Chatbots

Setiembre 2019

OBJETIVO

Bienvenidos al taller de chatbots!

Aquí veremos la potencia del generador de chatbots introducido en la versión GeneXus 16.

Crearemos un chatbot que atiende consultas en un Encuentro de tecnología y negocios.

Los chatbots generados con GeneXus ejecutan tanto en Web como Mobile.

ALGÚN AJUSTE PREVIO

Requerimientos previos del taller:

Versión de GeneXus:

Utilizaremos la versión GeneXus Beta.

Proveedor NLP:

En este taller usaremos Watson como Natural Language Processing (NLP) Provider. Obviamos la configuración de la cuenta en el NLP provider, dado que está listo de antemano para la realización del taller.

Base de conocimientos:

La Kb se encuentra en

https://samples.genexusserver.com/v16/versions.aspx?CourseGX29Chatbot,

Se debe hacer checkout de la versión **InitialState.**

Ejecución de la aplicación del taller:

Puedes ejecutar tanto Web como Mobile.

Web:

El objeto Web que es punto de entrada para nuestro chatbot, es el web panel *RUDICourseGX29ChatbotWebUI*. Basta con hacer Run de ese objeto en cada ciclo de pruebas.

En el caso de no usar SD, deja apagada la generación Generate Android= FALSE, Generate IOS=FALSE.

Mobile:

El objeto SD punto de entrada es *RUDICourseGX29ChatbotSDUI*.

En el caso de ejecutar Mobile, recomendamos usar tu propio dispositivo, si es Android.

En el ***Anexo: cómo ejecutar en mi dispositivo*** (abajo en este documento), se indican algunas consideraciones para la configuración.

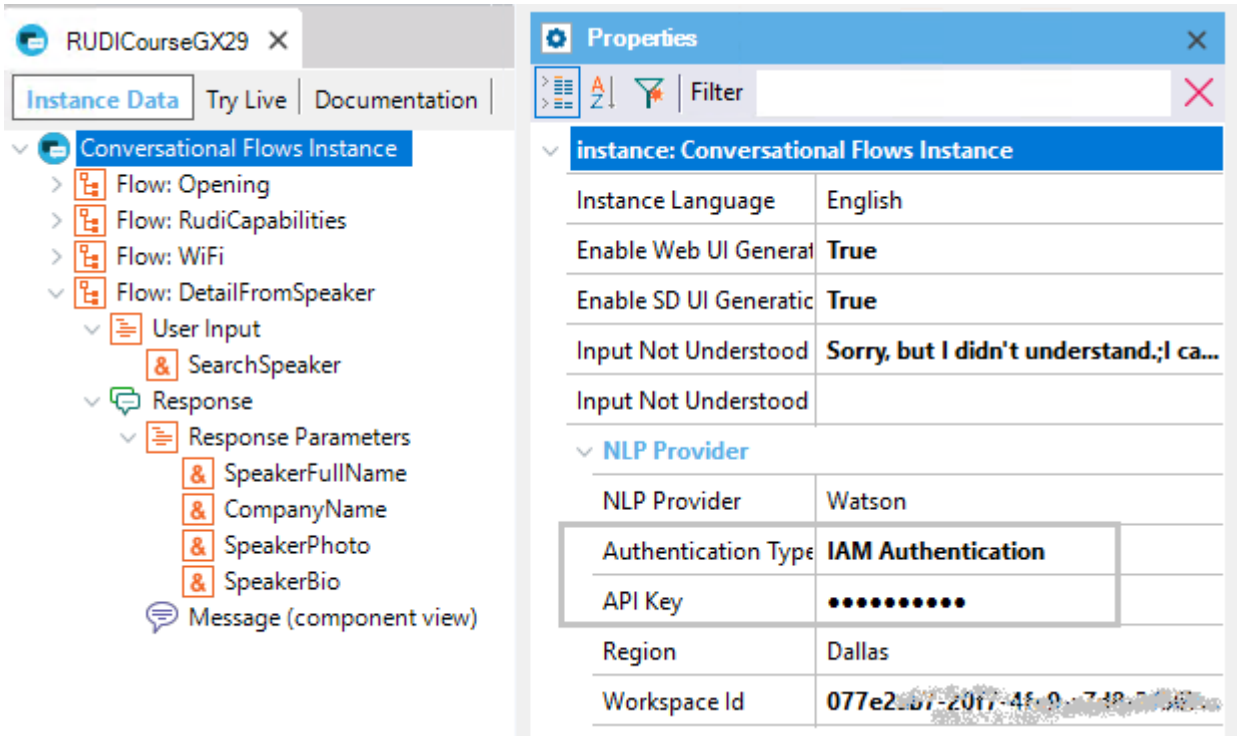
CONOCIENDO LOS CHATBOTS EN GENEXUS

Para comenzar, ejecutemos los siguientes pasos para dialogar con el chatbot.

El chatbot que tienes en la KB se llama **RUDICourseGX29** y aún no ha sido sincronizado con el Provider.

En el objeto **RUDICourseGX29**, en el nodo padre “Conversational Flows Instance”, configuremos en la propiedad **API Key** el valor dado en el taller. La propiedad Authentication Type debe ser igual a “IAM Authentication”, y la Region “Dallas”.

El workspace Id se asigna de manera automática.



Luego de salvar los cambios (lo mismo ocurre haciendo botón derecho sobre el objeto RUDICourseGX29 - seleccionando la opción Synchronize Chatbot), se genera un archivo de metadata que se impacta en el NLP provider, para generar el diálogo con sus intenciones y entidades[(1)].

Dada una intención del usuario se pueden distinguir diferentes objetos o tópicos, dentro de esa intención, que llamamos formalmente Entidades.

Las entidades se almacenan en el NLP provider, con sus posibles valores y sinónimos.

Para crear un chatbot, se necesita modelar su comportamiento (definir intenciones, entidades) y la respuesta que se muestra al usuario para cada una de esos intenciones.

NOTA: EL GENERADOR DE CHATBOTS FUNCIONA COMO UN PATTERN. A PARTIR DEL MODELO, EL PATTERN GENERA AUTOMÁTICAMENTE TODOS LOS OBJETOS NECESARIOS PARA RESOLVER LA CONVERSACIÓN ENTRE EL USUARIO Y EL NLP PROVIDER.

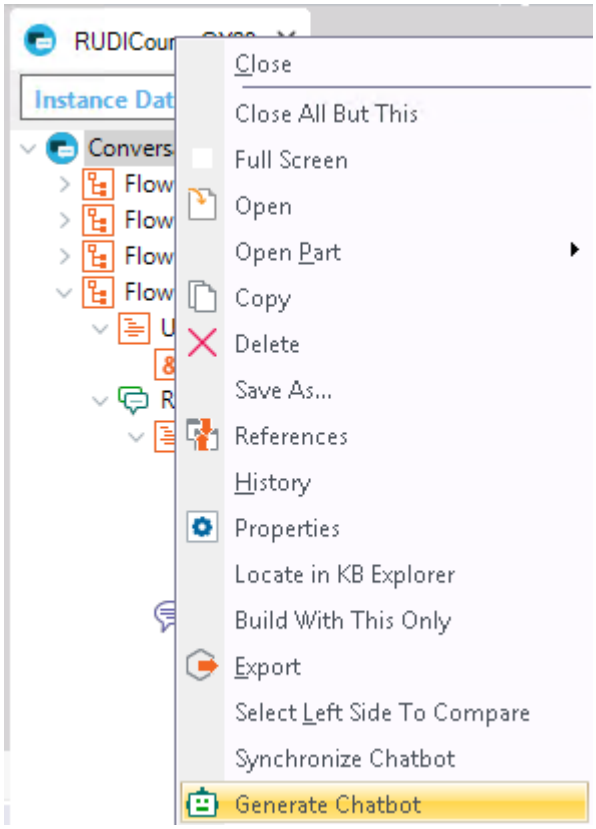
Ya entendimos los principales conceptos, y qué es el Chatbot Generator, ahora ¡empecemos a construir!

EL CHATBOT EN ACCIÓN

A través de nuestro chatbot, los usuarios podrán:

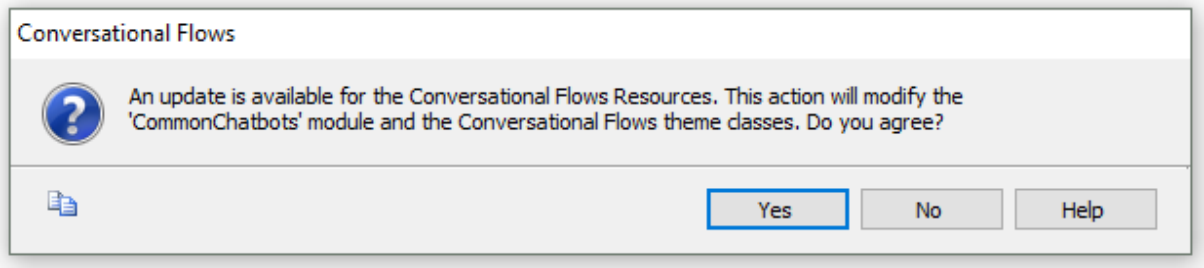
- Acceder a respuestas de preguntas frecuentes (ej: clave de WIFI)
- Saber dónde se encuentran algunos servicios
- Obtener datos de los oradores

Hagamos botón derecho sobre el objeto **RUDICourseGX29** (Objeto conversational Flows), y luego Generate ChatBot.



Nota: La opción “Generate Chatbot” hace que se calculen los objetos que se van a generar a partir de nuestra instancia (RUDICourseGX29) y se importen en la KB.

Puede aparecer el siguiente mensaje, que indica que se actualizarán los recursos del generador de chatbots (si no están actualizados).



Hagamos un **Rebuild all**.

Se va a solicitar el create de la BD. Observar que hay dos tablas, llamadas GXChatMessage y GXChatUser, que son usadas por el chatbot.

INICIALIZACIÓN

Las entidades han sido creadas en el NLP provider, pero sus valores deben ser inicializados.

Vamos a trabajar con las siguientes entidades:

- Floors
- Rooms
- Speakers

Los valores serán cargados en el NLP provider a través de una API provista por el módulo chatbots.

Abre el objeto **BatchEntities** para conocer el método usado para subir valores y entidades a Watson (“Chatbot.SendEntityValues”).

Ejecuta entonces, el proc BatchEntities (botón derecho - Run desde la solapa). Este proceso entrena el sistema, por lo cual si bien el request al NLP provider retorna de inmediato un HTTP Status 200, puede tardar unos minutos el proceso de entrenamiento.

Verás lo siguiente en el output de GeneXus:

Floors: [{"Id":"","Type":2,"Description":"200 "}]
Rooms: [{"Id":"","Type":2,"Description":"200 "}]
Speakers: [{"Id":"","Type":2,"Description":"200 "}]

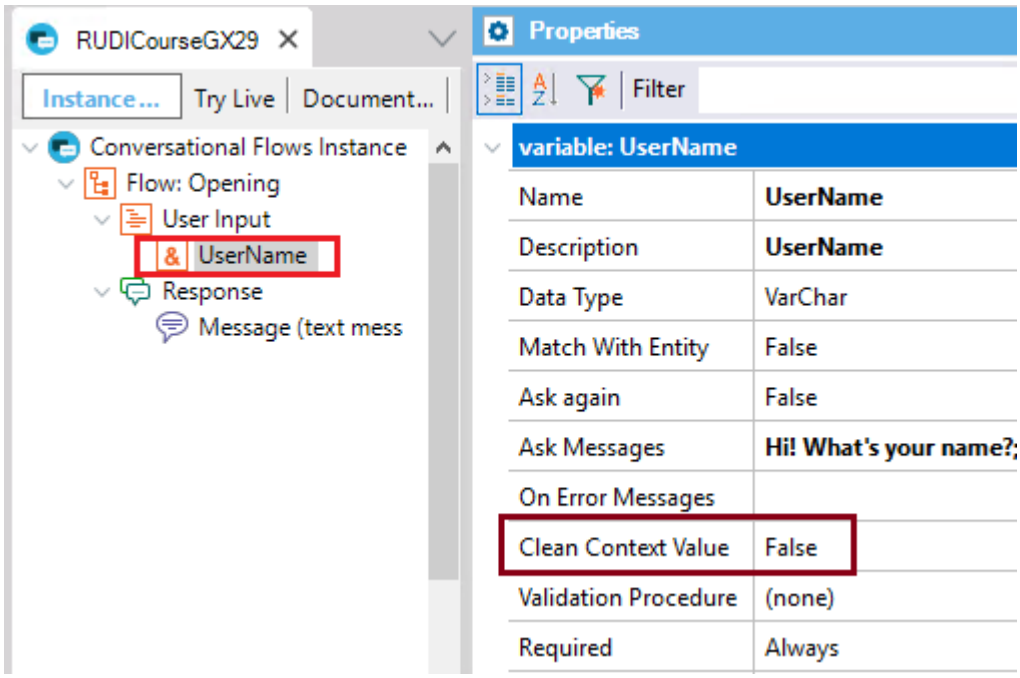
EMPEZANDO A CONSTRUIR EL CHATBOT

Veamos algunos flujos que son de particular interés:

1. SALUDO

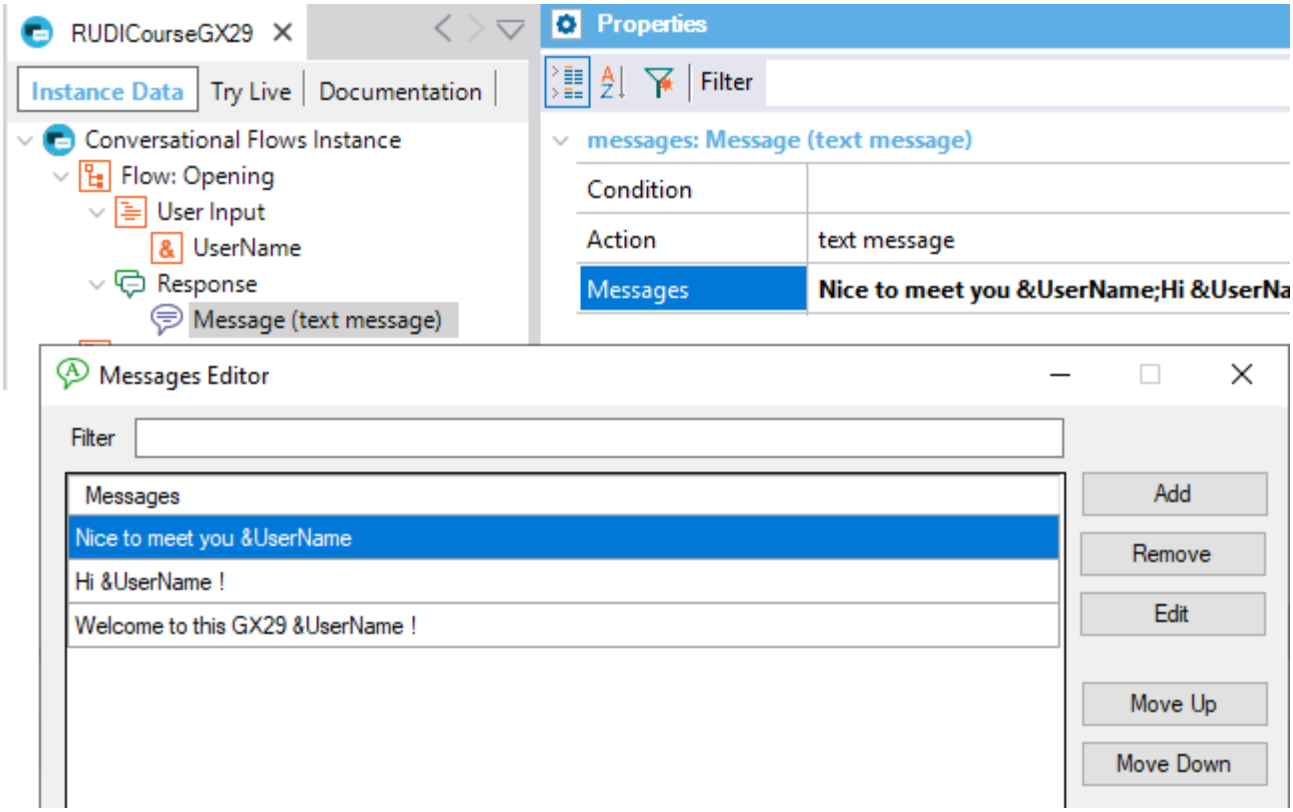
Antes que nada, veamos el flujo más sencillo, que es el del saludo (flujo "Opening").
Es un flujo que podría no tener ningún User Input (pero en nuestro caso queremos pedir aquí el nombre del usuario, para que nuestro chatbot sea más amigable), y devolver el saludo.

Observa el **User Input** "UserName" que tiene la propiedad **Clean Context Value**= FALSE. Eso significa que el nombre del usuario será guardado en el contexto[(2)] (herramienta fundamental en un chatbot para que el usuario no tenga que repetir información que ya ha proporcionado).



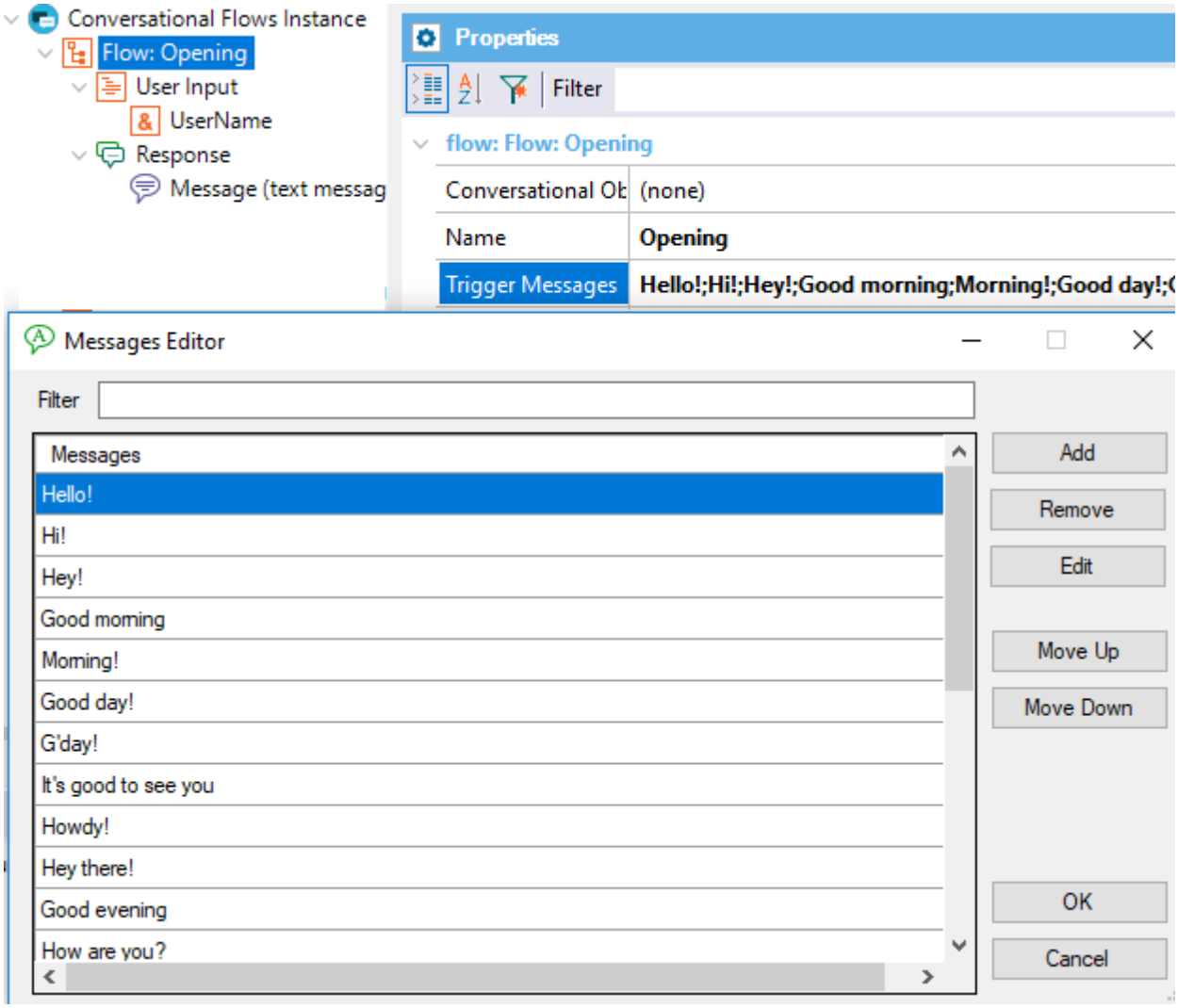
Observa también la respuesta. Es un nodo Message de tipo texto, en donde se le da

la respuesta al usuario, instanciando su nombre.



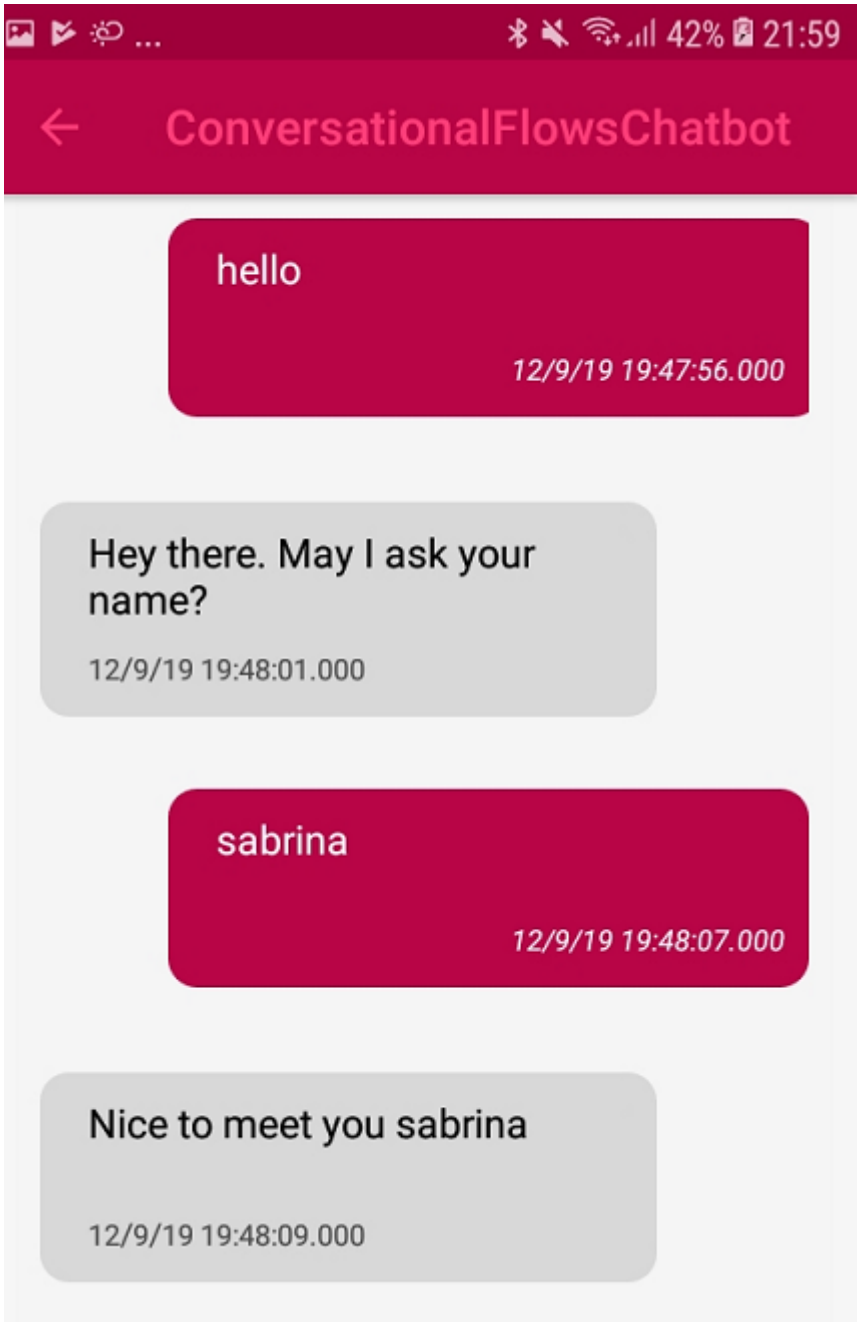
Cómo identifica el NLP Provider la intención del usuario, que en este caso es
un saludo?

Lo hace en base al entrenamiento que le dimos al flujo,
usando mensajes de ejemplo (llamados "**Trigger Messages**" en GeneXus).
Puedes verlos haciendo click en el nodo padre del flujo, en la propiedad Trigger Messages.



Para ver los cambios, hagamos F5.

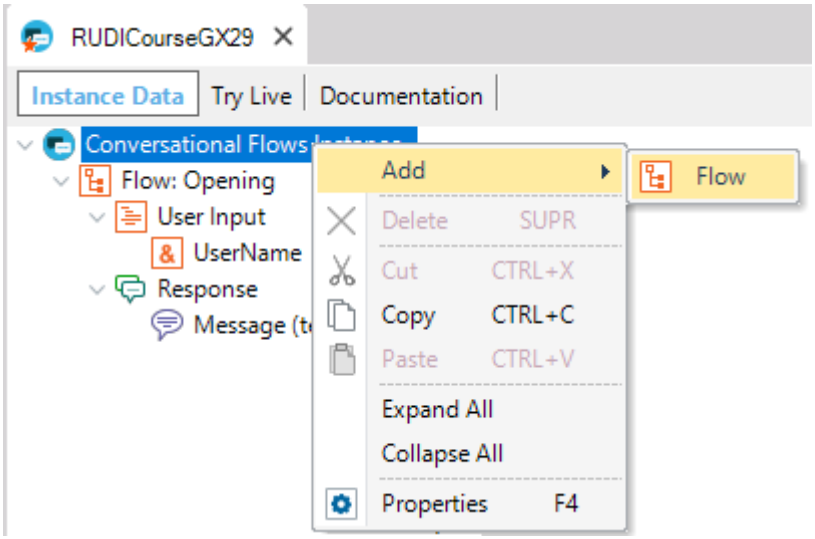
En ejecución verás algo como ésto:



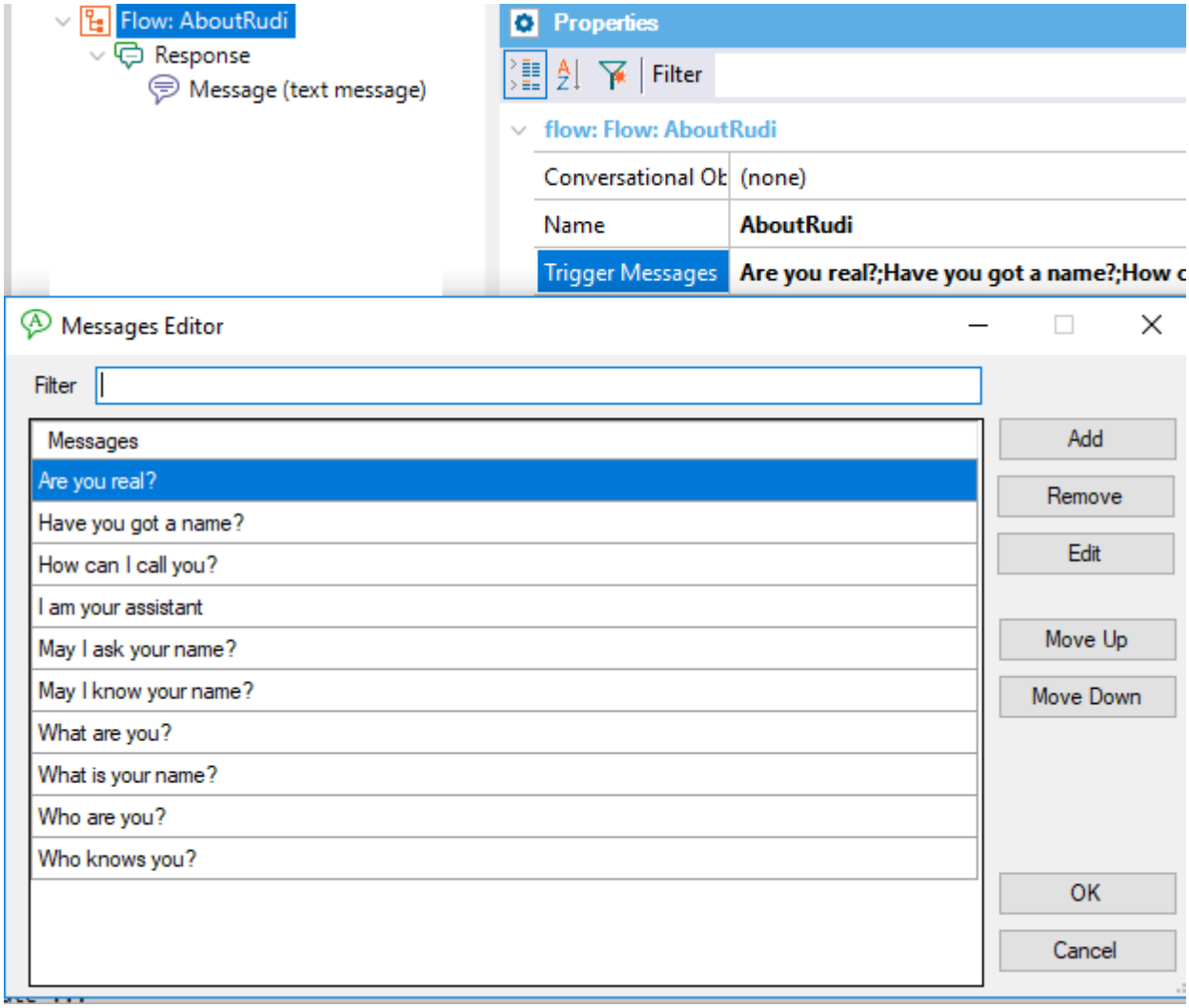
Ahora, construyamos un flujo similar, en donde el usuario pueda averiguar quién es RUDI.

Los pasos son éstos:

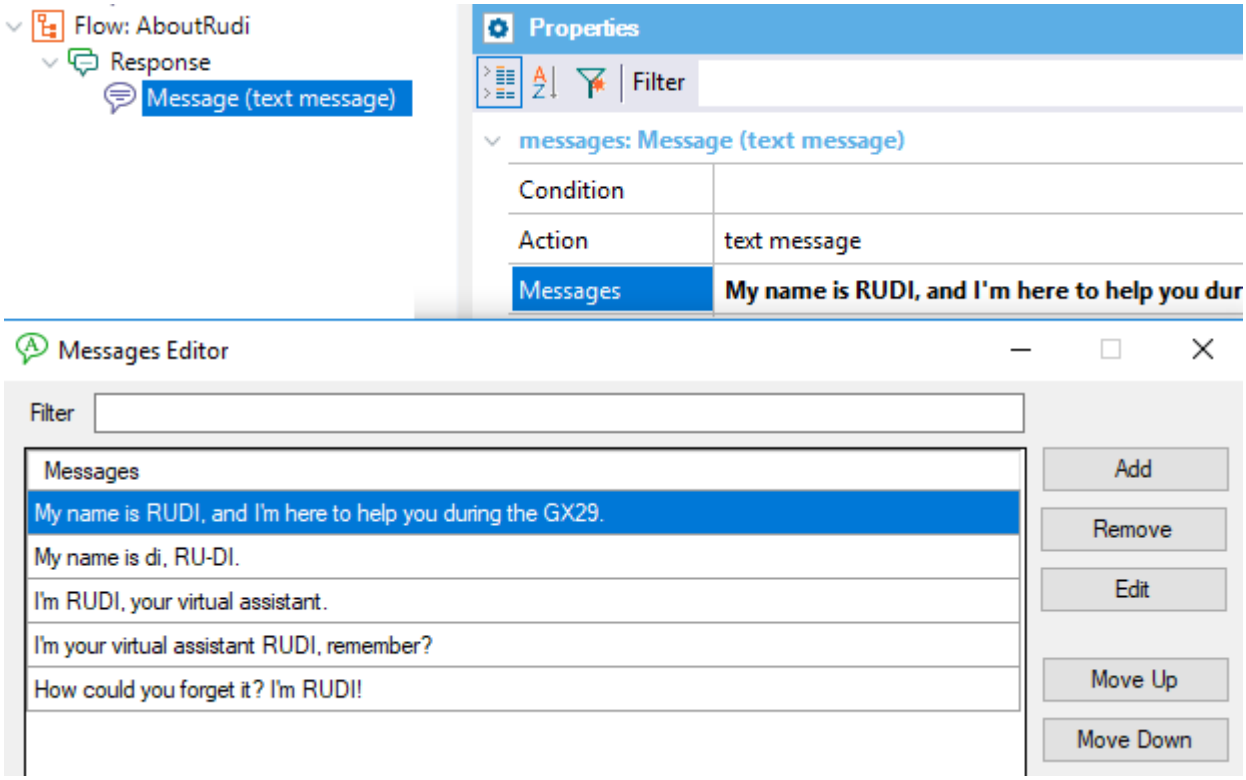
- Crear un nuevo Flow, llamado “AboutRudi”. Debes hacer botón derecho sobre el nodo **Conversational Flows Instance**, y luego Add - Flow.



- Agregar Trigger Messages (los que consideres necesarios – en un caso real, cuanto más completo y de calidad es el entrenamiento, mejores resultados dará la identificación de la intención del usuario).

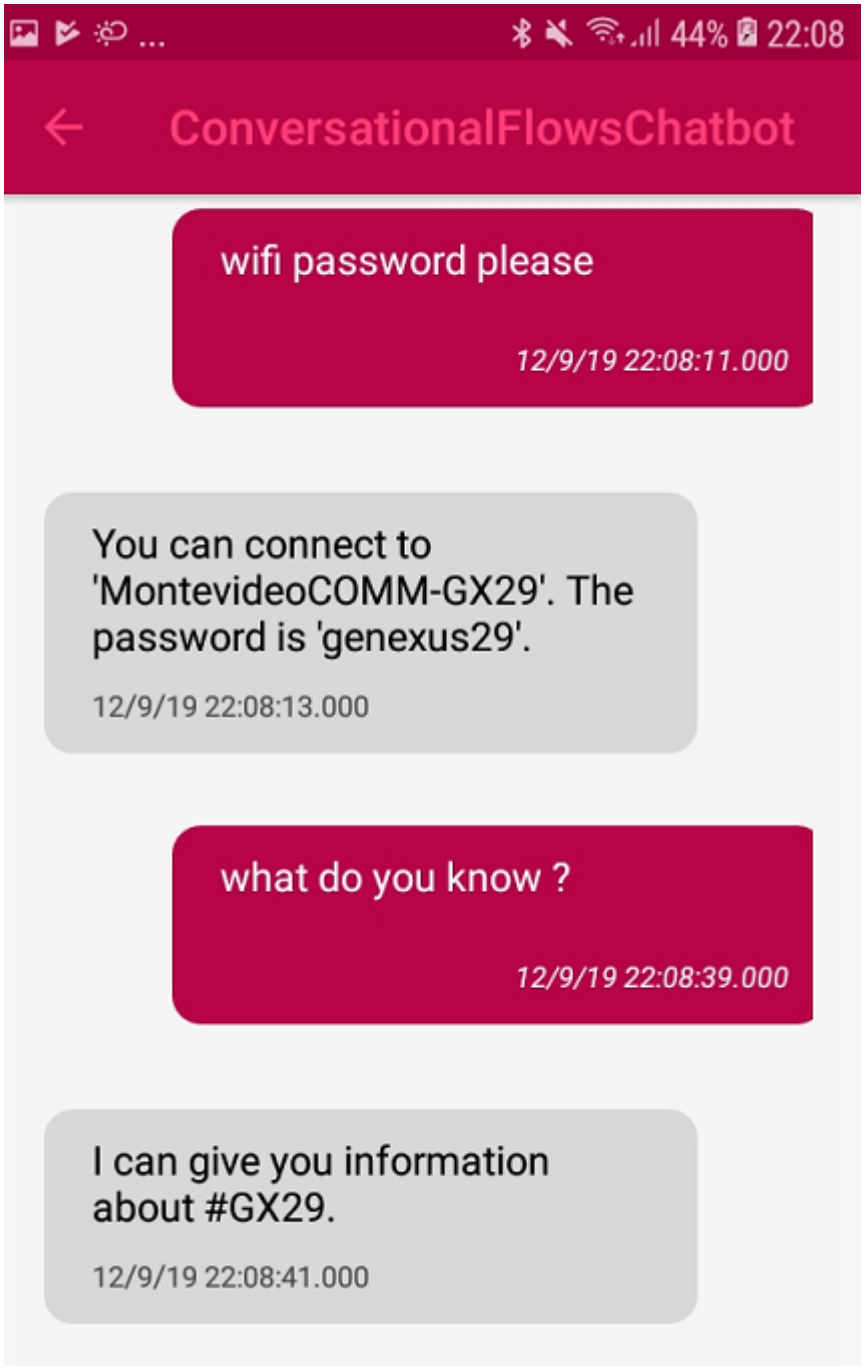


- Para agregar respuestas, debes hacer botón derecho sobre el nodo Response, y agregar un Message. Ahí puedes agregar todas las respuestas que quieras. Será aleatoria la respuesta dado al usuario, y justamente es lo que permite dar más “naturalidad” en la UX.



Hay varios flujos del mismo estilo que ya vienen creados en nuestro chatbot, por ejemplo, saber la clave de WIFI (flujo “wifi”), saber de qué habla RUDI (flujo “RudiCapabilities”), etc.

Probemos entonces cómo responde nuestro chatbot!



2. ACCEDER A SERVICIOS

Haremos un flujo en donde el usuario podrá preguntar por un servicio, por ejemplo, el baño. Para darle la respuesta, se le preguntará el piso en donde se encuentra.

- Debes crear un flujo de nombre “ServiceLocationBathRoomFloor”.

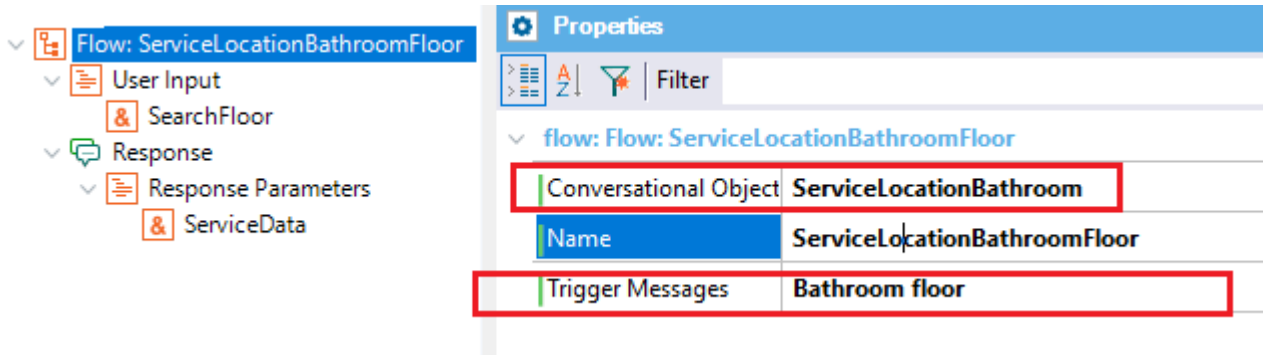
- En la propiedad Trigger Messages debes configurar un ejemplo (o todos los que quieras) de disparo de esta intención; por ej: “bathroom floor”.

  ¿Cómo se resuelve la respuesta?

  Cuando el usuario termina de ingresar los datos solicitados (en este caso el piso), se ejecuta el objeto dado en la propiedad **“Conversational Object”**.

- En la propiedad “Conversational Object” configuremos el procedimiento “ServiceLocationBathRoom” (es un proc que se encuentra en la KB); que dado el piso en donde está el usuario, retorna el baño más cercano.

  Observa que al momento de configurar esta propiedad, automáticamente se asigna un **User Input** (“SearchFloor”) y un **Response Parameter** (“ServiceData”). Estos coinciden con el parámetro de entrada y de salida respectivamente del proc “ServiceLocationBathRoom”.



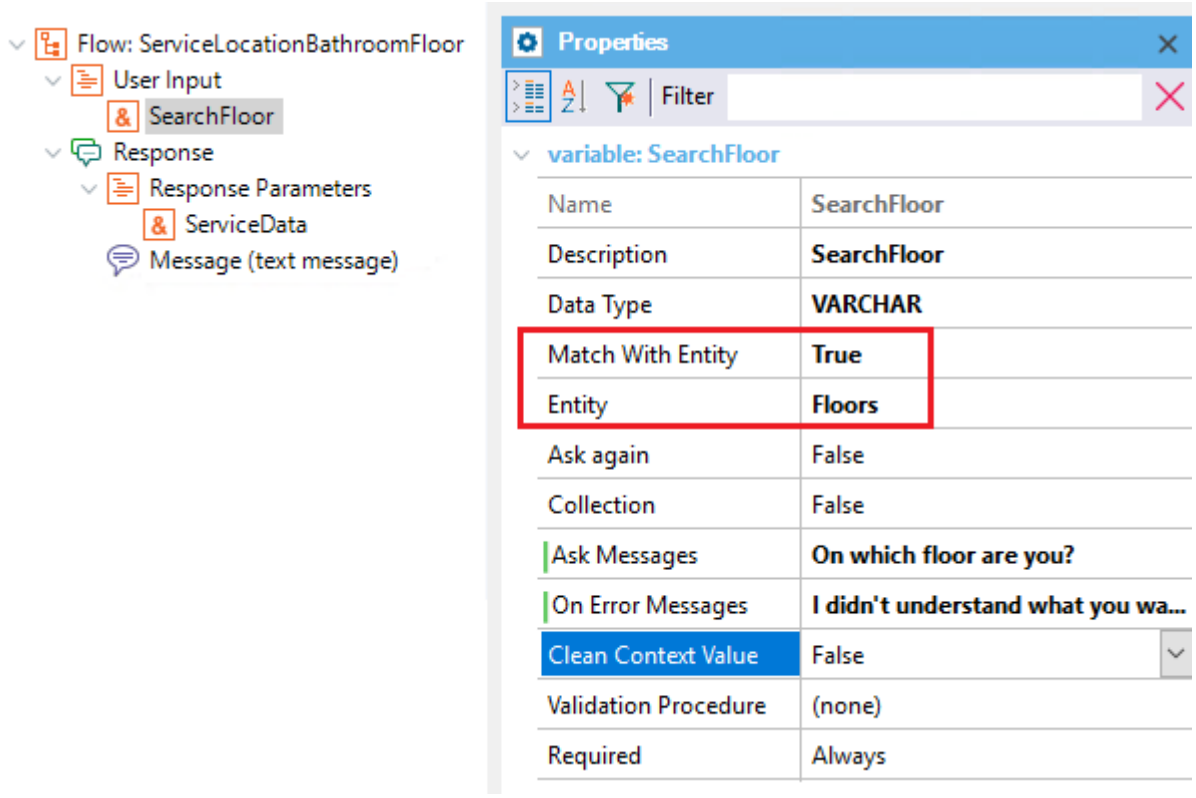
El User Input “SearchFloor” es donde el usuario indica el piso donde se encuentra. Dicho User Input debe tener **“Match with Entity” = TRUE**, y la entidad es **Floors**.

¿Qué significa Match with Entity?

Que se controlará que el usuario no ingrese un dato que no se encuentre dentro de los valores (y sus sinónimos) de esta entidad definida en el NLP Provider. Es como un “chequeo de integridad”, contra el Provider.

Puedes abrir el objeto “EntityFloors” en donde cargamos los valores de la entidad “Floors” en el Provider (pisos y posibles sinónimos) para clarificar cuáles son los datos en el NLP Provider.

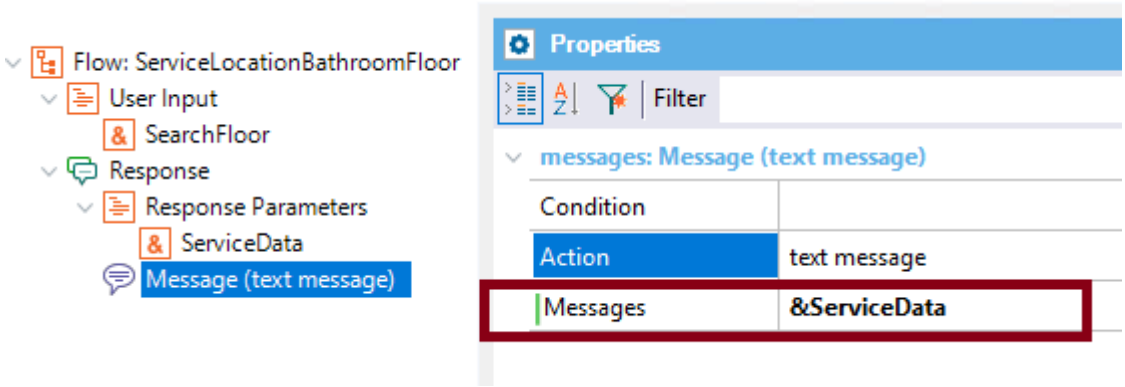
Es decir, que si el usuario ingresa un piso que no está dentro de los valores de la entidad, el Provider indica el error, y se le da el feedback al usuario con el mensaje indicado en la propiedad **On Error Messages** (Ej: “I didn't understand what you wanted to say with '&GXUserInput', but there are bathrooms on every floor.”)



- Lo que resta únicamente es agregar un Message para que responda al usuario cuando éste indica un piso válido. Lo que se responderá es la variable de salida de nuestro Conversational Object (&ServiceData).

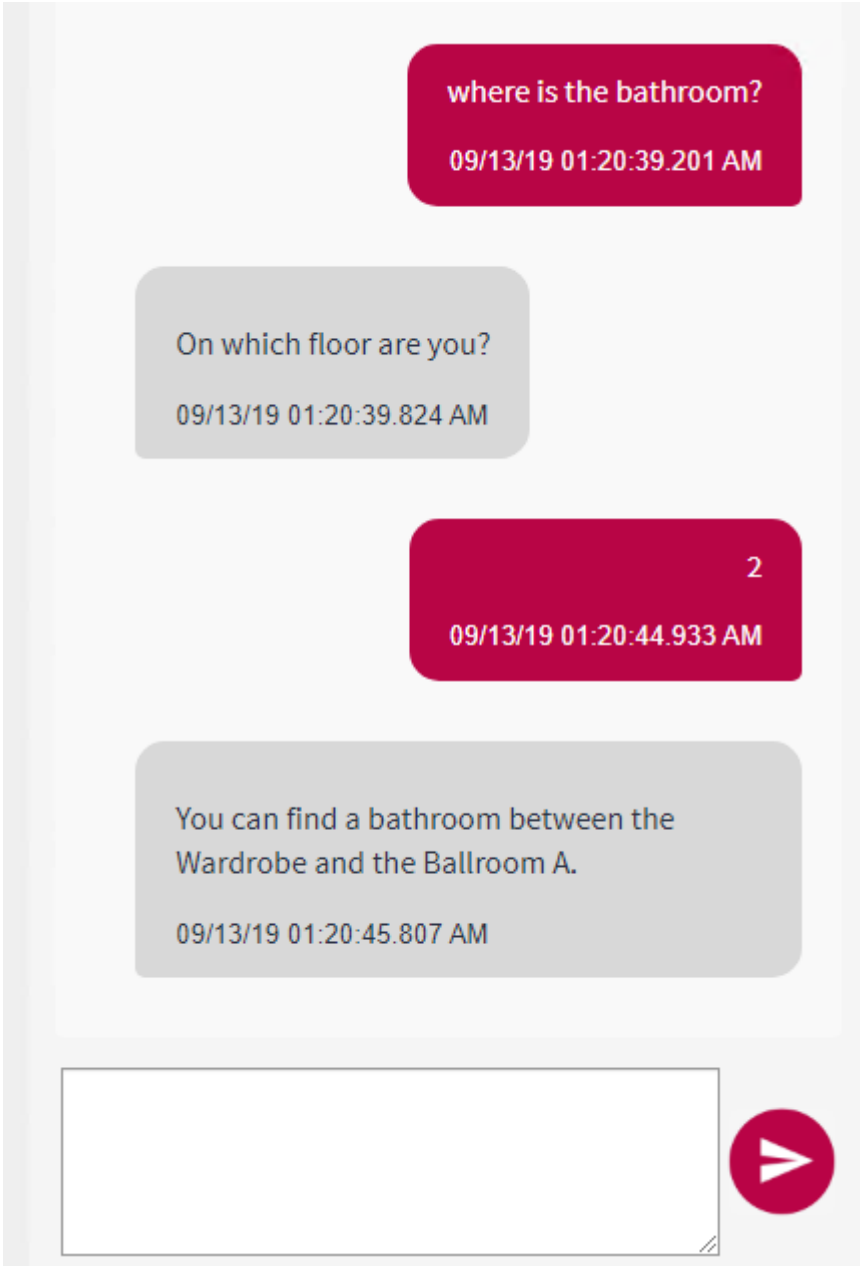
  Debes hacer botón derecho en el nodo “**Response**”, click en “**Add message**”, y agregar &ServiceData en la propiedad **Messages** del nodo

Message.



Hagamos Generate Chatbot (botón derecho sobre la solapa del Conversation Flows), y luego F5.

Listo, ya puedes probarlo en ejecución:



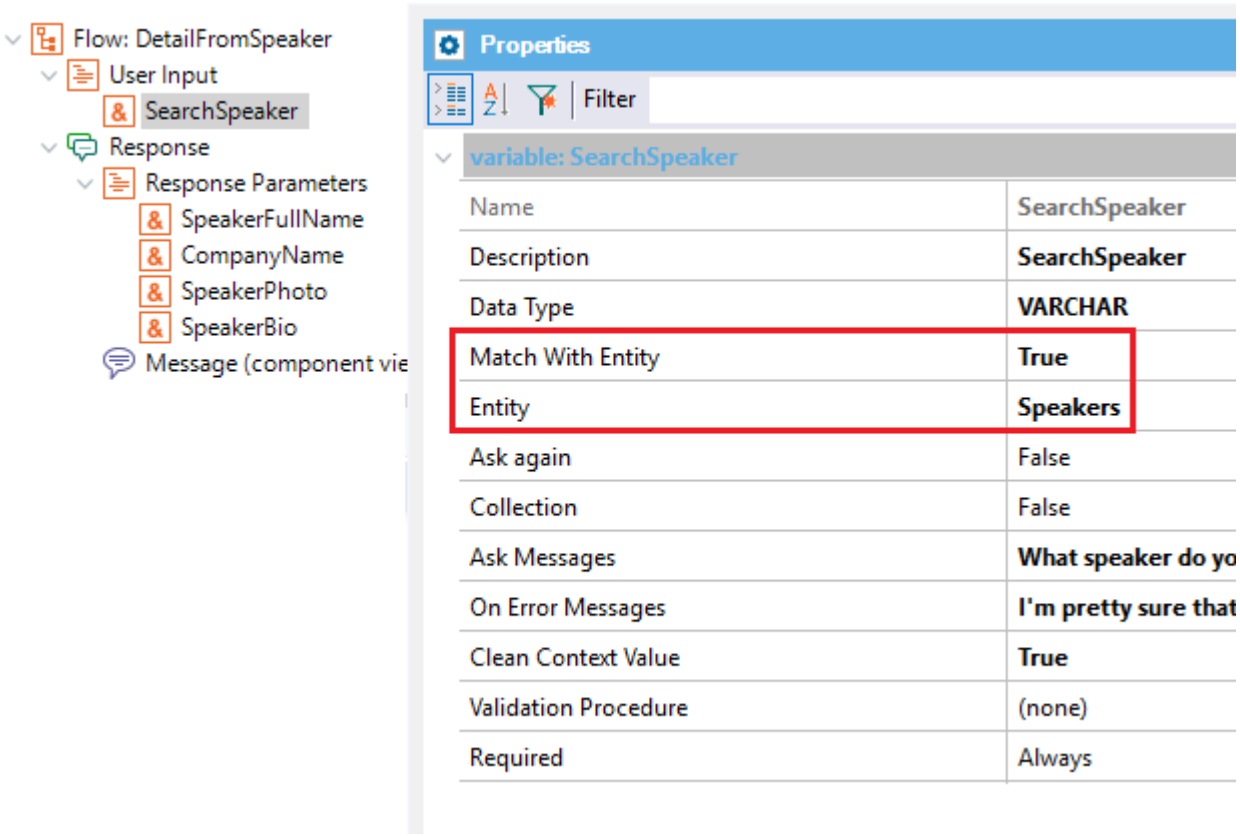
3. VER DETALLE DE UN ORADOR.

Este flujo (“DetailFromSpeaker”) tiene como objetivo mostrar en forma gráfica (no texto) los datos del orador.

Por lo tanto, tiene un **User Input** que es el nombre del orador. Si el orador se menciona en el mensaje inicial del usuario, éste no se solicita.

Observa que el orador tiene la propiedad **Match with Entity** = TRUE, y

Entity=**Speakers**.



En este ejemplo, la información del orador, se mostrará en un panel (Web o

Mobile) según corresponda.

A través de un Data Provider se obtiene la información del orador; el Data Provider

es invocado desde los paneles que serán usados para mostrar el resultado de la

consulta del usuario.

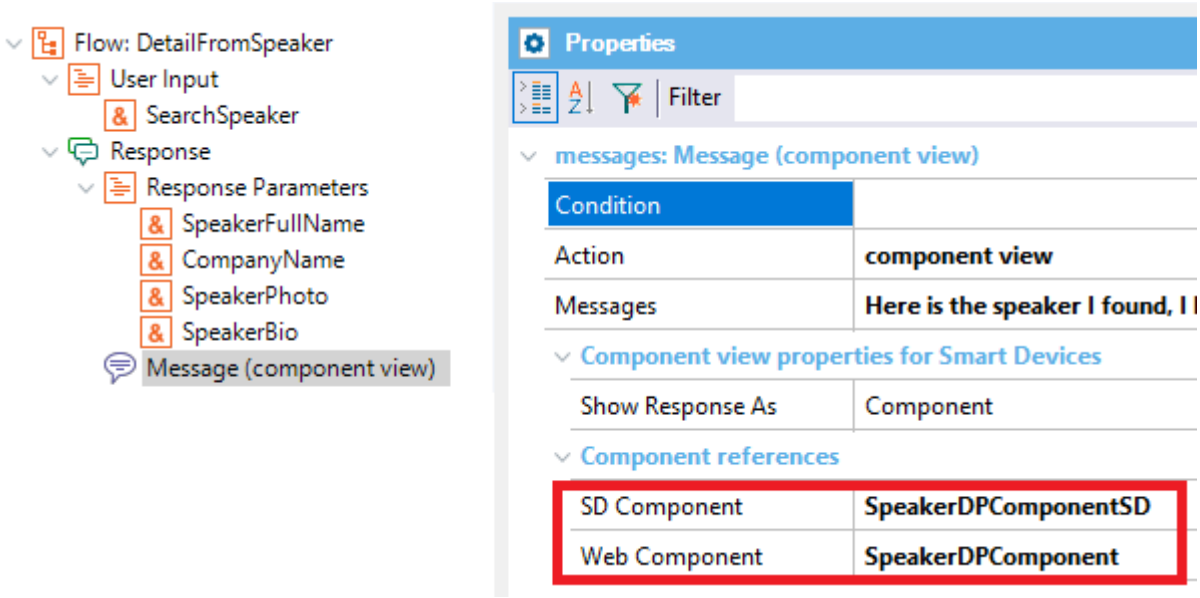
El objeto web que muestra el resultado es **SpeakerDPComponent**, y en caso de SD

es **SpeakerDPComponentSD**. Puedes abrirlos y ver la llamada al Data Provider

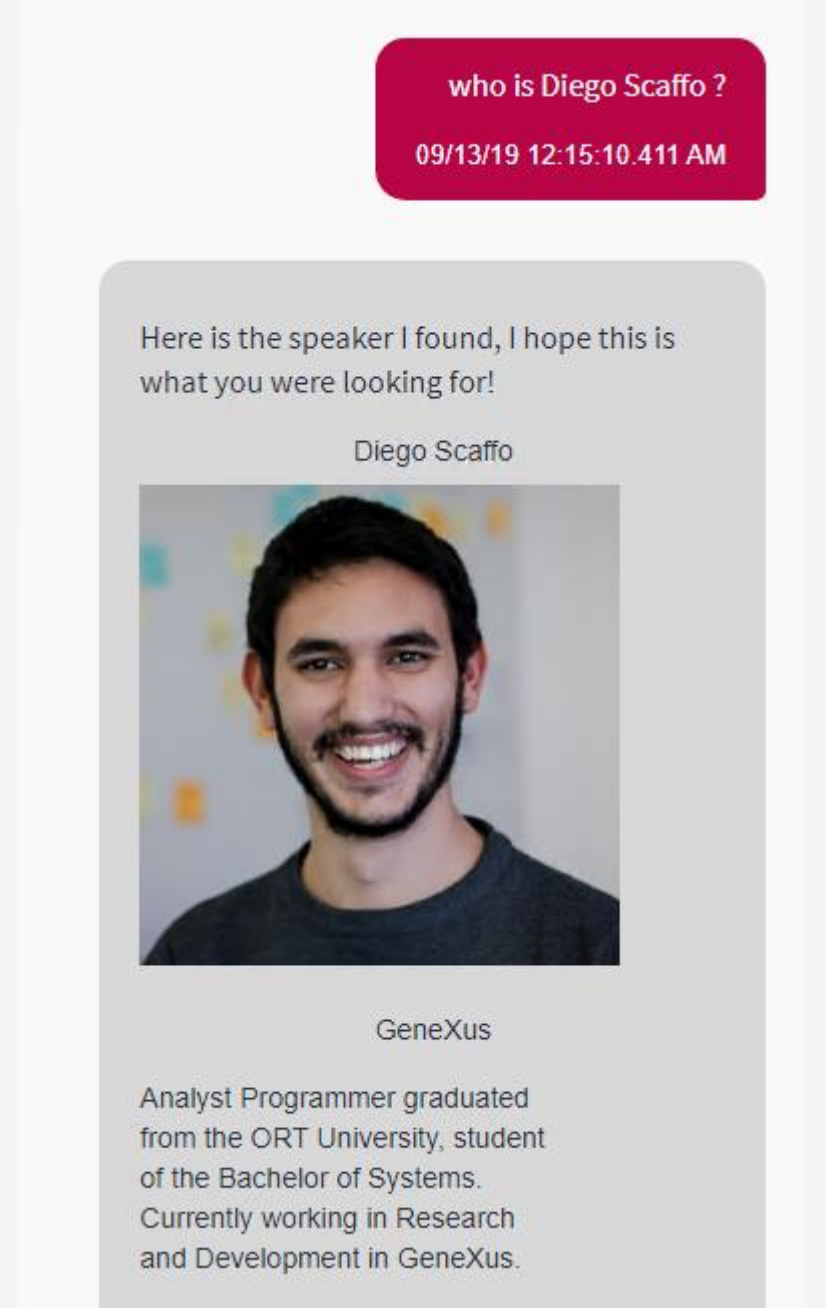
que dado un nombre de orador, devuelve sus datos.

Veamos la configuración en el flujo:

Los paneles mencionados anteriormente, donde se mostrarán los datos del orador, están configurados en el nodo Message, que en este caso tiene la propiedad **Action** = Component view. Los paneles SD y Web se configuran en las propiedades **SD Component** y **Web Component** respectivamente.



Probémoslo en ejecución en Web!


who is Diego Scaffo ?
09/13/19 12:15:10.411 AM
Here is the speaker I found, I hope this is what you were looking for!
Diego Scaffo
GeneXus
Analyst Programmer graduated from the ORT University, student of the Bachelor of Systems. Currently working in Research and Development in GeneXus.

Y en Mobile:

Hemos visto algunos conceptos de cómo construir un chatbot, pero hay mucho más! Te invitamos a revisar la documentación (abajo en este documento) y te agradecemos por haber participado!

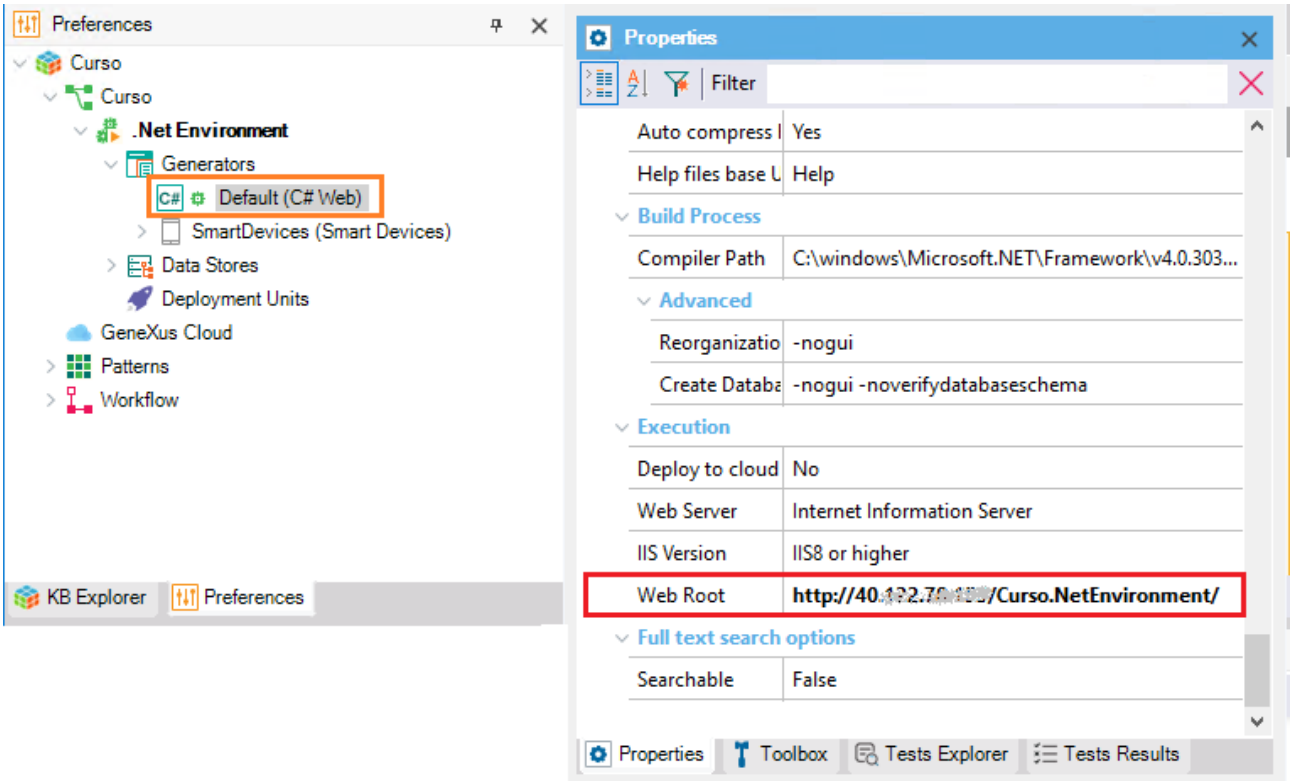
ANEXO: CÓMO EJECUTAR EN MI DISPOSITIVO

En este taller puedes usar un dispositivo Android.

Debes obtener la IP pública de la maquina (click aquí) y configurarla en las propiedades:

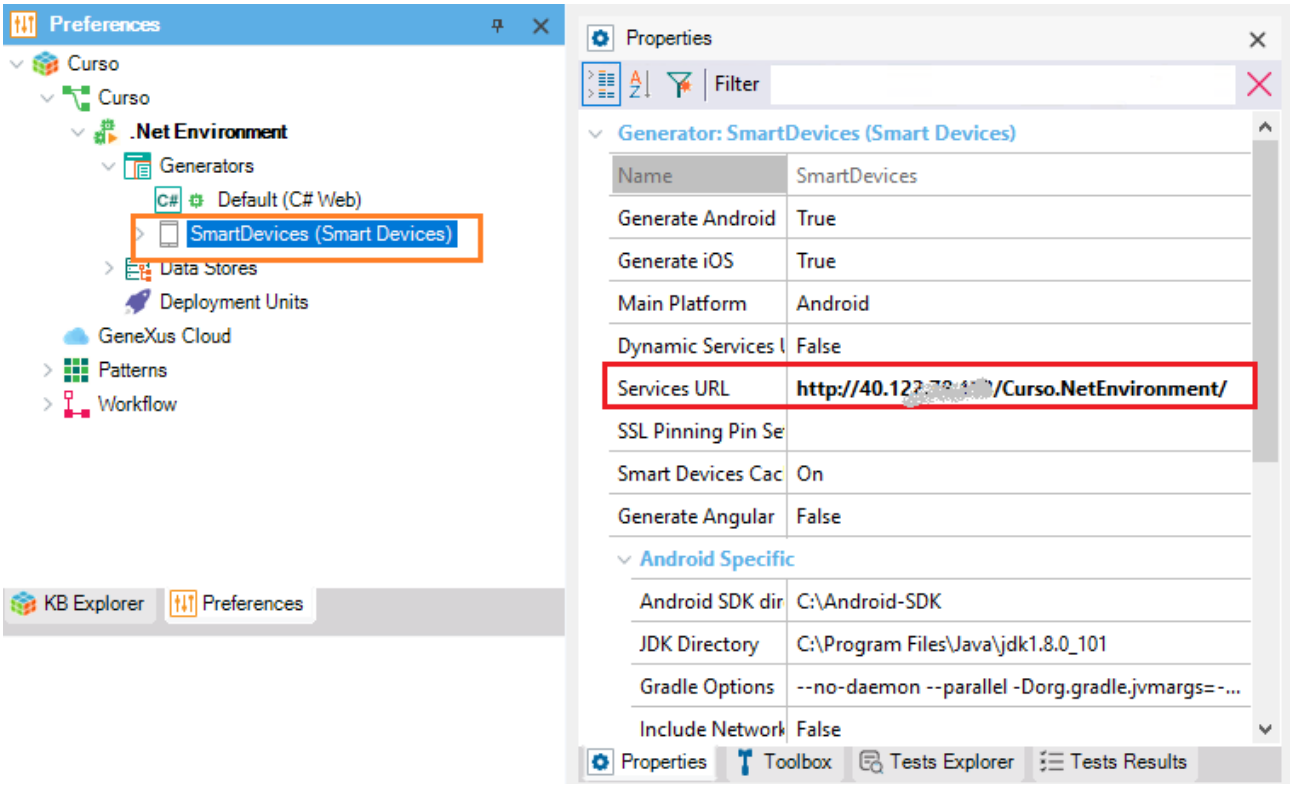
- Web Root (del generador C#)

  Cambia “localhost” por la IP pública de tu máquina.



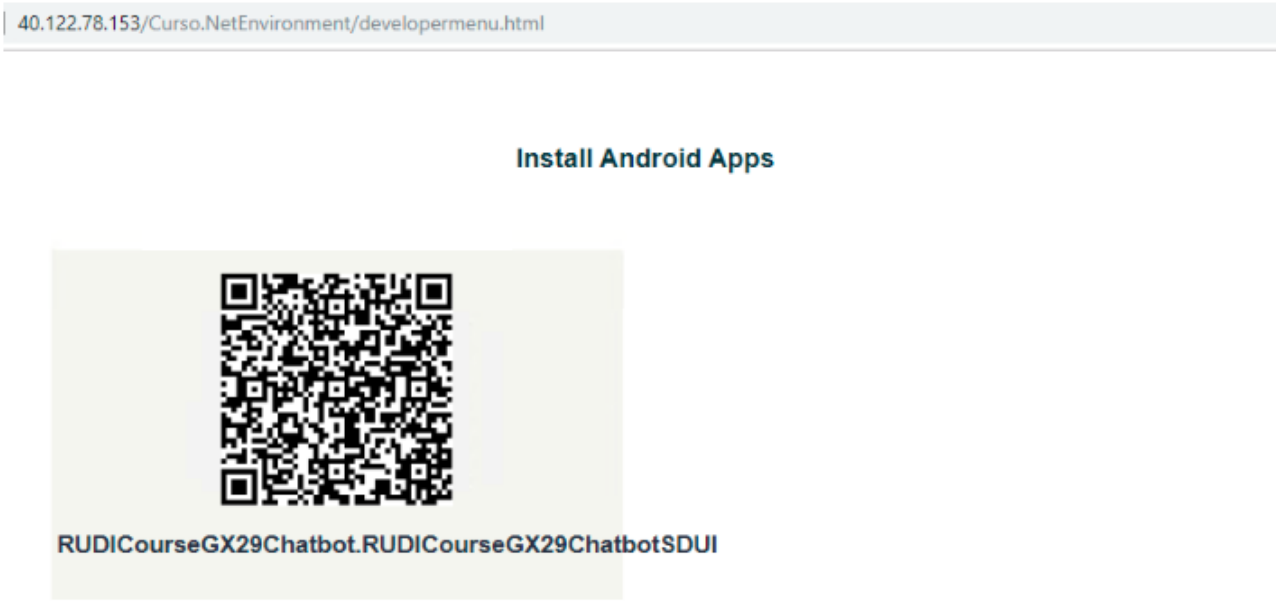
- Services URL (de Smart Devices)

  Cambia “localhost” por la IP pública de tu máquina.



El objeto **RUDICourseGX29ChatbotSDUI** (es el panel principal SD) y está como Startup Object, por lo cual podrás scannear el QR Code del mismo para instalar la app en tu dispositivo Android.

Los QR Codes están disponibles a través de la opción del menú de GeneXus **View -> Show QR Codes**. Verás una pantalla como la siguiente:



DOCUMENTACIÓN

Chatbots in GeneXus
Chatbot Generator
Chatbots Architecture
How to build a Chatbot using GeneXus
Chatbots Channels API
Chatbot Generator Try Live

REFERENCIAS

(1) Por ejemplo, dada la **intención** de “Obtener los datos de un orador”, una **entidad** de esa intención sería los “Oradores”, y el **valor de dicha entidad** por ejemplo, puede ser “Gaston Milano”.

Entonces, le llamamos entidad a una agrupación de varios valores. Alguno o varios de ellos serán instanciados en la consulta.

(2) El contexto es fundamental en un chatbot. Es el estado de la conversación, y se mantiene para no tener que pedir información repetida al usuario. En un chatbot en GeneXus, todas las variables que son User Inputs son parte del contexto. Mediante la propiedad **Clean Context value**, se puede limpiar el valor de la variable de contexto.
El contexto se puede manejar a través de la API de contexto.

GeneXus™
www.genexus.com

MONTEVIDEO - URUGUAY	Av. Italia 6201- Edif. Los Pinos, P1	(598) 2601 2082
CIUDAD DE MÉXICO - MÉXICO	Hegel N° 221, Piso 2, Polanco V Secc.	(52) 55 5255 4733
MIAMI - USA	7300 N Kendall Drive, Suite 470	(1) 201 603 2022
SÃO PAULO - BRASIL	Rua Samuel Morse 120 Conj. 141	(55) 11 4858 0300
TOKYO - JAPAN	2-27-3, Nishi-Gotanda	(81) 3 6303 9381
	Shinagawa-ku, Tokyo, 141-0031	(81) 3 6303 9980