

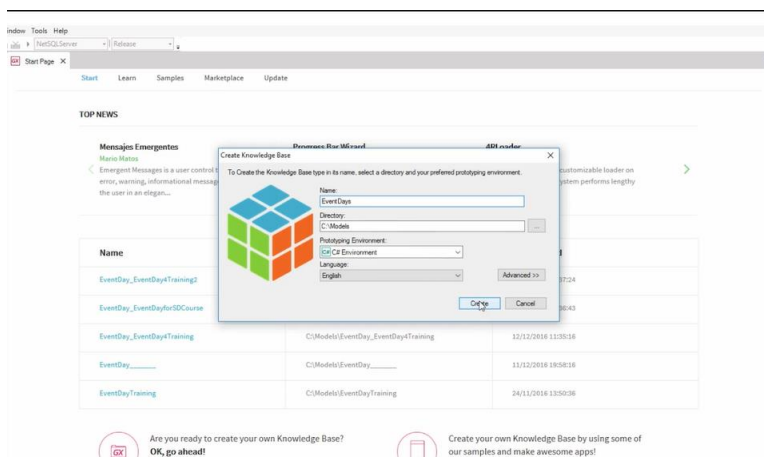
Demo: Starting to Develop the Web Application

GeneXus

Demo: starting to develop the application

Bien. Entonces ahora sí vamos a empezar a desarrollar una aplicación de cero con GeneXus; una aplicación que se parezca, para irnos acercando, a esta EventDay que veníamos viendo.

Estamos dentro de GeneXus 15; no estamos con ninguna KB abierta. Vamos a empezar por crear una nueva base de conocimiento, a la que le vamos a llamar EventDays. Vamos a crearla prototipando en C#, en este directorio de aquí. Damos Create...

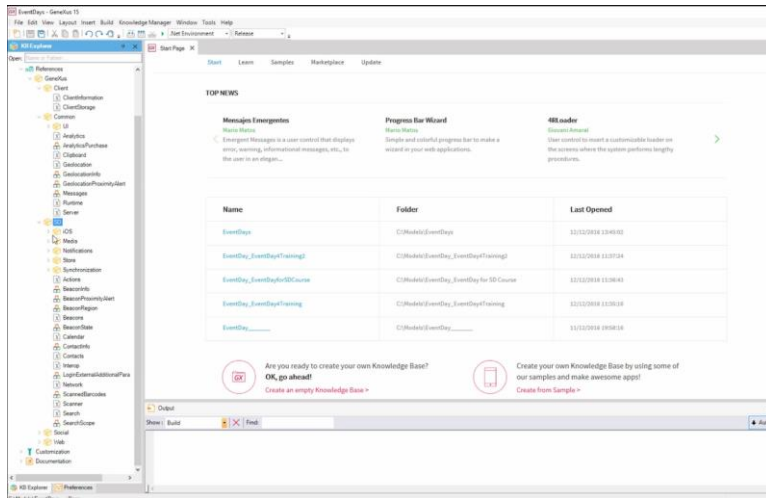


Ahora, cuando termine la creación de la base de conocimiento, vamos a ver todos los folders y módulos que va a importar automáticamente esa KB al crearse. Vamos a ver también los dominios que se van a crear automáticamente, etc.

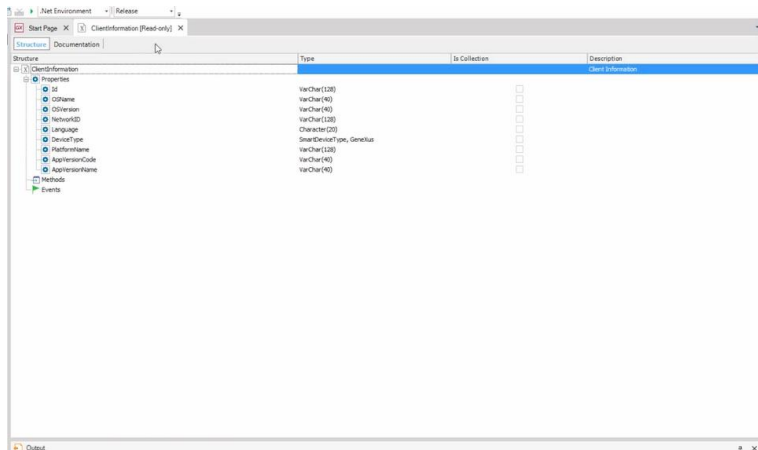
Terminó. Y vemos que nos creó un módulo Root, que contiene un folder GeneXus, dentro del cual aparecen otros dos folders: Common, con un objeto externo GlobalEvents que después vamos a mencionar mucho más adelante, y un folder Web, que va a implementar la MasterPage y demás objetos para la parte web de la aplicación que desarrollemos.

Y por otro lado, un tema muy interesante es observar este nodo References que aparece en el arbolito, que contiene este módulo GeneXus, con una serie de sub-módulos: Client, Common, prestemos especial atención a este, SD, que va a contener un montón de objetos externos y

SDTs que van a ser utilizados por nuestra aplicación para Smart Devices a la hora de interactuar con distintas funcionalidades del dispositivo.



Lo interesante a destacar de este nodo de aquí es que todo lo que está acá adentro va a ser Read-only; es decir, no va a poder ser modificado. Acá podemos ver que dice Read-only:



Y de hecho, ya va a venir implementado cuando generemos nuestra aplicación; cuando vayamos generándola a partir de las distintas etapas de nuestra prototipación, estos objetos no se van a volver a generar porque ya van a estar generados. Es decir, van a estar ya los binarios incorporados. Y esa es una de las grandes ventajas, porque entonces se va a optimizar el tiempo de generación.

Bueno, vamos a empezar por ver los dominios pre-definidos que ya vienen incorporados con la KB cuando la creamos. Vemos que hay un montón de dominios, en particular podemos ver este dominio Address, el dominio Email, y el Phone, por ejemplo... Todos estos dominios se van a conocer como **dominios semánticos**, porque van a tener incorporados cierta semántica que va a permitir que se tomen determinadas acciones cuando se encuentran estos dominios.

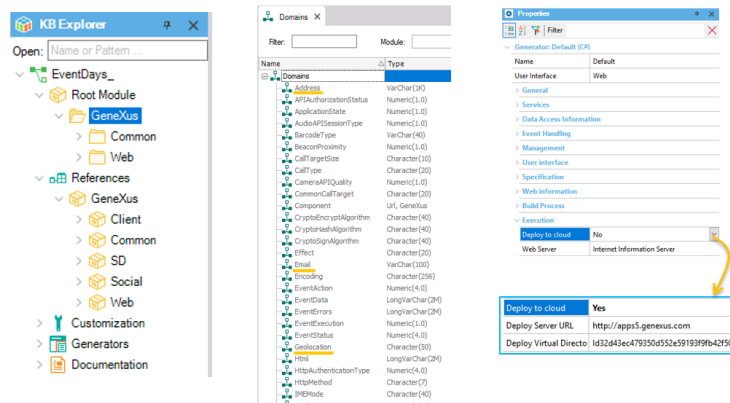
Ya lo vamos a ver ahora en un instante cuando prototipemos la aplicación.

Por otro lado, vamos a querer prototipar nuestra aplicación en la nube, no en forma local, porque de esa manera nos desentendemos de tener que tener todo el software, la base de datos, el manejador de base de datos, y demás en nuestro sistema de archivos, en nuestra red.

Vamos a Generadores... y vamos a posicionarnos sobre el generador web de nuestro Environment... y allí vamos a encontrar, abajo del todo, la propiedad **Deploy to cloud**, que por defecto vemos que está en No, pero la vamos a llevar a Yes. Y vemos que aparece la URL del Deploy Server, y aquí el directorio virtual. Y a partir de acá nos vamos a desentender entonces del software y de la base de datos, que en particular van a ser manejados en la nube.

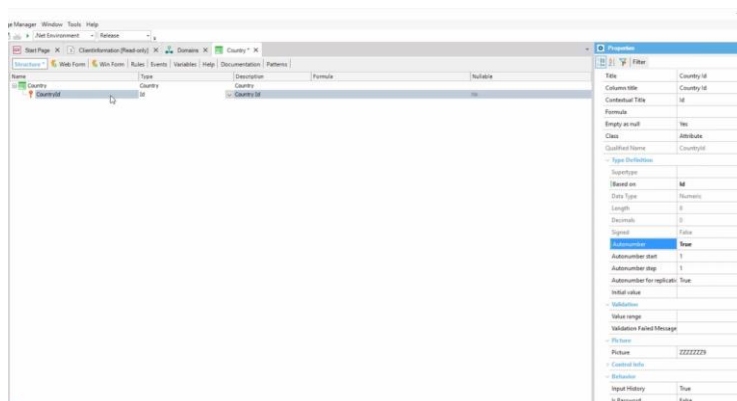


Demo: New knowledge base

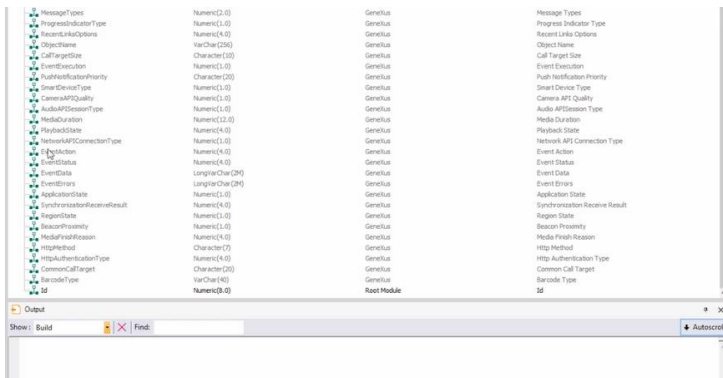


Bien. Entonces vemos que tenemos, por ahora, un solo generador en nuestro Environment: el generador default, que es C#, porque así lo elegimos.

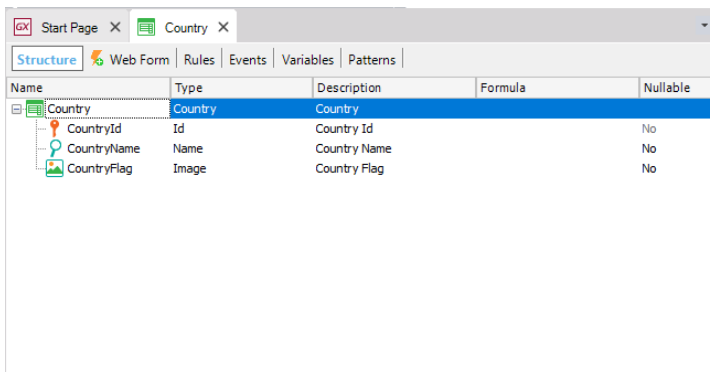
Vamos a empezar por crear dos transacciones. La primera transacción va a ser una transacción para almacenar o registrar a los países de origen de nuestros oradores. Entonces vamos a agregar un atributo CountryId como identificador; vamos a definirlo del dominio Id, numérico de 8. Vamos a decir que este CountryId vamos a querer que sea autonumerado, así que vamos a ir a la propiedad Autnumber del atributo, a ponerla en True. El dominio lo vamos a dejar sin autonumerar.



Podemos ir, de hecho, a la solapa de dominios, y ver que ahora aparece, en negrita, el nuevo dominio que ingresamos (los que son predefinidos están en gris).



Vamos a definir CountryName; también vamos a dar de alta un dominio Name, de tipo VarChar de 20. Y por último, CountryFlag, que vamos a decir que sea de tipo Image. Grabamos.



Y luego vamos a crear otra transacción: la transacción Speaker, para registrar a los oradores de nuestro evento. Identificador SpeakerId... luego vamos a ingresar SpeakerName, el nombre del orador de dominio Name... SpeakerSurname, que vamos a definirle dominio VarChar, esta vez de 40. Vamos a definir un atributo fórmula, SpeakerFullName, de tipo VarChar de 60: lo vamos a definir SpeakerSurname (le aplicamos el método trim para sacar los espacios en blanco), dejamos un espacio en blanco, y concatenamos con SpeakerName (también trim para sacarle los espacios en blanco). Luego definimos el atributo SpeakerImage, para poder guardar la foto del orador... SpeakerCVMini, Curriculum Vitae mini, que lo vamos a definir de tipo VarChar de 1024.

Por otro lado vamos a colocar el país del orador, CountryId, y su nombre. Vamos a especificar un teléfono para el orador, y veamos que cuando hago tab, me está apareciendo "Phone, GeneXus". Acá estamos viendo que se trata del dominio Phone que es un dominio semántico, como ya vamos a ver, que está dentro del módulo GeneXus que habíamos visto. Bien. Vamos a poner también SpeakerAddress, pasa lo mismo, y SpeakerEmail, sucede lo mismo. Grabamos.

New transactions

Domains:

- Id = Numeric(8)
- Name = VarChar(20)
- Surname = VarChar(30)

Ids

autonumber

Name	Type	Description	Formula	Nullable
Speaker	Speaker	Speaker		
SpeakerId	Id	Speaker Id		No
SpeakerName	Name	Speaker Name		No
SpeakerSurname	Surname	Speaker Surname		No
SpeakerFullName	VarChar(60)	Speaker Full Name	SpeakerSurname.trim() + ' ' + SpeakerName.trim()	No
SpeakerImage	Image	Speaker Image		No
SpeakerCVMini	VarChar(1K)	Speaker CVMini		No
CountryId	Id	Country Id		No
CountryName	Name	Country Name		No
SpeakerPhone	Phone, GeneXus	Speaker Phone		No
SpeakerAddress	Address, GeneXus	Speaker Address		No
SpeakerEmail	Email, GeneXus	Speaker Email		No

Pero además de tener esta estructura, vamos a querer ingresar algunas reglas para impedir, por ejemplo, que el SpeakerName quede vacío. No queremos que el SpeakerName quede vacío, entonces este error lo disparamos si SpeakerName es vacío (IsEmpty).

Por otro lado, lo mismo vamos a querer hacer con respecto al apellido del orador: no queremos que quede vacío. Entonces, lanzamos un error si SpeakerSurname está vacío.

Y por último, vamos a desplegar una regla Message, esta no va a ser un error, cuando el Curriculum Vitae quede vacío; pero va a ser un mensaje, no vamos a impedir que se grave un registro, un orador, si dejó vacío el CV. Simplemente le vamos a avisar al usuario que no debería haberlo dejado vacío.

New transactions

Domains:

- Id = Numeric(8)
- Name = VarChar(20)
- Surname = VarChar(30)

Ids

autonumber

Name	Type	Description	Formula	Nullable
Speaker	Speaker	Speaker		
SpeakerId	Id	Speaker Id		No
SpeakerName	Name	Speaker Name		No
SpeakerSurname	Surname	Speaker Surname		No
SpeakerFullName	VarChar(60)	Speaker Full Name	SpeakerSurname.trim() + ' ' + SpeakerName.trim()	No
SpeakerImage	Image	Speaker Image		No
SpeakerCVMini	VarChar(1K)	Speaker CVMini		No
CountryId	Id	Country Id		No
CountryName	Name	Country Name		No
SpeakerPhone	Phone, GeneXus	Speaker Phone		No
SpeakerAddress	Address, GeneXus	Speaker Address		No
SpeakerEmail	Email, GeneXus	Speaker Email		No

```

Error( 'The Speaker Name must not be empty')
if SpeakerName.IsEmpty();

Error( 'The Speaker Surname must not be empty')
if SpeakerSurname.IsEmpty();

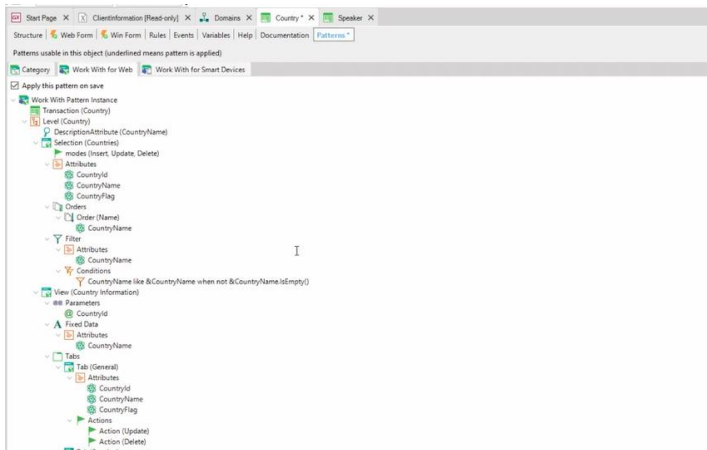
Msg( 'Curriculum Vitae should not be empty')
if SpeakerCVMini.IsEmpty();

```

Grabamos. Tenemos entonces dos transacciones.

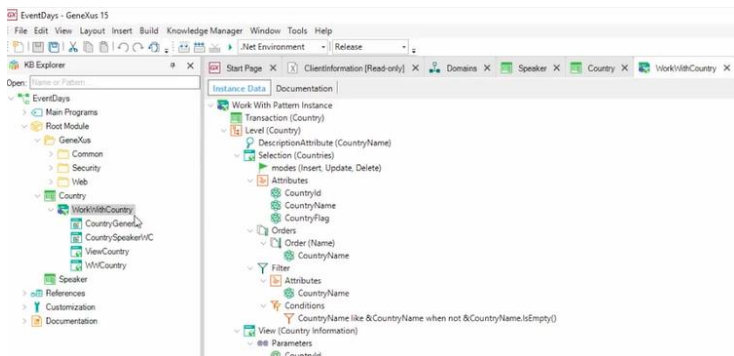
Si empezamos a desarrollar entonces el backend web de nuestra EventDay, podríamos entonces pensar que lo primero que nos gustaría hacer es aplicarle el pattern Work With for Web a estas dos transacciones. ¿Cómo aplicábamos el pattern?

Recordémoslo. Estamos en la transacción Country, vamos a la solapa Patterns, y allí encontramos el tab Work With for Web... y simplemente marcábamos este checkbox...

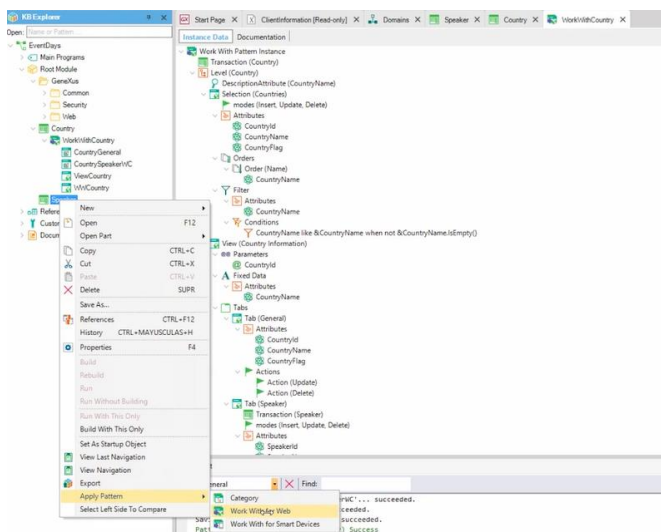


...y al grabar se iban a generar tanto el archivo de instancia, que es este archivo donde se determina qué cosas se van a mostrar en el Work With, es decir se personaliza, como los objetos que implementan cada parte del Work With: el List y el View.

Entonces acá si hacemos doble click vemos que tenemos el archivo de instancia, que no es un objeto, y debajo sí tenemos los objetos que implementan ese patrón; objeto GeneXus: objetos Web Panel, Web Component, etc.

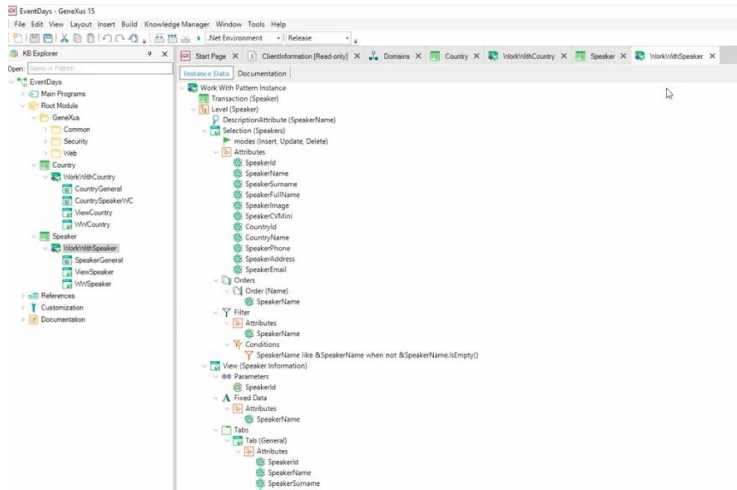


Por otro lado, lo mismo para el Speaker; podríamos hacer lo mismo, o directamente con botón derecho... Apply Pattern... Work With for Web...



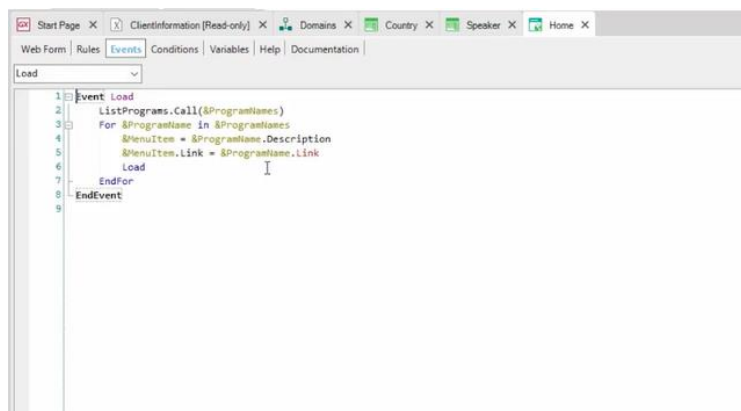
...y automáticamente entonces se va a aplicar el pattern Work With a esta transacción para web.

Si miramos, vemos que acá tenemos el archivo de instancia y lo abrimos... vemos que el archivo de instancia se puede abrir de forma independiente, tanto dentro de la transacción yendo a la solapa Patterns, como independientemente.



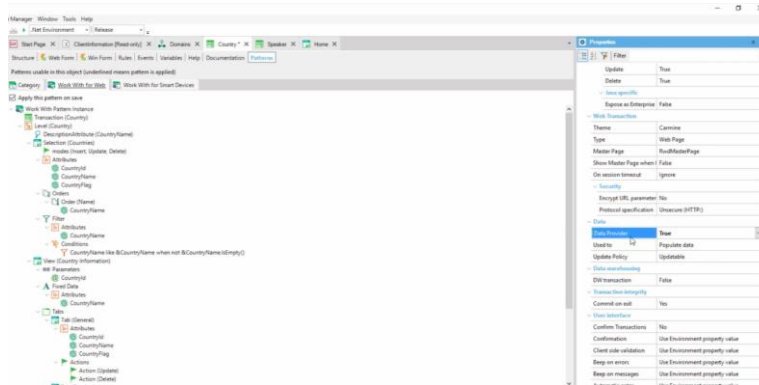
Bien. Entonces tenemos aplicado el Work With. Ahora vamos a ponerle el Autonumber en True a SpeakerId.

Vamos a querer ejecutar la aplicación para web, esta especie de backend web que estamos empezando a construir. Una cosa que pasó automáticamente cuando aplicamos el patrón fue que dentro del folder Web se crearon nuevos objetos, entre ellos el objeto Home, que por defecto va a llamar a estos dos Work With. Va a estar implementado allí dentro en forma automática la invocación; en este ListPrograms se cargan los programas a ser llamados, que van a ser los dos Work Withs y se van a mostrar allí.



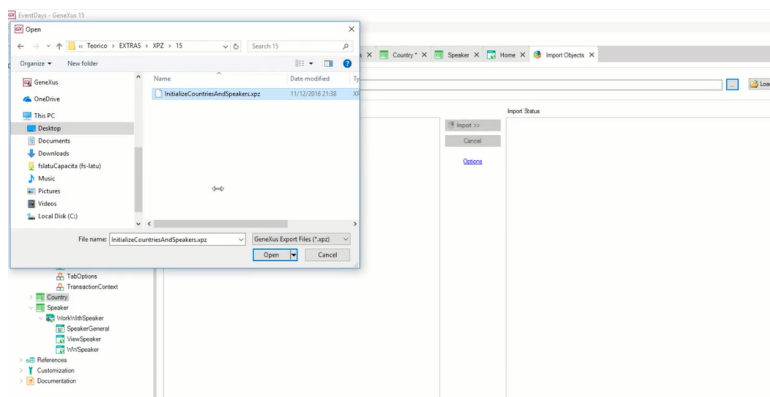
Pero además, voy a querer inicializar con datos estas transacciones. No lo quiero hacer a mano, no queremos perder tiempo ingresando sus valores. A partir de GeneXus 15, uno lo que haría para inicializar esas transacciones con datos sería ir a la propia transacción, y la propiedad **Data Provider** ponerla en True... dejar la propiedad **Used to** para Populate data, porque es lo que queremos, es decir poblar con datos, y al salvar se va a crear un Data

Provider asociado a la transacción. Ahí, en ese Data Provider, es donde vamos a declarar cómo se carga, qué registros vamos a cargar entonces en esa transacción, y eso se invocaría solo cuando diéramos el primer F5; así cuando se reorganice la base de datos, automáticamente se va a invocar a ese Data Provider para cargar automáticamente la tabla con datos.

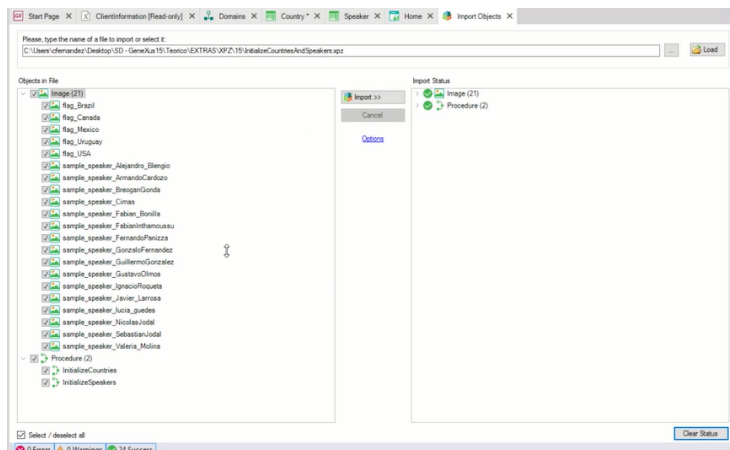


No lo voy a hacer, porque no me da el tiempo, y ya tenía yo cargados datos a la vieja usanza.

Voy a importar dos procedimientos que hacían lo mismo. Entonces lo que voy a hacer es ir a buscar... en XPZ tengo un .xpz...

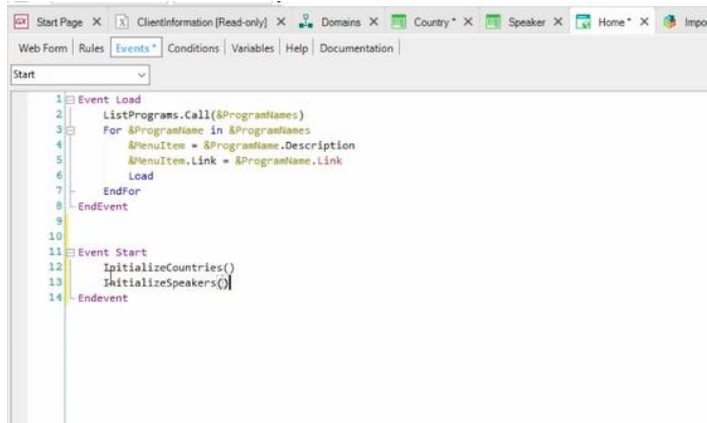


...que me va a cargar esos dos procedimientos para inicializar con datos esas dos transacciones, las tablas, y me va a cargar las imágenes de las banderas de los países y de los oradores. Entonces importo...



Esto en realidad lo estamos haciendo así simplemente porque yo ya lo tenía hecho, la idea sería usar el Data Provider asociado a la transacción.

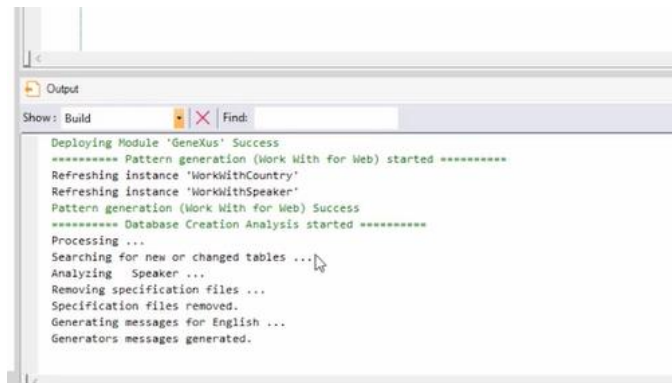
Bueno, ya fue exitosa la importación, entonces lo único que voy a hacer ahora es en el Home, en el evento Start, invocarlos: InitializeCountries e InitializeSpeakers.



```
1 | Event Load
2 |   ListPrograms.Call(&ProgramNames)
3 |   For &ProgramName in &ProgramNames
4 |     &MenuItem = &ProgramName.Description
5 |     &MenuItem.Link = &ProgramName.Link
6 |   Load
7 | EndFor
8 | EndEvent
9 |
10 |
11 | Event Start
12 |   InitializeCountries()
13 |   InitializeSpeakers()
14 | EndEvent
```

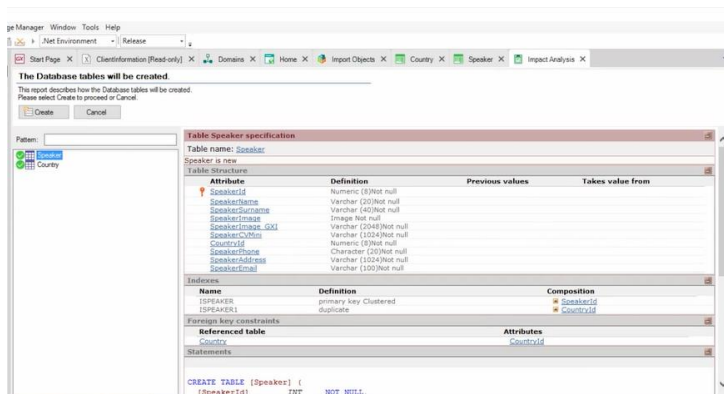
Grabo.

Entonces ahora queremos probar el backend web, para recién después pasar al desarrollo de nuestra aplicación para Smart Devices. Lo que voy a hacer es simplemente un F5, y vamos a prestar atención a esta ventana de Output y lo que nos va informando.



```
Deploying Module 'GeneXus' Success
***** Pattern generation (Work With for Web) started *****
Refreshing instance 'WorkWithCountry'
Refreshing instance 'WorkWithSpeaker'
Pattern generation (Work With for Web) Success
***** Database Creation Analysis started *****
Processing ...
Searching for new or changed tables ...
Analyzing Speaker ...
Removing specification files ...
Specification files removed.
Generating messages for English ...
Generators messages generated.
```

Bueno, vemos que nos pide reorganizar la base de datos. ¿Dónde está reorganizando esta base de datos? En la nube.



The Database tables will be created.

Table Speaker specification

Table name: Speaker

Speaker is new

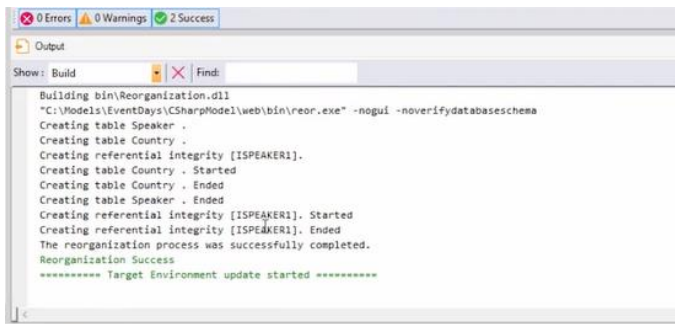
Attribute	Definition	Previous values	Takes value from
SpeakerId	Numeric (8)Not null		
SpeakerName	Varchar (20)Not null		
SpeakerSurname	Varchar (40)Not null		
SpeakerImage	Image Not null		
SpeakerImage_GDI	Varchar (2048)Not null		
SpeakerCountry	Varchar (1024)Not null		
CountryId	Numeric (8)Not null		
SpeakerPhone	Character (120)Not null		
SpeakerAddress	Varchar (1024)Not null		
SpeakerEmail	Varchar (100)Not null		

Name	Definition	Composition
ISPEAKER	primary key Clustered	SpeakerId
ISPEAKER1	duplicate	SpeakerId

Referenced table	Attributes
Country	CountryId

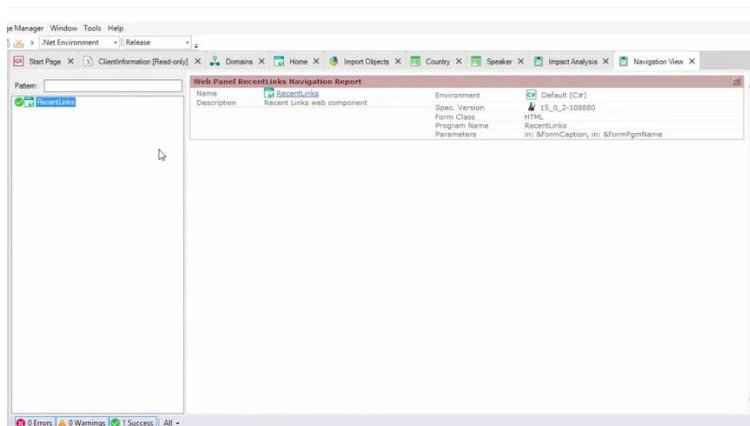
```
CREATE TABLE [Speaker] (
[SpeakerId] INT NOT NULL
```

Vemos que está empezando a generar los programas C#...acá está empezando a reorganizar... ya creó la tabla Speaker, la tabla Country...

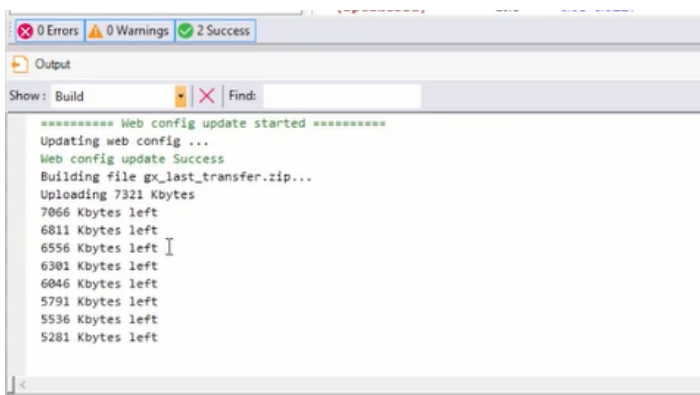


```
0 Errors 0 Warnings 2 Success
Output
Show: Build Find:
Building bin\Reorganization.dll
"C:\Models\EventDays\CSharpModel\web\bin\reor.exe" -negui -noverifydatabaseschema
Creating table Speaker .
Creating table Country .
Creating referential integrity [ISPEAKER1].
Creating table Country . Started
Creating table Country . Ended
Creating table Speaker . Ended
Creating referential integrity [ISPEAKER1]. Started
Creating referential integrity [ISPEAKER1]. Ended
The reorganization process was successfully completed.
Reorganization Success
***** Target Environment update started *****
```

Comienza la especificación... acá nos muestra el listado de navegación... vemos que empieza a mostrarnos las navegaciones...

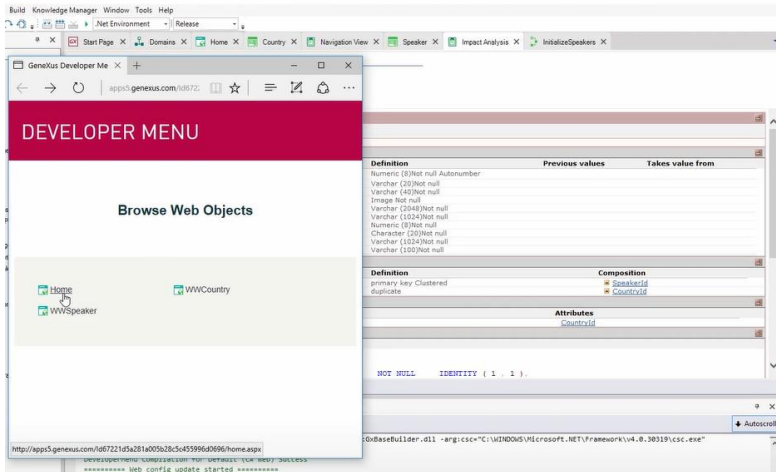


Está empezando a subir la aplicación a la nube; nos informa los bytes que van quedando.



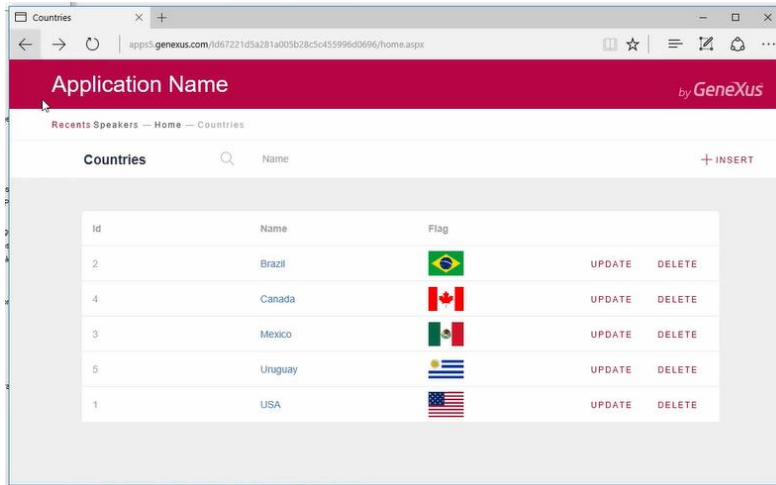
```
0 Errors 0 Warnings 2 Success
Output
Show: Build Find:
***** Web config update started *****
Updating web config ...
Web config update Success
Building file gx_last_transfer.zip...
Uploading 7321 Kbytes
7066 Kbytes left
6811 Kbytes left
6556 Kbytes left
6301 Kbytes left
6046 Kbytes left
5791 Kbytes left
5536 Kbytes left
5281 Kbytes left
```

Comienza la ejecución...bueno y aquí terminó, y vemos que levantó el Developer Menu, con el Home y los dos Work Withs que teníamos.

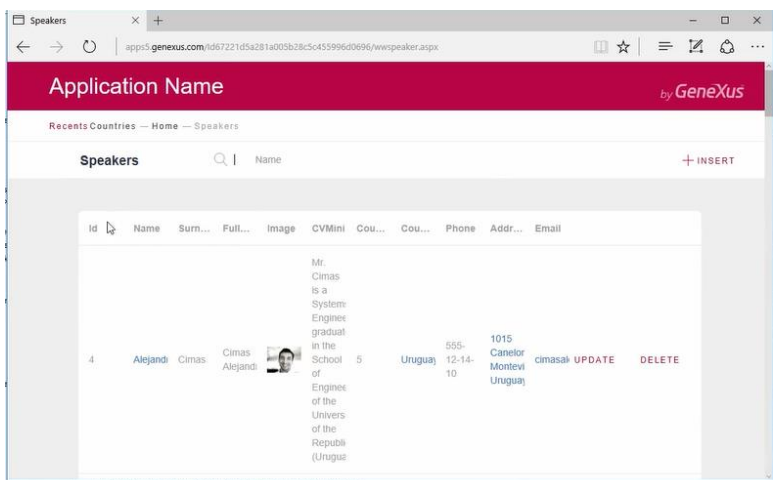


Si vamos al Home se va a ejecutar entonces el evento Start, que va a invocar a esos dos procedimientos para inicializar con datos los objetos.

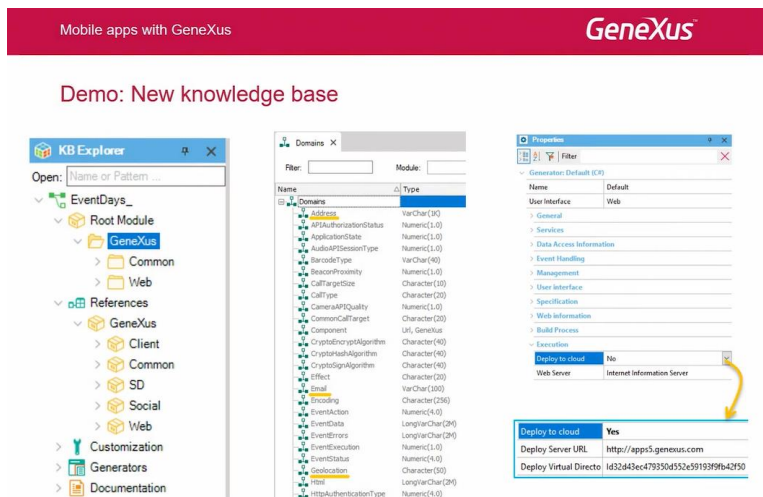
Y acá vemos, entonces, a los países cargados...



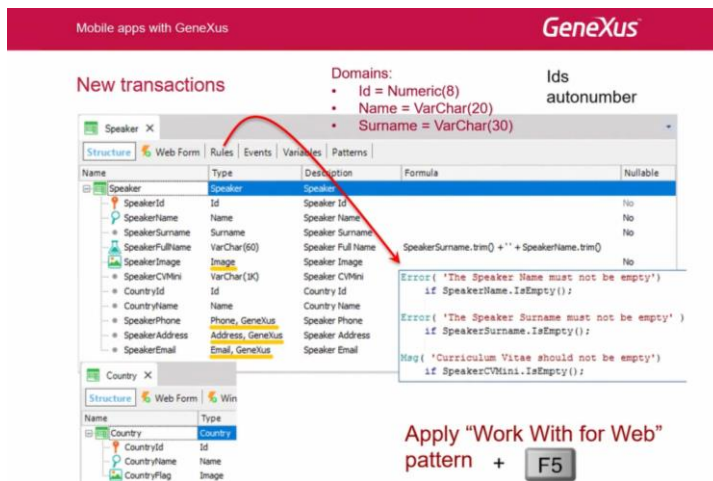
Y aquí a los oradores.



Entonces: habíamos dicho que habíamos hecho esta serie de cosas:



Habíamos aplicado el Work With for Web a las dos transacciones...



Y ahora lo que vamos a hacer es lo mismo, aplicar el Work With, pero para Smart Devices, y vamos a ver qué es lo que sucede.

