

Aplicação GX que interage com um RAG Assistant



Alejandra Caggiano

Vimos anteriormente como interagir a partir de uma base de conhecimento GeneXus com um Chat assistant.

Aplicação GX que interage com um RAG Assistant



The screenshot displays the GeneXus Enterprise AI Course interface. On the left, a dark sidebar contains the GeneXus logo and a navigation menu with items: UNANIMO, Products, and Chat with GeneXus Training (highlighted). The main content area is titled "GeneXus Enterprise AI Course" and "Chat with GeneXus Training". It features a question input field containing "What is a transaction object?" and a blue button labeled "ASK THE ASSISTANT". 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

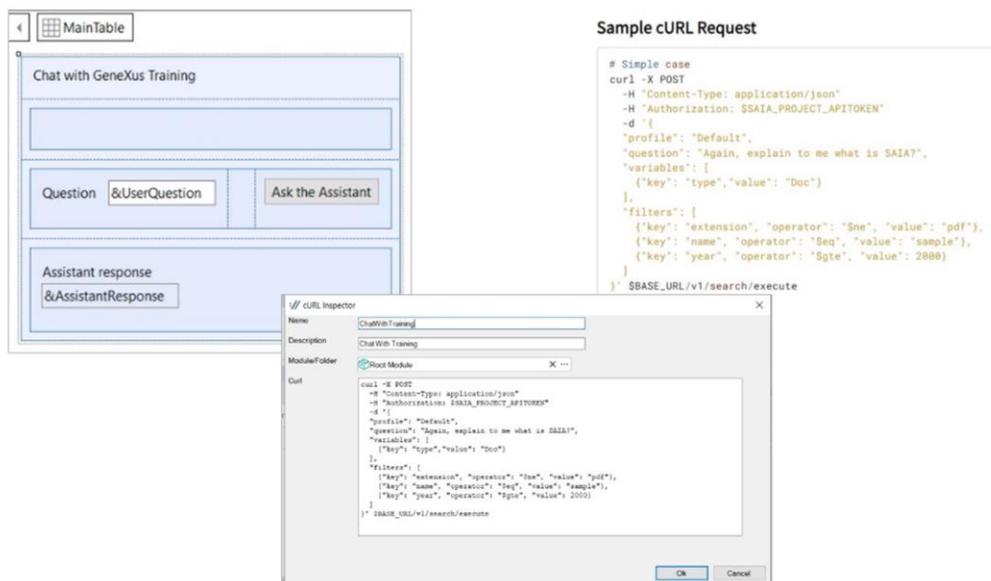
A transaction object in the context provided refers to a specific type of object created in GeneXus to represent real-world entities or actions that need to be recorded or processed within a system. In this context, it is mentioned that for every object from reality identified, a transaction object will 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 ampliar agora essa mesma base de conhecimento para que também possa interagir com nosso RAG Assistant chamado ChatWithGXTraining. Desta forma, permitiremos ao usuário final realizar perguntas sobre o referido conteúdo.

Aplicação GX que interage com um RAG Assistant



Se formos ao GeneXus, já criamos o web panel chamado ChatGXTraining, onde o usuário final pode inserir uma consulta e perguntar ao assistente para obter a resposta.

Temos duas variáveis. A variável &UserQuestion, sobre a qual o usuário fará sua consulta, e a variável &AssistantResponse, que será a responsável por exibir a resposta recebida.

Nosso objetivo então é codificar o evento associado ao botão para que ao pressioná-lo chame um procedimento que envie a consulta e receba a resposta.

Para criar este procedimento já com a base da estrutura do request, vamos carregar a cURL sample através do menu Tools / Application Integration / cURL Inspector.

Vamos colocar como nome ChatWithTraining e colamos o sample, disponível na documentação 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)

Neste caso precisamos interagir com a api que nos permite conversar com documentos, então o sample que colamos corresponde à cURL Request.

Aplicação GX que interage com um 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-GV0QCLi2YHYxBBGAshAg1CilSa9lFgeZKcQr5S0xsehbjpg
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
```

Pressionamos Ok e vemos o procedimento criado com a estrutura base que necessitamos.

A primeira coisa que indicamos é o protocolo de conexão. E para isso, como já sabemos, declaramos o método Secure com o valor 1 que corresponde ao protocolo HTTPS.

A seguir indicamos o Host que corresponde ao conteúdo da variável &BASE_URL, e que depende por sua vez da instalação de GeneXus Enterprise AI.

Bem. Precisamos então indicar o Project api token, que como também já sabemos, o copiamos da plataforma e o colamos.

Vamos agora ao corpo da consulta.

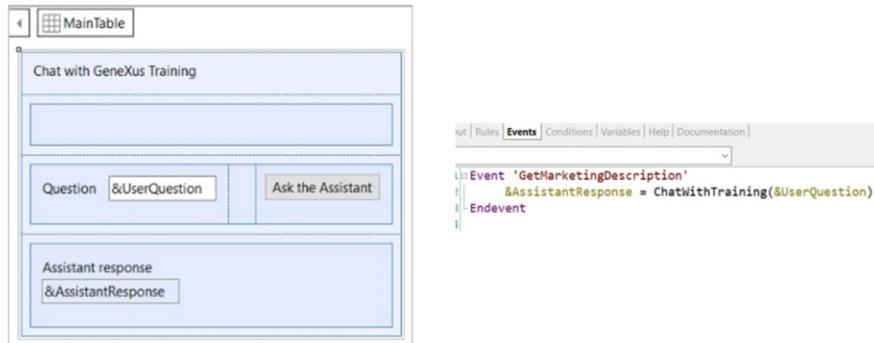
Em primeiro lugar, indicamos o "profile", ou seja, o nome do assistente, que neste caso é ChatWithGXTraining.

Assim como analisamos no exemplo anterior, este procedimento deverá receber a consulta do usuário final para enviá-la ao assistente e obter a resposta.

Então declaramos a correspondente regra Parm, onde o parâmetro de entrada é a pergunta do usuário, e o parâmetro de saída, ou retorno, é a resposta do assistente,

Poderíamos também indicar filtros para a consulta, mas não o faremos neste caso.
Bem. Depois indicamos a execução do POST, completando o path.

Aplicação GX que interage com um RAG Assistant



The image shows a screenshot of a GX application interface and its underlying rule logic. The interface, titled "MainTable", contains a chat window titled "Chat with GeneXus Training". The chat window has a text input field, a "Question" label, a text input field containing "&UserQuestion", and a button labeled "Ask the Assistant". Below the input field is a label "Assistant response" and a text input field containing "&AssistantResponse".

The rule logic is displayed in a window with a menu bar (File, Rules, Events, Conditions, Variables, Help, Documentation) and a dropdown menu. The code is as follows:

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

Agora devemos completar sua chamada a partir do web panel. Para isso, vamos até o evento associado ao botão e chamamos o procedimento, enviando como parâmetro a variável &UserQuestion com a pergunta do usuário, e recebendo sua resposta na variável &AssistantResponse, ambas presentes no form.

Para testá-lo, pressionamos F5 e executamos o web panel.

Aplicação GX que interage com um RAG Assistant



Lembremos que nosso RAG assistant foi alimentado apenas com 18 documentos, o que não é muita informação, e por esse motivo faremos perguntas simples cujas respostas se encontrem no âmbito desses documentos. Isso nos permitirá testar a aplicação.

Em primeiro lugar, vamos perguntar: O que é uma base de conhecimento GeneXus? Vejamos a resposta.

Bom. Perguntemos agora: O que é um objeto transação?

Muito bem. E se perguntarmos, o que é um atributo fórmula em GeneXus?

Perfeito. As respostas recebidas foram claras e corretas. Consideramos válido.

Agora vamos observar o menu. Temos acesso ao assistente para chat e ao assistente para conversar com documentos.

Desta forma então, conseguimos uma aplicação GeneXus que interage com dois assistentes de inteligência artificial criados através do GeneXus Enterprise AI.

GeneXus[™]
by **Globant**

training.genexus.com