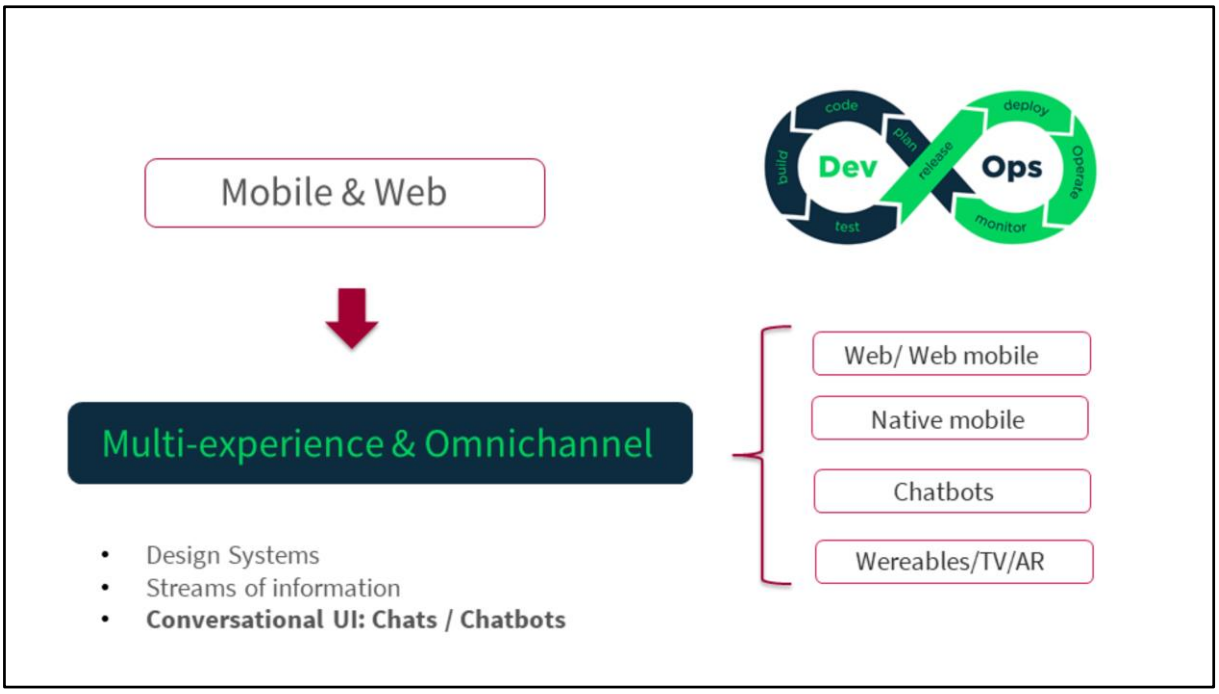
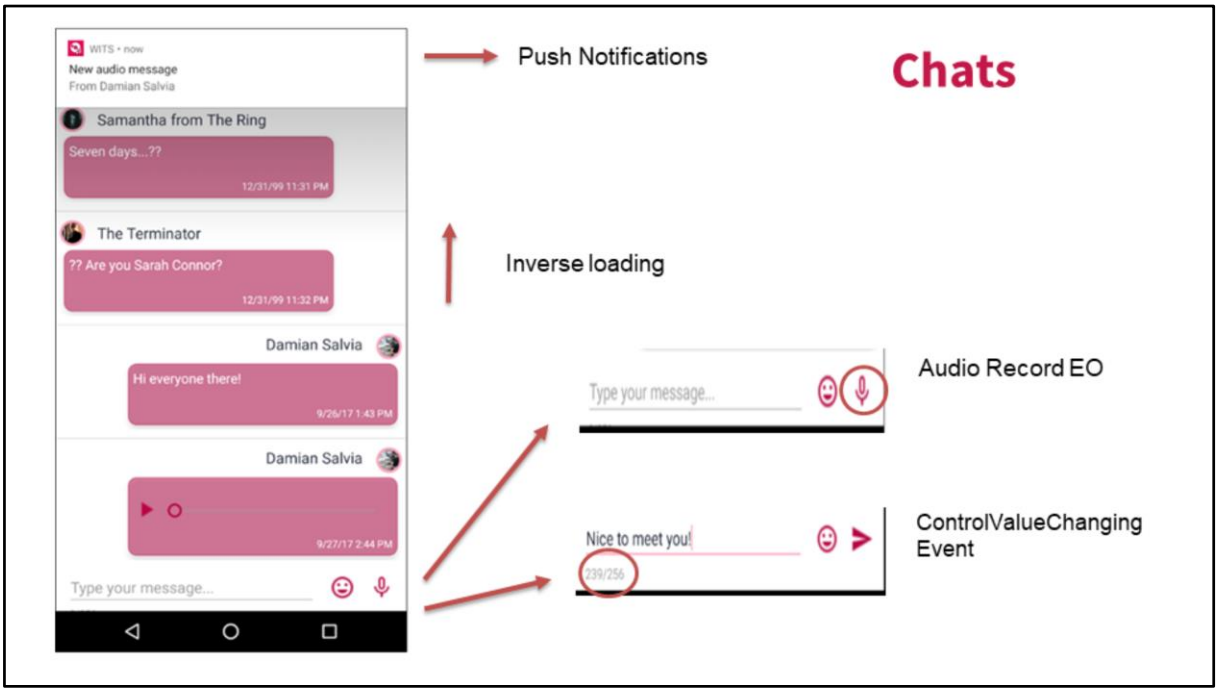

GeneXus™
The power of doing

# Conversational UI:
# Chats / Chatbots

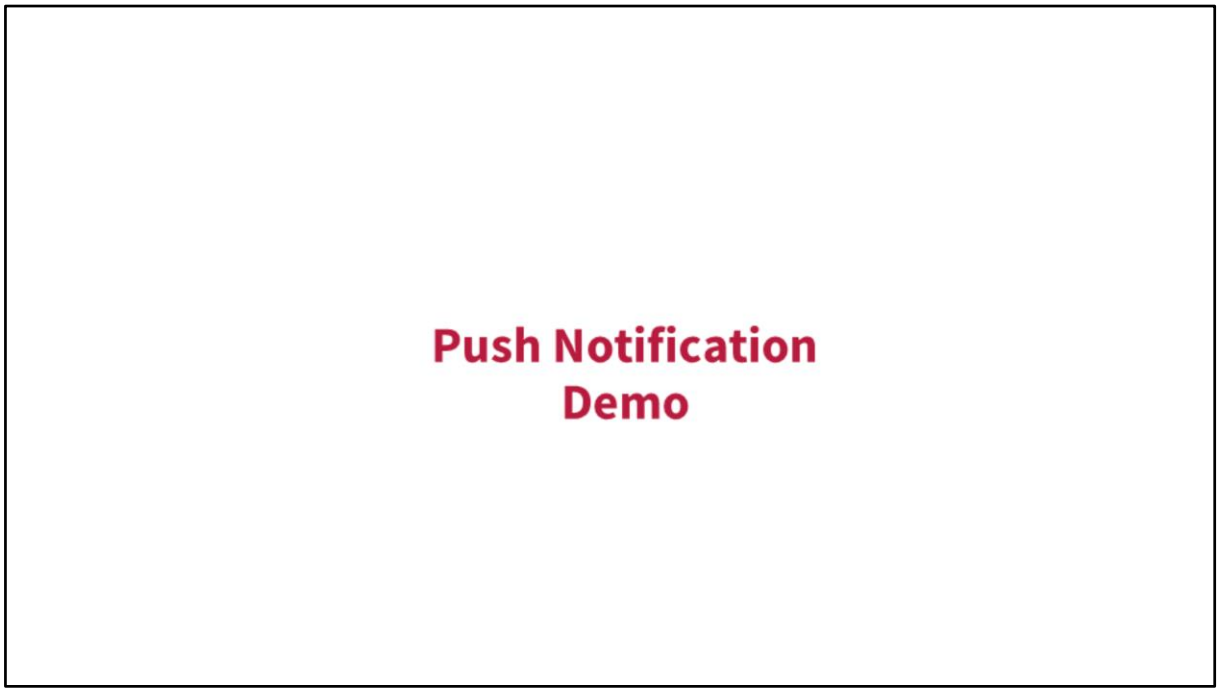
GeneXus 16




Ahora veremos el siguiente punto.
Las características para implementar Chats simples y la facilidad para implementar Chatbots.

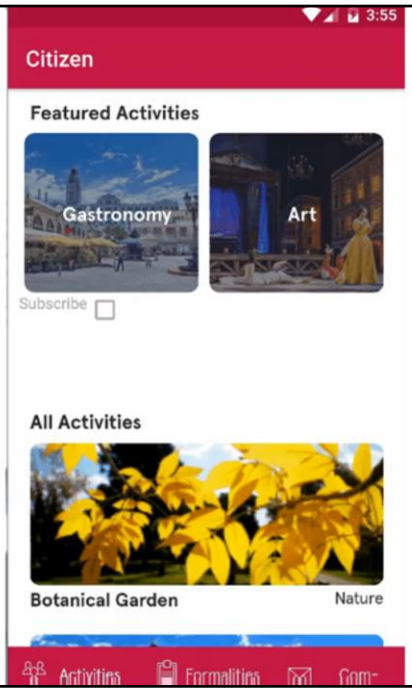



Para implementar un Chat, contamos con:
Inverse Loading, es uina propiedad de un Grid que permite visualizar la última información de abajo hacia arriba.
Audio Record, es un External Object que permite grabar una pieza de audio.
ControlValueChanging, es un evento que permite saber si el usuario ha cambiado el valor de un campo habitable, en este ejemplo se puede utilizar para llevar el conteo de los caracteres del mensaje que se va a enviar.
Y las Push Notifications, permiten informar al usuario sobre algún evento mediante una alerta, en este contexto informamos que hay una nuevo mensaje recibido.

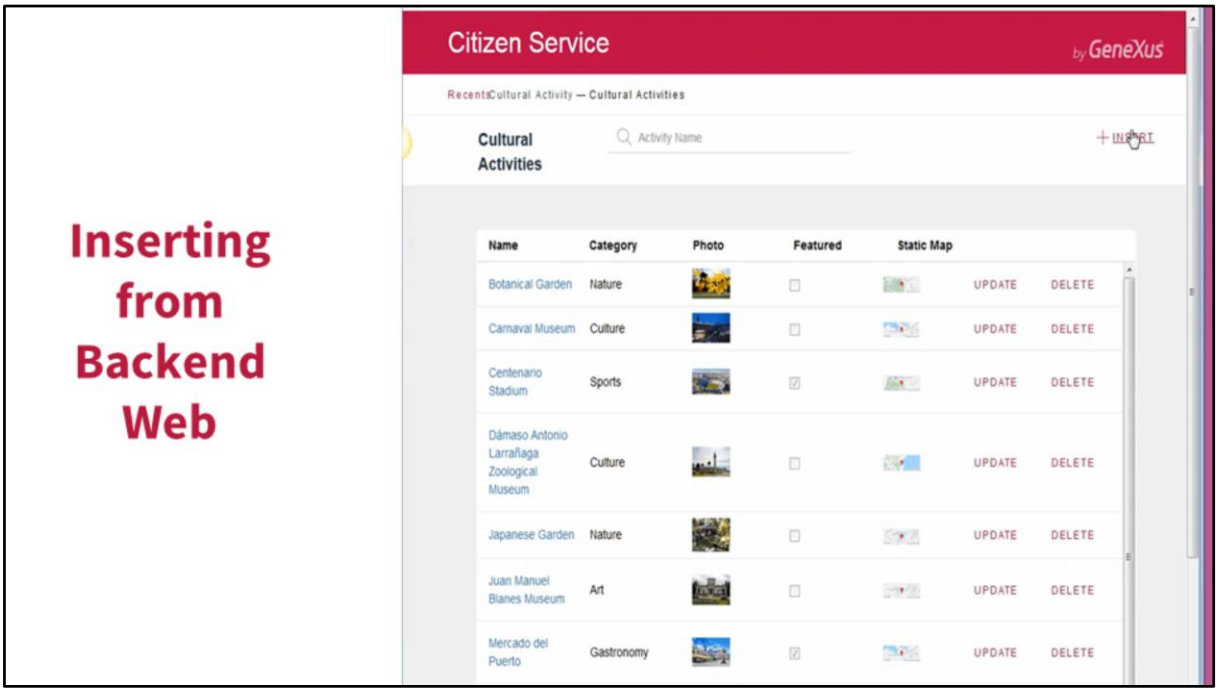



Nos enfocaremos ahora en las Push Notification y la implementación de la socket API. (Sobre las novedades de la versión 16, hablaremos más tarde)
Veamos un ejemplo en ejecución.

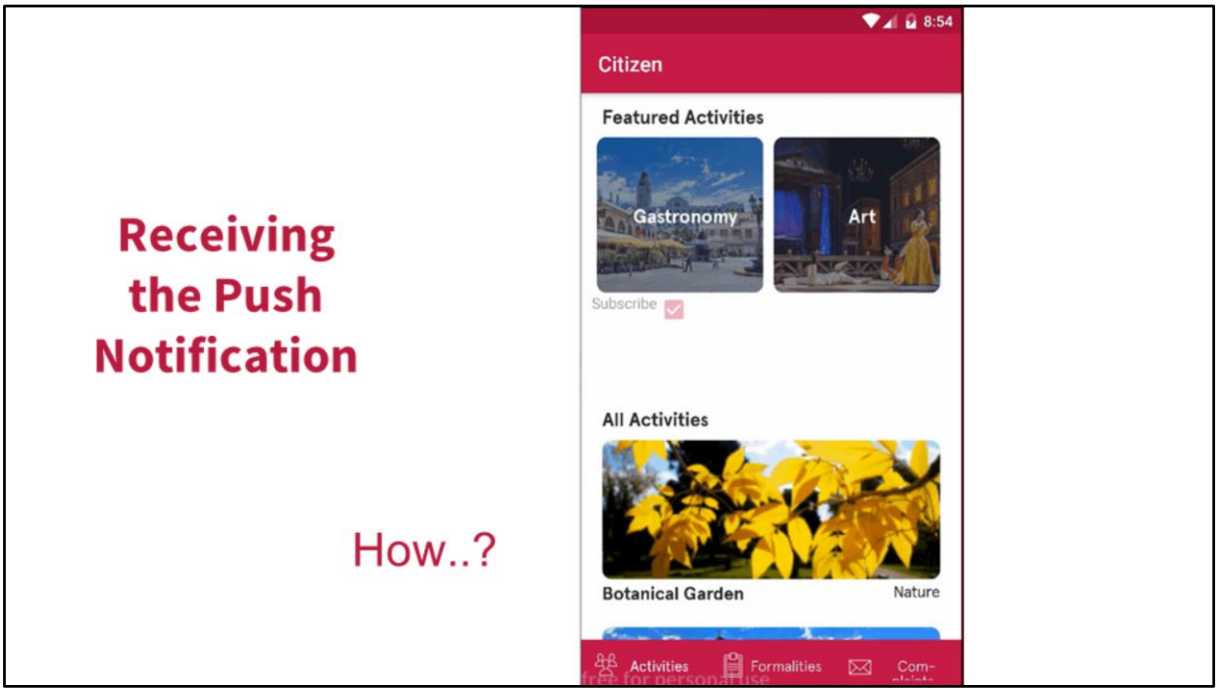
## Subscribing



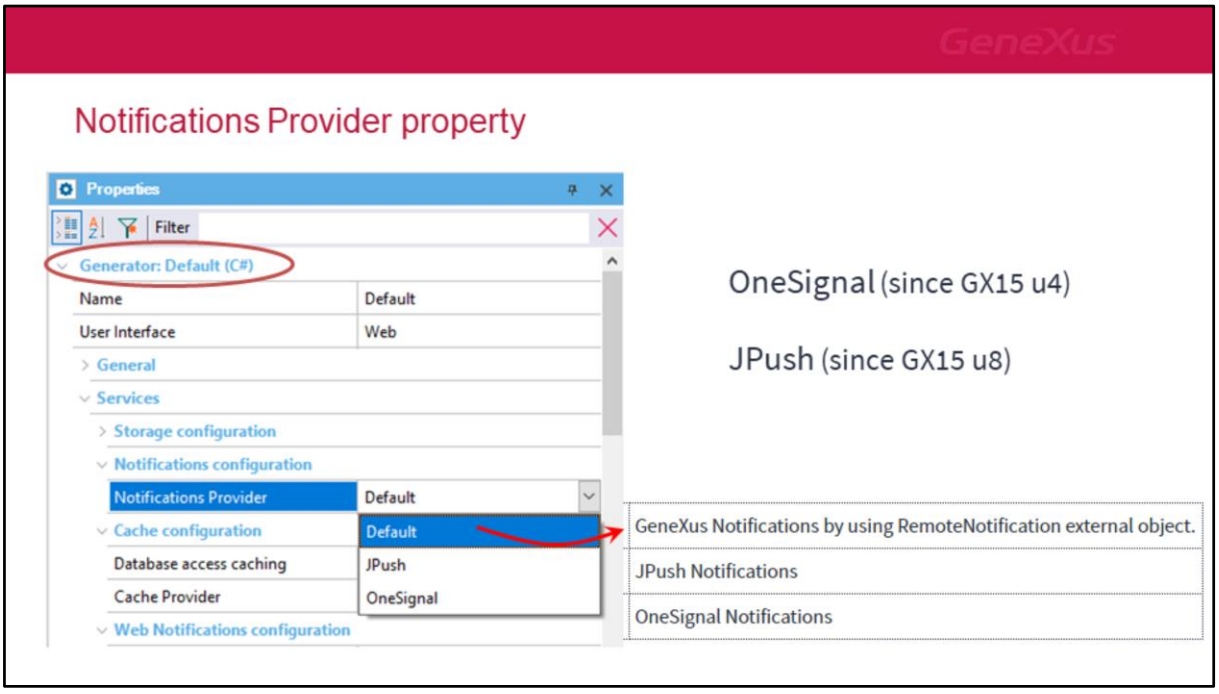

Desde nuestra aplicación Citizen el usuario tiene la opción de suscribirse para recibir notificaciones sobre las nuevas actividades que hay.



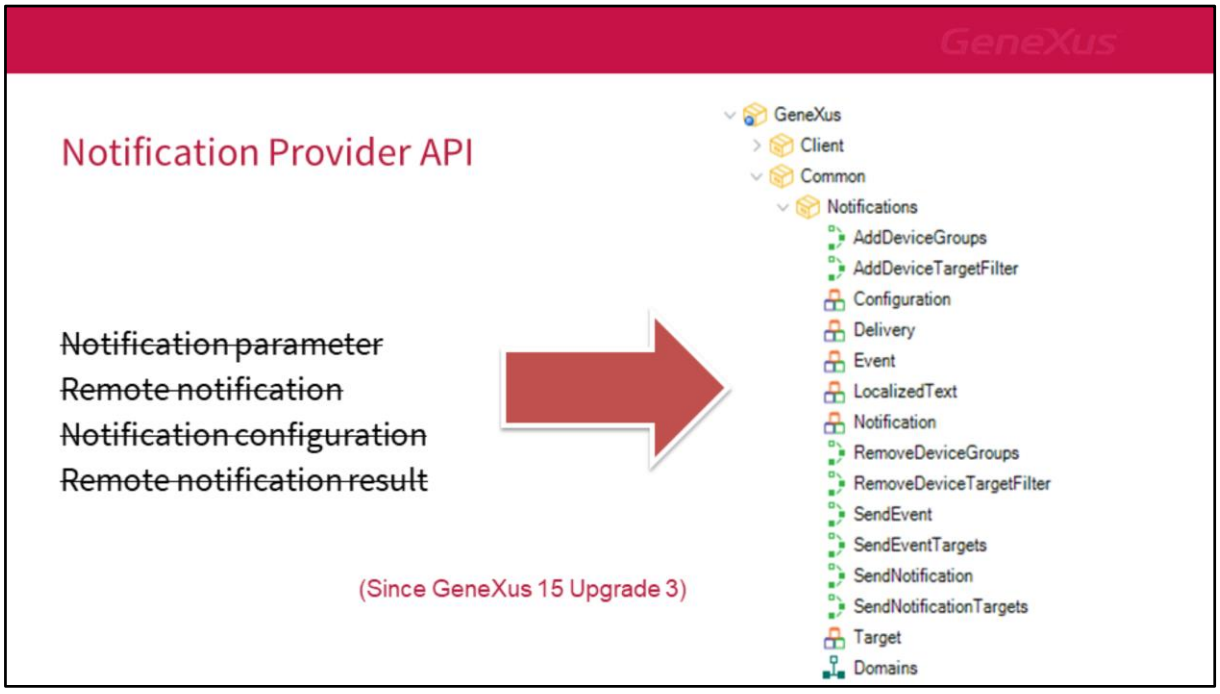

Posteriormente, desde backend web alguien hace la inserción de una nueva actividad y al momento de que nosotros realizamos la confirmación, se envía esta notificación al usuario.



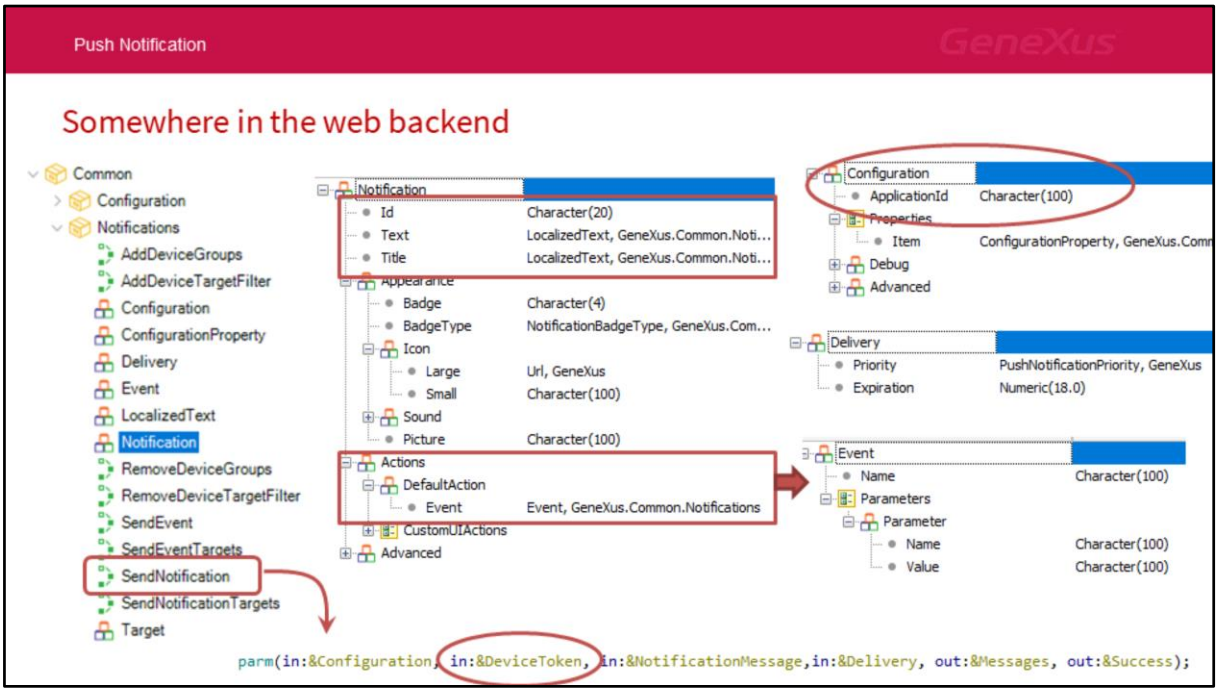

El usuario, en su dispositivo recibe la notificación y al dar tapp sobre ella verá el detalle de esta actividad que puede ser de su interés.
¿Y cómo se realiza esto?




Para ello contamos con la propiedad Notification Provider a nivel de generador web que nos permite elegir el proveedor interno con el que manejaremos las notificaciones, al elegir un proveedor u otro, se nos desplegaran diversas propiedades para poder configurar.



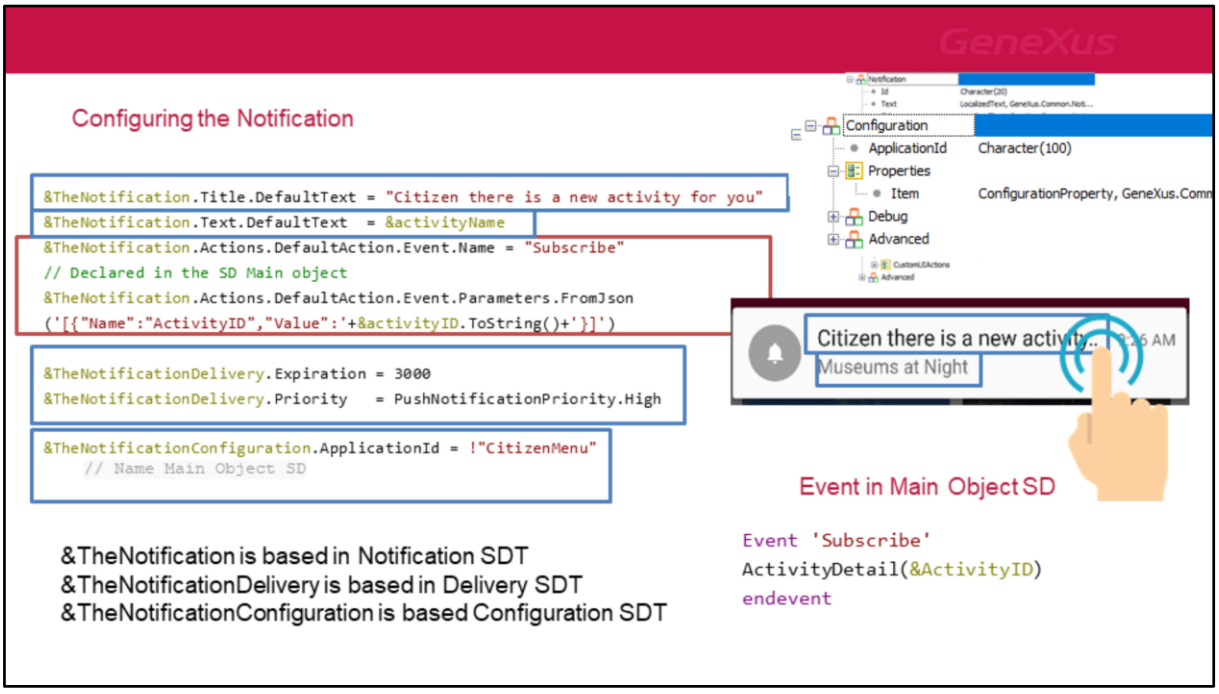

Anteriormente usábamos varios External objects para el manejo de las notificaciones, ahora las unificamos en una sola API de nombre “notification Provider” que encontramos en el módulo GeneXus > Common > Notifications.




Con esta API podemos configurar la notificación utilizando variables del tipo SDT y enviándolas al backend web.

Detengamos a analizar los SDT que utilizaremos:
El SDT Notification nos permite configurar el contenido de la notificación, así como la acción que desencadenará algún evento, por ejemplo, abrir un SD panel pasando parámetros.
Configuration, permite indicar el nombre del objeto main SD, donde se ejecutará la notificación en su ítem Application ID.
Delivery, permite configurar la prioridad de una notificación y expiración de la misma.

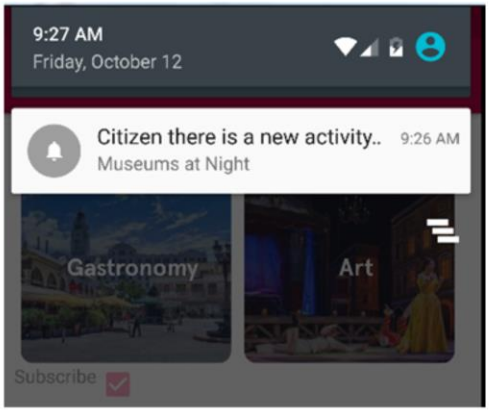
Por último, tenemos algunos procedimientos para poder enviar la notificación, en este caso tenemos el ejmplo del SendNotification, que se encarga de mandar una notificación al dispositivo con las configuraciones realizadas.




Dentro de la notificación podemos configurar:
El título
El texto que se va a mostrar
Así como el evento que fue definido en el objeto main SD y el que se encarga de abrir el detalle de la actividad una vez que el usuario dio el tapp sobre la notificación.
Después configuramos la prioridad y expiración de la notificación.
Y por último, configuramos el AplicationID, el cual es el nombre de nuestro objeto main SD.

## Sending the Notification

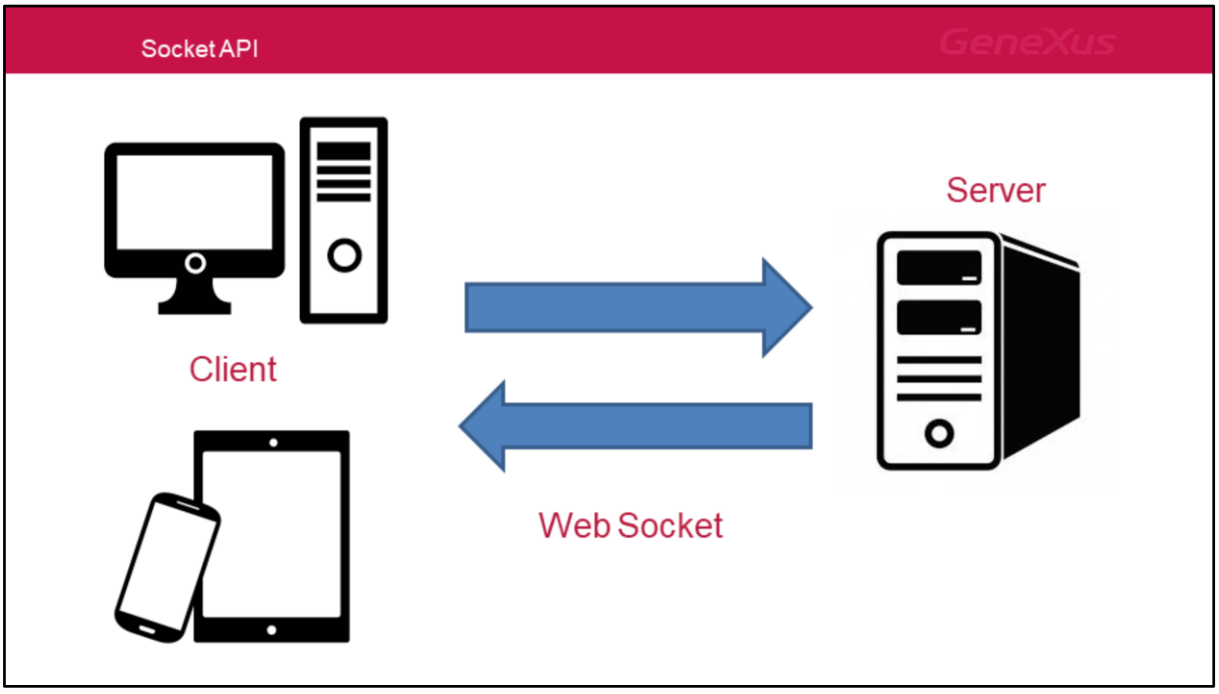
```
For each Device
   Where DeviceSubscribe= true

  GeneXus.Common.Notifications.SendNotification
  (
       &TheNotificationConfiguration,
       DeviceToken, // Target device token
       &TheNotification,
       &TheNotificationDelivery,
       &OutMessages,
       &IsSuccessful
  )
endfor
```



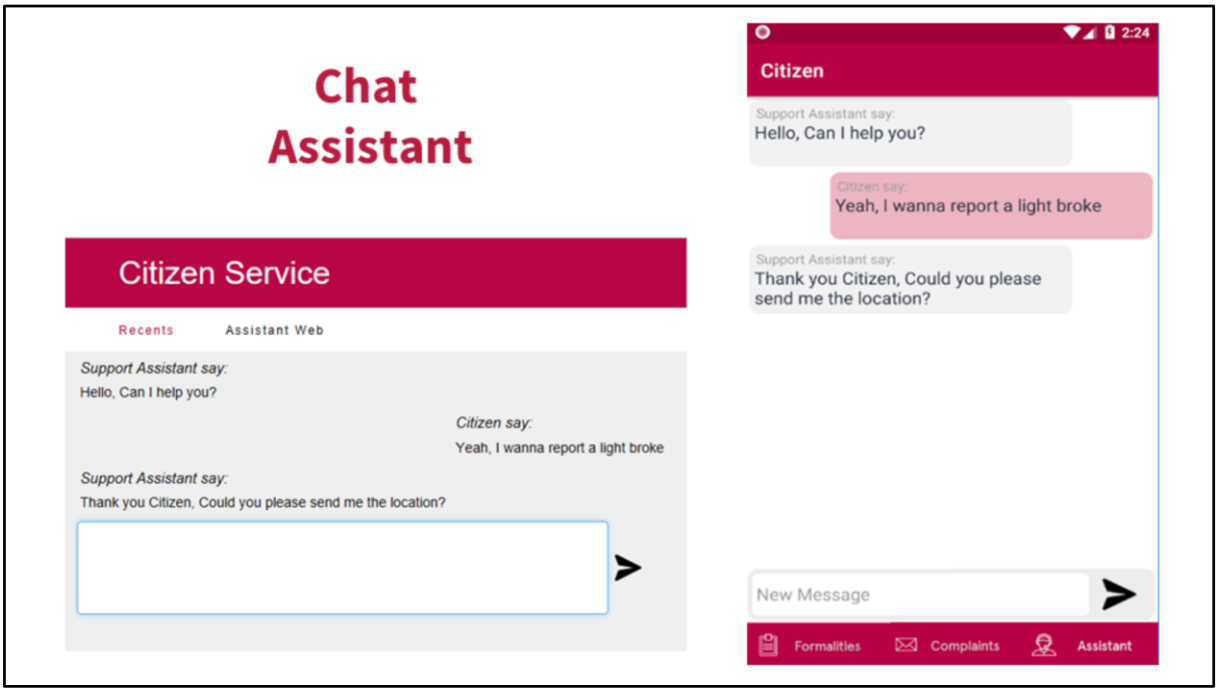

Por último, debemos llamar al procedimiento SendNotification basando las variables que configuramos e indicando al dispositivo que recibirá esta notificación.
En este ejemplo, serán los usuarios que se suscribieron.

## Socket API

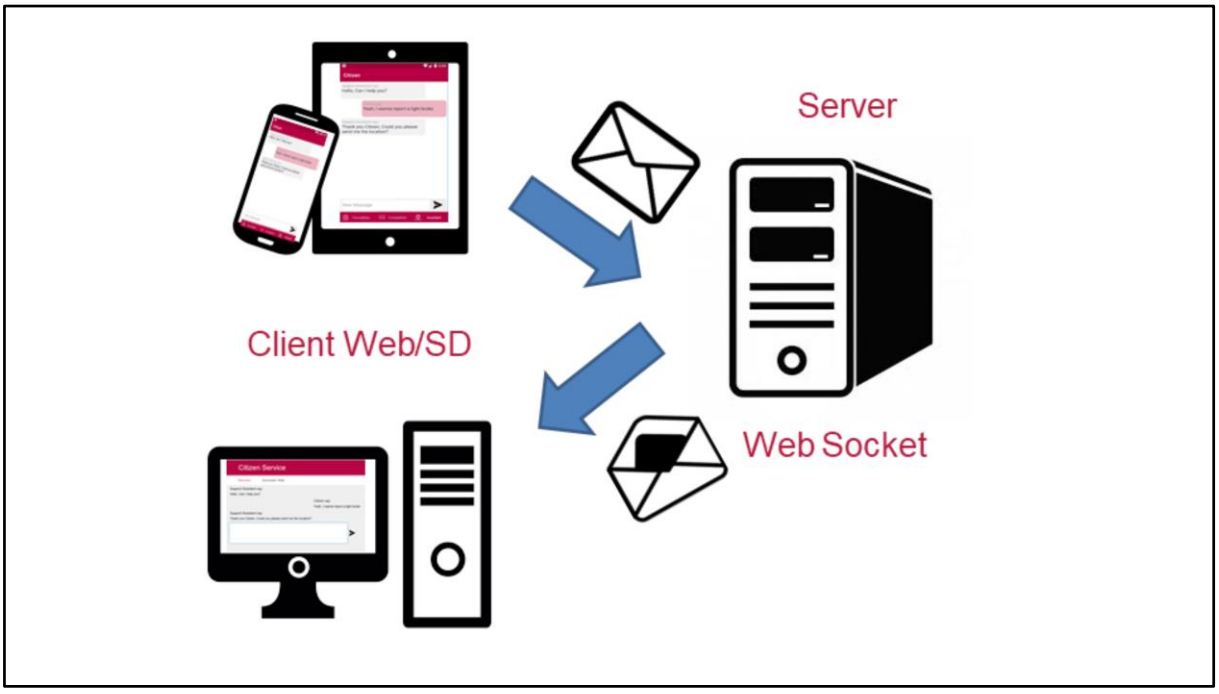
Pasemos al siguiente tema…



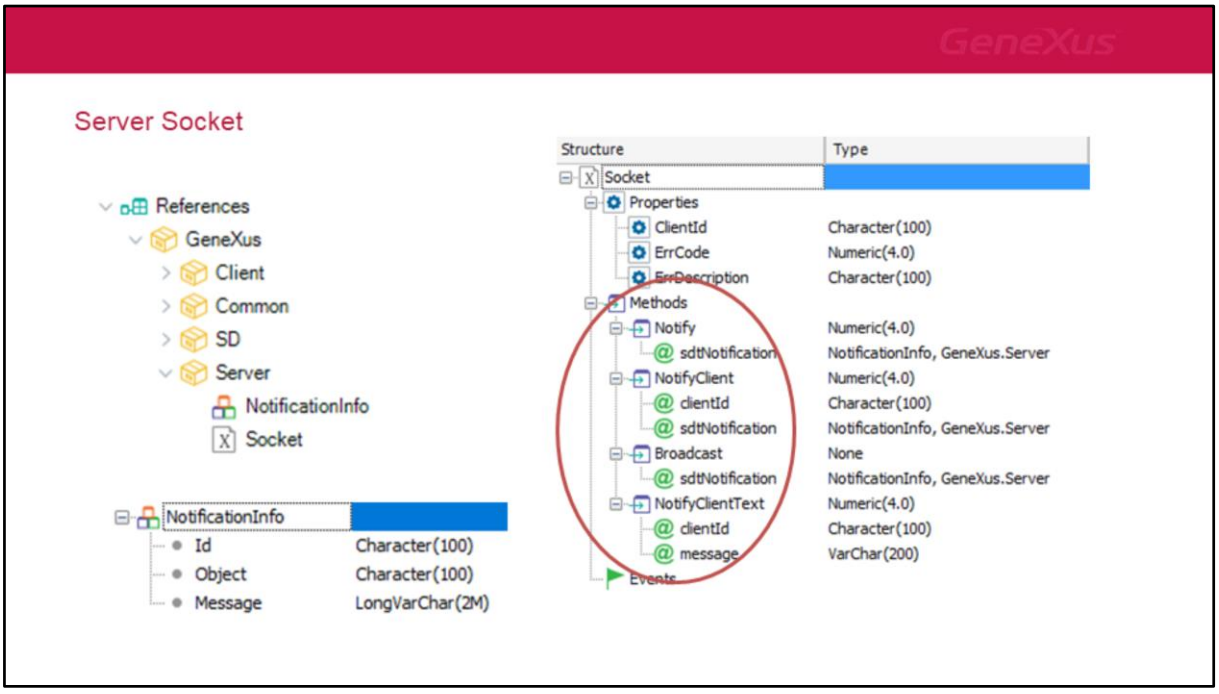

Sobre la Socket API, cabe destacar que nos permite establecer una comunicación bidireccional entre el cliente y el servidor mediante los Web Socket (que son un protocolo de comunicaciones que permiten un canal de comunicación sobre una conexión TCP)




Un ejemplo de uso es un Chat de asistencia (Chat Assistant) que nuestro ejemplo le permite al usuario hacer un reporte en tiempo real desde su móvil y un asistente le apoyará desde la parte web.



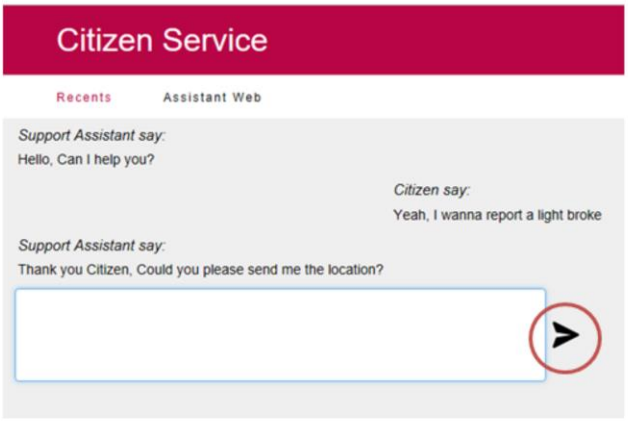

El funcionamiento es el siguiente:
El cliente SD envía mensaje al Server, quien se encarga de enviarlo al cliente web, el cliente Web refrescará en su pantalla para mostrar el nuevo mensaje y viceversa.



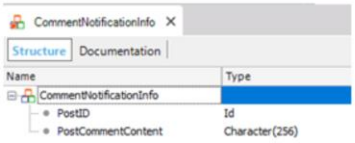

Para ello contamos con Server Socket, que nos permite hacer la comunicación del Server hacia los clientes enviando una notificación utilizando alguno de los métodos.
Por ejemplo:
Notify, nos permite enviar una notificación a un usuario que originó una acción.
NotifyClient, permite enviar una notificación a un usuario específico.
BroadCast, permite enviar notificación a todos los usuarios
Y NotifyClientText, permite enviar texto sin formato a un cliente en específico.

NotificationInfo nos permite especificar la información a la notificación que se enviará, como Id de la notificación, el Objeto que recibirá esa notificación y el mensaje.

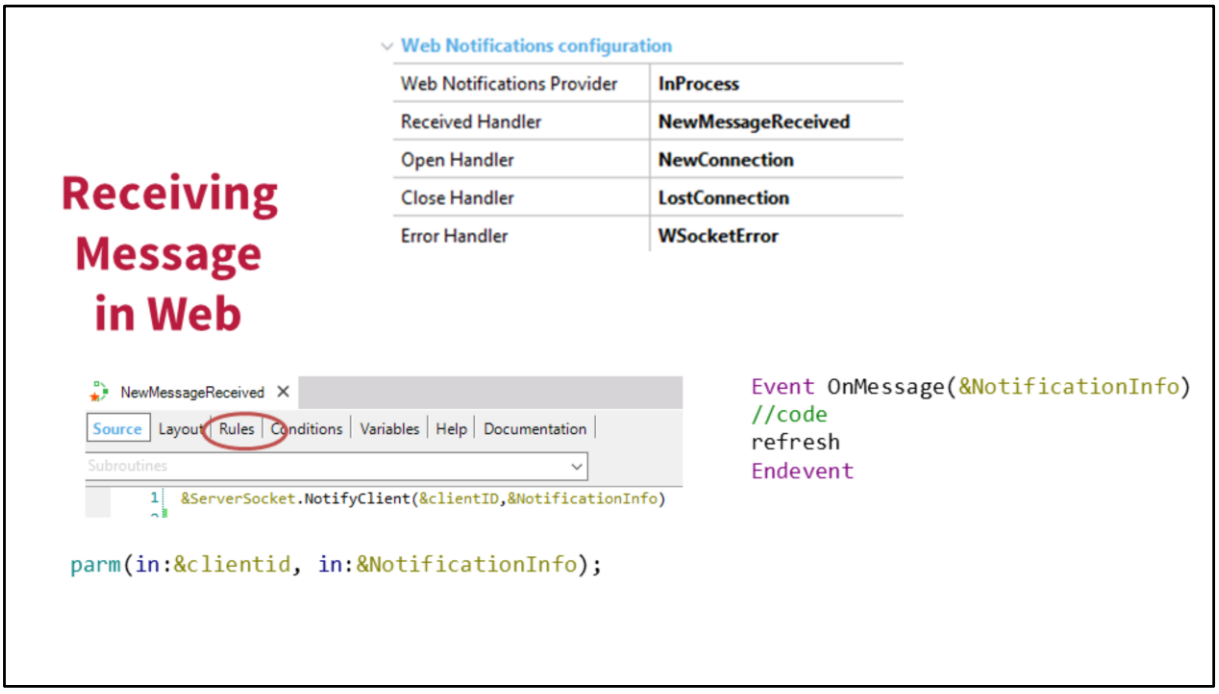
## Sending Message from Web




```
Event 'Send'
&commentNotificationInfo.PostId= random()
&commentNotificationInfo.PostCommentContent= &NewMessage
&NotificationInfo.Message= &commentNotificationInfo.ToJson()
&ServerSocket.NotifyClient(&clientID,&NotificationInfo)
Endevent
```

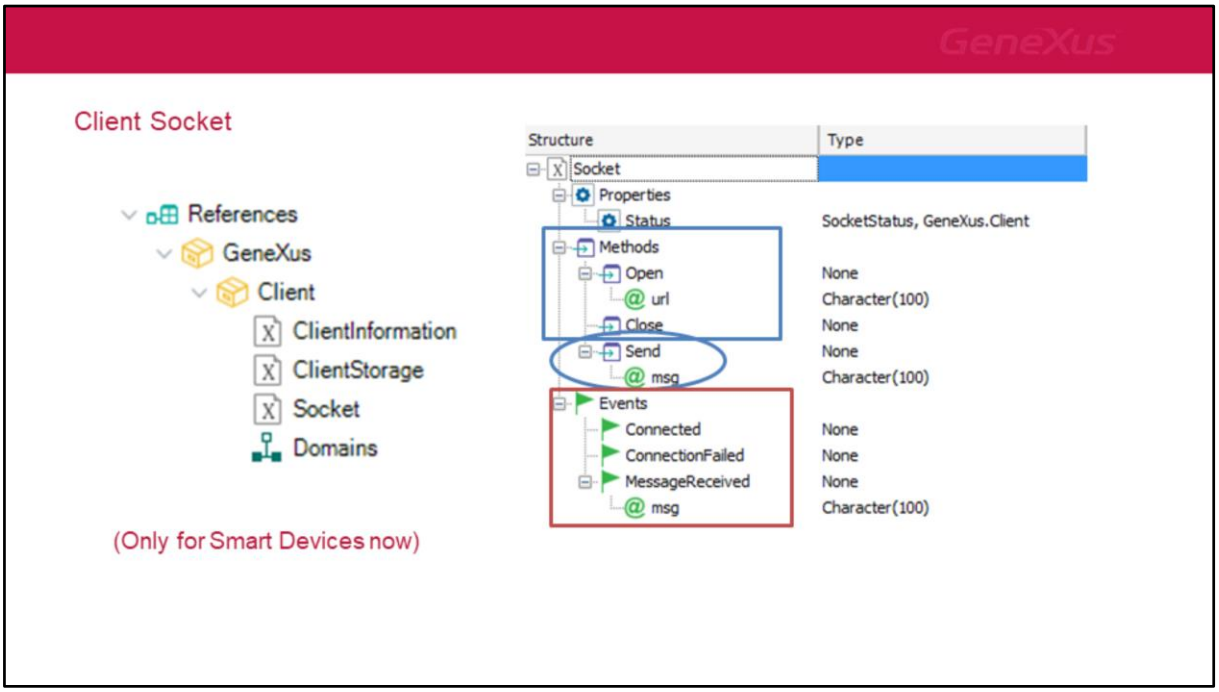



En nuestro ejemplo, enviamos un mensaje nuevo desde el chat web al usuario con el que estamos conversando.
Cabe destacar que el SDT CommentNotificationInfo lo creamos para enviar nuestro mensaje en un formato Json.




Posteriormente, para recibir la respuesta necesitamos configurar a nivel de la KB, los procedimientos que manejará la conexión y el mensaje recibido.

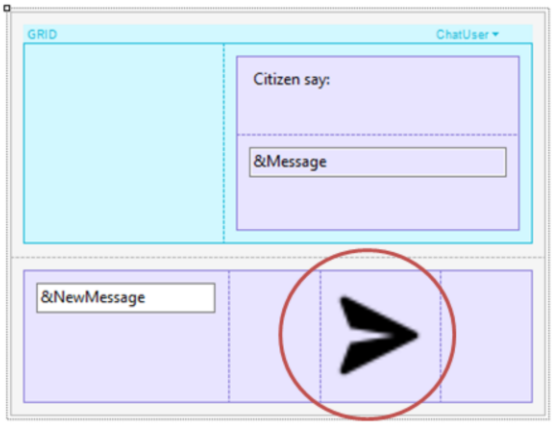
Veamos que en nuestro caso, tenemos el procedimiento “NewMessageReceived”.
Por último, para mostrarlo utilizamos el evento “Message” en nuestro Web Panel y hacemos un Refresh para cargar el nuevo mensaje que se mostrará.




Por otro lado, para la comunicación entre el cliente y el servidor contamos con Client Socket, que tiene métodos para establecer la conexión y un método para enviar el mensaje.
Así mismo contamos con eventos que se ejecutan al establecer o no la conexión y el recibir un mensaje.

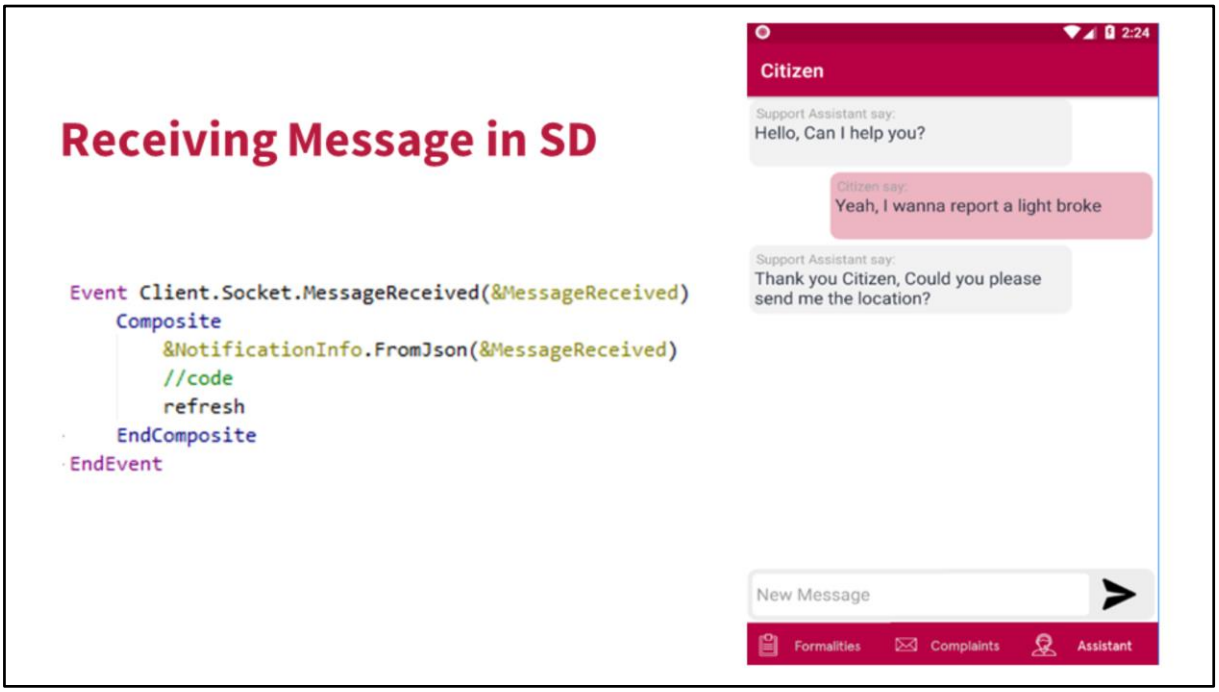
Cabe destacar que ahora Client Socket está disponible solo para generador Smart Devices.

## Sending Message from SD

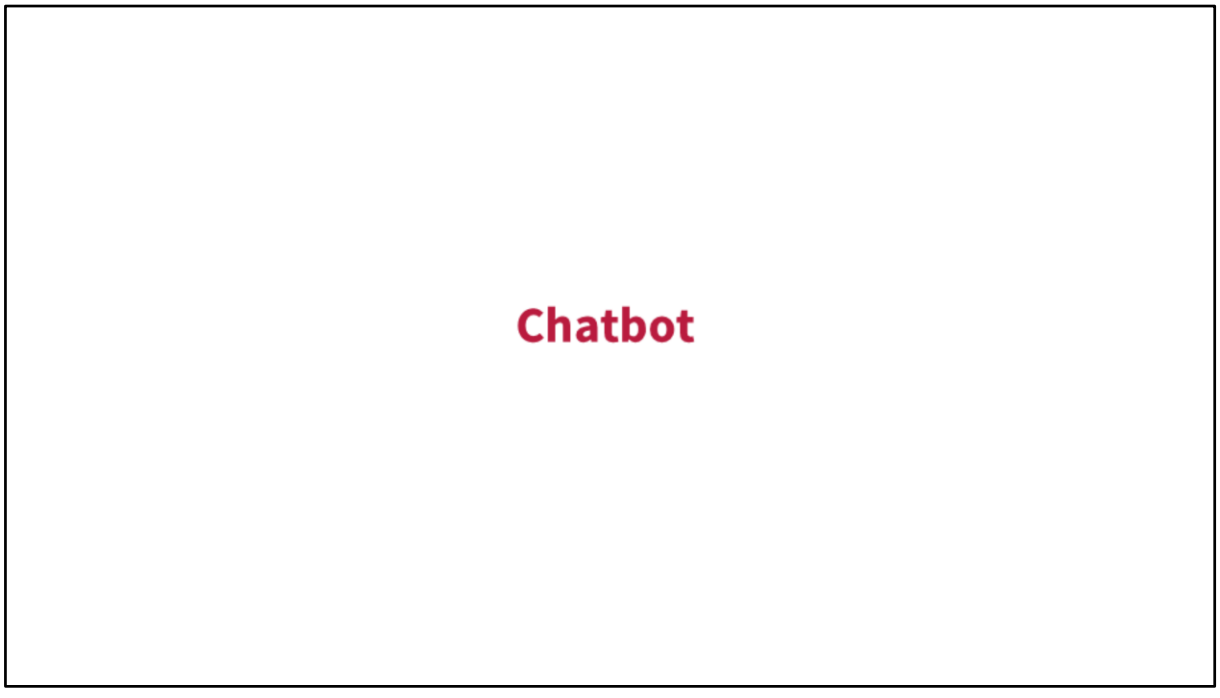



```
Event ImageSend.Tap
    GeneXus.Client.Socket.Send(&NewMessage)
Endevent
```

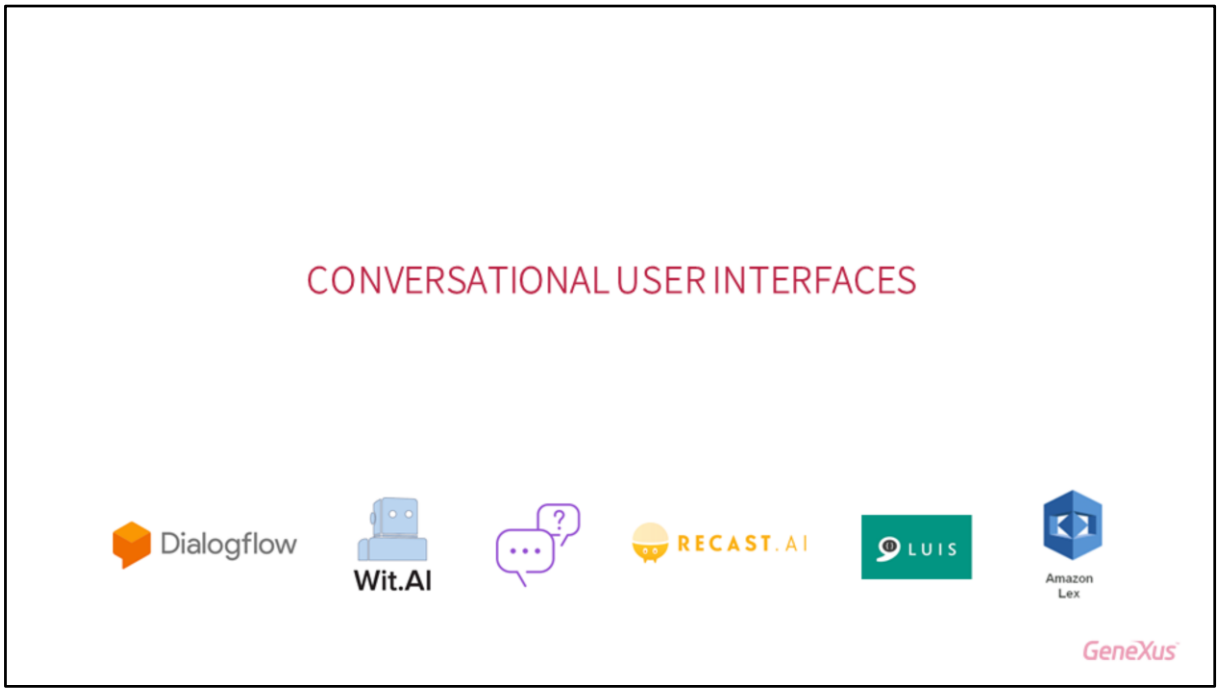
Siguiendo con nuestro ejemplo, desde la parte móvil configuramos el envío utilizando el método Send, basando la variable del nuevo mensaje.




Y para recibir los menajes utilizamos el evento “MessageReceived” para posteriormente desplegarlos al usuario.

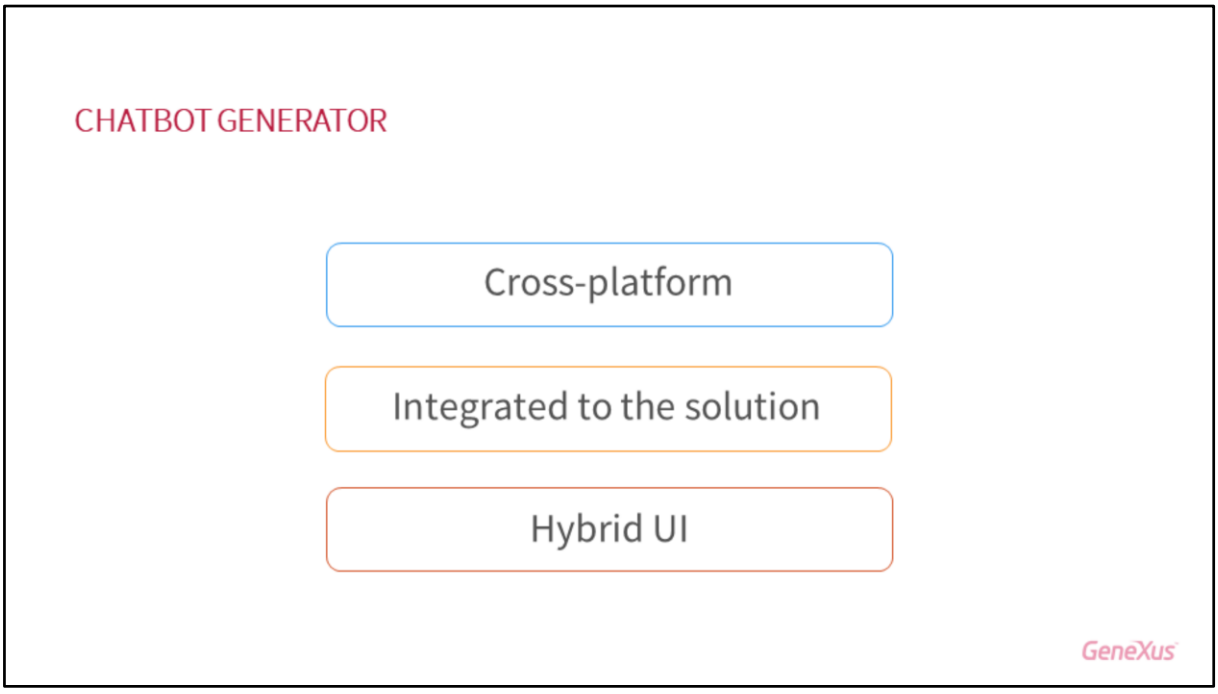



Buen día, les voy a contar un poco acerca de lo que tenemos en la versión 16 para crear un Chatbot.




Un Chatbot, básicamente es un tipo de interfaz de usuario conversacional en donde el usuario interactúa directamente con un robot y está fuertemente basado en lo que son plataformas de procesamiento del lenguaje natural.

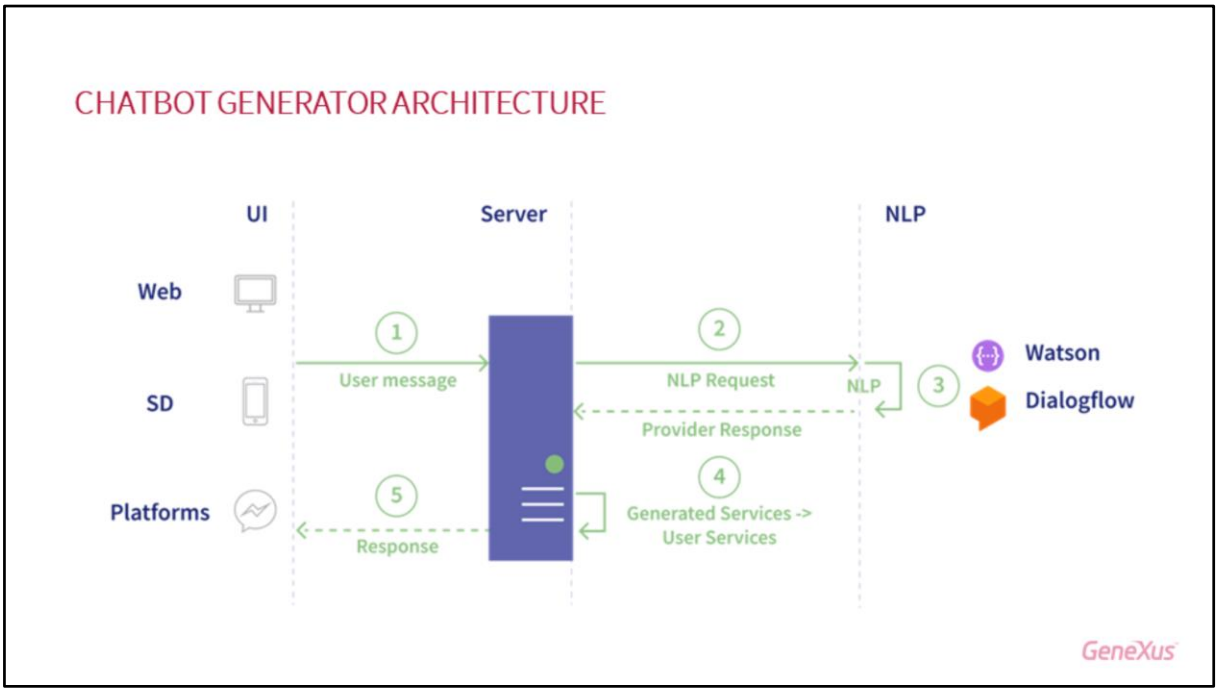
Acá tenemos algunas de ellas: DialogFlow, Watson Assistant, Recast.AI, Luis (la de Microsoft), hoy por hoy estamos soportando DialogFlow y Watson Assistant, pero estamos trabajando para soportar otras.



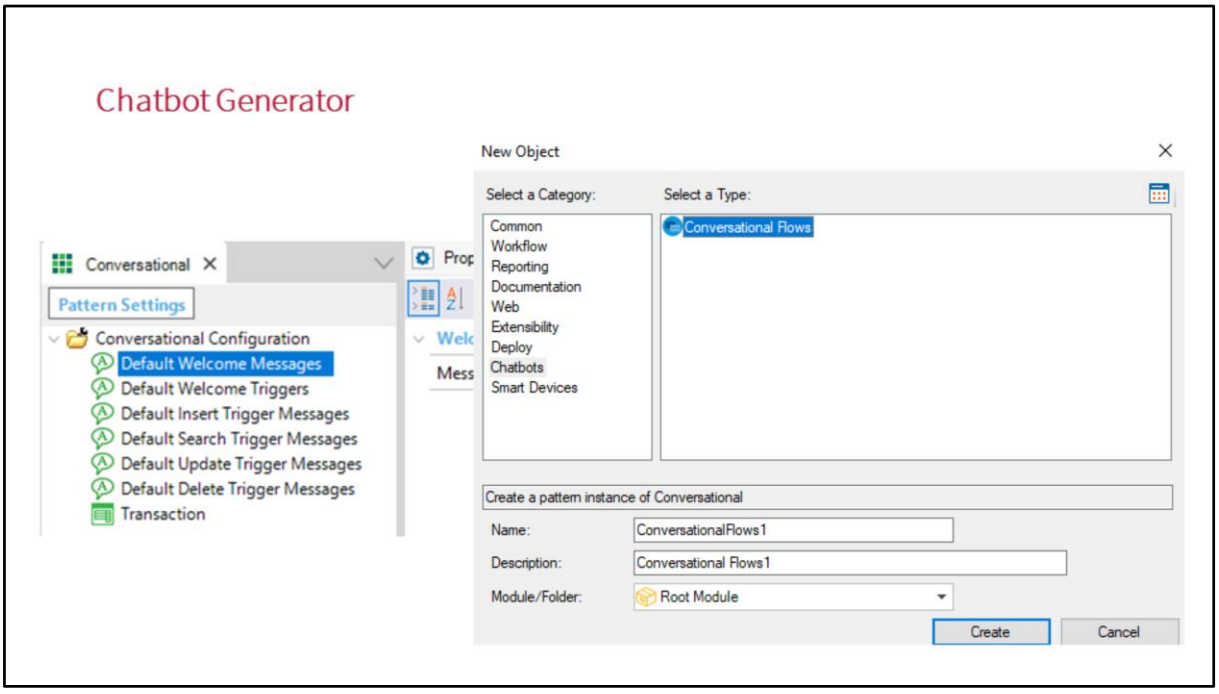

El Chatbot Generator nos permite crear una solución, como yo les decía, hoy contra DialogFlow o Watson Assistant, pero en miras de soportar cualquier tipo de solución de procesamiento de lenguaje natural, por eso nos va a esconder toda la complejidad que tiene eso por detrás.

Vamos a hacer un modelo para hacer un Chatbot, vamos a generar ese objeto y vamos a tener nuestro chatbot en cualquiera de estar plataformas.

Además es una solución integrada en GeneXus y  vamos a poder hacer soluciones hibridas, es decir soluciones no solo de chatbots que interactúan por texto sino que se va a poder devolver paneles y objetos con interfaz gráfica, tanto para consulta como para persistir datos, por ejemplo, en el sitio web de una Universidad permitir a los estudiantes consultar acerca de las asignaturas, de los exámenes, pero también que le sea posible ingresar un formulario para inscribirse en un examen, por ejemplo.
Eso lo vamos a ver un poquito mas adelante.



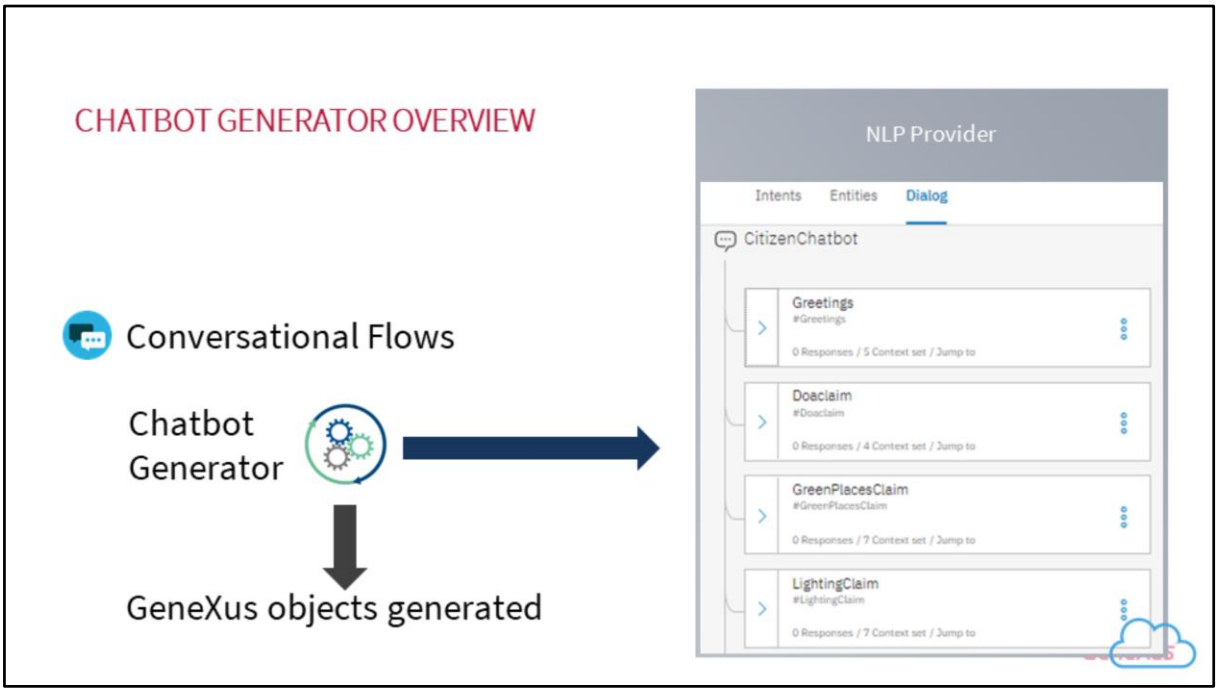

Primero les quiero comentar como es la arquitectura de esta solución, básicamente tenemos tres componentes:
La parte de User Interface
El Servidor donde se procesan a los objetos y la solución NLP (la plataforma NLP que yo les comentaba anteriormente) entonces desde la UI, desde la interfaz (es donde tenemos el Chatbot), el usuario ingresa un mensaje, ese mensaje se procesa en el servidor, se arma el paquete necesario, dependiendo de la solución NLP que estemos usando, (de eso no nos tenemos que preocupar porque esos objetos se generan automáticamente)
Luego, ese Request viaja a la solución NLP, allí se decide, se toma la decisión, se interpreta la consulta del usuario, se decide cual va a ser la respuesta, esa respuesta se envía nuevamente al servidor, en el servidor se decide que es lo que se le va a contestar al usuario según el modelo que tengamos definido en GeneXus y le viaja la respuesta al usuario, que puede ser una respuesta de tipo texto o puede ser una respuesta gráfica (un Web Panel, un SD Panel, etc)




¿Cómo trabajamos con el Chatbot Generator en la versión 16 (o en la 15 U12 también inclusive, para web)?

Creamos un objeto nuevo que tenemos en GeneXus que se llama conversational flow, creamos ese objeto y a partir de ahí podemos empezar a diseñar nuestro Chatbot y nuestro modelo.

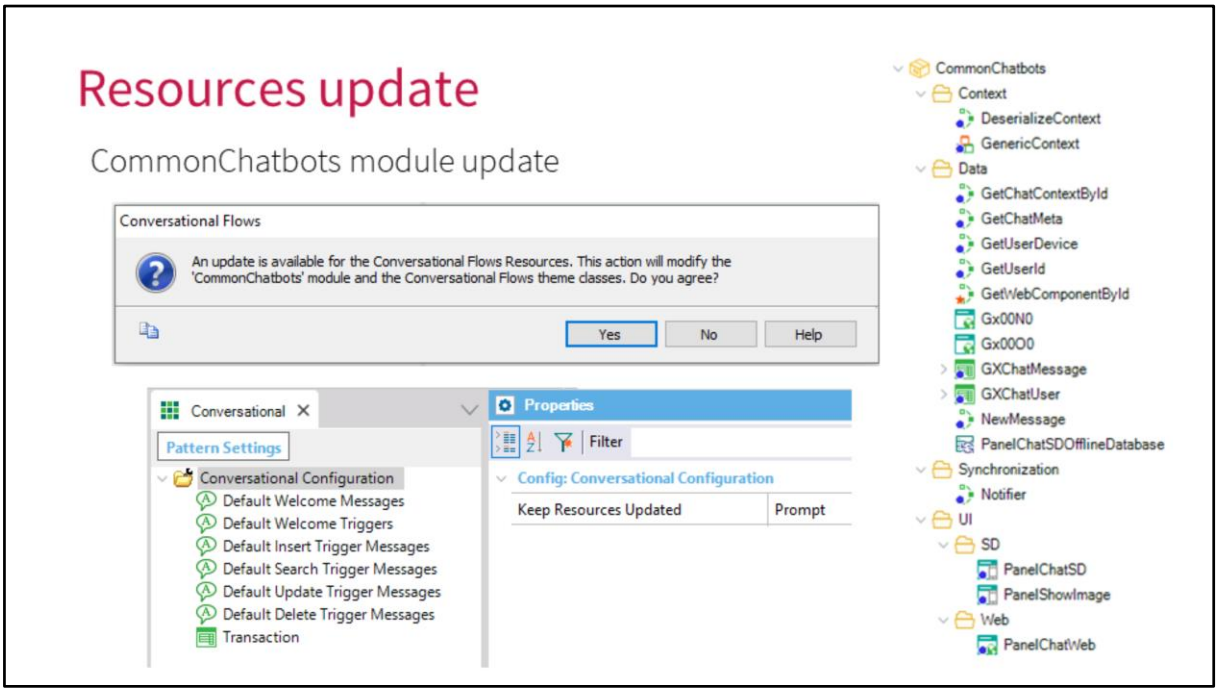
Les quería comentar que esto tiene un funcionamiento similar a lo que es el mecanismo de un pattern, en el sentido en que vamos a aplicar el pattern, (vamos a generar este objeto) y el mecanismo es muy similar en el sentido en que: se generan objetos dentro de nuestra base de conocimientos (similar a lo que ocurre en el Pattern WorkWith) pero también se generan componentes del lado de la solucion NLP.
¿Cómo funciona? Debido a que funciona con el mecanismo de un Pattern… fíjense que vamos a tener los Pattern Settings, clásicos pero específicos para este caso, para lo que es el Chatbot Generator.
En los Pattern Settings van a poder configurar diferentes cosas genéricas a todas las instancias (las instancias son los objetos conversational Flow que les comentaba anteriormente)




Entonces, yo les decía recién que lo que se va a generar es:
Del lado de GeneXus se van a generar objetos necesarios para lo que es el diálogo con el chatbot y del lado del proveedor (como ven acá) se va a generar un diálogo.

(En este caso esta captura es de Watson Assistant, pero también se soportan lo mismo para dialog flow como les decía), del lado del proveedor se va a crear el diálogo necesario con todos los intents y las entidades (que después vemos que es)

Todo lo que es necesario del proveedor, se va  a crear automáticamente, no me voy a tener que preocupar yo de hacerlo, y del lado de GeneXus se van a crear los objetos con toda la lógica para interactuar con este diálogo que se ha creado del lado del proveedor.

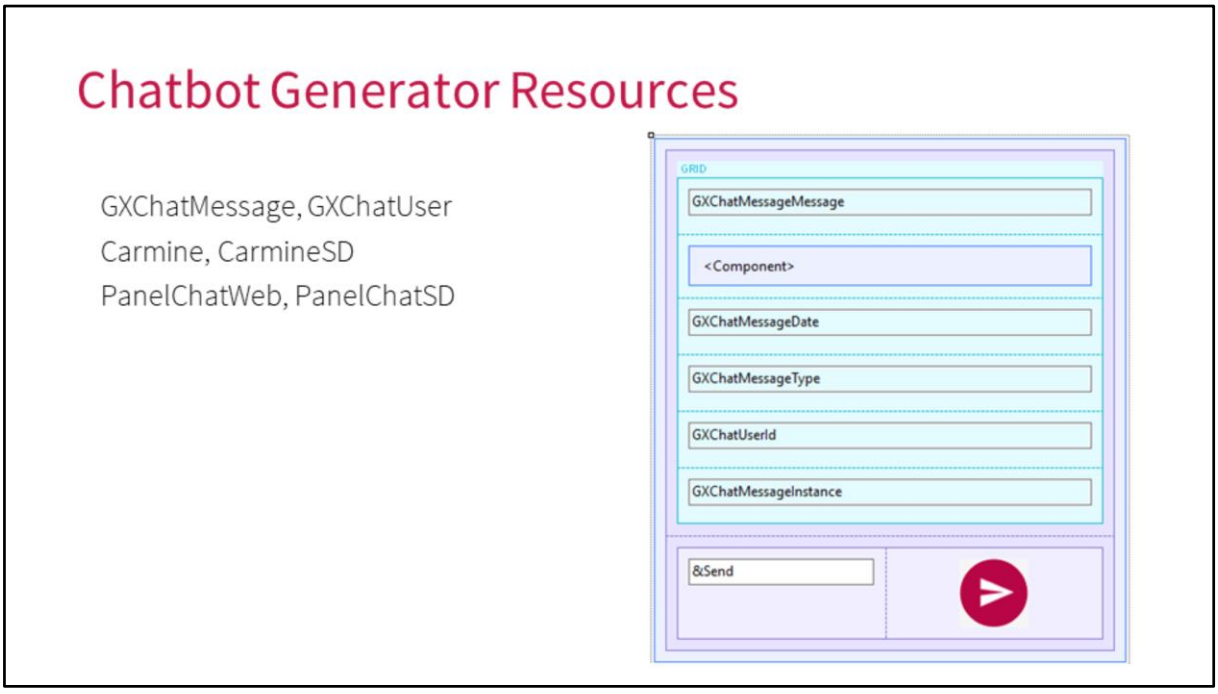



¿Cuáles son los objetos que vamos a tener en la base de conocimientos a raíz de aplicar el Chatbot Generator en estas instancias, en estos objetos conversational flows?

Primero tenemos:
Recursos, los recursos son ejemplos que nos permiten empezar a trabajar y tener ya nuestro Chatbot definido, sea para Web o sea para Smart Devices.
Los recursos están en un módulo que se llaman CommonChatbots, los objetos de este módulo, de estos recursos, se actualizan en cada Build.
Si ustedes desean actualizarlos, esto es en función de un propiedad que tenemos justamente en los Pattern Settings, que es “Keep Resources Updated”,pueden decirle que si (que siempre se actualicen esos recursos), que no (que no quieren actualizar esos recursos) una vez que se importaron por primera vez, después quieren que permanezcan siendo así y no quieren que se vuelvan a importar o que nos pregunte…

¿Por qué querría yo no volverlos a importar?
Porque es posible que yo haya hecho modificaciones a estos recursos y no quiero que se me pasen por arriba, de todas maneras, lo mas recomendable es hacer un Save as de estos recursos, seguir trabajando en paralelo con mis objetos y que estos recursos se actualicen por si hay alguna mejora en algún Build y haremos el merge nosotros manualmente.

¿quedan dudas de esto? ¿se entiende?



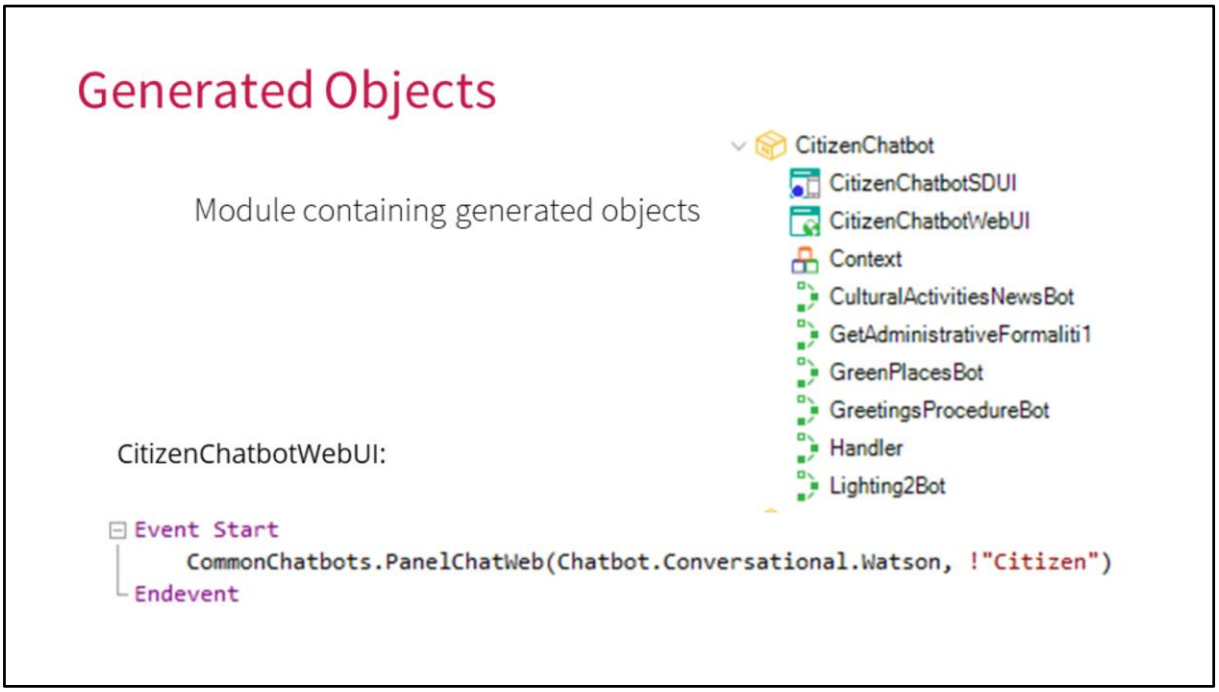

Les voy a contar algunos de los recursos.

Un par de los recursos importante son Las Tablas, son dos tablas que van a ver que se crean en la base de datos de la aplicación, que es la GxChatMessage y GxChatUser.
La tabla GxChatMessage, contiene el mensaje, la metadata, referencia al usuario, información necesaria justamente que tiene que ver con la conversación, con el chat.
Y GxChatUser contiene el usuario y el device ID.

Estas tablas van a ver que como el panel SD (el panel Chat SD), que es uno de los recursos que tenemos para representar al chatbot, en SD van a ver que es Offline, entonces estas tablas quedan persistidas en lo que es el dispositivo.
Eso es a los efectos de que si el usuario está desconectado, pueda ver la conversación.

En el Carmine y en el CarmineSD van a tener clases que son para darle estilo, para darle estética a los recursos que yo les decía, al panel web y al panel SD que son ejemplos de Chatbots.

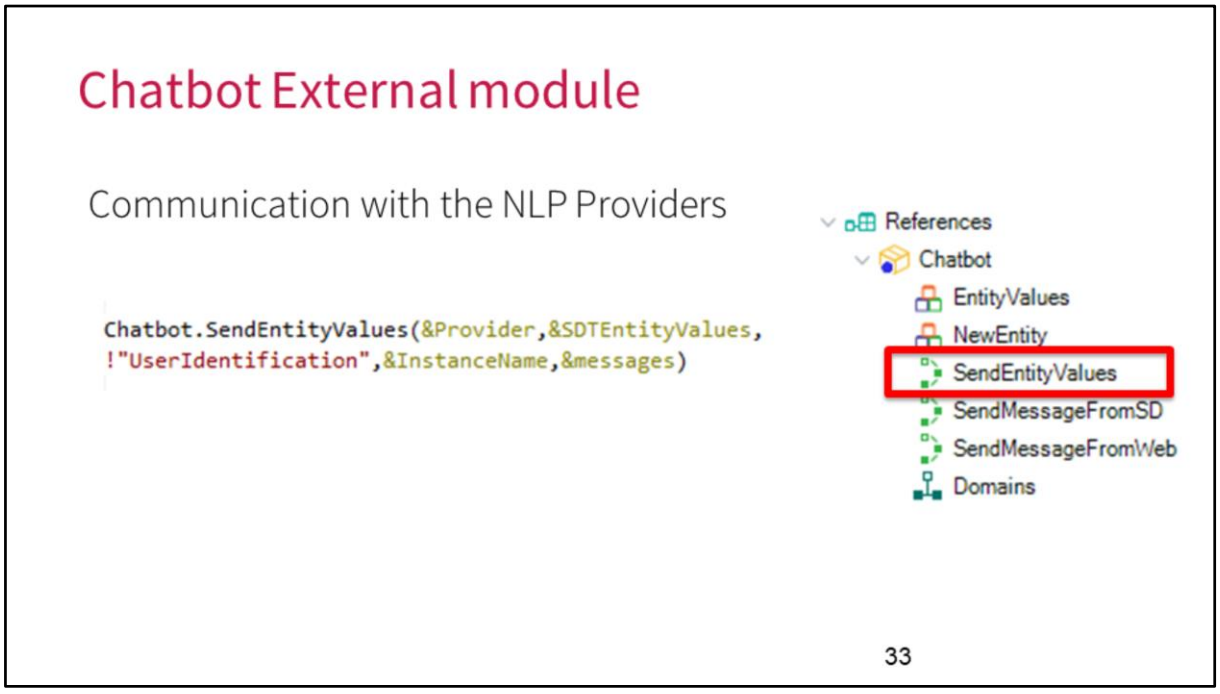
Estos son los objetos que justamente yo les comentaba, que los vamos a ver un poco mas adelante.
Esta es una captura del Panel Chat Web, que ya tienen ustedes como recurso (hay ejemplo que lo van a estar viendo en el práctico) pero es simplemente una pantalla donde se muestra el mensaje y tiene un componente dentro de un grid que justamente lo que nos permite es mostrar componentes gráficos en la conversación.




Eso eran los recurso, y los recursos justamente como yo les decía, los pueden ustedes editar, cambiar a gusto, después vienen los objetos generados, que ahí si, son objetos que siempre se generan cada vez que generan la instancia, se generan estos objetos con lo cual no puedo o no debería yo cambiarlos porque voy a perder esos cambios.

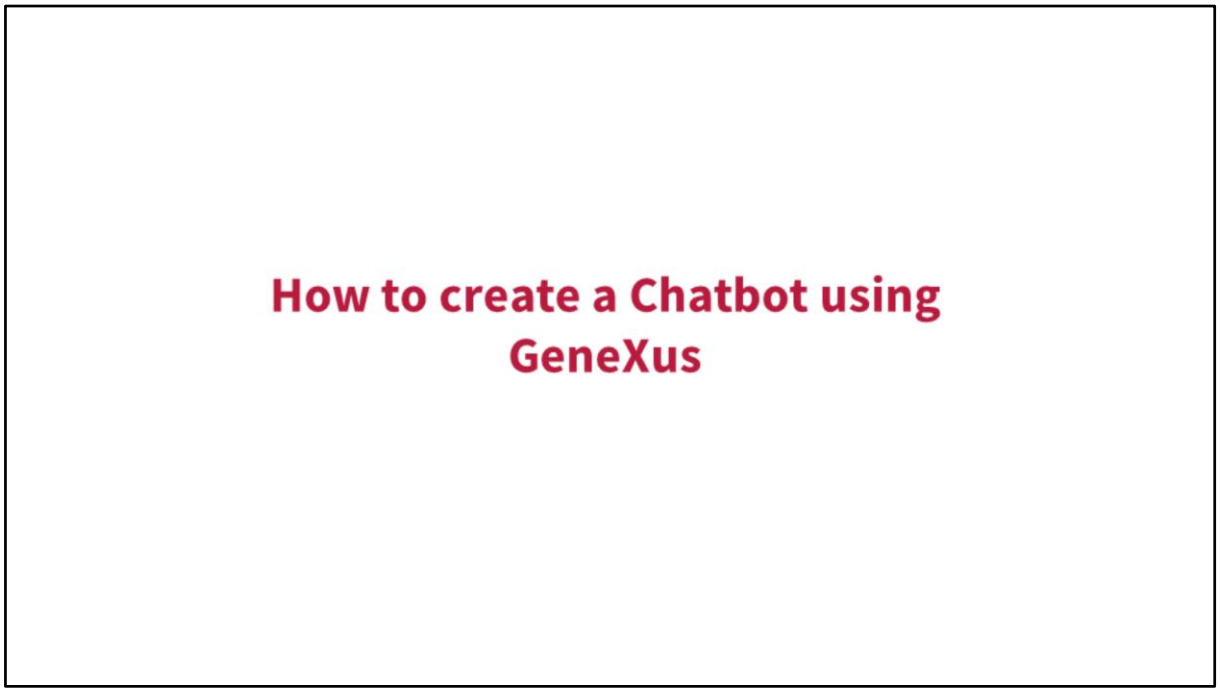
Los objetos generados los tengo bajo un módulo que lleva el nombre de la instancia, el nombre de mi chatbot, de mi conversational flow.

En particular, hay un par de objetos que llevan el nombre de la instancia seguido por WebUI o SDUI que llaman al Panel Web o SD según corresponda (fíjense como lo llaman)
Este es el módulo de los recursos, se llama al objeto por el chatweb o SD pasándole como parámetro el provider y el nombre de la instancia, digo esto porque de repente ustedes después quieren llamar a este objeto, a este recurso de otra forma, por ejemplo, como si fuera un componente tener en cuenta con que parámetros se está llamando.

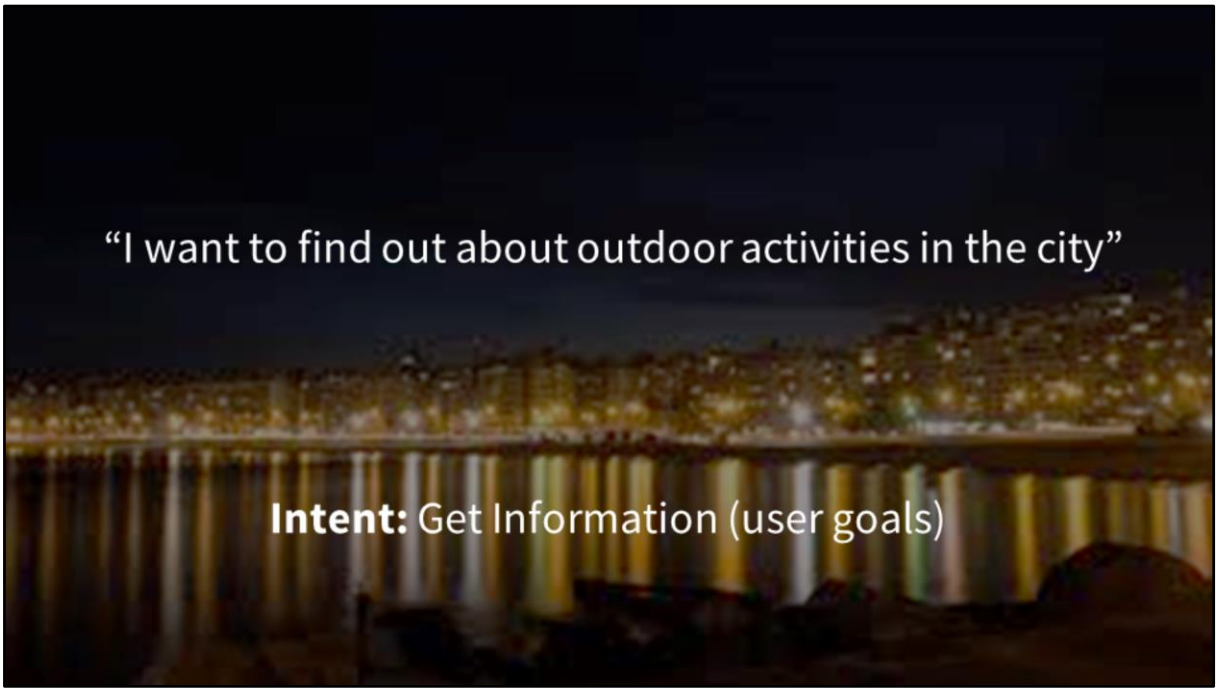



Y otras de las partes, hablamos de los recursos, hablamos de los objetos generados, otros de los componentes que se van a ver que se incorporan en su KB es un módulo externo que contiene todo lo que es la interacción con objetos, digamos con la lógica que permite hablar con el chat. Pero se exponen algunos métodos interesantes y quiero contarles este: SendEntityValues, este método es interesante si ustedes quieren dentro del provider actualizar lo que son las entidades.

Vamos a ver lo que es una entidad, pero básicamente una entidad es información adicional que permite al provider identificar la intención del usuario, son grupos de valores con sinónimos, que esa información esta dentro del provider, entonces a través de este método vamos a poder mandar al provider valores y sinónimos de una entidad determinada.

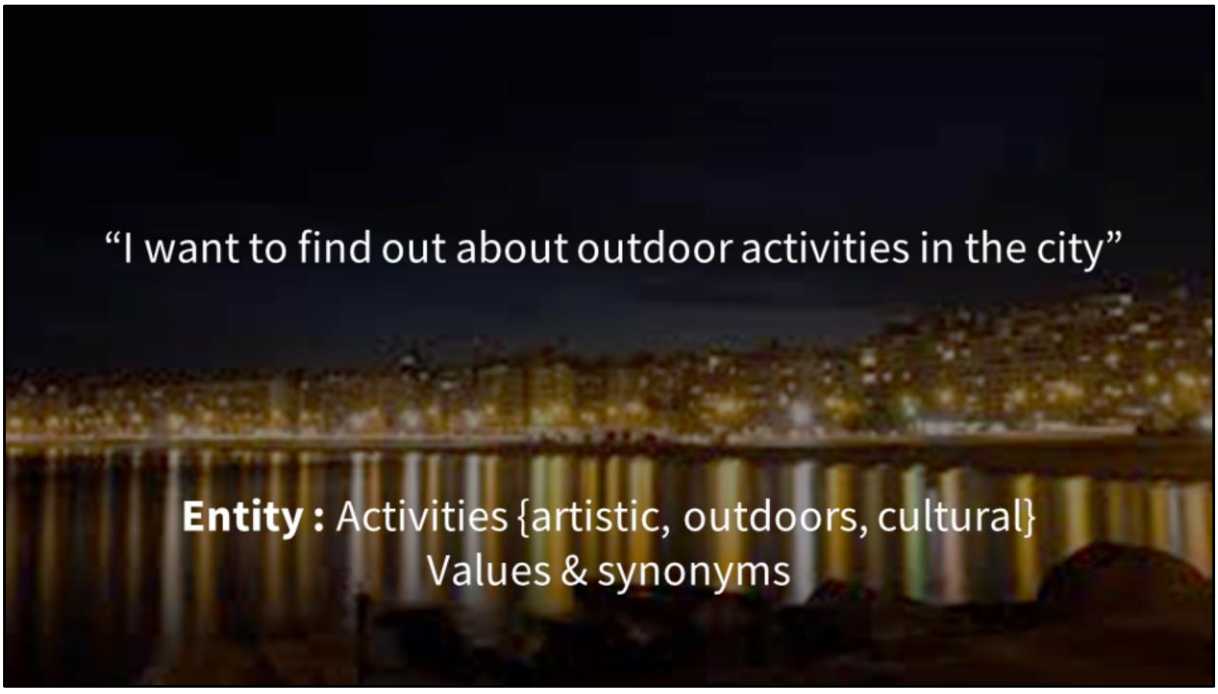



¿Cómo crear un Chatbot en GeneXus?

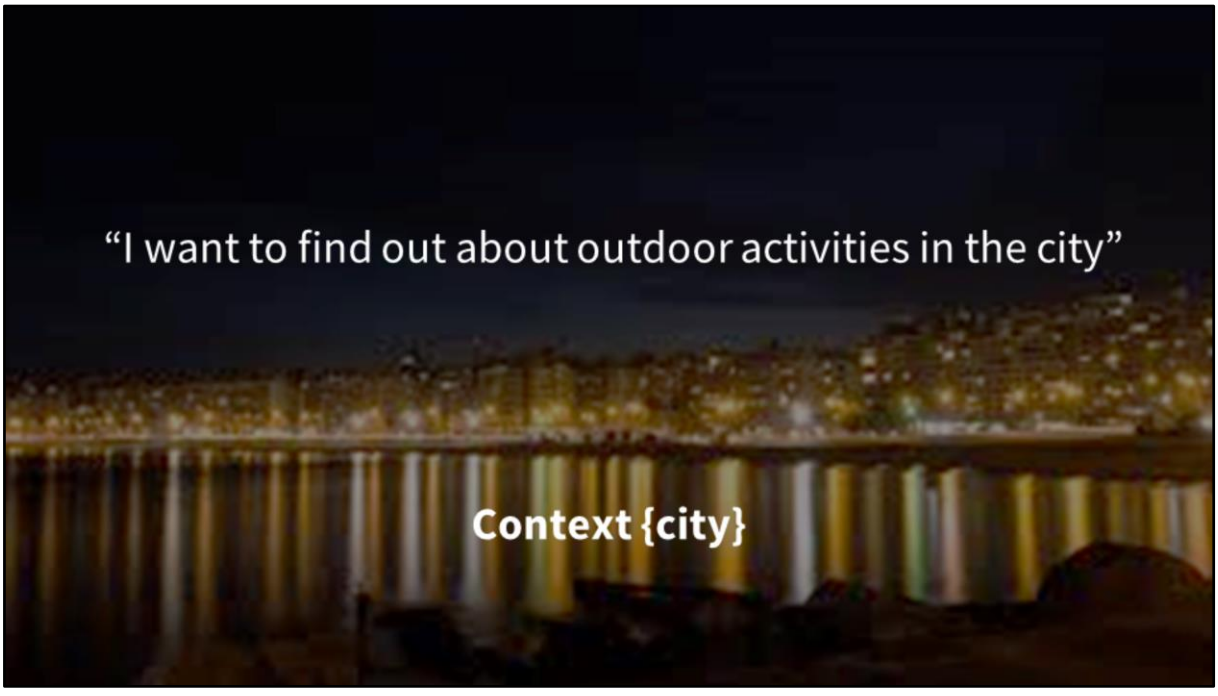



Hoy les estaba mencionando un par de conceptos que son: entidades, intenciones y otro es contexto, les voy a comentar a que me refiero con cada una de esas cosas.

El Intent: es el propósito del usuario a la hora de hacer una consulta, por ejemplo, quiero obtener información acerca de actividades en la ciudad, ese es el Intent.

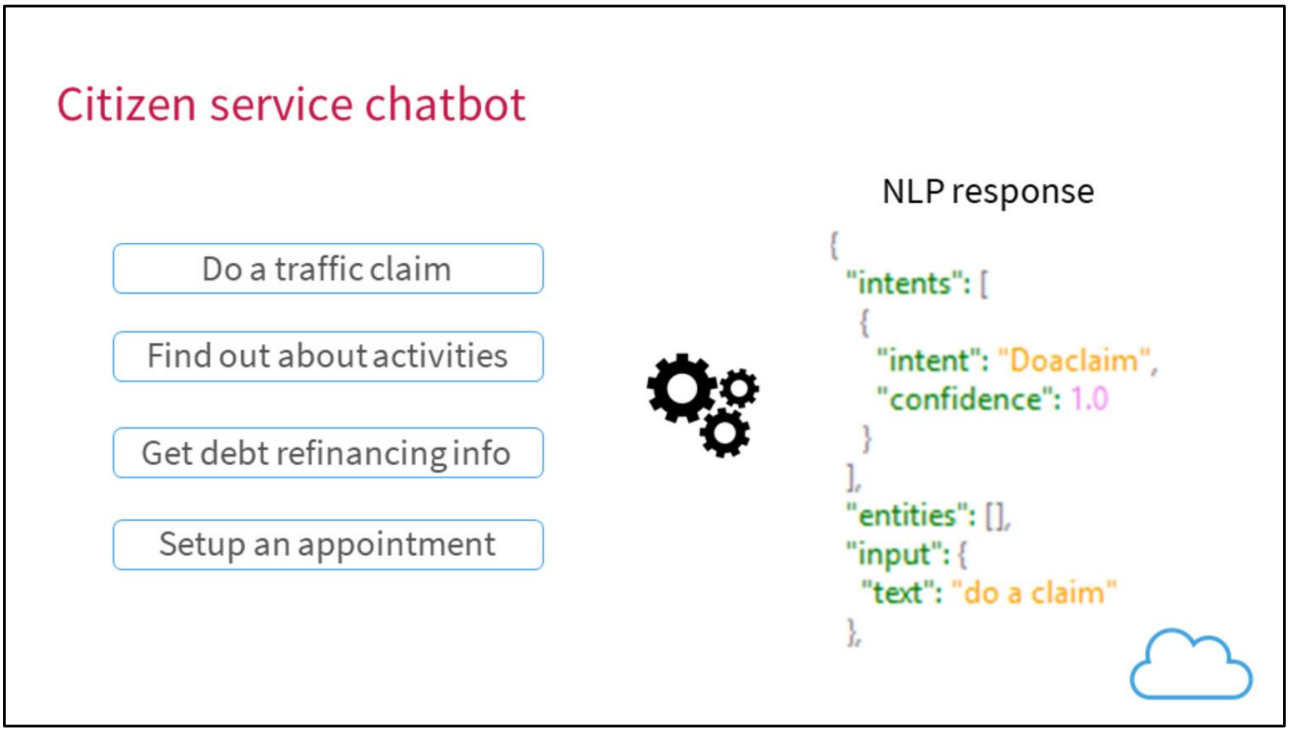



La entidad (Entity): es información adicional que nos permite identificar, permite al provider identificar la intención del usuario o mejor dicho, ayuda, le da extra información para identificar esa intención, por ejemplo, en este caso el usuario quiere obtener información de actividades pero que son al aire libre (outdoor).
Entonces, la entidad es un grupo de valores, cada valor puede tener “n” sinónimos, en este caso puede ser el tipo de actividades, las actividades pueden ser culturales, artísticas, educativas, etc. Y cada uno de esos valores tendrá sus sinónimos y esta es información que estaá en el provider.




Y otro concepto es el contexto (Context), toda conversación tiene un contexto (si yo estoy hablando con una persona, al rato o al minuto de hablar, en realidad no se precisan muchos, ya se sabe de que estamos hablando, los dos entendimos de que estamos hablando) eso sucede también en una conversación con un robot, se tiene que mantener un contexto, el robot no le puede preguntar “n” veces al usuario lo mismo.

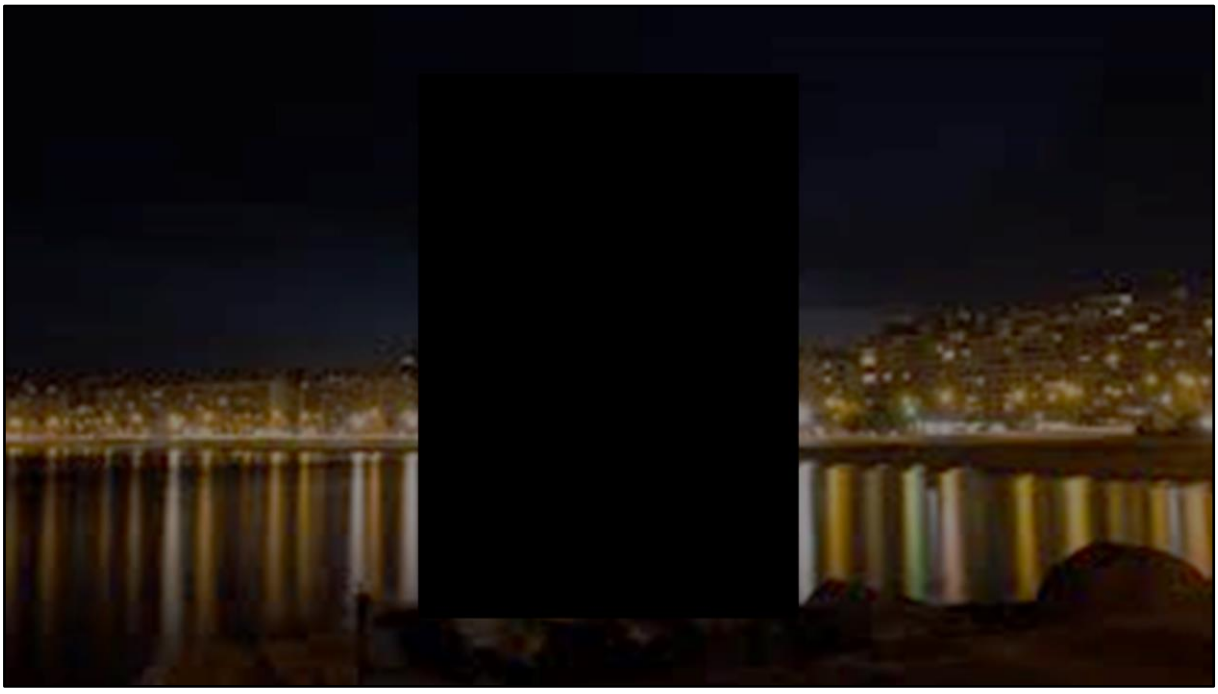
En esta caso puede ser la ciudad, un ejemplo, si ya quedó claro de que ciudad estamos hablando, el bot no le va a volver a preguntar cuál es la ciudad a la que se refiere.




Con esos tres conceptos nosotros ya podemos crear nuestro chatbot.

Lo que tenemos que hacer primero es, preguntarnos a nosotros mismos cuales son las intenciones que va a cubrir ese chatbot, por ejemplo, si es servicio al ciudadano las intenciones que pueden cubrir es, cómo hacer un reclamo por un problema en el tránsito, cómo hacer una consulta acerca de refinanciar una deuda, cómo obtener información de actividades, re tramitar la licencia de conducir, etc.

Un detalle importante es que esas intenciones van a ser interpretadas, el que va interpretar la intención es el provider, con cierto nivel de confianza.



Entonces…

Les voy a mostrar un video, que después van a ver en el práctico el caso.

Esto es un Machine Learning, es inteligencia artificial (Procesamiento del lenguaje natural) entonces él, de alguna forma interpreta, digamos… no es certero, no es matemático, de alguna forma interpreta la consulta del usuario con un nivel de confianza, cuanto más alto sea el nivel de confianza, ahí en ese caso sí dice “El Intel del usuario es tal…” el usuario quería preguntar acerca de las actividades culturales o de las actividades artísticas.

No es matemática, no es matemática pura… es Procesamiento del lenguaje natural.

En éste caso, fíjense que acá en el Chatbot los saludamos y lo primero que nos dice el Chat bot es “yo se contestar de tres cosas”:

- Te puedo contestar acerca de una pregunta que me hagas
- Se agendar una entrevista
- Y te puedo aceptar un reclamo

O sea, el Chatbot obviamente va a hablar de un conjunto acotado de cosas, no habla de todo por supuesto y está bueno aclarar al usuario de primera de que es lo que puede conversar este Chat.

El usuario le dice : “quiero hacer una consulta” y el bot le pregunta: “yo se contestar acerca de actividades y de trámites administrativos”
Esto evidentemente es una entidad, o sea, actividades y trámites administrativos son dos valores de una entidad, que es lo que sabe contestar al usuario.
Esos valores, esos sinónimos estarán en el provider, quiere decir que el usuario me va a decir

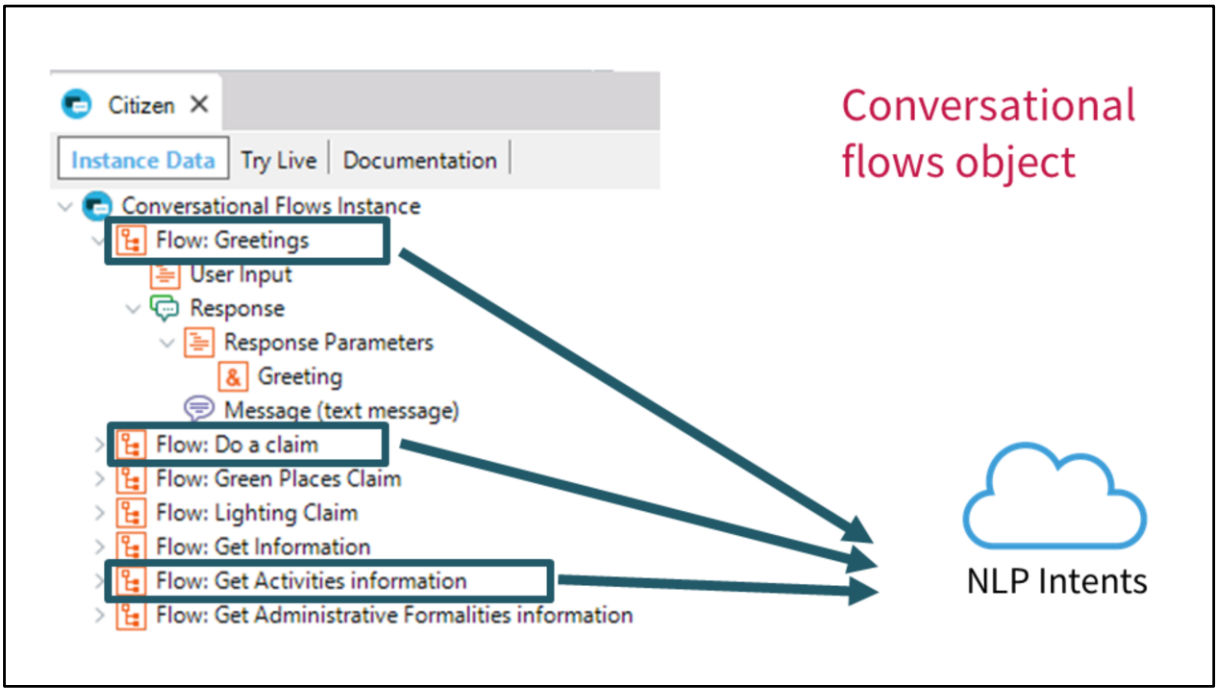
“quiero hablar de actividades o quiero hablar de trámites administrativos”, lo importante es que me diga uno de esos valores o algunos de los sinónimos de esos valores para que nos entendamos., para que se sepa de que está hablando.

El usuario ahora dice “About Activities, Thanks” y de ahí se infiere el valor que está en el Provider, se infiere que el usuario quiere preguntar de actividades.
Lo siguiente que le pregunta el Bot es: “Bien, Actividades ¿pero de qué tipo? ¿Culturales, artísticas o naturales?
Esta es otra entidad con tres valores y cada uno de los valores tendrá sus sinónimos, el usuario puede poner uno de esos sinónimos y se interpreta, se entiende.

Después vamos a ver como está definido el tema de las entidades en el Provider.

Una vez que se entendió ese Intent, como hablábamos hoy, con un nivel de confianza suficiente… entonces se le devuelve al usuario la salida, la respuesta a esta intención, que es un panel donde se muestran todas las actividades que el usuario quería ver.
Hasta ahí cumplimos con una intención, ahora vamos a ver la siguiente, donde el usuario selecciona una de estas actividades
Selecciona, por ejemplo: Jardín Botánico, el Bot interpreta que el usuario quiere hacer una (porque así está modelado, por supuesto) visita guiada.
El Bot le dice “¿cuándo querés agendar la visita?
El usuario responde: “Tomorrow, 5pm”
Y perfecto, se agenda la visita guiada.

En esta caso, lo que hicimos fue grabar datos y en el otro caso, era una consulta.

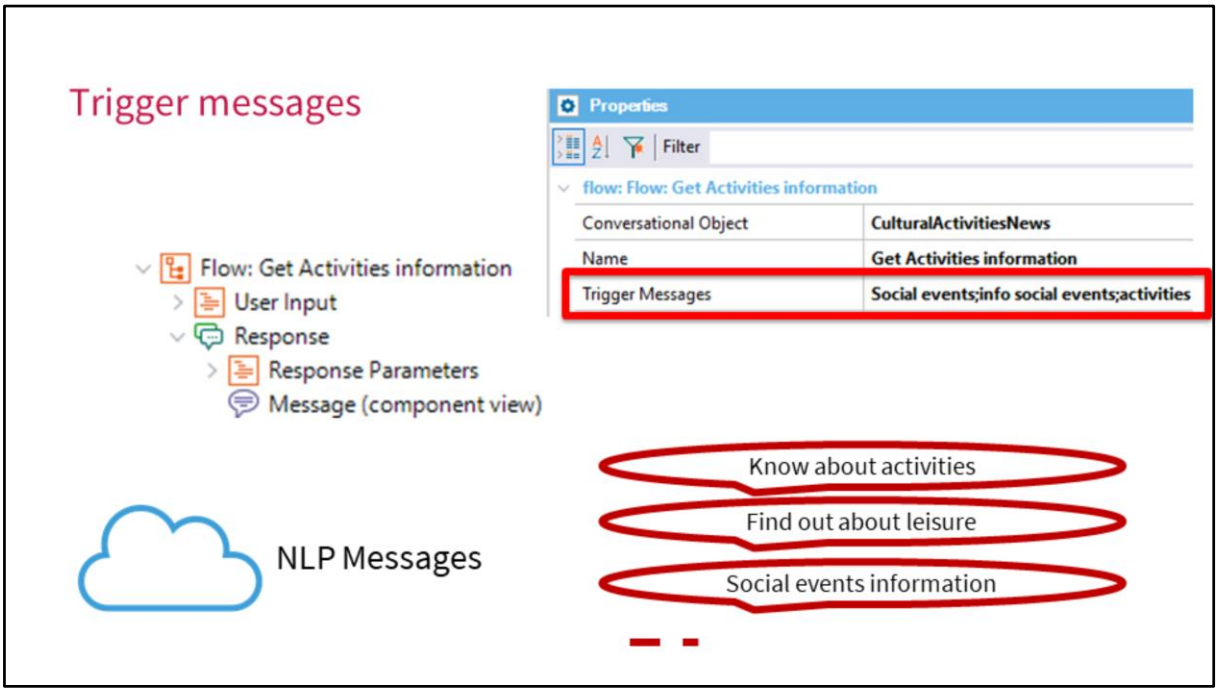



Entonces, ¿Cómo hicimos todo esto?

Tenemos un objeto nuevo en GeneXus que se llama conversational flows, que es el que representa nuestro bot, nuestro chatbot, tiene una vista así, de árbol, donde cada uno de los flujos (cada uno de los nodo flow) es un Intent en el proveedor.

Por ejemplo, en el caso de servicio al ciudadano voy a hacer un flujo para saludar, un flujo para hacer un reclamo, para hacer un reclamo por un problema de saneamiento, para consultar acerca de actividades al aire libre o de lo que fuere.
Todos esos Intents los plasmo yo en flows, dentro de mi chat, dentro de esta representación en mi conversational flows.

Lo importante es que cuando yo genere este objeto, estos Intens se me crean automáticamente en el provider**.**




Ahora viene un poco lo que tu preguntabas…

Yo tengo que ayudar al Provider a entender la intención, porque hay mil formas de preguntar acerca de actividades en la ciudad, se me pueden ocurrir… mil no, pero unas cuantas seguro. Entonces tengo que enseñar, entrenar al provider para que pueda identificar correctamente las intenciones que he definido, eso se hace a travez de la propiedad Trigger messages.

Al nivel del nodo Flow, tengo la propiedad Trigger messages, donde yo voy a poner las posibles formas en las que un usuario podría hacerme la pregunta de cómo obtener información:

- Know about activities
- Find out about leisure
- Social events information

Esta información se va sacando de logs y se va aprendiendo, no es que a mi se me ocurran como desarrollador todas las formas de preguntar, sino que seguramente se usen Logs y un poco en un proyecto que sea piloto se va aprendiendo y sacando las formas en las que el usuario trató de preguntar y capaz que falló, entonces se va agregando y se sigue entrenando, todo lo que es Machine Learning requiere un proceso fuerte de entrenamiento pero obviamente no tenemos que ser exhaustivos en todas las formas de decir esto, porque como está por debajo en el procesamiento de lenguaje natural, todo lo que es semánticamente equivalente a cualquiera de estas, trae los mismos resultados.

- Cuando tu hablas de que va aprendiendo… ¿esas preguntas yo las voy incorporando pero me implica generar de nuevo la KB? ¿o puede ser online? La voy agregando a medida que…
- Bueno, hoy trabajamos desde GeneXus exclusivamente pero cuando agregamos Trigger messages basta con salvar, te acordás que, yo estoy trabajando en GeneXus y cuando yo salvo

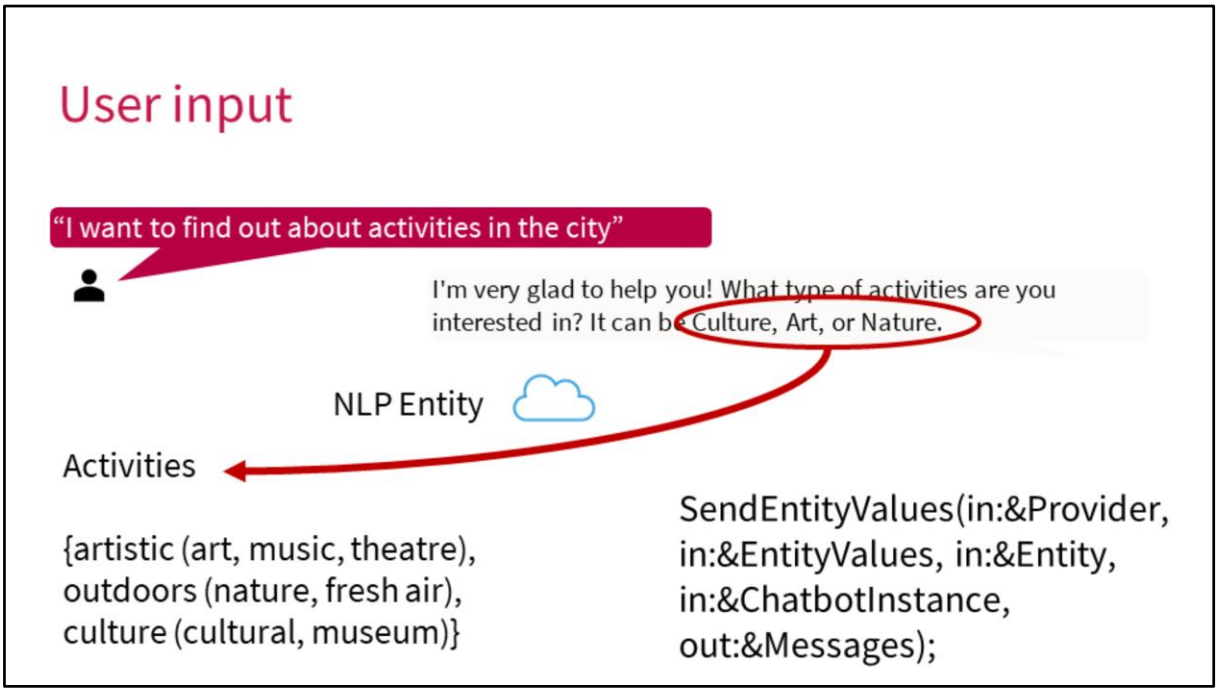
la instancia, se arma en realidad un Json con esto nuevo que estoy agregando, viaja al Provider (Watosn, etc) y agrega a nivel del Intent este ejemplo (se le llaman ejemplos en Watson Assistant).

Hoy es unidireccional, desde la KB hacia el Provider.

- Para eliminar…. Ahí es una forma de agregar, ¿y si tenes que eliminar y infiere algo que no es lo correcto?
- Si, también ¿verdad Diego?, lo importante es que hoy es unidireccional, no es bidireccional que es lo que estamos viendo para mejorar, pero los cambios que hagas en tus modelos se impactanen el Provider.
- Repito por las dudas, en ese caso si vos volvés a definir en los Trigger messages, cada vez que vos haces una modificación, esos Trigger messages cuando guardas, lo que se sincroniza es lo que esta en la instancia, entonces es mandatorio lo que esta definido en GeneXus,.

Lo único que no se modifica al sincronizar, son los valores que tengas definidos en las entidades, eso no se va a perder, pero si vos eliminas un Trigger messages de un Flow, ese texto no va a estar como ejemplo de esa intención en el proveedor.

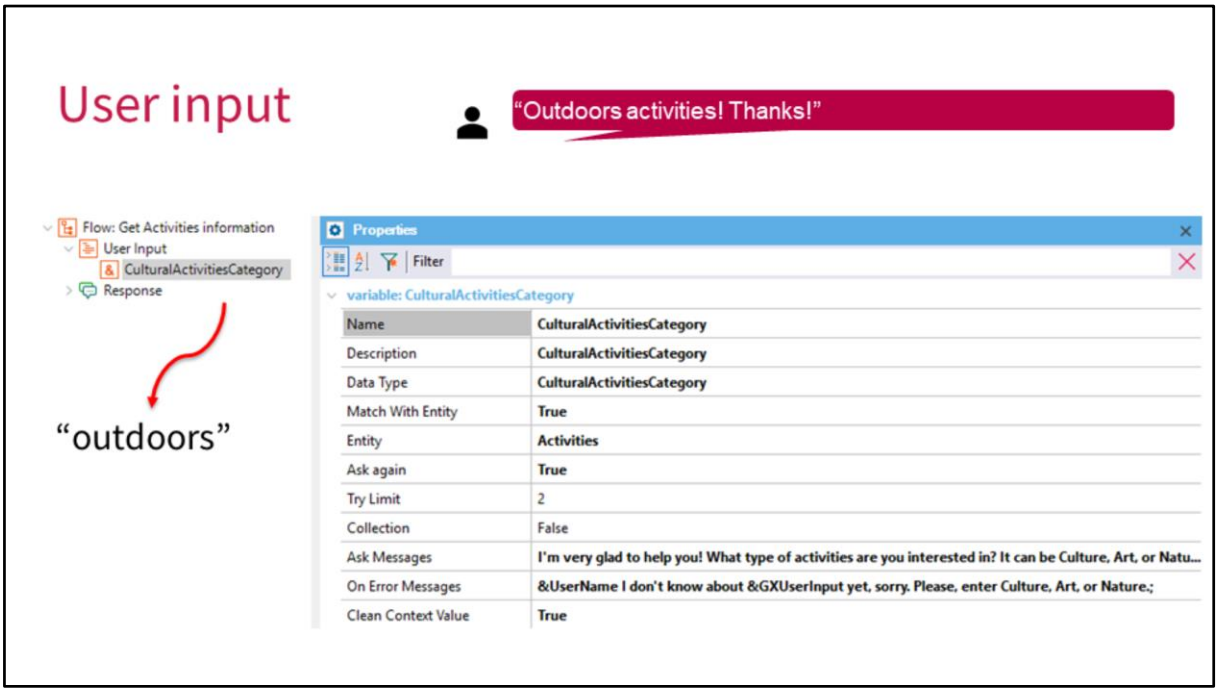
- Yo probé un poco estoy lo que vi es que siempre GeneXus me genera el proveedor, el objeto en Inglés y que el proveedor soporta otros lenguajes, no encontré donde se podían cambiar.
- Si, por default estamos generando en Inglés, tenemos que agregar la propiedad para que puedas definir el idioma, pero lo vamos arreglar, vamos a agregar los otros idiomas que tengas así no queda siempre en inglés .




Seguimos con el ejemplo,

Lo primero que el usuario preguntaba cuando decía que quería hacer ciertas actividades, se acuerdan que el bot le contestaba "bueno, te puedo contestar acerca de cultura, arte y naturaleza"

Habíamos dicho que esto era una entidad, una entidad que ya dijimos, la tenemos en assistant o dialog flow o el provider que sea, y esto se crea automáticamente, las entidades con sus valores y sus sinónimos y a través de este método que hace un ratito les decía "SendEntityValues" yo puedo cargar los valores de esas entidades en el provider.




Ahora, siguiendo con el flujo ¿cómo hice yo para preguntarle al usuario "yo le puedo contestar acerca de esto, esto y esto? ¿Cómo hicimos para pregunarle?
Bueno, tenemos un User input, en nuestro flujo un User input, tenemos una propiedad Ask messages, en esta propiedad le decimos cual es la pregunta que le vamos a hacer al usuario, ¿para que? Para que conteste… El bot le va a hacer esta pregunta para que el usuario conteste .

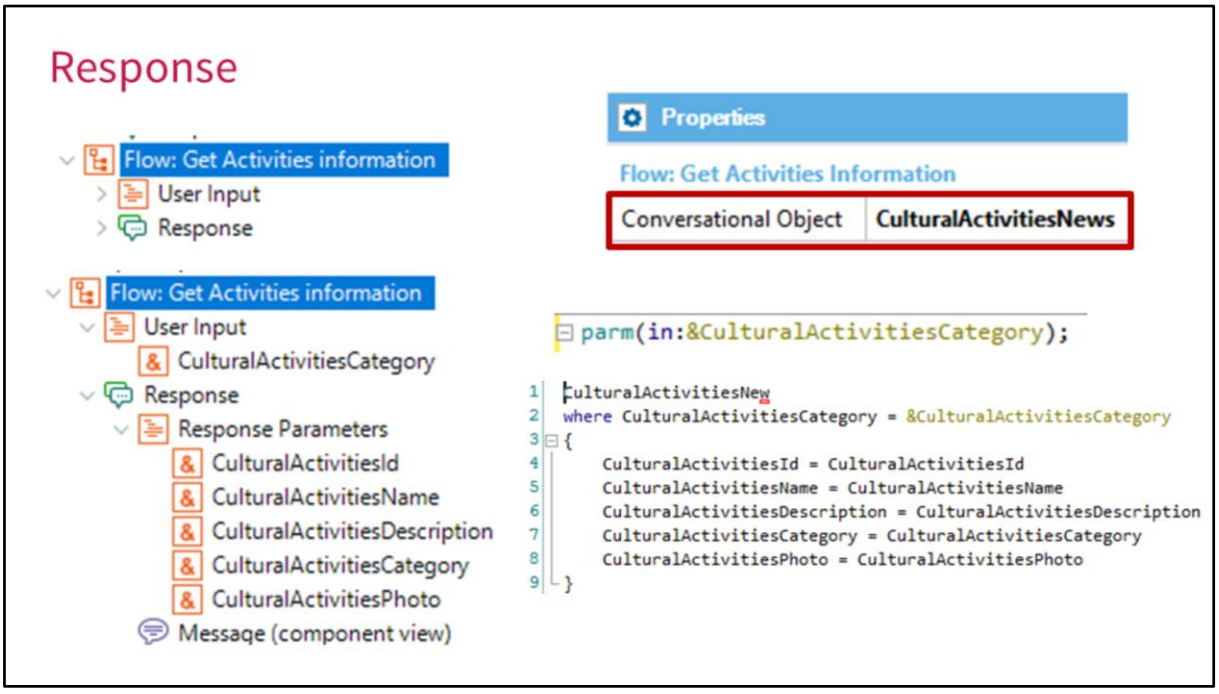
En particular este User input tiene la propiedad Match With Entity con el valor True ¿esto que quiere decir? Que de lo que el usuario diga, se va a extraer el valor de una entidad, ¿de cual entidad? De esta… la que está especificada en la propiedad Entity
Si el usuario le erra, en el sentido que no ingresa nada que tenga que ver con esa entidad, entonces podemos manejarlo a través de la propiedad Try Limit, que es la cantidad de veces que yo le voy a decir al usaurio "no, no te entendí, se contestar acerca de cultura, arte, naturaleza" , eso se lo voy a preguntar 2 veces, porque tengo Try Limit 2
Y en On Error Messages tengo mensajes donde le dice justamente "no, de eso no te se contestar, solo se contestar de esto, esto y esto"

Fijense en particular, que acá dice "UserName" (dice la variable User Name, como si fuera una variable), esto es porque User Name está en el contexto, yo alguna vez hablé con el usuario y se como se llama, por eso le digo "fulano" no se contestarte acerca de about GXUserInput, GXUserInput también está en el contexto y es lo que el usuario me acaba de decir, por ejemplo, el usuario me dice "yo quiero hacer actividades con la bicicleta", no entendí lo que dice, entonces yo le digo "Pedro, I don´t know about actividades con la bicicleta, ¿por qué? Porque no está dentro de los valores de la entidad.

Bien, cuando se instancia el valor, cuando se entiende el valor de la entidad, quedo instanciada en esta variable , yo voy a precisar esto obviamente porque después tengo que filtrar, tengo que

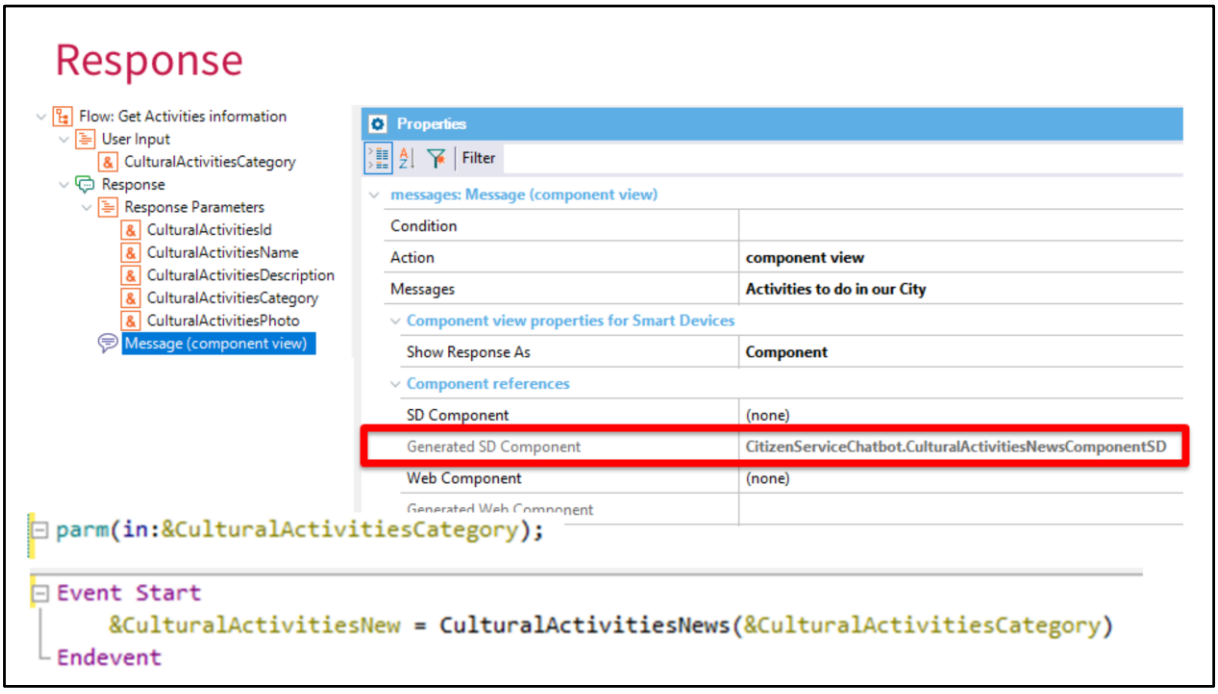
ejecutar un objeto que me de las actividades que están según este filtro por supuesto, así que eso lo voy a precisar y queda guardado para mi como diseñador, como programador de este modelo…

- Ahí donde está On Error message, vi que se puede generar no solamente un mensaje sino muchos… un listado de mensajes, en ese caso ¿Cómo hace para responder uno u otro, como se elije un mensaje …? **(26:06)**
- En realidad, lo que va a hacer es tirártelo de forma “Randomica” digamos, es aleatorio, no es uno atrás del otro sino de esa colección se tira uno. No siempre es el miso, eso hace que el Run time te tire mensajes siempre distintos y que suene un poco más inteligente.
Que no parezca un robot.

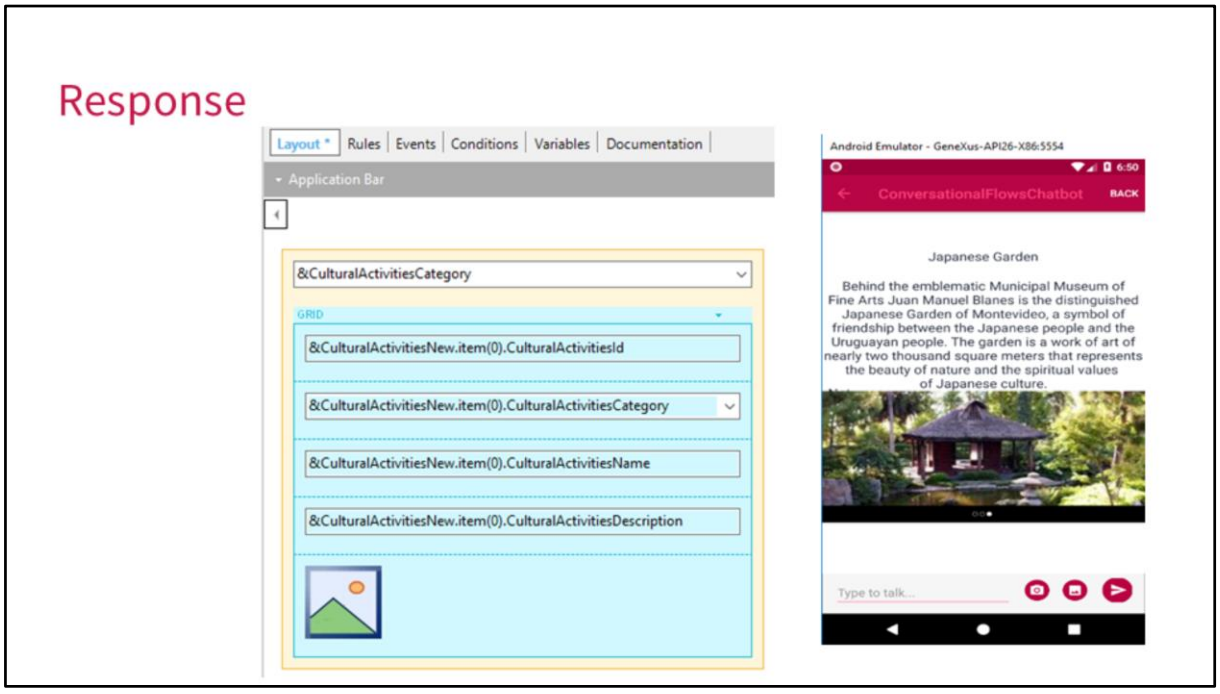



Bueno, entonces la respuesta…
Ya se lo que quiere el usuario, ahora voy a buscar una respuesta, para la respuesta lo que voy a hacer es ejecutar (se ejecuta en realidad automáticamente) un objeto que está indicado en la propiedad Conversational Object.
En este caso (podría ser un procedimiento) hicimos un Data Provider que obviamente va a recibir como parámetro el tipo de actividad que quiere realizar el ususario y vamos a buscar todas las actividades filtradas por ese valor.

Importante, cuando nosotros asignamos la propiedad Conversational Object, automáticamente se infiere en mi flujo, todo lo que son los Input de ese Data Provider, de ese objeto y el Output de ese Data Provider. O sea, yo originalmente tenía un flujo así…"pelado" digamos, le asigné la propiedad Conversational Object con un Data Provider, este Data Provider recibe como parámetro el tipo de actividad y tiene como salida una colección, entonces automáticamente en el flujo tengo (se infiere) la entrada y la salida…




¿Eso para que?, porque tengo la posibilidad de que se genere en forma automática, un objeto con esa información, ¿para qué? Para la respuesta, se va a generar un Component, en este caso un panel SD con esa información que está en mi modelo.
Básicamente este objeto lo que va a hacer es llamar a mi Data Provider…



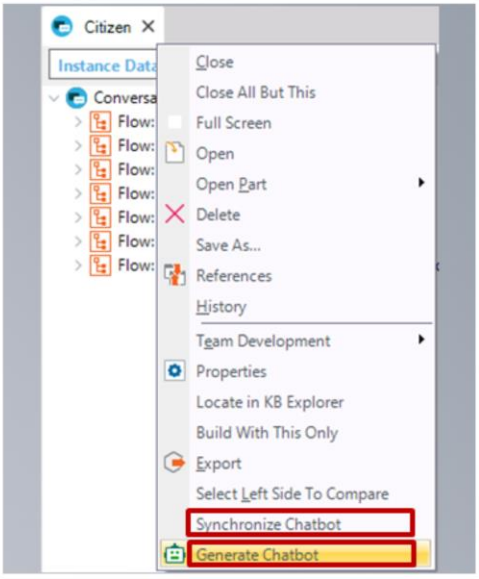

¿y que tiene en la pantalla?
La pantalla lo que tiene es los parámetros de salida del Data Provider, que en realidad como se trata de una colección es un Grid, es un Grid con todas las actividades disponibles, filtradas justamente por lo que quiere hacer el usuario.

No tengo obligación de usar el panel generado automáticamente, puedo hacer un panel creado por mi.

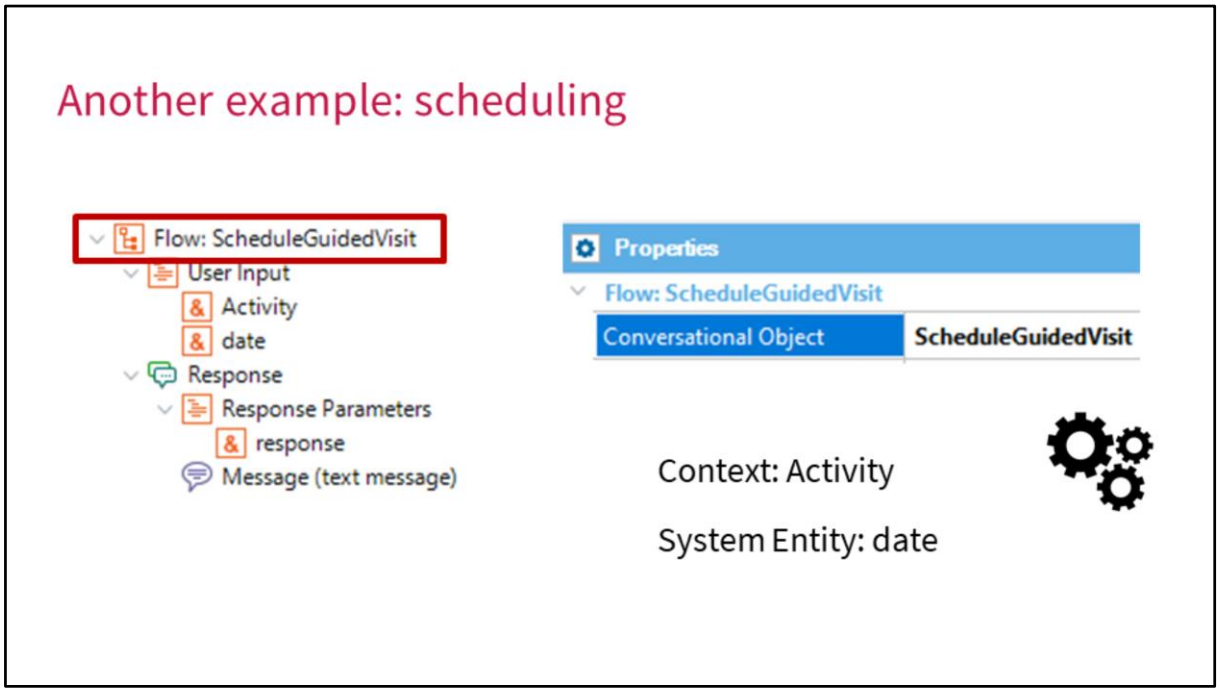
# Impacting model changes

Synchronize Chatbot

Generate Chatbot




Entonces, una vez que cree mi modelo, ¿Cómo se impacta, cómo se genera este modelo? Bueno, cuando salvamos la instancia que yo comentaba ahora, se impacta mi modelo en el proveedor pero también se puede forzar ese impacto con la opción Synchronize Chatbot, hacemos botón derecho sobre la instancia Synchronize Chatbot y se impacta mi modelo en el proveedor. Tenemos la opción Generate Chatbot, que también ocurre en el Build si es necesario y lo que hace es generar todos los objetos en mi KB, todos los objetos generados de los que yo les hablaba hoy, esos bjetos se generan en la opción Generate Chatbot.

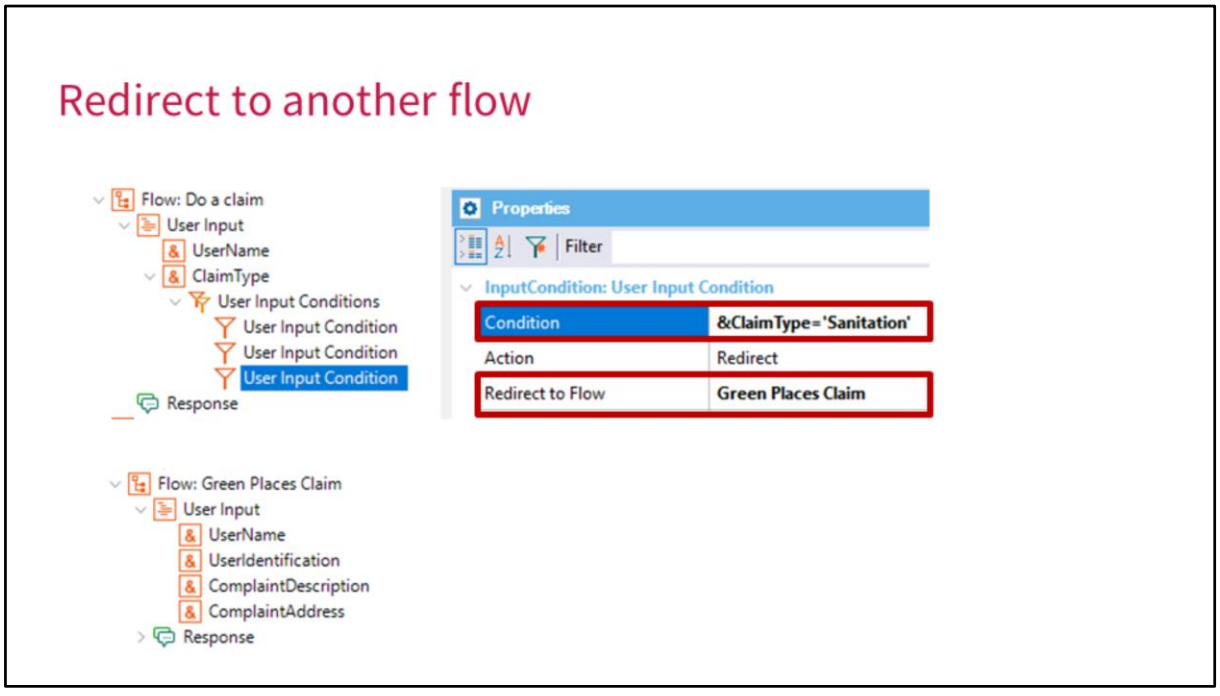



Y el otro ejemplo que estábamos viendo hoy, es el de agendar una visita guiada.
Acá me pareció importante (es igual al otro, no tiene mucha diferencia), solo que me pareció importante traerlo por dos detalles nomás:

La actividad no es necesaria solicitarla al usuario porque la actividad está implícita, está en el contexto, ya hablamos, ya se sabe cual es la actividad, por lo cual cuando quiera "Schedular" la visita guiada obviamente es la actividad de la cual estamos hablando y eso es importante justamente de lo que hablábamos hoy, para que parezca una conversación con una persona.

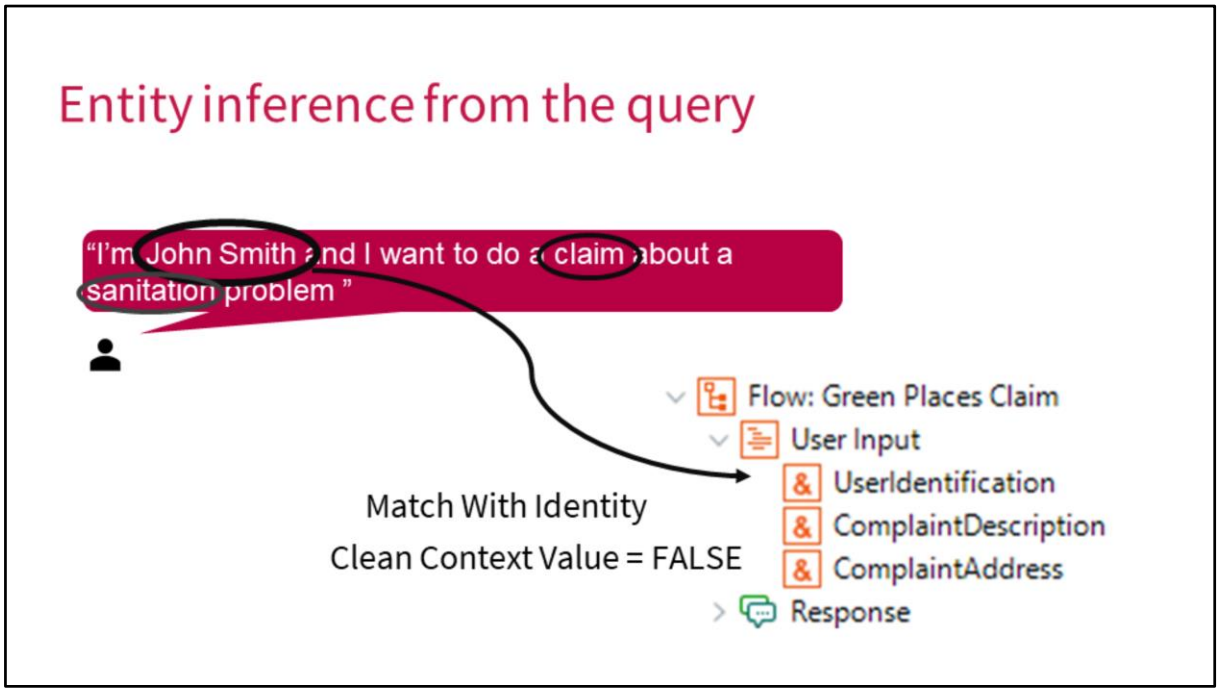
… Y el User Input de fecha, se acuerdan que cuando se le pregunto al usuario ¿cuándo queres agendar? El ususario dijo: "Tomorrow 5pm", todo eso es interpretable, ¿por qué? Porque eso está definido como una entidad del sistema dentro del proveedor y el proveedor se encarga de interpretar esa expresión, que sea mañana, de tarde… bueno, de tarde no, obviamente, refiriéndose a las fechas, cualquier expresión que se pueda referir a una fecha.
Y el objeto conversacional en este caso era un procedimiento simplemente.

El objeto conversacional puede ser cualquier tipo de objeto GeneXus.



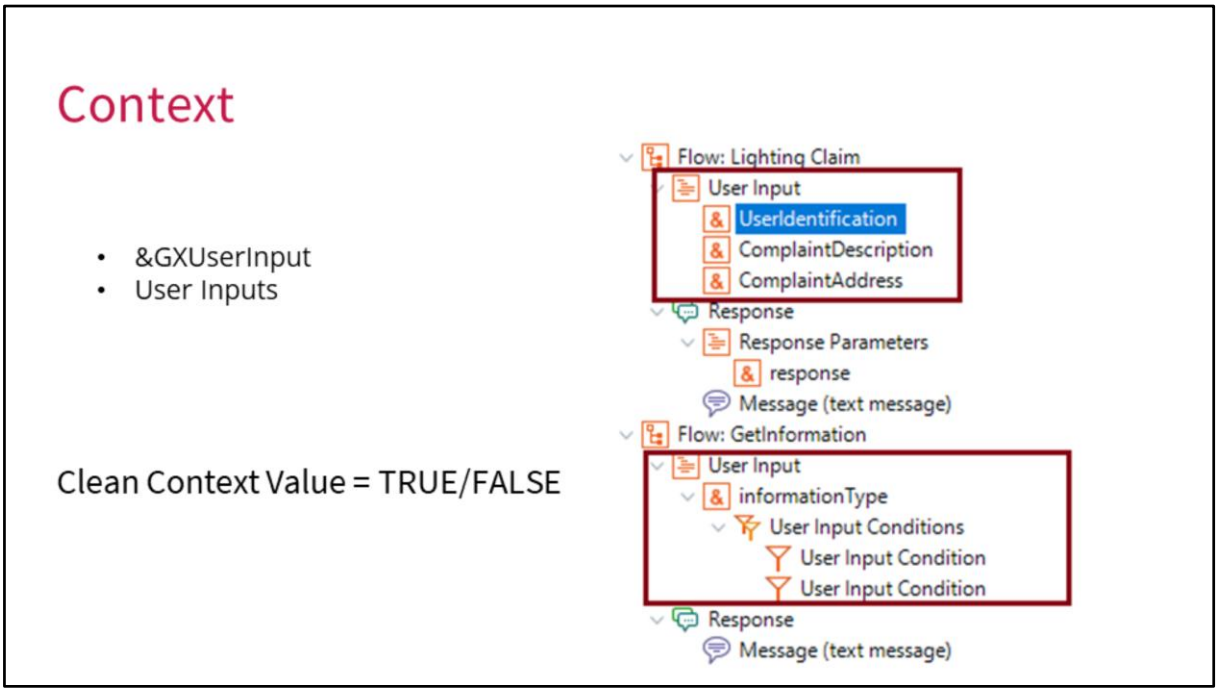

Bueno, los flujos, si bien resuelven una intención, puede pasar en algún caso que el flujo por si mismo no la resuelva y que sea necesario redirigir a otro flujo, por ejemplo… supónganse que se quiere hacer un reclamo y según el tipo de reclamo se le quiere pedir diferente información al usuario. Si es un reclamo acerca de saneamiento, por ejemplo se le quiere pedir "¿dónde hay un problema de saneamiento?", en ese caso lo que se hace es, dentro del flujo tengo "User Input" que es el tipo de reclamo y tengo una "Condition" para el User Input y simplemente le digo "bueno, si el tipo de reclamo es de saneamiento, lo voy a redirigir a Action- Redirect, a un flujo que en mi caso se llama Green Places Claim y acá le pido la información pertinente al reclamo acerca de Saneamiento, le pido su identificación y dónde ocurrió el problema, etc.

Importante que el usuario puede directamente entrar por acá (ComplaintAddress), el usuario en vez de decir "quiero hacer un reclamo" puede decir "quiero hacer un reclamo por un problema de saneamiento, entonces el Intent que se reconoce directamente es "hacer un reclamo por saneamiento" no es que estén anidados y necesariamente tengan que entrar por el primero.



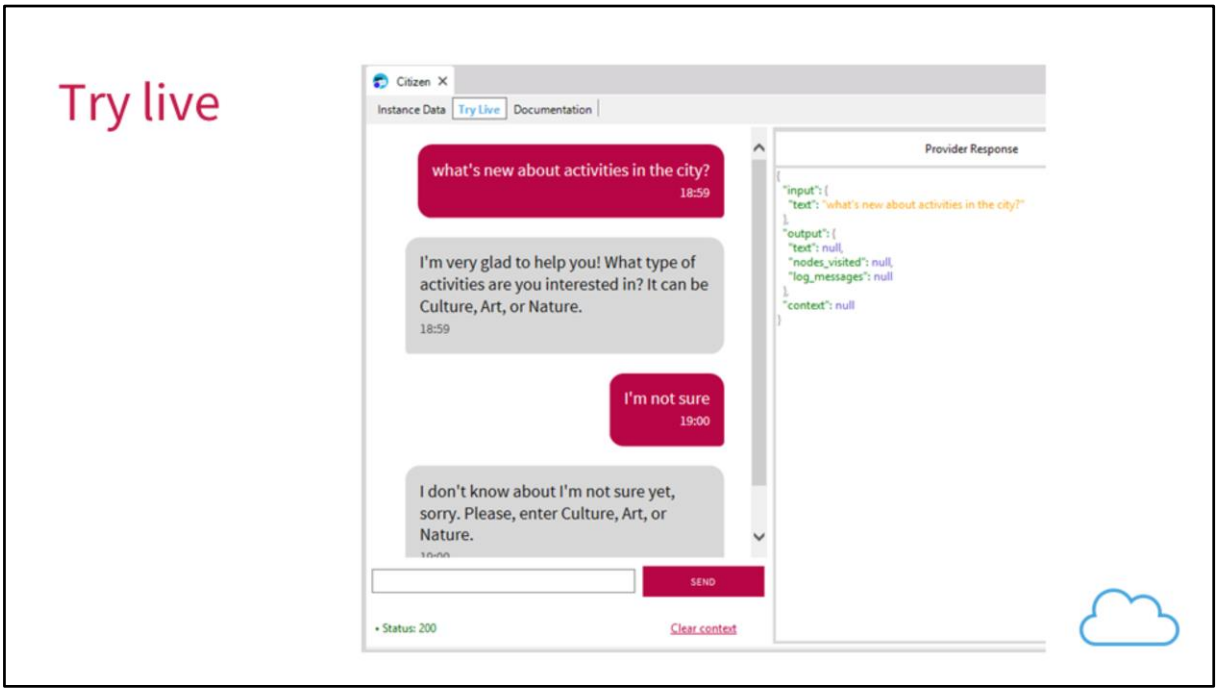

Y este es este caso, dice "Soy fulano y quiero hacer un reclamo acerca de saneamiento", se interpreta y entra directamente por el flujo que corresponde.
Pero fíjense acá algo importante, que el usuario ya dijo quien era, entonces para que parezca humano esto, yo no le puedo preguntar de nuevo ¿Quién sos? Porque ya me dijo quien es de entrada.
Entonces este User Identification, este User Input tiene match Identity, hay una entidad en el proveedor donde tengo todos los usuarios registrados, ya se quien es, por lo cual no se lo voy a pedir de nuevo, ya queda instanciado el User Identification con este señor.

Y sería bueno acá, no limpiar el contexto, quedarme con el User Identification porque de repente el usuario hace otro reclamo o lo que fuere y no se lo voy a solicitar.




Entonces ahora les cuento muy breve porque no tenemos mucho tiempo…

Qué es el contexto que venimos hablando, el contexto se maneja a nivel de los User Input, todo lo que son User Input pueden quedar persistidos en el contexto como les decía hoy, para no pedírselo de nuevo.
Eso es a través de la propiedad Clean Context Value, si yo no limpio el contexto en la sesión, me quedo con ese valor y además tenemos el contexto GXserInput que es lo que el ususario me acaba de decir, que está muy interesante poder repetirle lo que me dijo, para que parezca una conversación mas humana.




Y les cuento brevemente que tienen dentro de GeneXus un Try Live que es un poco para ver como respondería el proveedor, vamos derecho contra el proveedor, entonces vamos, le tiramos la pregunta y por lo menos vemos…
No ejecuta objetos ni SD ni web, pero por lo menos sí nos tira de este lado, la respuesta del provider y tenemos por lo menos más contexto de lo que está pasando, es muy útil.

- ¿Hay forma de settear el contexto antes de arrancar, en el caso de tenerlo en una aplicación en donde vos sabes quién es? Ya podes settear el User Input y ese tipo de opciones..
- Sí, podemos grabar el contexto, lo que pasa que no lo tenemos Built in, ¿ No Diego?
- En realidad hoy se puede hacer pero “a mano”, tenemos pensado agregar métodos a la Api que ya tenemos para que se pueda modificar más fácilmente, de hecho si puedo voy a complementar con una cosita.. No se si ibas a preguntar algo más..
- No, no, del tema del idioma… ¿Cómo lo resolvieron en la aplicación del evento?
- En la aplicación del evento, está entrenado totalmente en Inglés y usando las traducciones en Run Time que nos da GeneXus ahí, este módulo nuevo que tenemos, lo que hacemos es pasar a inglés todos los mensajes que llegan, entonces tenemos muchos “Confidence”, lo que mencionaba Sabrina de la confianza así tenemos mucha mas certeza a la hora de contestar. Y después obviamente pasamos la traducción para atrás para el idioma en el Devices.

Lo otro que sí quería mencionar es, hoy cuando preguntaste cuando se agrega un objeto nuevo ¿Qué es lo que pasa?, es decir además del caso en el que se sincroniza guardar, además lo que estamos queriendo agregar a la Api también, es la posibilidad en Run Time poder agregar esos objetos, entonces puedes tener tu propio “Panel” donde digas “este mensaje va para esta intención” y lo puedas solucionar sin tener que ir a abrir GeneXus, abrir Convertational Fows y hacer esas modificaciones.

- Supongamos que estoy haciendo un Chatbot para un supermercado y defino una entidad que se llama “producto” y resulta que tengo miles de productos, ¿es razonable hacer eso cuando

tenemos mucho volumen de datos? ¿y si lo testearon, hasta cuanto aguanta un chatbot de subirle cantidad de entidades?

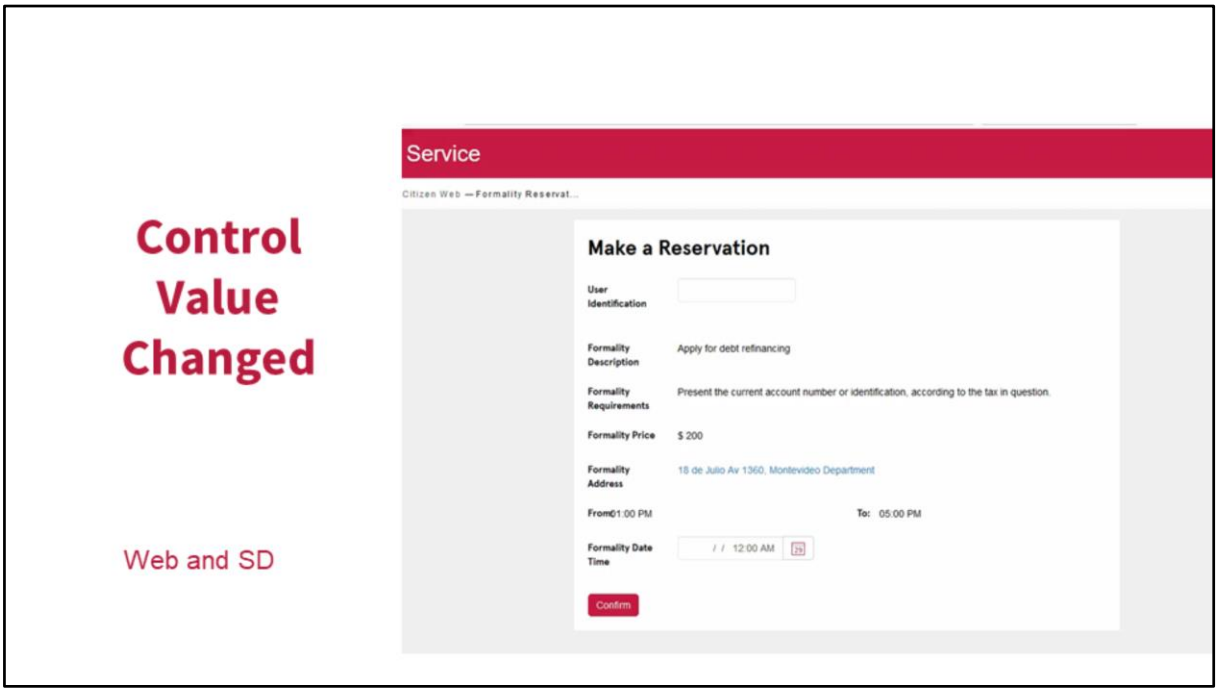
- Bueno, es una pregunta bastante complicada en realidad porque justamente no parece la mejor práctica subirle mil, dos mil datos a la misma entidad por un tema de entrenamiento, ahí hay que ver en cada modelo, cuál es la mejor forma de resorverlo.

Esto la otra vez me lo preguntaban ya por casos un poco más extremos, cuando tengo 1 millón de usuarios ¿Qué hago? Necesito, es decir, en ese sentido es mejor buscar la manera de resolverlo ya con otro tipo de parámetros.

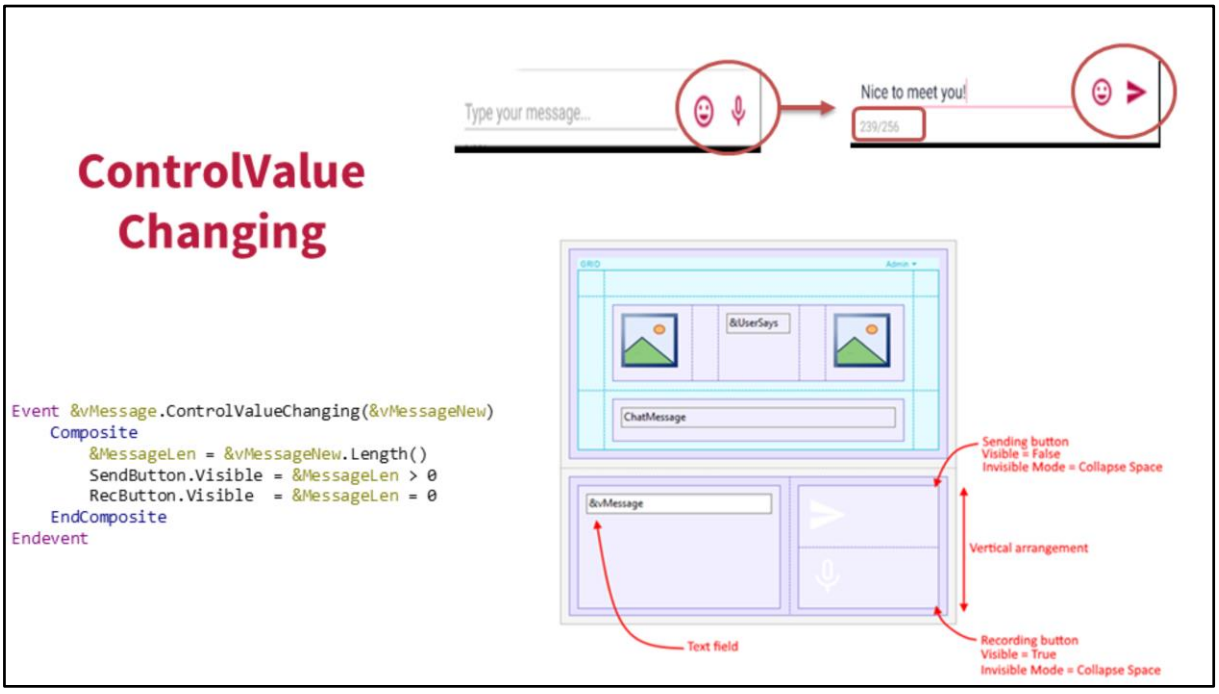
En este caso de los productos, de hecho vas a tener que diez productos se llaman prácticamente igual y eso te va a dar a confusión, entonces hay que ver bien como modelarlo al caso.

- ¿Está pensado sacar de la interfaz de GeneXus del editor de Chatbot? Por ejemplo, cuando con el tema de las traducciones…
- Por ahora no
- Dentro de los proveedores ¿hay uno recomendado que tenga más funcionalidades que otro? Yo vi en la documentación que en algunas partes decía solo válido para Watson, por ejemplo.
- No no, es Watson Assistant y DialogFlow y no tenemos una preferencia para uno u otro, estamos avanzando…
- ¿No hay mas funcionalidad en unos que en otros?
- No, en algunos casos en particular nos proveen, por ejemplo, el tema de condicionar del lado del proveedor (Condicionar los flujos del lado del proveedor) Watson Assistan lo tiene y DialogFlow no, pero lo podes resolver usando DialogFlow también, es decr, siempre vas a terminar eligiendo un poco en base a tu ambiente, si estás trabajando con todo cosas de IBM, entonces seguramente vas a ir seguramente por Watson Assistent, pero en realidad ahí lo que te conviene es probar y ver cual te funciona mejor, justamente una de las cosas que te permite esto, es que todo lo que tenes definido en tu instancia de Convertational Flow, cambias una propiedad y luego decis “ahora pruebo con DialogFlow” , “ahora pruebo con Watson Assistant”, no tenes que tocar nada y seguir probando. Entonces, tenes las herramientas como para vos decidir cual es el mejor para tu caso.
- ¿Cuándo quieren hacer un proceso así de Humand y de Middle, cómo se hace para que después de que el chat hable con una persna, vuelva a encausar la conversación? No se puede…
- Si, eso hoy habría que hacerlo a “mano” pero se puede hacer, es decir, en el caso de que se esté equivocando mucho o en el caso de que la intención sea hablar cn una persona, se pondrán los controles para decir “ahora no le mandes mas mensajes al bot” “mandale el mensaje a una tabla de mensajes en lo que se esté hablando con una persona” y mediante todo lo que explicó Maxi, notificando de un lado y del otro para que los mensajes se vayan refrescando de los dos lados. Es decir, se puede hacer sí, hoy a mano, no tenemos la opción disponible, vamos a ir viendo un poco también ese tema porque nos interesa tenerlo.


Veamos ahora algunas novedades dentro de Smart Devices.

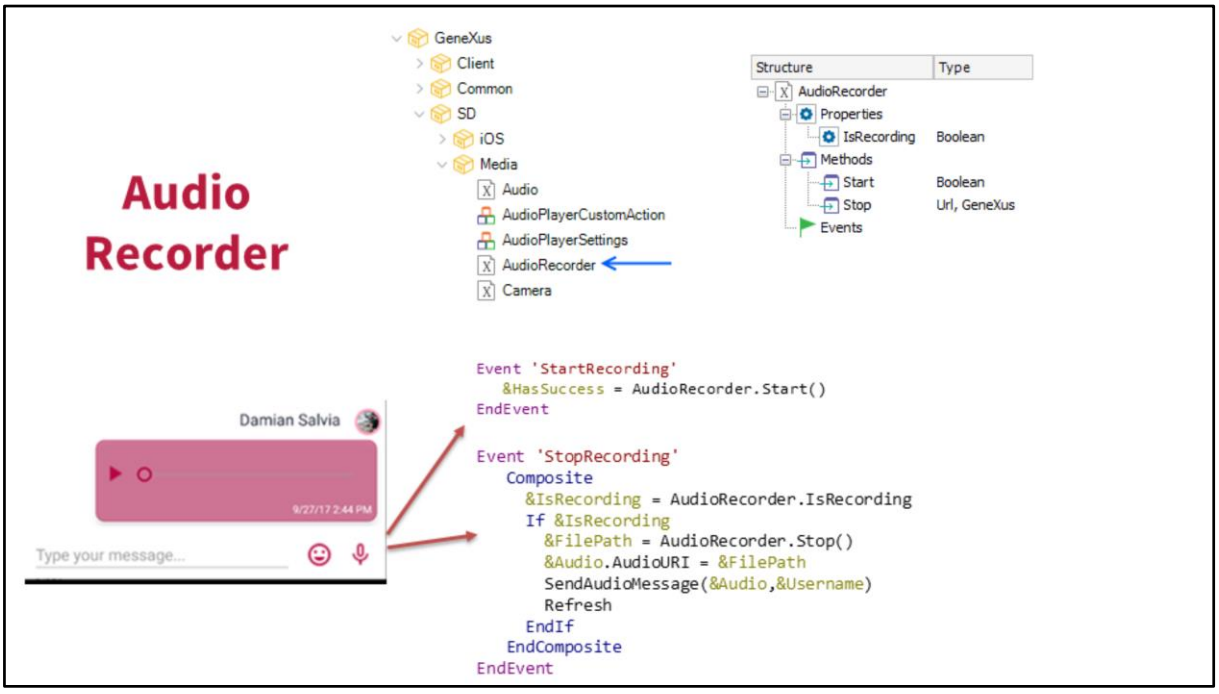



Control Value Changed, es un evento que nos permite ejecutar algún código justo en el momento en el que el usuario terminó de ingresar un valor.
Aplica tanto para web y SD.

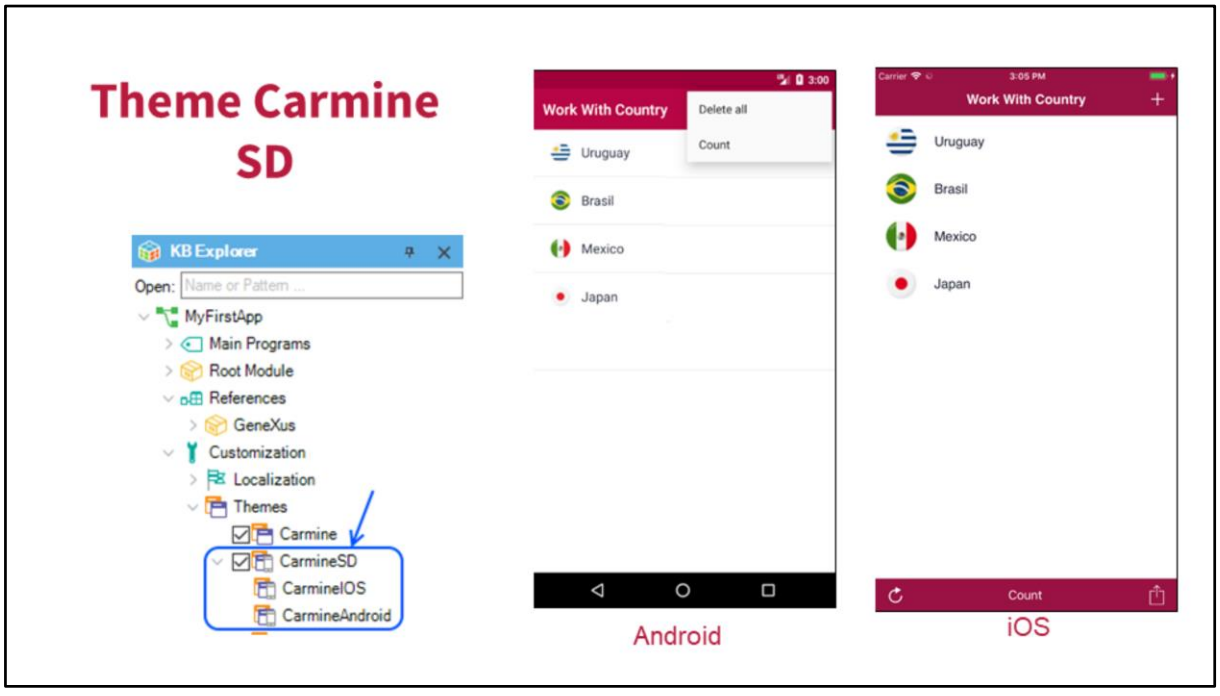



El evento ControlValue Changing nos permite validar si el usuario está cambiando el valor de un campo editable y ejecutar un código.

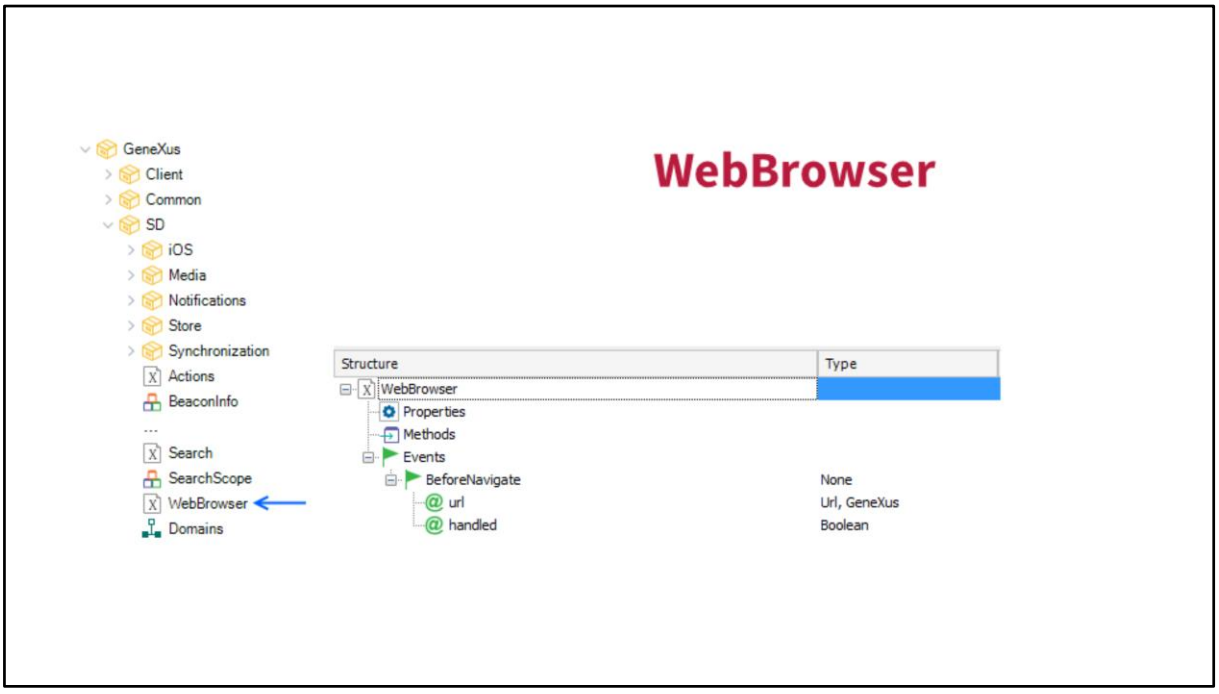
Por ejemplo, cuando el usuario no ha escrito aún tiene habilitado el botón para grabar un audio y cuando comienza a redactar el texto, aparece solamente el botón de enviar.
También lo podemos utilizar para llevar el conteo de las palabras que vamos a enviar.




Audio Recorder es un external Object que nos permite grabar una pista de audio, utilizando sus métodos Start o Stop.




Desde el Upgrade 6 de GeneXus 15, contamos con el tema Carmine SD, el cual nos permite tener un look & feel moderno y sofisticado.




Web Browser es un External Object que nos permite realizar una URL embebida en nuestra aplicación Smart Devices.

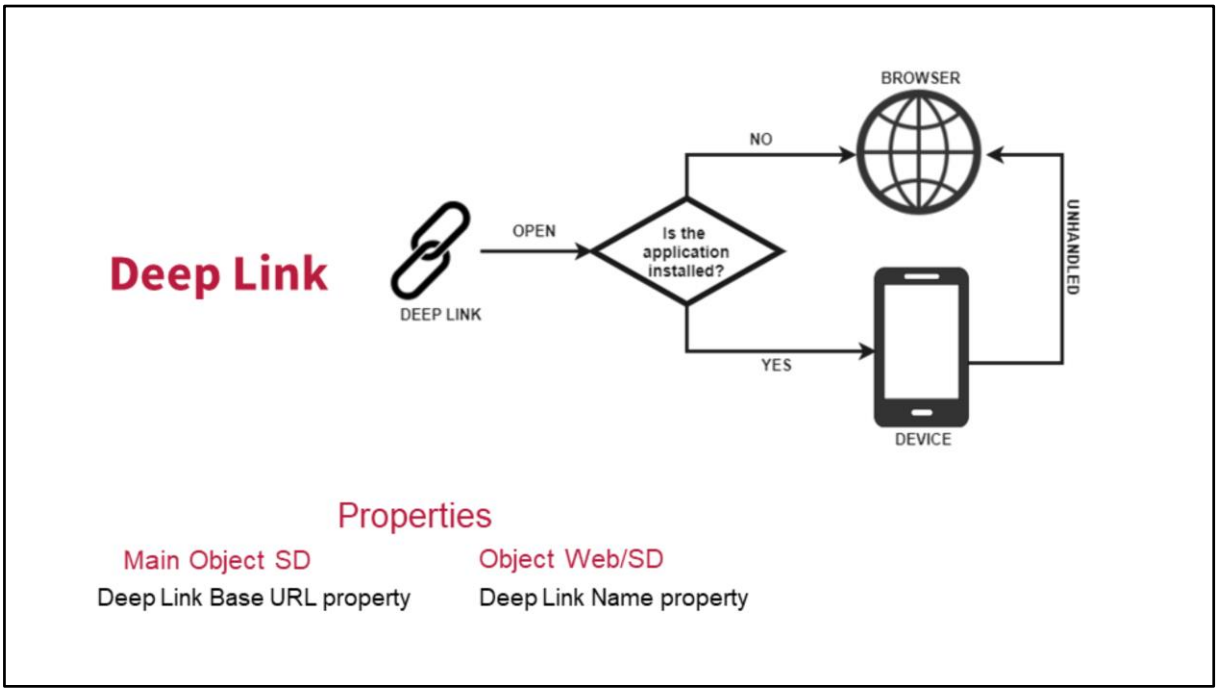
# Sample: WebBrowser

Before Navigate to URL or Web app Embedded

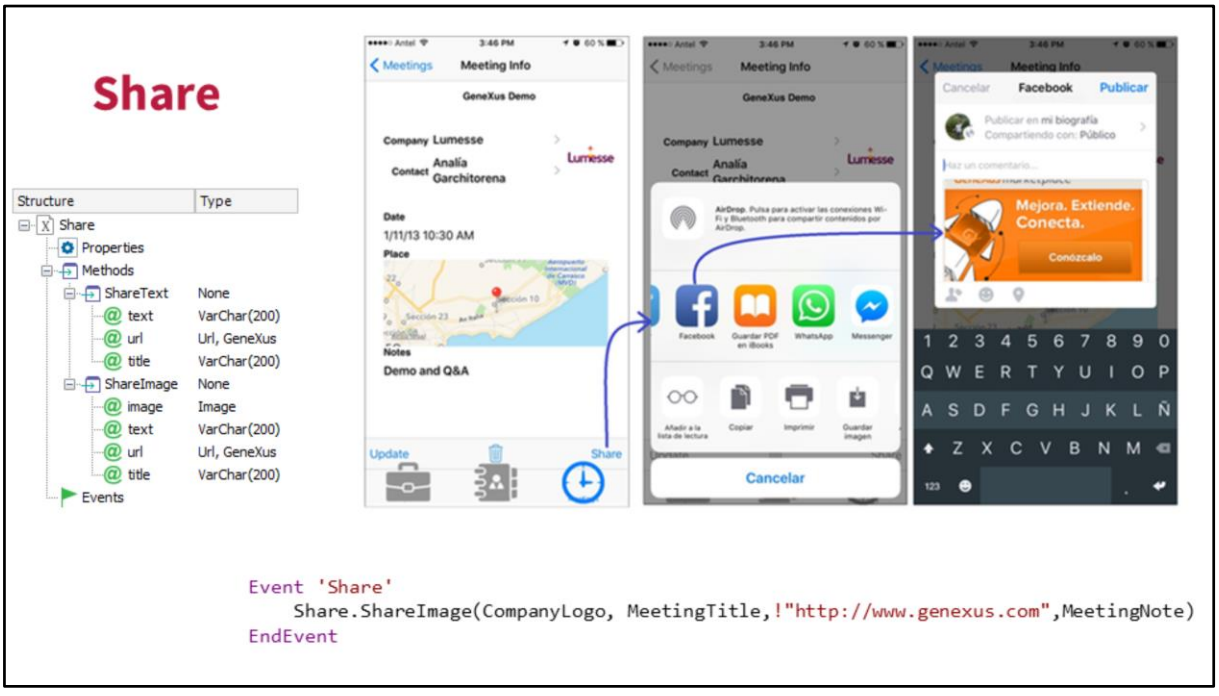
```
Event GeneXus.SD.WebBrowser.BeforeNavigate(&Url, &Handled)
   composite
     if &Url = //something
        //<your_code>
       &Handled = true
       return
     endif
   endcomposite
EndEvent
```

Web app Embedded in SD app

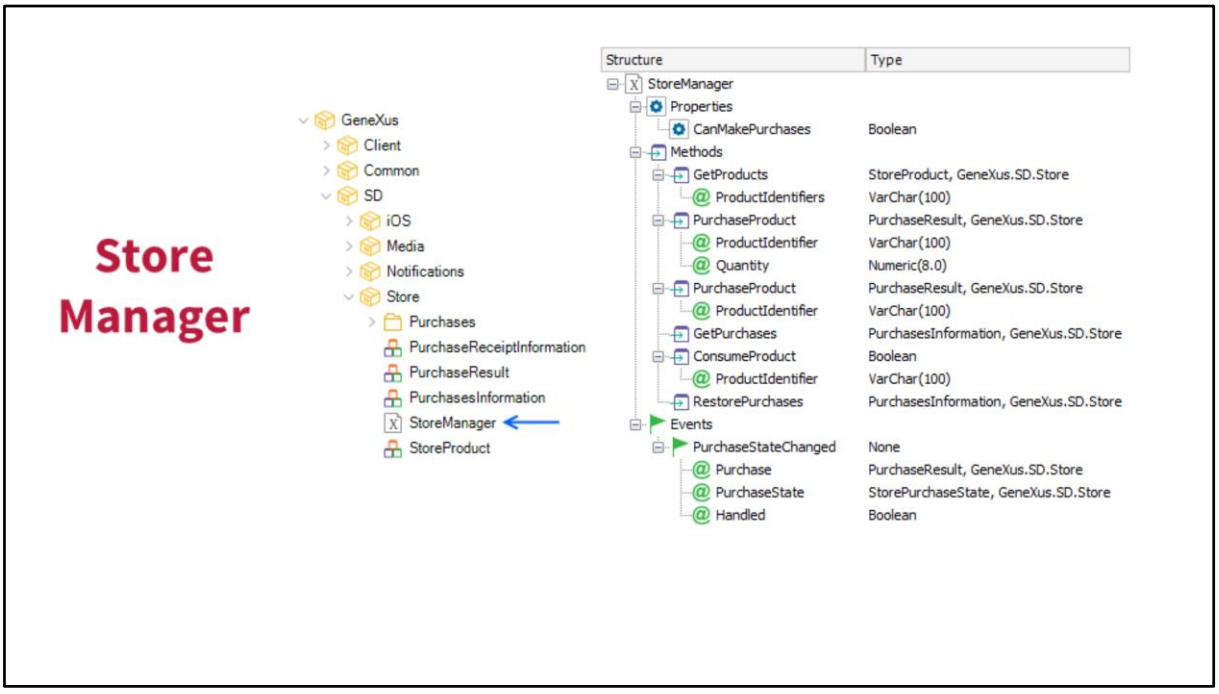

Para manejar la navegación, utilizamos el evento Before Navigate, el cual se dispara antes de navegar a una URL o aplicación web embebida.
Con esto controlamos, por ejemplo, si el usuario está entrando a una URL externa.



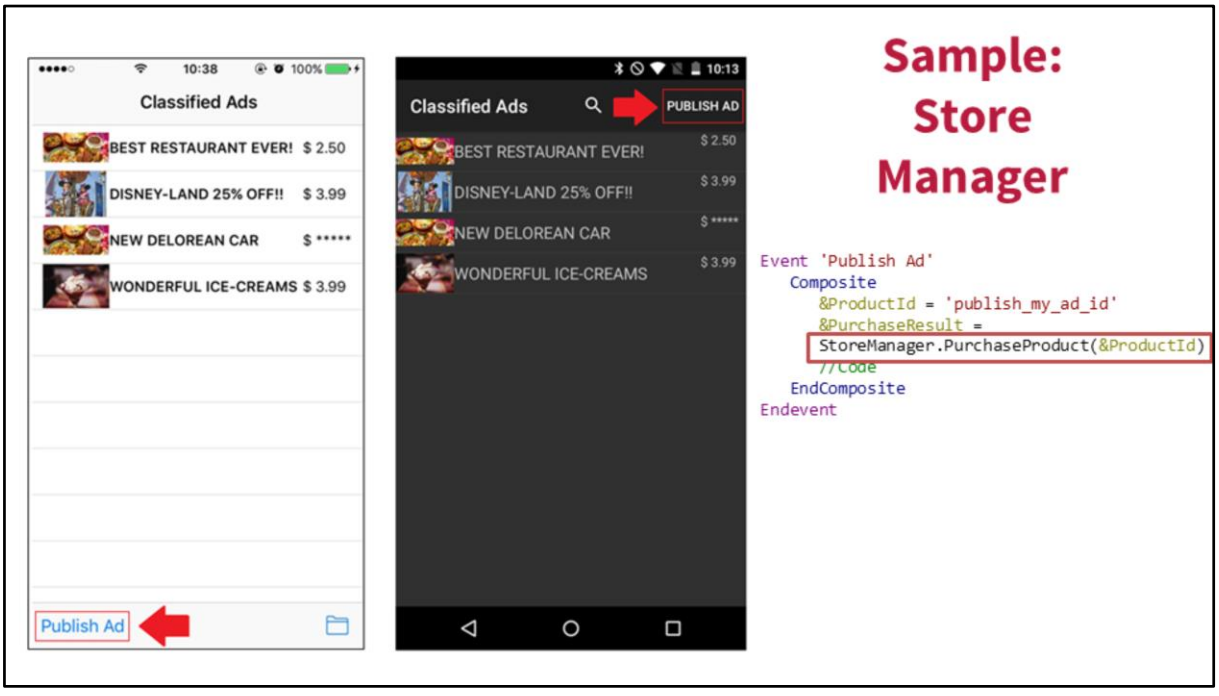

Deep Link nos permite elegir cómo abrir una URL, qué hacer en la aplicación Smart Devices si la misma está instalada o en el navegador web.
Para utilizarlo, configuramos las propiedades correspondientes a nivel de cada objeto.



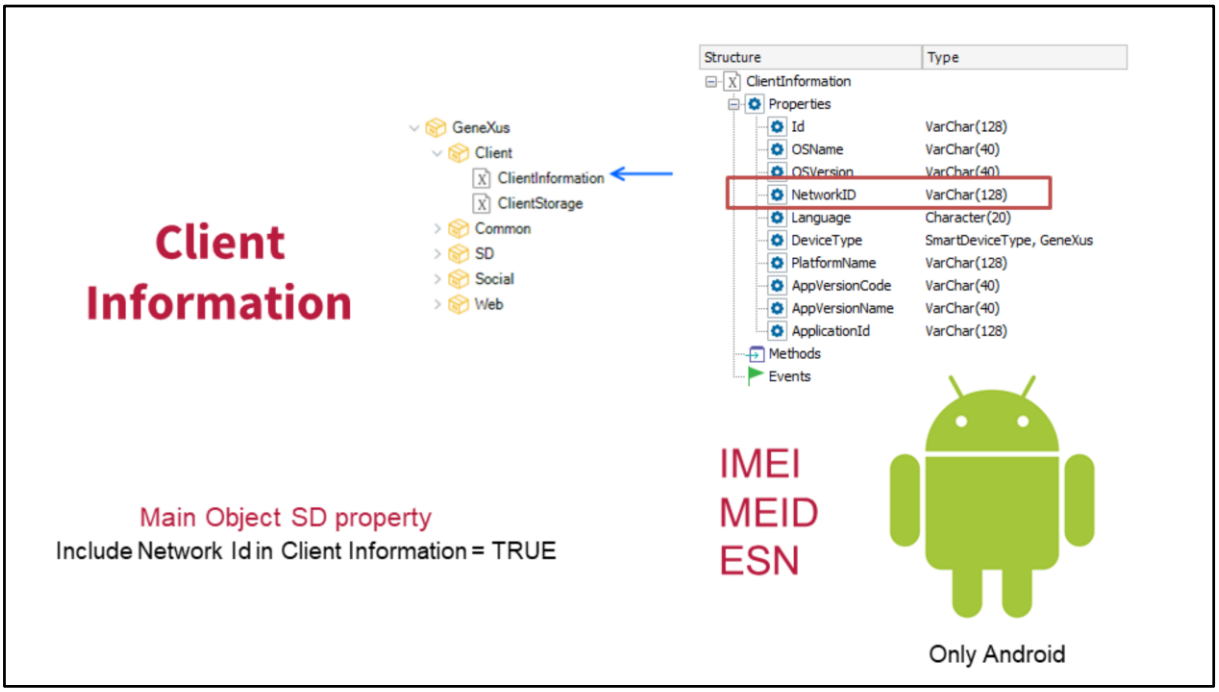

Share, es un External Object que nos permite compartir texto o imagen con alguna aplicación externa que lo permita.



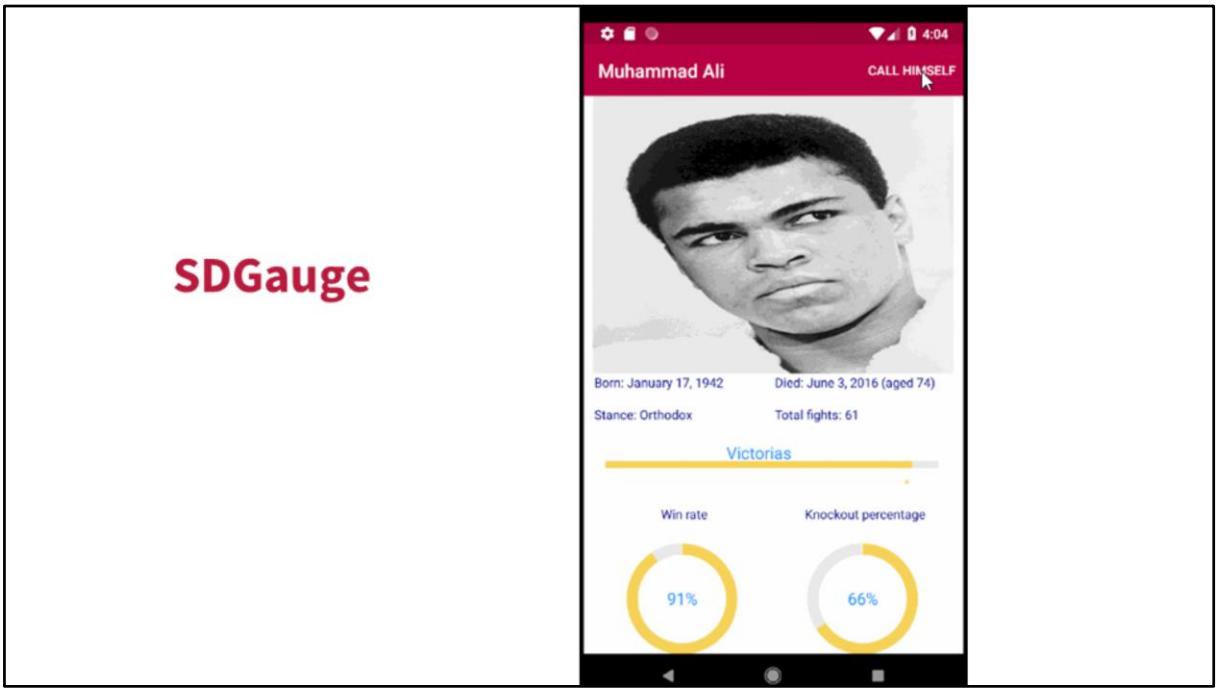

Store Manager es un external Object que permite administrar las aplicaciones de compras.



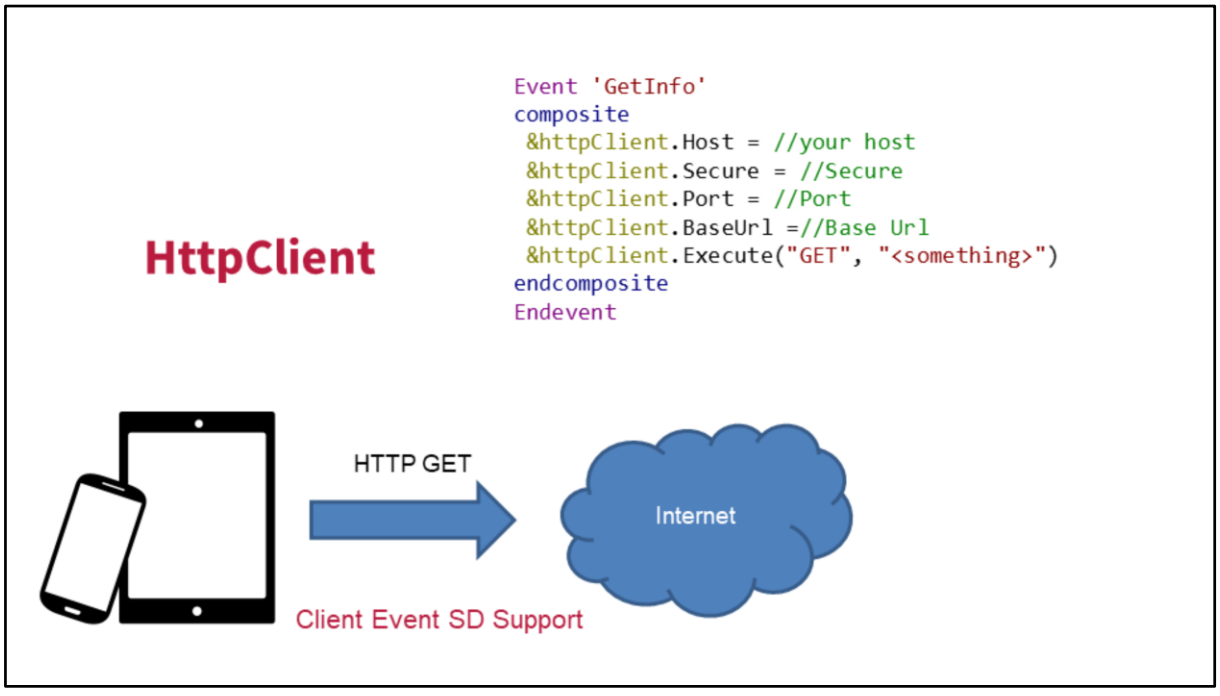

En este ejemplo, apreciamos que al momento de dar tap al botón Publish Ad, se despliega la opción de completar la compra.



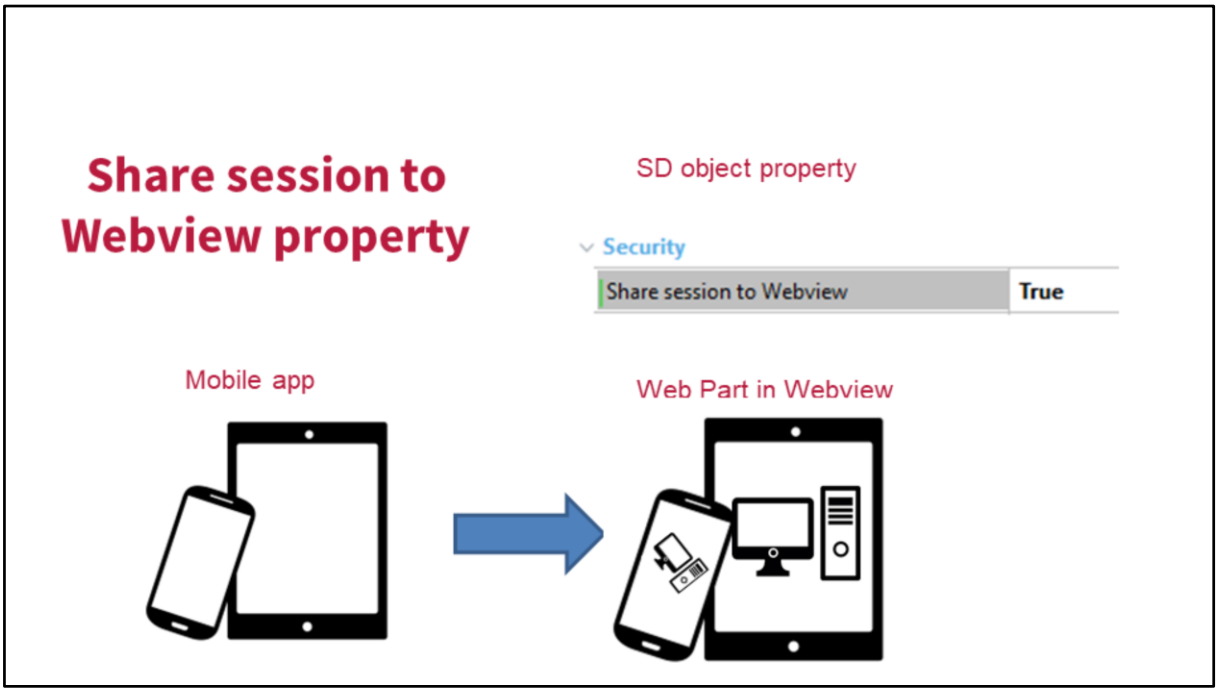

Client Information es un External Object que sirve para recuperar la información del Devices. Una de las características importantes, es su propiedad Network ID, que en conjunción con la propiedad Include Network ID, nos permite recuperar, por ejemplo, el IMEI del dispositivo.



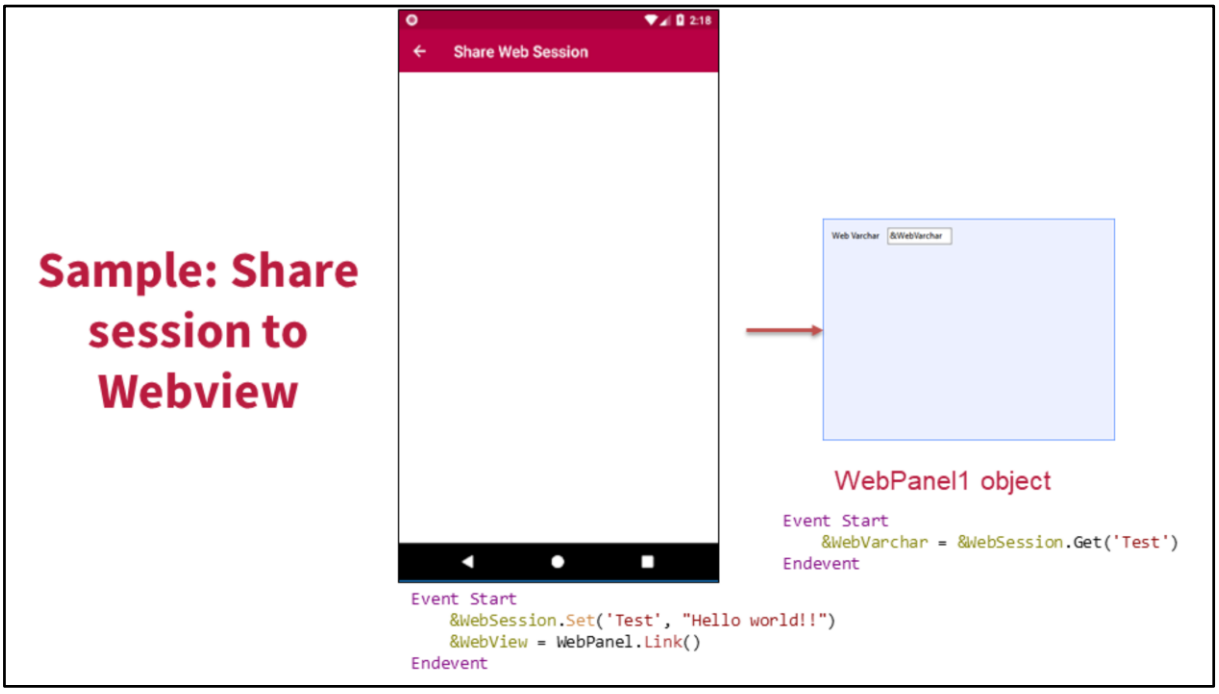

SDGauge es un control que nos permite desplegar información en forma de rangos, linear o circular.
Ahora en esta versión, el control cuenta con animación para mejorar la experiencia del usuario.




A partir de la versión 15, el tipo de dato HttpClient se puede utilizar desde aplicaciones Smart Devices y a partir del Upgrade 12, se permite el manejo desde los eventos del lado del cliente.

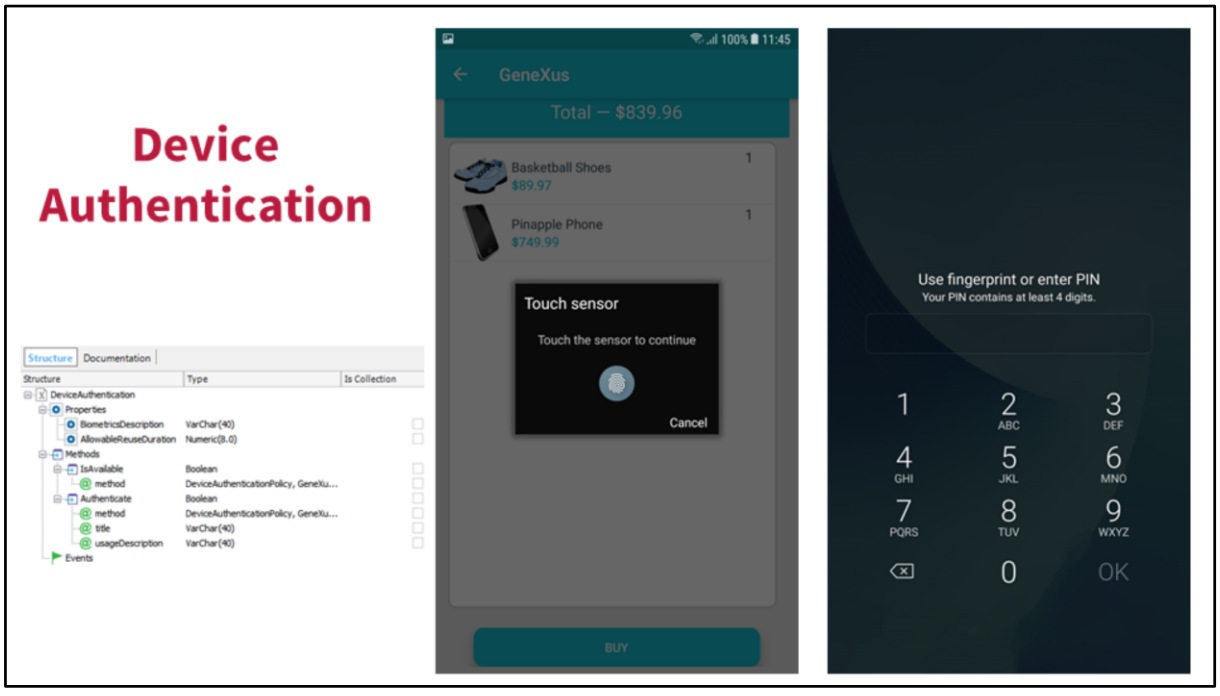



La propiedad Share sesión t Webview permite compartir sesión entre la aplicación móvil y su parte web.

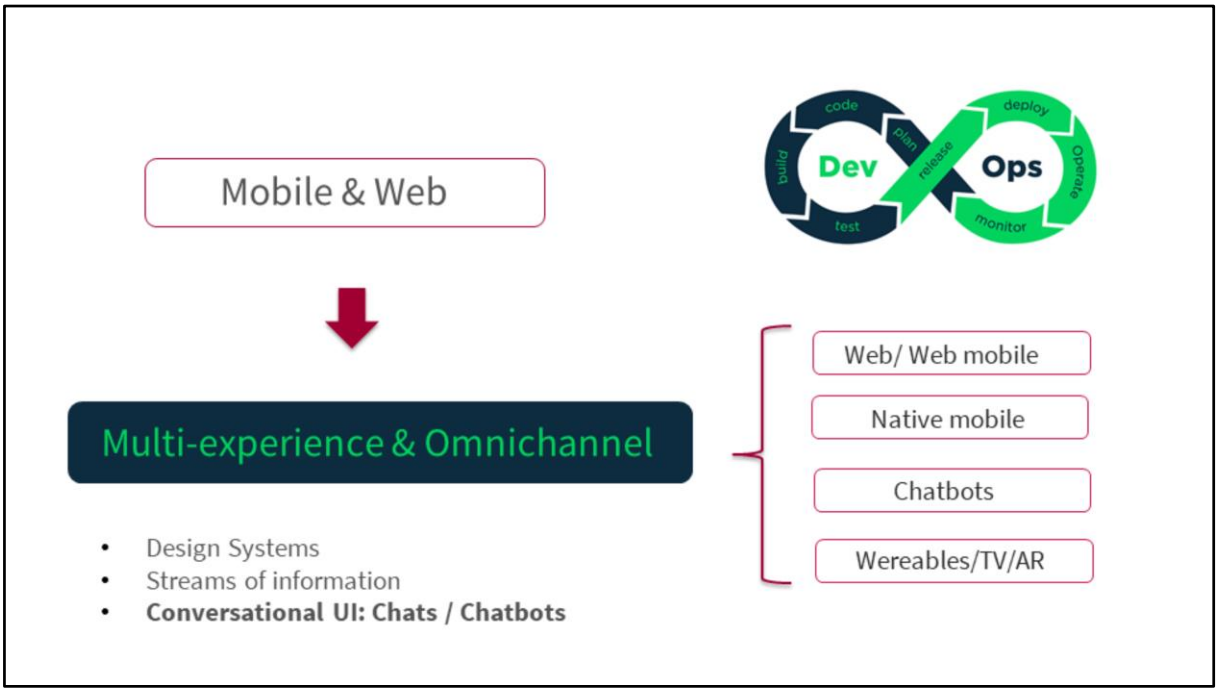



Por ejemplo, desde el objeto SD, seteamos la web sesión.
Posteriormente en un Web Panel obtenemos el valor de esa sesión.

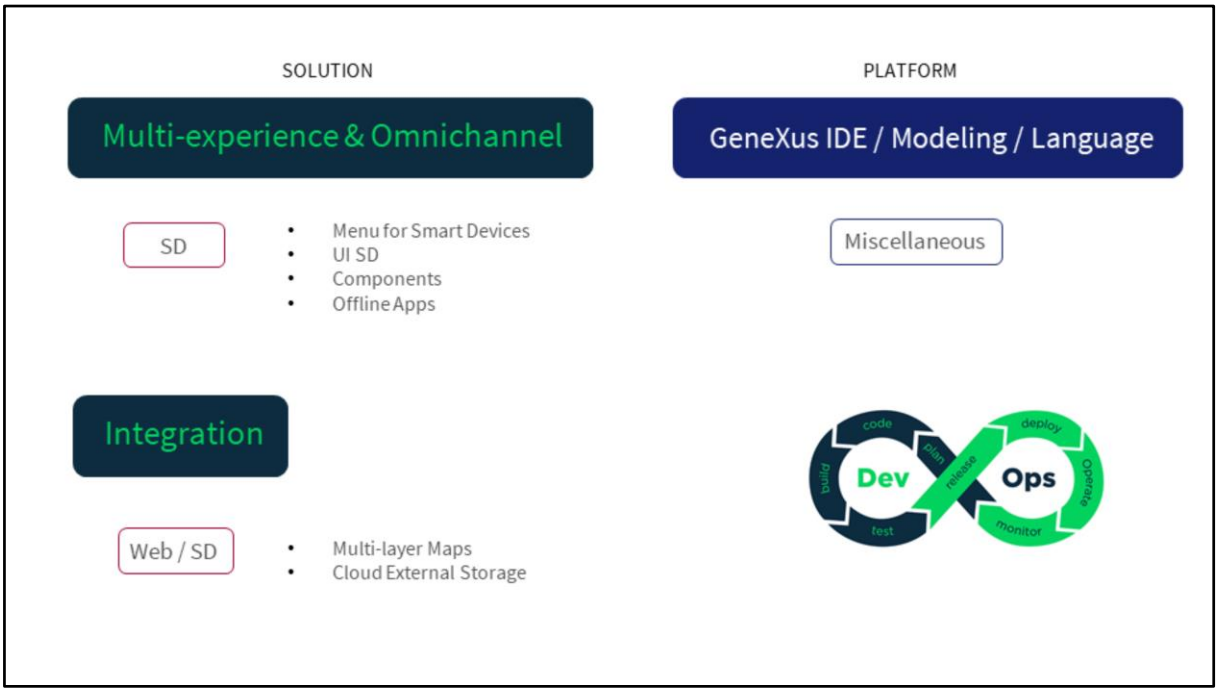
Y por último, mostramos el Web Panel dentro del Webview.




Device Authentication permite al navegador tener acceso a los métodos de autenticación del dispositivo.
Cabe destacar que cada plataforma maneja los propios.




Vimos todas las facilidades para desarrollar un chat simple y un chat bot.




En lo que sigue, veremos más funcionalidades que se han incorporado a partir de los diferentes Upgrade de GeneXus 15 hasta llegar a la versión 16.
Estudiaremos lo nuevo que podemos hacer con los mapas y algunas mejoras que han hecho en los almacenamientos externos de archivos multimedia.

Posteriormente, veremos algunos aspectos relacionados con la plataforma GeneXus.


GeneXus™
The power of doing