


# Aplicación GeneXus que interactúa con un RAG assistant



Alejandra Caggiano

Hemos visto anteriormente cómo interactuar desde una base de conocimiento GeneXus con un Chat assistant.

## Aplicación GeneXus que interactúa con un RAG assistant



The screenshot displays the 'GeneXus Enterprise AI Course' interface. On the left, a dark sidebar contains the 'UNANIMO' logo and a menu with 'Products' and 'Chat with GeneXus Training'. The main content area is titled 'Chat with GeneXus Training' and features a search input field containing the question 'What is a transaction object?' and a blue 'ASK THE ASSISTANT' button. Below the input, the assistant's response is displayed in a white box with a light blue border. The response is structured with a bold heading 'Assistant response' followed by three paragraphs of text explaining transaction objects in GeneXus.

**GeneXus** **GeneXus Enterprise AI Course**

**UNANIMO**

Products

Chat with GeneXus Training

**Chat with GeneXus Training**

Question  ASK THE ASSISTANT

**Assistant response**

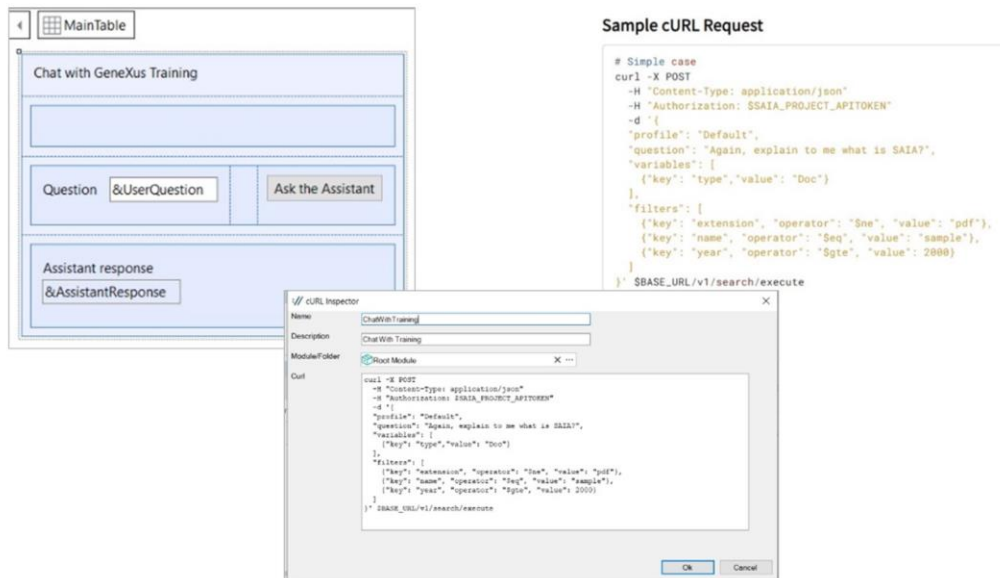
A transaction object in the context provided refers to a specific type of object created in GeneXus to represent real-world entities or actions that need to be recorded or processed within a system. In this context, it is mentioned that for every object from reality identified, a transaction object will be created. These transaction objects are used to model and manage data related to customers, attractions, countries, and other entities mentioned.

In the design of the first transaction, it is important to define the attributes and properties that will be associated with the transaction object. These attributes will capture relevant information about the entity or action being represented, such as customer details, attraction information, country data, etc. By creating transaction objects, developers can effectively map real-world concepts into the system and perform operations on them, such as data retrieval, manipulation, and storage.

Overall, transaction objects play a crucial role in structuring and organizing data within a system, allowing for the representation and management of various entities and actions. They serve as a bridge between the real world and the digital system, enabling the system to interact with and process information related to different aspects of reality.

Vamos a ampliar ahora esa misma base de conocimiento para que también pueda interactuar con nuestro RAG Assistant de nombre ChatWithGXTraining. De esta forma habilitaremos al usuario final a realizar preguntas sobre dicho contenido.

## Aplicación GeneXus que interactúa con un RAG assistant



Si vamos a GeneXus, ya hemos creado el web panel de nombre ChatGXTraining., donde el usuario final puede ingresar una consulta y preguntar al asistente para obtener la respuesta. Tenemos dos variables:

La variable &UserQuestion, sobre la cual el usuario planteará su consulta, y la variable &AssistantResponse que será la encargada de mostrar la respuesta recibida.

Nuestro objetivo entonces es codificar el evento asociado al botón para que al presionarlo se llame a un procedimiento que envíe la consulta y reciba la respuesta.

Para crear ese procedimiento ya con la base de la estructura del request, vamos a cargar el cURL sample a través del menú Tools / Application Integration / cURL Inspector.

Ponemos como nombre ChatWithTraining y pegamos el sample, disponible en la documentación técnica de GeneXus Enterprise AI.

[https://wiki.genexus.com/enterprise-ai/wiki/?8,Table+of+contents%3AEnterprise+AI,](https://wiki.genexus.com/enterprise-ai/wiki/?8,Table+of+contents%3AEnterprise+AI)

En este caso necesitamos interactuar con la api que nos permite chatear con documentos, así que el sample que pegamos corresponde al cURL Request.

## Aplicación GeneXus que interactúa con un RAG assistant



```
out Rules | Conditions | Variables | Help | Documentation |
Parm(in:&UserQuestion, out:&AssistantResponse);

1 //curl -X POST -H "Content-Type: application/json" -H "Authorization: $SAIA_PROJECT_APITOKEN" -d '{ "profile": "Default
2
3 &HttpClient.Secure = 1
4 &HttpClient.Host = "api.qa.saia.ai"
5
6 &HttpClient.AddHeader("Authorization", "Bearer default_OuK5BwzqSLjNEHwUf-GV0QCLi2YHYxBBGAshAg1CilSa9lFgeZKcQr5S0xsehbjbg
7 &HttpClient.AddHeader("Content-Type", "application/json")
8
9 &HttpClient.AddString('{"profile": "ChatWithGXTraining", "question": "' + &UserQuestion.Trim() + '" }')
10
11 &HttpClient.Execute("POST", "/v1/search/execute")
12
```

Presionamos Ok, y vemos creado el procedimiento con la estructura base que necesitamos.

Lo primero que indicamos es el protocolo de conexión. Y para eso, como yo sabemos, declaramos el método Secure con el valor 1 que corresponde al protocolo HTTPS.

Luego indicamos el Host que corresponde al contenido de la variable &BASE\_URL, y que depende a su vez de la instalación de GeneXus Enterprise AI.

Bien. Necesitamos luego indicar el Project api token, que como también ya sabemos, lo copiamos de la plataforma y lo pegamos.

Vamos ahora al cuerpo de la consulta. En primer lugar, indicamos el "profile", o sea, el nombre del asistente que en este caso es ChatWithGXTraining.

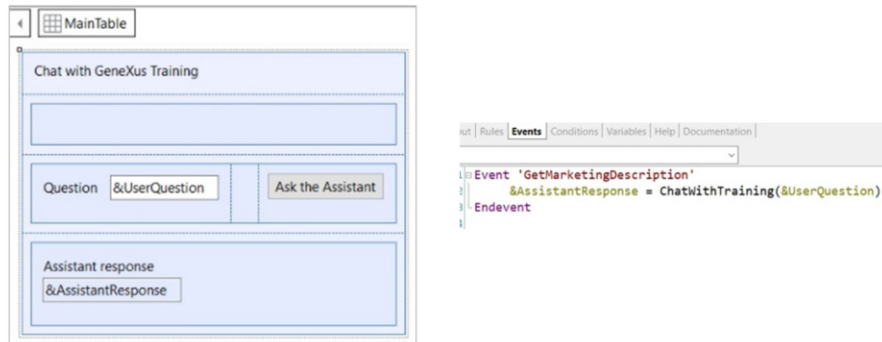
Al igual que analizamos en el ejemplo anterior, este procedimiento deberá recibir la consulta del usuario final para enviársela al asistente y que se obtenga la respuesta.

Así que declaramos la correspondiente regla Parm, donde el parámetro de entrada es la pregunta del usuario, y el parámetro de salida, o retorno, es la respuesta del asistente, Podríamos también indicar filtros a la consulta, pero no lo haremos en este caso.

Bien. Luego indicamos la ejecución del POST, completando el path



## Aplicación GeneXus que interactúa con un RAG assistant



The image shows a GeneXus web panel on the left and its associated rule code on the right. The web panel, titled 'MainTable', contains a 'Chat with GeneXus Training' section. It features a text input field for a question, a button labeled 'Ask the Assistant', and a text area for the assistant's response, which is bound to the variable '&AssistantResponse'. The rule code on the right is as follows:

```
1 | Event 'GetMarketingDescription'  
2 |   &AssistantResponse = ChatWithTraining(&UserQuestion)  
3 | Endevent  
4 |
```

Ahora debemos completar su llamada desde el web panel. Para eso, vamos al evento asociado al botón y llamamos al procedimiento, enviándole como parámetro la variable &UserQuestion con la pregunta del usuario, y recibiendo su respuesta en la variable &AssistantResponse, ambas presentes en el form.

Para probarlo, presionamos F5, y ejecutamos el web panel.

## Aplicación GeneXus que interactúa con un RAG assistant



Recordemos que nuestro RAG assistant fue alimentado solamente con 18 documentos, lo cual no es mucha información, y por ese motivo haremos preguntas simples cuyas respuestas se encuentren en el alcance de esos documentos. Esto nos permitirá testear la aplicación.

En primer lugar, vamos a preguntar: ¿Qué es una base de conocimiento GeneXus? Veamos la respuesta.

Bien. Preguntemos ahora: ¿Qué es un objeto transacción?

Muy bien. Y si preguntamos ¿qué es un atributo fórmula en GeneXus?

Perfecto. Las respuestas recibidas fueron prolijas y correctas. Lo damos por válido.

Ahora observemos el menú. Tenemos acceso al asistente para chat y al asistente para chatear con documentos.

De esta forma entonces hemos logrado una aplicación GeneXus que interactúa con dos asistentes de inteligencia artificial creados a través de GeneXus Enterprise AI.

**GeneXus**<sup>™</sup>  
by **Globant**

[training.genexus.com](https://training.genexus.com)