menos de 15 segundos, tendremos que esperar ese tiempo desde la eliminación para que se haga la sincronización. Listo, ya quedó la info actualizada en el server.



Vamos a programar ahora un panel que nos permita realizar la sincronización cuando el usuario lo desee, sin tener que esperar el tiempo establecido.
Ya tengo creado un Panel, llamado DBSynchronization, este por ahora tiene solamente una imagen y un texto.

Lo agregamos al menú TravelAgency y grabamos.

Queremos que cuando el usuario haga Tap sobre la imagen se lance la sincronización, vamos a programar entonces el evento Tap.

En este evento como es del lado del cliente, vamos a usar composite, recordar que siempre la sincronización se va a iniciar desde el lado del cliente.

Entonces escribimos Syncrhronization.Send(), esto va a realizar el envío de los datos del dispositivo hacia el server
y Syncrhronization.Receive() para hacer la recepción.

Además vamos a brindarle al usuario algún feedback sobre la acción que esta realizando, para esto vamos a utilizar “Progress”.
Progress es un objeto externo que nos permite mostrar como un panel donde vemos lo que esta pasando.
Para el titulo vamos a poner "Synchronizing Database“, este va a ser el titulo.
Y vamos a poner una descripción en Progress.Description.

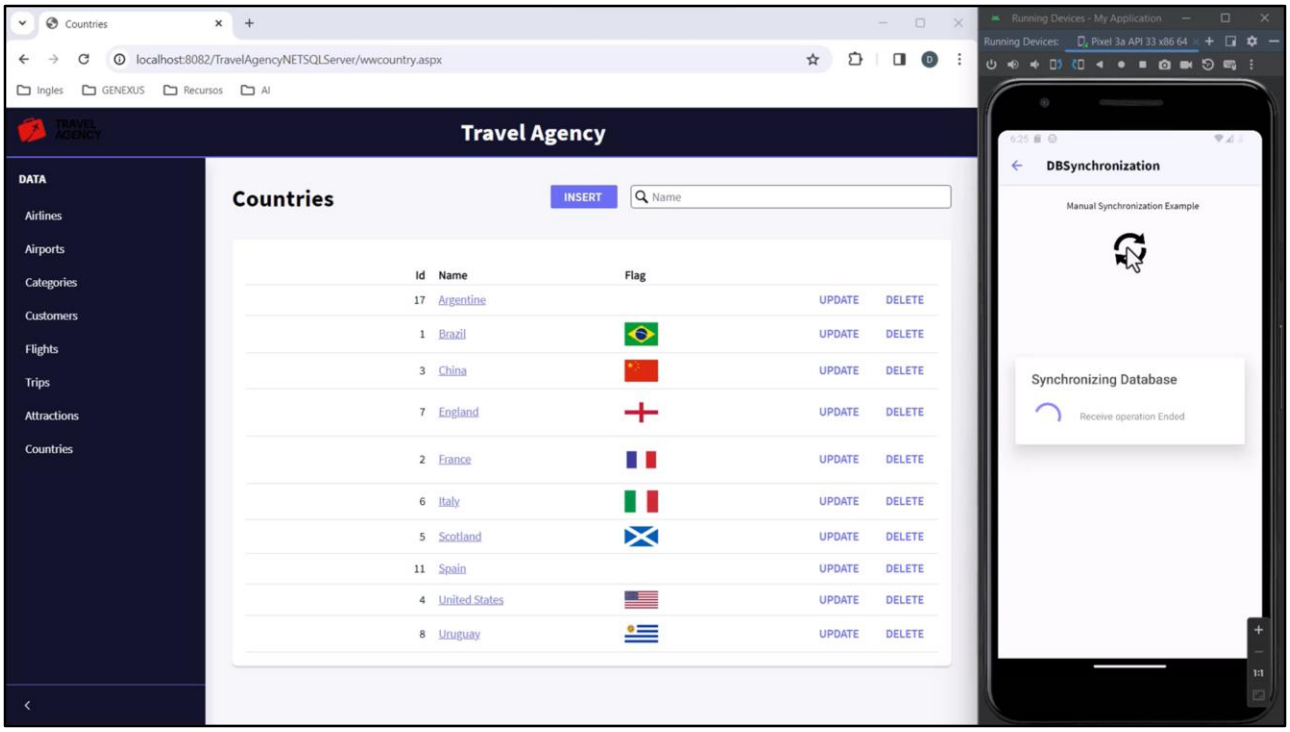
Tenemos que indicar el tipo de progreso que vamos a usar, vamos a usar indeterminado, que se usa cuando no sabemos o no podemos indicar el grado de avance del proceso.

Luego mostramos el control con Progress.Show()

A continuación haremos el Send, cuando termine podríamos cambiar la descripción, entonces vamos a poner un texto en Progress.Description

Luego hacemos el Receive, y nuevamente cambiamos la descripción para indicar que terminó.

Y lo ultimo que tenemos que hacer es ocultar el control, entonces ponemos Progress.Hide() que es el método que lo oculta.



Vamos a ejecutar la aplicación.

Bien. Ya tenemos la aplicación en el emulador y en web.

Vamos a agregar nuevamente un país desde la aplicación mobile y confirmamos.
Si actualizamos en la aplicación web, vemos la sincronización al momento porque el primer send es automático, pero si inmediatamente agregamos otro país, y vamos a la aplicación web, no nos aparece.
No vamos a esperar los 15 segundos a que sincronice, lo vamos a hacer en forma manual.
Hacemos tap sobre el icono, ahí nos indica que el send ya terminó.
Y ahora ya vemos el país ingresado.

Ahora vamos a borrarlo desde el dispositivo, confirmamos. No vamos a esperar, vamos a synchronization hacemos tap y ahora si, apenas refrescamos ya vemos los cambios.

Bien, con esta pequeña demo pudimos ver como configurar algunas propiedades del objeto Offline Database y además vimos como es posible utilizar la API Syncrhonization para realizar las operaciones de envío y recepción a pedido del usuario.


GeneXus by Globant
GeneXus™
by Globant
training.genexus.com