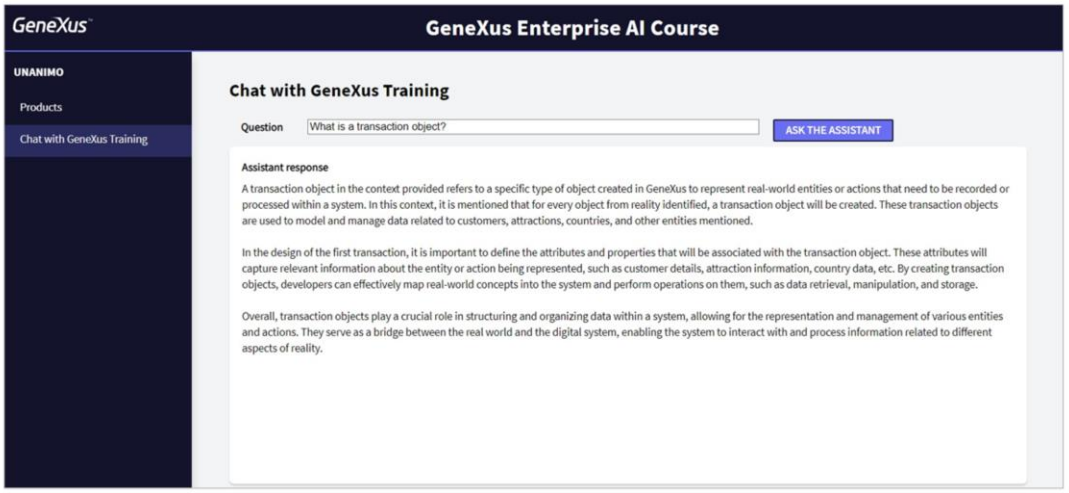
GeneXus by Globant

Aplicação GX que interage com um RAG Assistant

Alejandra Caggiano

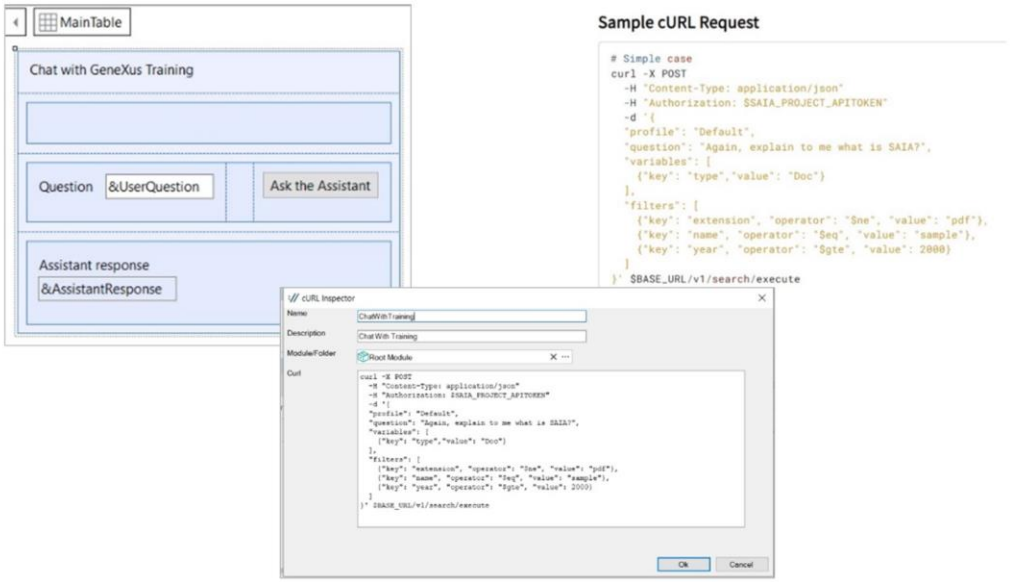
Vimos anteriormente como interagir a partir de uma base de conhecimento GeneXus com um Chat assistant.

Aplicação GX que interage com um RAG Assistant



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,

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

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.

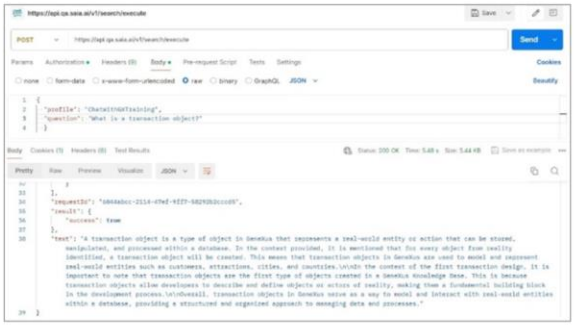
Assim como analisamos no exemplo anterior, este procedimento deverá receber a consulta do usuário final para enviá-la ao assistente e obter a resposta.

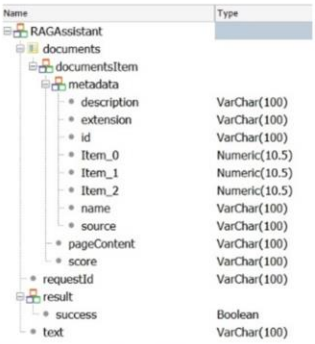
Então declaramos a correspondente regra Parm, onde o parâmetro de entrada é a pergunta do usuário, e o parâmetro de saída, ou retorno, é a resposta do assistente,

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






```
//curl -X POST  -H "Content-Type: application/json"  -H "Authorization: $SAIA_PROJECT_APITOKEN"  -d '{  "profile": "Defaul

&HttpClient.Secure = 1
&HttpClient.Host = "api.qa.saia.ai"

&HttpClient.AddHeader(!"Authorization", !"Bearer default_OuK5BwzqSLjNEHwUf-GV0QCLI2YHYxBBGAshAg1CiLSa9lFgeZKcQr5S0xsehbjbg
&HttpClient.AddHeader(!"Content-Type", !"application/json")

&HttpClient.AddString(!'{ "profile": "ChatWithGXTraining", "question": "' + &UserQuestion.Trim() + '" }')

&HttpClient.Execute(!"POST", "/v1/search/execute")

&RAGAssistant.FromJson(&HttpClient.ToString())

&AssistantResponse = &RAGAssistant.text
```

Até aqui então o envio da consulta. Agora devemos receber a resposta do assistente. Se lembrarmos da saída que vimos a partir do Postman quando testamos este assistente via api, vemos que esta saída corresponde a uma estrutura, e a resposta do assistente fica no item “text”.

Então a salvamos como um arquivo json, para depois importá-lo na KB e gerar automaticamente o tipo de dado estruturado que necessitamos para receber a resposta.

Voltamos então à KB e o importamos. Vamos para Tools / Aplication Integration / Json import. Colocamos como nome RAGAssistant e carregamos o arquivo.

Vemos que foi criado o tipo de dado estruturado. O abrimos e observamos o item “text”, onde receberemos a resposta do assistente.

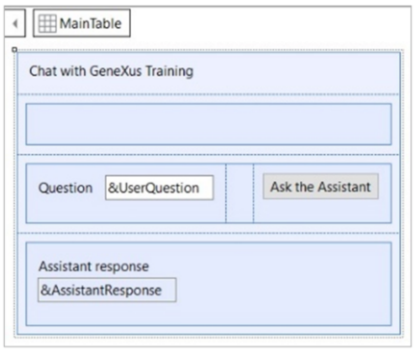
O que devemos fazer então? Devemos criar uma variável baseada neste tipo de dado e carregar ali a resposta do assistente.

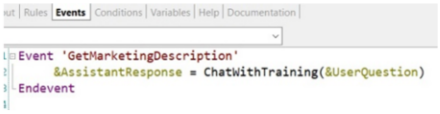
No procedimento, definimos então a variável &RAGAssistant e carregamos a resposta from Json.

Já temos a resposta completa, mas o que precisamos é do conteúdo do item “text”. Então, para nossa variável de retorno, &AssistantResponse, carregamos com esse conteúdo.

Bem. Já concluímos o procedimento.

Aplicação GX que interage com um RAG Assistant





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

GeneXus™
by Globant

training.genexus.com