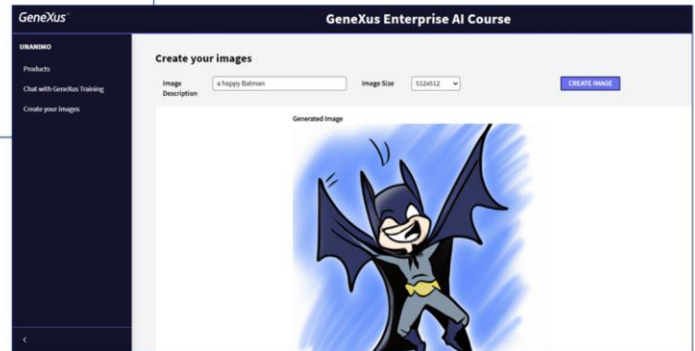
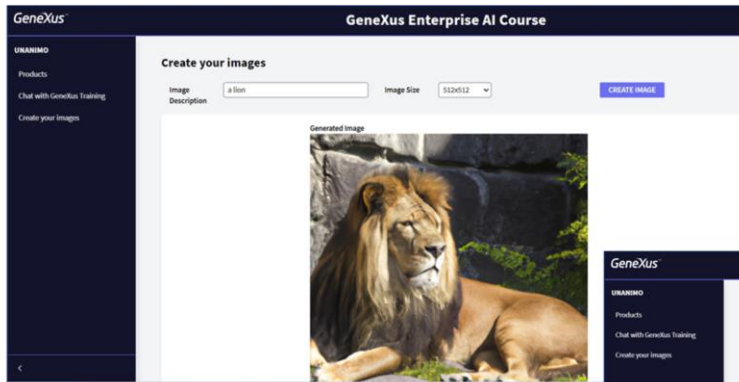


Aplicación GeneXus que interactúa con IA para generar imágenes



Alejandra Caggiano

Aplicación GeneXus que interactúa con IA para generar imágenes



Ya tenemos nuestra aplicación GeneXus que interactúa con un Chat assistant y con un RAG assistant. Vamos a habilitarle ahora la posibilidad de generar imágenes.

Aplicación GeneXus que interactúa con IA para generar imágenes

- GeneXus Enterprise AI: Proxy API
- DALL-E-2

Para eso vamos a utilizar la Proxy API para que interactúe con un sistema de inteligencia artificial que nos permita generar imágenes a partir de una descripción en lenguaje natural.

Aplicación GeneXus que interactúa con IA para generar imágenes

The screenshot shows a web application interface with a header bar containing '<No action group selected>'. Below the header is a navigation bar with a 'MainTable' button. The main content area is titled 'Create your images' and contains a form with three input fields: 'Image Description' with a placeholder '&ImageDescription', 'Image Size' with a dropdown menu '&ImageSize', and a 'Create image' button. Below the form is a section titled 'Generated Image' which displays a small thumbnail of a green landscape with a sun.

DALL-E-2 - cURL

```
curl --location 'https://api.saia.ai/proxy/openai/v1/images/generations' \
-H 'Content-Type: application/json' \
-H 'Authorization: $SAIA_API_TOKEN' \
-d '{
  "model": "dall-e-2",
  "prompt": "a halloween pumpkin",
  "size": "1024x1024"
}'
```

Vamos entonces a GeneXus, a nuestra base de conocimiento.

Ya tenemos creado un web panel de nombre `CreareImages`, donde habilitamos al usuario final a ingresar una breve descripción, y seleccionar el tamaño de la imagen a generar.

La funcionalidad entonces que queremos resolver es el código en el evento asociado al botón `Create Image`. Y para eso debemos crear un procedimiento que reciba la descripción y tamaño de la imagen como parámetros de entrada y devuelva la url de la imagen generada. A partir de esa url mostraremos la imagen en la variable `&GeneratedImage` presente en el form del web paneo.

Vamos a `Tools / Integration application / cURL Inspector`.

Ponemos como nombre `GenerareImages` y cargamos el cURL sample, que en este caso corresponde a la interacción de la Proxy API con DALL-E-2

Aplicación GeneXus que interactúa con IA para generar imágenes

```
Layout Rules * Conditions Variables Help Documentation |
1 Parm(in: &ImageDescription, in: &ImageSize, out:&ImageURL);
2
3
4 //curl --location 'https://api.saia.ai/proxy/openai/v1/images/generations' -H 'Content-Type: application/json'
5 &HttpClient.Secure = 1
6 &HttpClient.Host = "api.qa.saia.ai"
7 &HttpClient.AddHeader("Content-Type", "application/json")
8 &HttpClient.AddHeader("Authorization", "Bearer default_OuK5BwzqSLjNEHwUF-GV0QLI2YHYxBBGashAg1CiLSa91F")
9 &HttpClient.AddString("{\"model\":\"dall-e-2\", \"prompt\": \"' + &ImageDescription.Trim() + '\" , \"size\": \"' + &ImageSize + '\"}");
10
11 &HttpClient.Execute("POST", !"/proxy/openai/v1/images/generations")
12
```

Como ya sabemos, esto genera el procedimiento base donde completaremos la definición de la conexión con los datos necesarios, según nuestro contexto.

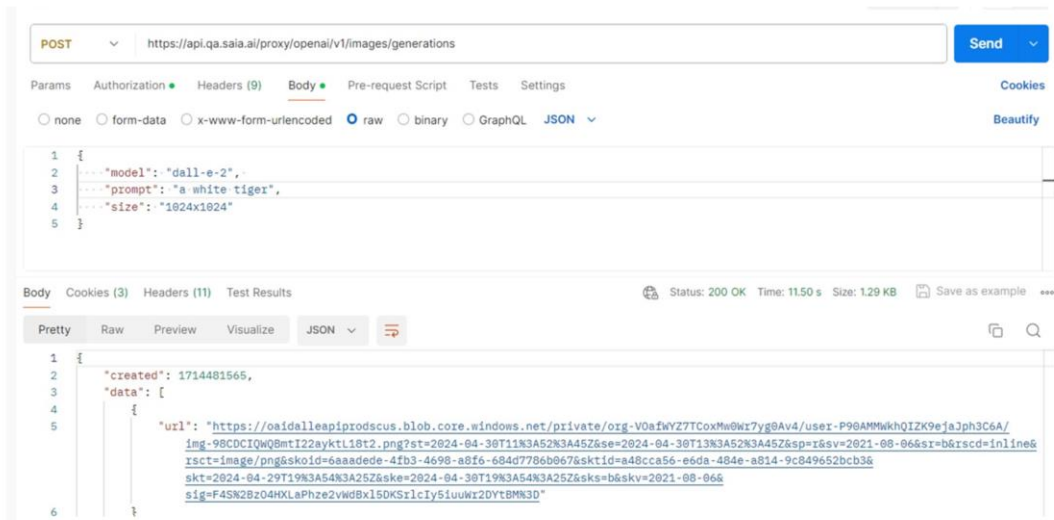
En primer lugar, declaramos la regla Parm para que reciba la descripción y tamaño de la imagen, y devuelva la url de la imagen generada.

Analicemos el source:

Definimos el protocolo de conexión HTTPS, definimos el host, el project api token, y definimos el cuerpo de la solicitud de forma tal que considere los parámetros de entrada que corresponden a la descripción y tamaño de la imagen requerida por el usuario final.

Luego definimos el POST para su ejecución.

Aplicación GeneXus que interactúa con IA para generar imágenes



The screenshot displays a Postman interface for a REST client. The top section shows a POST request to the URL `https://api.qa.saia.ai/proxy/openai/v1/images/generations`. The request body is a JSON object with the following content:

```
1 {
2   "model": "dall-e-2",
3   "prompt": "a white tiger",
4   "size": "1024x1024"
5 }
```

The bottom section shows the response body in JSON format, with a status of 200 OK. The response contains a timestamp and a list of image URLs:

```
1 {
2   "created": 1714481565,
3   "data": [
4     {
5       "url": "https://oaidalleapiprodscus.blob.core.windows.net/private/org-V0afWY27TCoxMw0W17yg0Av4/user-P96AMMwkhQIZK9ejaJph3C6A/img-98CDIQW0BmtI22aykt118t2.png?st=2024-04-30T19%3A52%3A45Z&se=2024-04-30T19%3A52%3A45Z&sp=r&sv=2021-08-06&sr=b&rscd=inline&rsct=image/png&skoid=6aaadede-4fb3-4698-a8f6-684d7786b067&sktid=a48cca56-e6da-484e-a814-9c849652bcb3&skt=2024-04-29T19%3A54%3A25Z&ske=2024-04-30T19%3A54%3A25Z&skv=2021-08-06&sig=F45N2Bz04HXLaPhze2vhd8x15DKSr1cIy5Iuuwr2DYtBMN3D"
6     }
7   ]
8 }
```

¿Qué sucede con la respuesta recibida?

Repetimos el mismo proceso que realizamos en los ejemplos anteriores. Desde Postman salvamos la respuesta de la solicitud e importamos el archivo en nuestra base de conocimiento, utilizando la opción Tools / Application integration / Json import.

Aplicación GeneXus que interactúa con IA para generar imágenes

Name	Type
MyImages	
• created	Numeric(10.5)
data	
dataItem	
• url	VarChar(100)

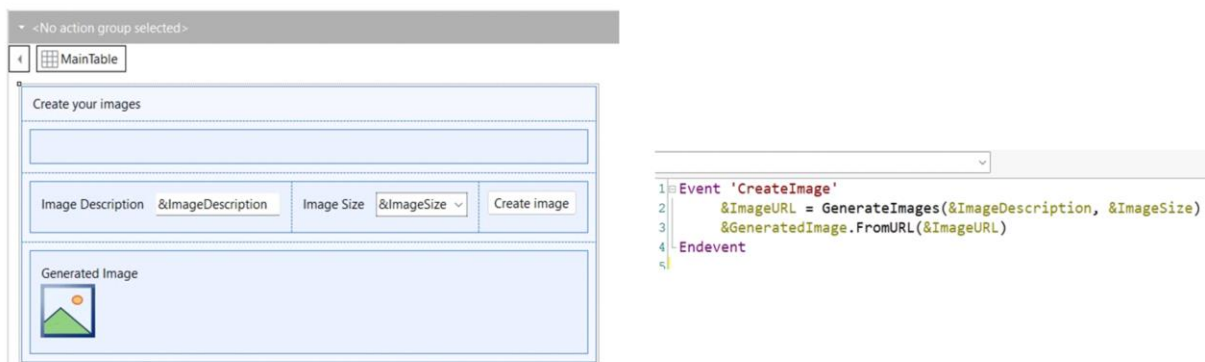
```
1 //curl --location 'https://api.saia.ai/proxy/openai/v1/images/generations' -H 'Content-Type: application
2
3 &HttpClient.Secure = 1
4 &HttpClient.Host = "api.qa.saia.ai"
5
6 &HttpClient.AddHeader("Content-Type", "application/json")
7 &HttpClient.AddHeader("Authorization", "Bearer default_OuK5BwzqSLjNEHwUf-GV0QCLi2YHYxBBGAShAg1CiLSa91F
8
9 &HttpClient.AddString(!{"model": "dall-e-2", "prompt": " " + &ImageDescription.Trim() + " " , "size": " " +
10
11 &HttpClient.Execute("POST", !"/proxy/openai/v1/images/generations")
12
13
14 &MyImages.FromJson(&HttpClient.ToString())
15
16 &ImageURL = &MyImages.data.Item(1).url
17
```

Ponemos como nombre MyImages, y esto genera el siguiente tipo de dato estructurado, donde el elemento url del primer item de la colección "data" corresponderá a la url de la imagen generada y será el parámetro de salida de nuestro procedimiento.

Así que definimos la variable &MyImages, basada en el sdt, y la cargamos con la respuesta de la solicitud aplicando el método FromJson, como ya hemos visto en los ejemplos anteriores.

Finalmente cargamos la variable &ImageURL con el valor del elemento "url" del primer item de la colección "data".

Aplicación GeneXus que interactúa con IA para generar imágenes



The image shows a GeneXus web panel on the left and its corresponding code on the right. The web panel, titled "MainTable", contains a form with the following elements:

- A header "Create your images" above a text input field.
- Two input fields: "Image Description" with placeholder "&ImageDescription" and "Image Size" with a dropdown menu "&ImageSize".
- A "Create image" button.
- A section labeled "Generated Image" containing a small image icon of a landscape with a sun and mountains.

The code on the right is an event handler for the "CreateImage" event:

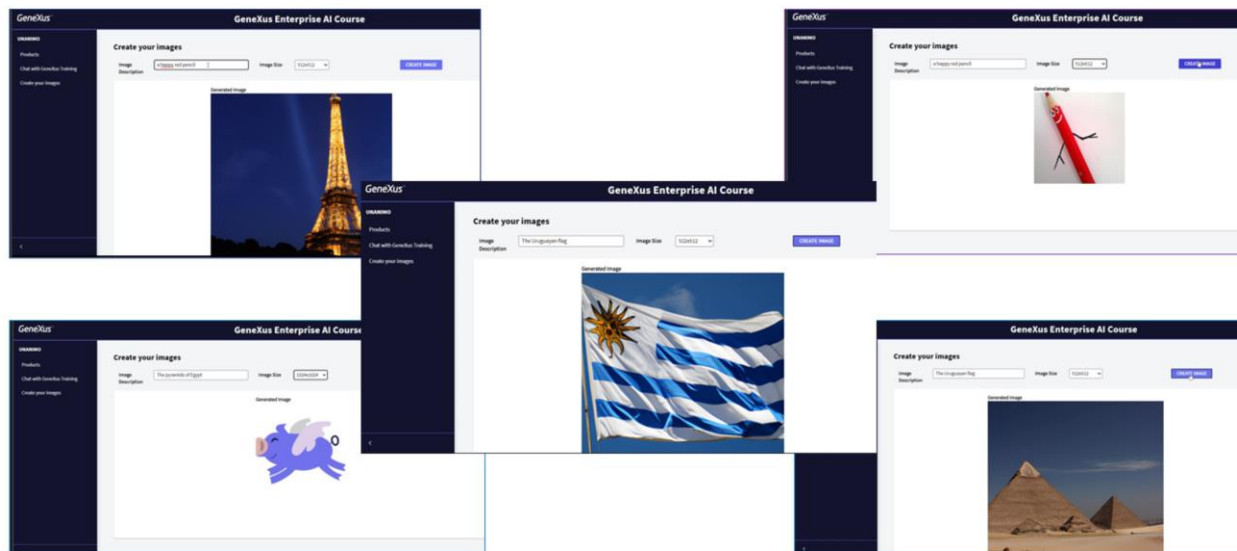
```
1 Event 'CreateImage'  
2   &ImageURL = GenerateImages(&ImageDescription, &ImageSize)  
3   &GeneratedImage.FromURL(&ImageURL)  
4 Endevent  
5
```

Una vez definido este procedimiento debemos llamarlo desde el evento asociado al botón del web panel. Así que hacemos la llamada y recibimos su salida en la variable &ImageURL.

Por último, a la variable &GeneratedImage, presente en el form, la cargamos con la imagen que se obtiene de la url recibida.

Para probarlo, presionamos F5.

Aplicación GeneXus que interactúa con IA para generar imágenes



En primer lugar, vamos a pedir una imagen de la torre Eiffel de noche, y elegimos como tamaño 512x512. Si la imagen que recibimos no nos gusta, pedimos otra.

Veamos otros ejemplos algo más creativos, como puede ser un lápiz rojo sonriendo... o un cerdito azul volando, de tamaño 256x256

Las pirámides de Egipto... y la bandera Uruguaya

GeneXus[™]
by **Globant**

training.genexus.com