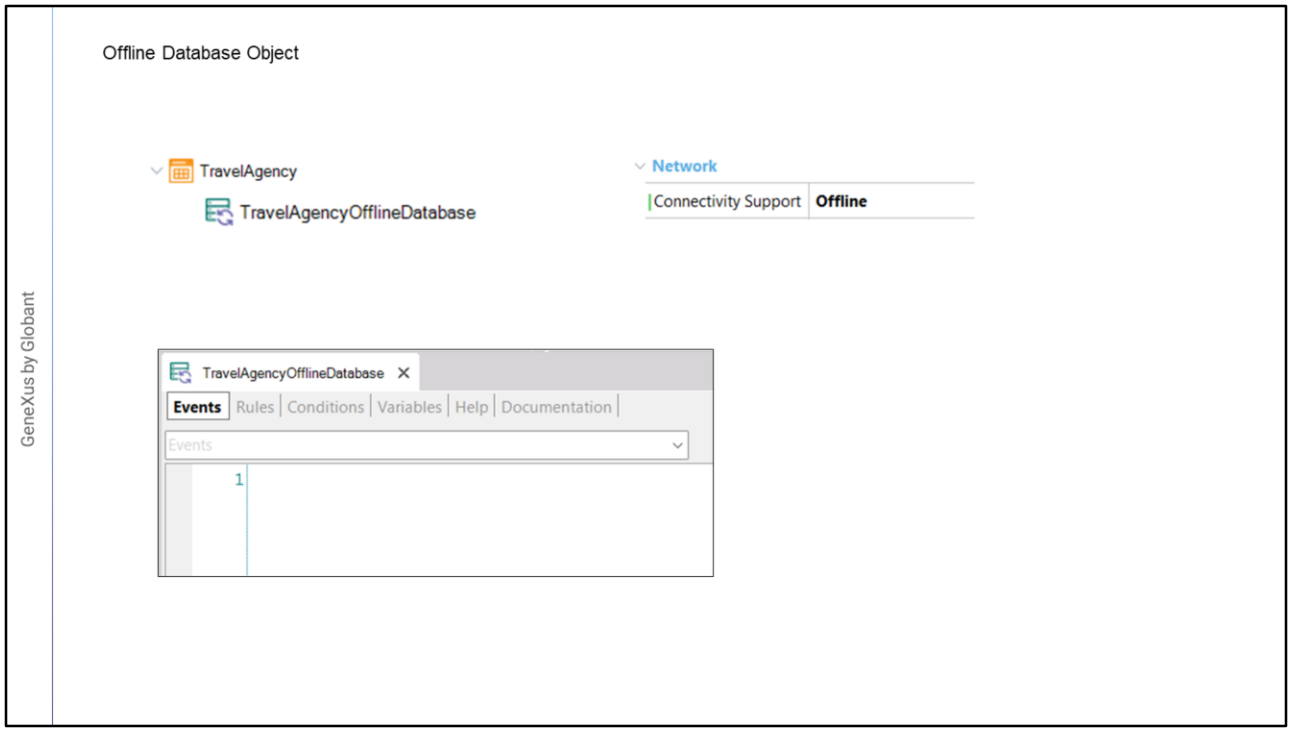
GeneXus by Globant

Offline Database Synchronization

Diego Marranghello

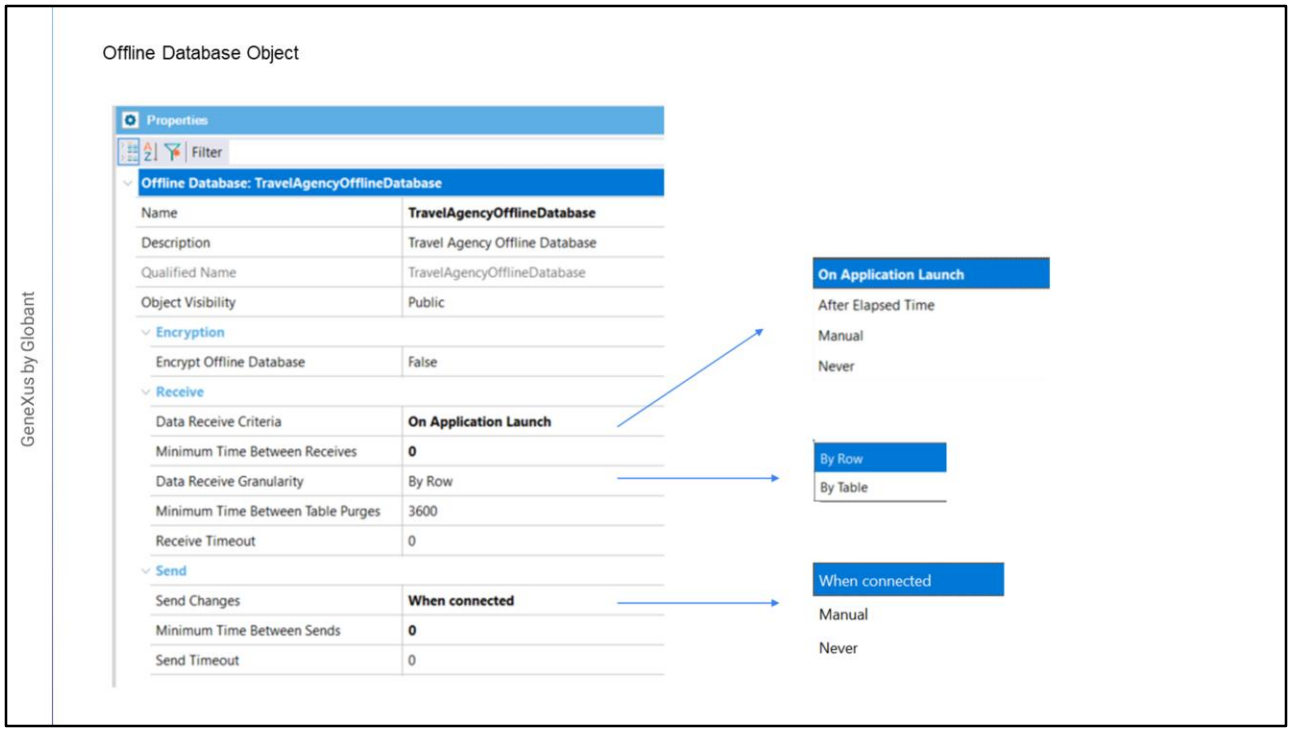


Para cada objeto main que possui a propriedade Connectivity Support = Offline, é criado um objeto denominado Offline Database.

Ao ser criado, podemos ver quais são as tabelas que serão criadas na BD local e são criados também, na linguagem nativa do dispositivo, os programas necessários para a criação desta base de dados local.

Este objeto é o responsável por determinar quando ocorre a sincronização e quais são os dados que interessam quando são sincronizados com as tabelas do server.

O objeto Offline Database também possui eventos e condições que permitem determinar seu comportamento.



Vamos ver algumas das propriedades mais importantes do objeto Offline Database.

No grupo Receive temos a propriedade “Data Receive Criteria” que será usada para o recebimento de dados, os valores que podemos utilizar são:

-Quando é iniciada a aplicação, que é o valor default, indica que a aplicação iniciará o recebimento de dados quando ela for iniciada.

-Com o valor After Elapsed Time, o recebimento será feito quando for cumprido o tempo indicado na propriedade Minimum Time Between Receives, expresso em segundos, e também será realizado quando a aplicação for inicializada.

-O valor Manual indica que o recebimento será realizado apenas de forma manual, aqui deverá ser desenvolvida uma ação para iniciar o recebimento.

-Por último, o valor Never indica que nunca será realizado o recebimento de dados por padrão e GeneXus não irá gerar os programas de sincronização necessários, os quais deverão ser implementados manualmente se for necessário.

Como já vimos, temos a propriedade “Data Receive Granularity”, que nos permite estabelecer qual será o mecanismo de recebimento a implementar, pode ser by Row que é o valor padrão ou by Table.

Depois temos a propriedade Send Changes que nos permitirá estabelecer quando

queremos realizar o envio dos dados do dispositivo para o servidor, os valores que temos são:

-When Connected, que é o valor default, indica que o envio será realizado de forma imediata quando for detectado que há conexão.

-O valor Manual indica que o envio será realizado apenas de forma manual, aqui deverá ser desenvolvida uma ação para iniciar o envio.

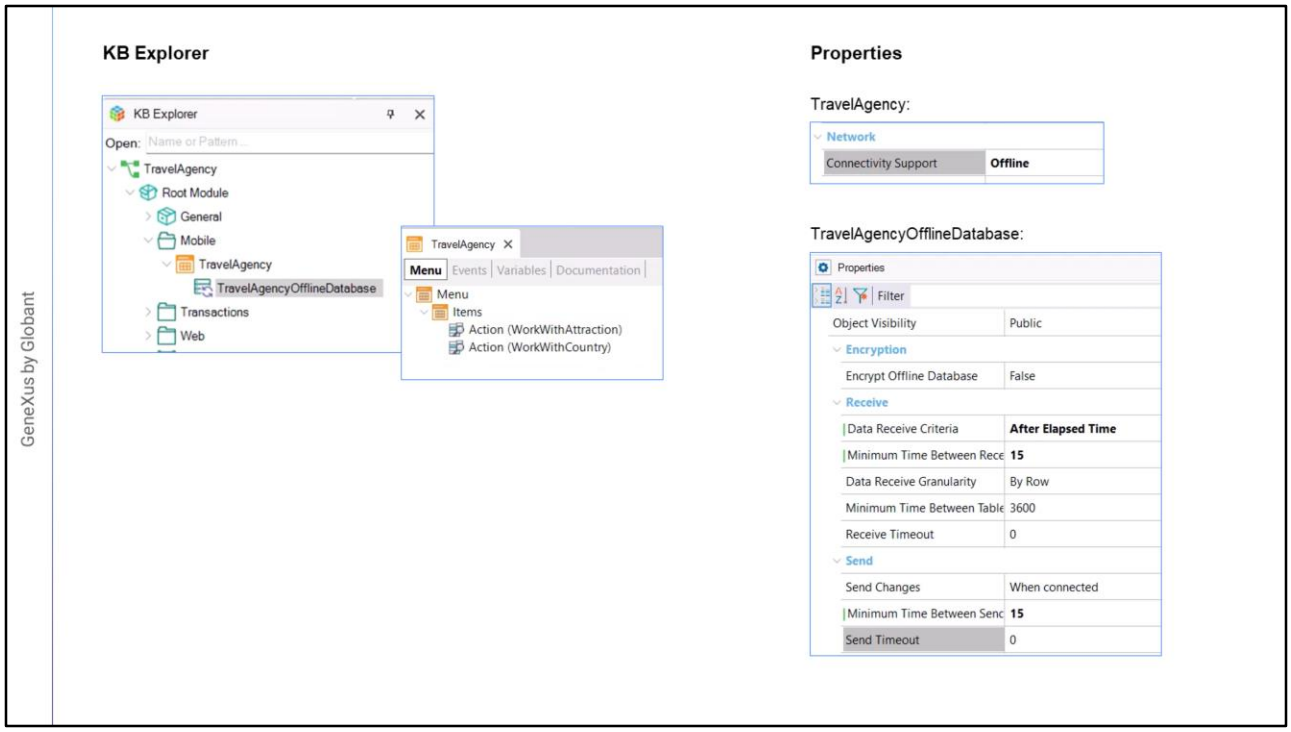
-E o valor Never indica que não serão enviados dados de forma automática por GeneXus, os registros modificados não serão armazenados na tabela auxiliar GXPendingEvent, portanto fica a responsabilidade do lado do desenvolvedor, caso seja necessário o envio, de programar toda a lógica necessária.

Seja no caso de usar para o envio ou recebimento o valor Manual ou para dar mais funcionalidade ao usuário, precisamos ser capazes de desenvolver ações que realizem a sincronização.
Para isso temos a API Synchronization.

Esta API não se encontra nas References como o restante das APIs, mas faz parte da gramática.
Possui os métodos Send e Receive para a sincronização, ServerStatus para determinar o estado do server e ResetOfflineDatabase que retorna a base de dados local ao seu estado inicial, seja fazendo um Create Database para esvaziar as tabelas ou carregando uma base de dados pré-carregada.

Com o uso desta API poderíamos, por exemplo, desenvolver um panel de opções para que o usuário dispare a sincronização quando ele quiser.
Aqui podemos ver um exemplo muito simples onde são utilizados os dois métodos em um mesmo evento.

Vejamos um exemplo em GeneXus.



Criamos parte de uma aplicação para uma agência de viagens, onde criamos um objeto Menu e como itens os objetos WorkWith de Atração e de País.

Tendo este objeto a propriedade Connectivity Support como Offline foi gerado o objeto TravelAgencyOfflineDatabase.

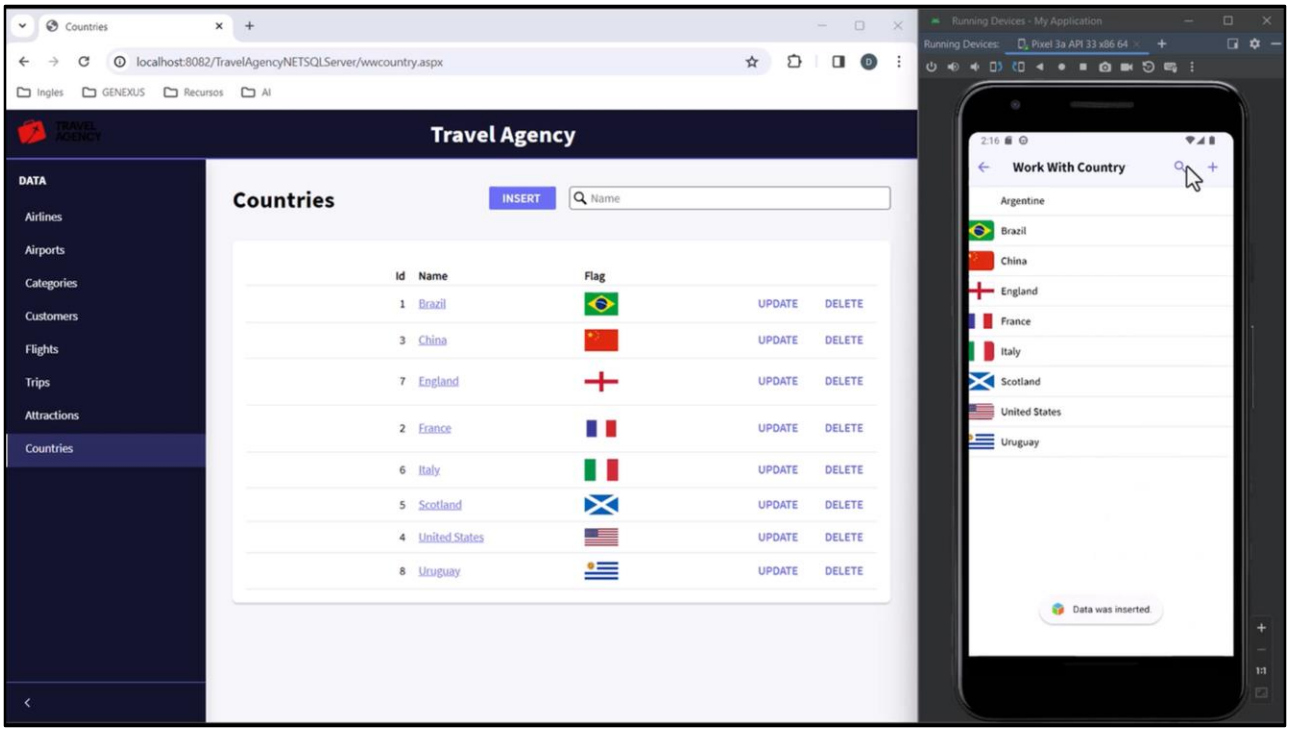
Vamos configurar este objeto para que a sincronização seja realizada a cada 15 segundos.

Vamos às propriedades e na propriedade Data Receive Criteria colocaremos o valor After Elapsed Time e em Minimum Time Between Receives colocaremos o valor 15.

Agora no grupo Send, na propriedade Minimum Time Between Sends também vamos colocar o valor 15.

Com esta configuração, o tempo mínimo que haverá entre um envio/recebimento de dados em nossa aplicação será de 15 segundos.

Vamos testar isso, salvamos e fazemos um rebuild all da aplicação.



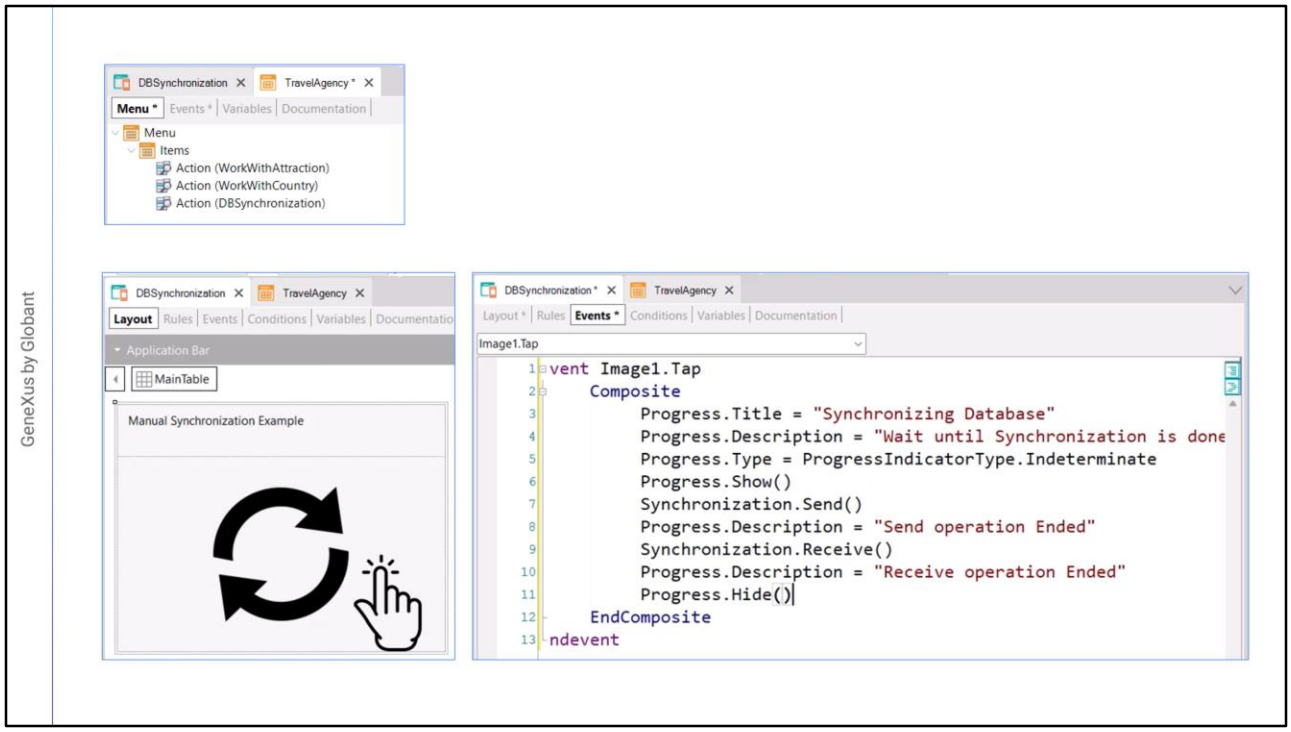
Bem, já temos as mudanças.

Vamos executar em mobile e na web, entrando em ambos os casos no Work With de Country.
Vamos criar um novo país a partir do mobile.

E ao atualizar em web, vemos que imediatamente atualiza a informação com o país que acabou de inserir, isto ocorre porque o primeiro send é automático. Mas se antes dos 15 segundos inserirmos outro registro, ao atualizar na tela web ainda não aparece. Isso ocorre porque não passaram os 15 segundos que configuramos, entre uma inserção e outra. Por não ter passado esse período, teremos que aguardar esse tempo para a sincronização.

Vamos atualizando o panel e ali aparece o país registrado.

Agora entramos no primeiro país criado e vamos excluí-lo. Confirmamos.
Vamos para a web e atualizamos. Como entre a última inserção que fizemos e a exclusão se passaram menos de 15 segundos, teremos que aguardar esse tempo desde a exclusão para que seja feita a sincronização. Pronto, a informação já está atualizada no server.



Vamos programar agora um panel que nos permita realizar a sincronização quando o usuário quiser, sem ter que esperar o tempo estabelecido.
Já tenho um Panel criado, chamado DBSynchronization, este por enquanto só tem uma imagem e um texto.

O adicionamos ao menu TravelAgency e gravamos.

Queremos que, quando o usuário toque na imagem, seja iniciada a sincronização, então vamos programar o evento Tap.

Neste evento como é do lado do cliente, vamos usar composite, lembre-se que sempre a sincronização vai começar a partir do lado do cliente.

Então escrevemos Synchronization.Send(), isto irá realizar o envio dos dados do dispositivo para o server
e Syncrhronization.Receive() para fazer o recebimento.

Além disso, iremos fornecer ao usuário algum feedback sobre a ação que está realizando, para isso utilizaremos “Progress”.
Progress é um objeto externo que nos permite mostrar como um panel onde vemos o que está acontecendo.
Para o título vamos colocar "Synchronizing Database“, este será o título.
E vamos colocar uma descrição em Progress.Description.

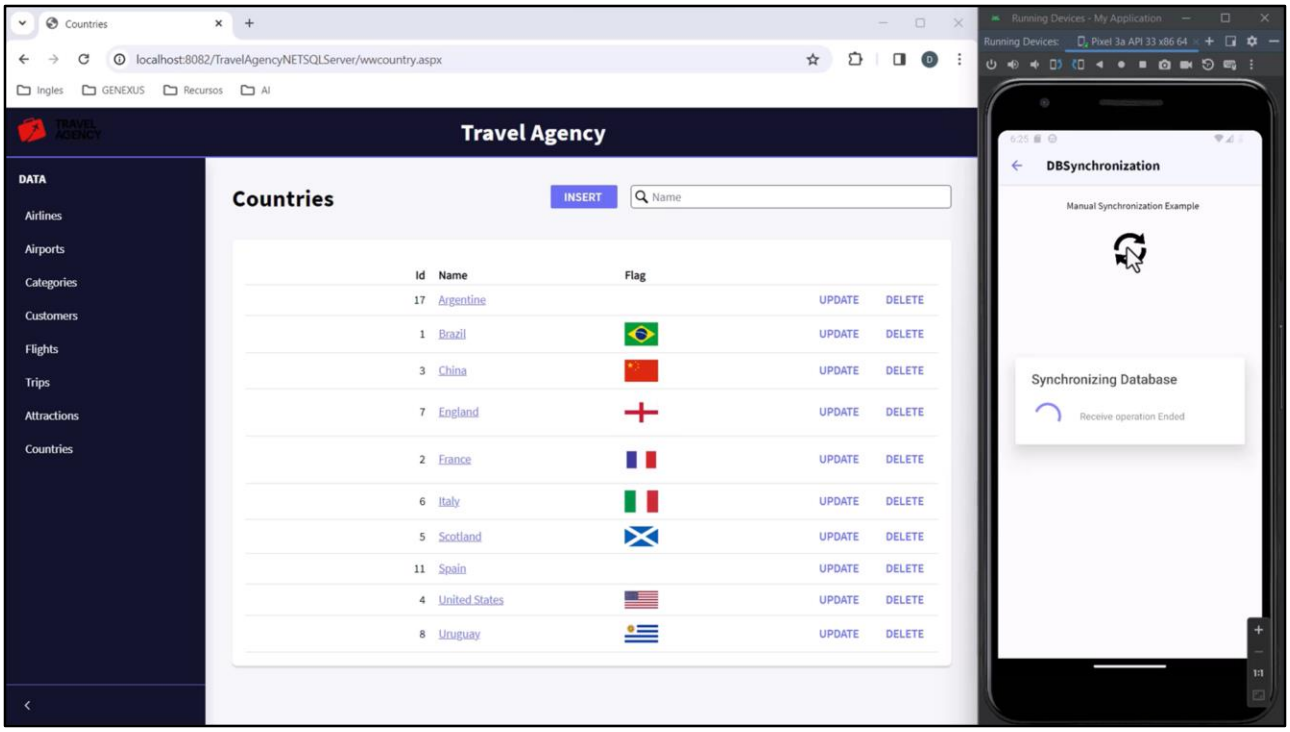
Temos que indicar o tipo de progresso que vamos usar, vamos usar indeterminado, que é usado quando não sabemos ou não podemos indicar o grau de avanço do processo.

Depois mostramos o controle com Progress.Show()

E a seguir faremos o Send, quando terminar poderíamos mudar a descrição, então vamos colocar um texto em Progress.Description

Depois fazemos o Receive, e novamente alteramos a descrição para indicar que terminou.

E a última coisa que temos que fazer é ocultar o controle, então colocamos Progress.Hide() que é o método que o oculta.



Vamos executar a aplicação.

Bem. Já temos a aplicação no emulador e na web.

Vamos adicionar novamente um país a partir da aplicação mobile e confirmamos.
Se atualizarmos na aplicação web, vemos a sincronização no momento porque o primeiro send é automático, mas se imediatamente adicionarmos outro país, e formos para a aplicação web, ele não aparece.
Não vamos esperar os 15 segundos para que sincronize, vamos fazer isso de forma manual.
Tocamos no ícone, ali nos indica que o send já foi finalizado.
E agora já vemos o país inserido.

Agora vamos excluí-lo a partir do dispositivo, confirmamos. Não vamos esperar, vamos para synchronization, tocamos e agora sim, apenas atualizamos e já vemos as mudanças.

Bem, com esta pequena demonstração pudemos ver como configurar algumas propriedades do objeto Offline Database e também vimos como é possível utilizar a API Syncrhonization para realizar as operações de envio e recebimento a pedido do usuário.


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