Diseño del Prompt para el monitoreo de Logs

Estoy necesitando implementar un proyecto ASP.NET Core Web Api con .NET 9 para el monitoreo de unas Tables de Azure Storage para dos aplicaciones AppSalud y LinaChatbot

El objetivo del proyecto es leer unos Logs que están en unas tablas, para detectar:

- errores de servicios externos (outbound)
- errores internos recurrentes de la aplicación
- tiempos de respuesta elevados según el tipo de servicio

Criterio de aceptación 1: Identificación errores y tiempos de respuestas elevados

Dado que se requiere analizar los Logs en las diferentes tablas del Azure Storage,
Cuando se obtengan los registros de las tablas,
Entonces se debe implementar un LLM que analice los registros obtenidos para identificar erros de servicio externos, internos, y tiempos de respuestas elevados.

Criterio de aceptación 2: Identificación errores y tiempos de respuestas elevados

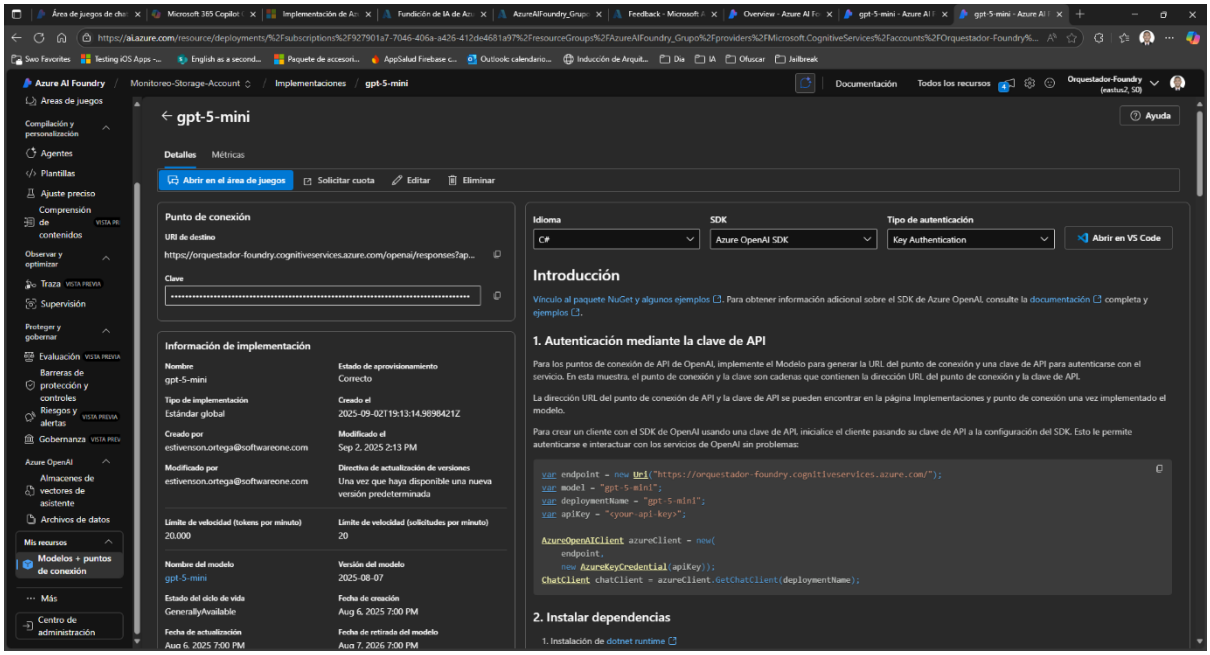
Dado que se requiere enviar una notificación
Cuando cuando el criterio de aceptación 1 se cumpla
Entonces se debe enviar una notificación ya sea por (Telegram, correo, etc.)

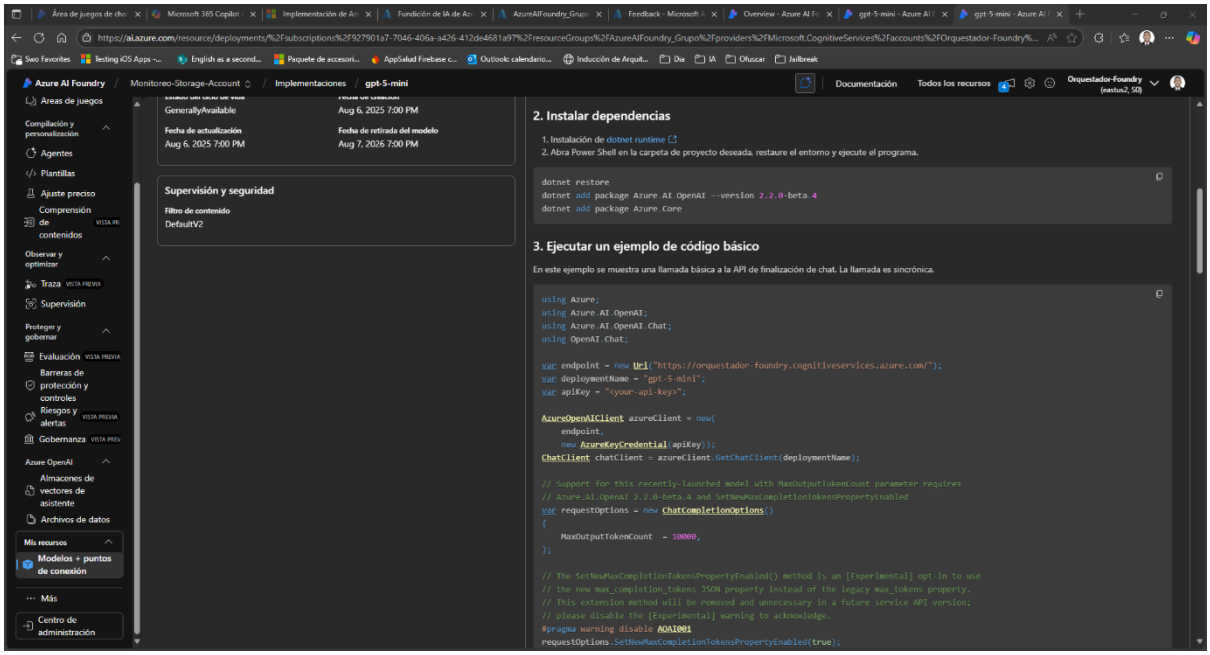
Recursos

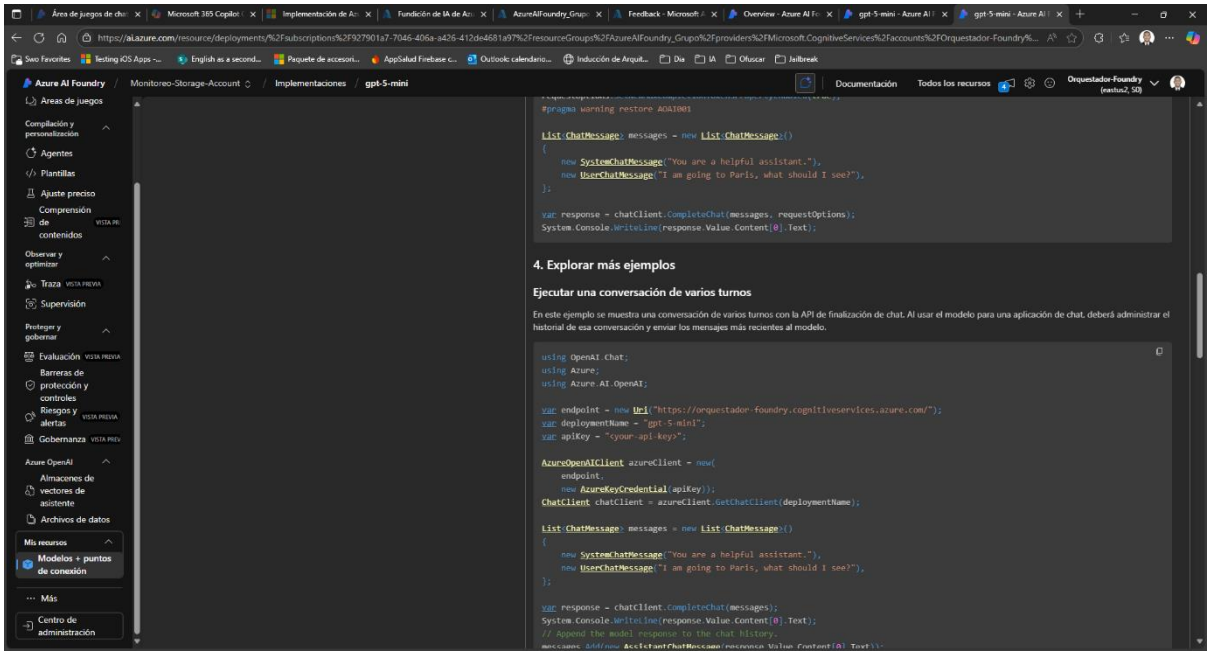
Para AppSalud este son los datos que se tienen:
Cuneta de almacenamiento:
DefaultEndpointsProtocol=https;AccountName=appsaluddevstorage;AccountKey=tgw633R2bnHIWaJnuhgkEUaLk0tUU6bv0MSjkpIvBQYpz2ZcYSMQWN/ngwQDjMLzTQynp5VgDyOAg==;EndpointSuffix=core.windows.net;
Listado de tablas a analizar: AppLog, Log, LogCsAuthenticate

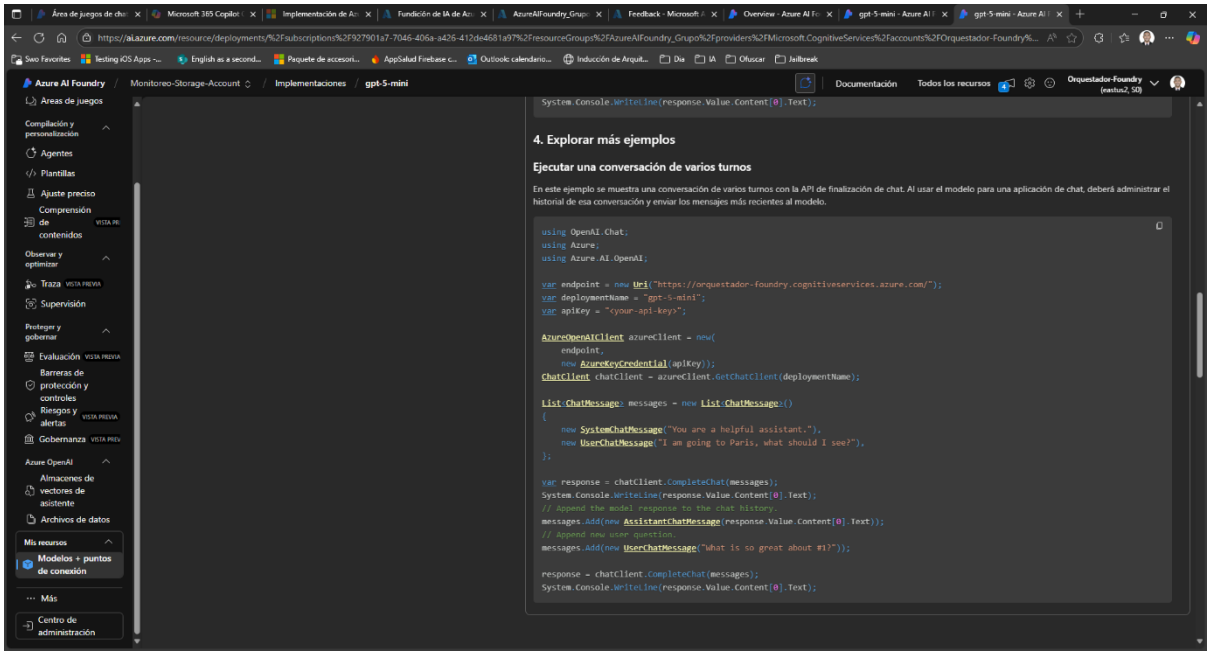
Para LinaChatbot este son los datos que se tienen:
Cuneta de almacenamiento:
DefaultEndpointsProtocol=https;AccountName=Linadevstorage;AccountKey=tgw633R2bnHIWaJnuhgkEUaLk0tUU6bv0MSjkpIvXRTBQYpz2ZcYSMQWN/ngwQDjMLzTQynp5VgDyOAg==;EndpointSuffix=core.windows.net;
Listado de tablas a analizar: AppLog, Log, LogCsAuthenticate

En cuanto al LLM esto es lo que se tiene:






#pragma warning restore AOAI001
List<ChatMessage> messages = new List<ChatMessage>()
{
new SystemChatMessage("You are a helpful assistant."),
new UserChatMessage("I am going to Paris, what should I see?"),
};
var response = chatClient.CompleteChat(messages, requestOptions);
System.Console.WriteLine(response.Value.Content[0].Text);
4. Explorar más ejemplos
Ejecutar una conversación de varios turnos
En este ejemplo se muestra una conversación de varios turnos con la API de finalización de chat. Al usar el modelo para una aplicación de chat, deberá administrar el historial de esa conversación y enviar los mensajes más recientes al modelo.
using OpenAI.Chat;
using Azure;
using Azure.AI.OpenAI;
var endpoint = new Uri("https://orquestador-foundry.cognitiveservices.azure.com/");
var deploymentName = "gpt-5-mini";
var apiKey = "<your-api-key>";
AzureOpenAIClient azureClient = new(
endpoint,
new AzureKeyCredential(apiKey));
ChatClient chatClient = azureClient.GetChatClient(deploymentName);
List<ChatMessage> messages = new List<ChatMessage>()
{
new SystemChatMessage("You are a helpful assistant."),
new UserChatMessage("I am going to Paris, what should I see?"),
};
var response = chatClient.CompleteChat(messages);
System.Console.WriteLine(response.Value.Content[0].Text);
// Append the model response to the chat history.


System.Console.WriteLine(response.Value.Content[0].Text);
4. Explorar más ejemplos
Ejecutar una conversación de varios turnos
En este ejemplo se muestra una conversación de varios turnos con la API de finalización de chat. Al usar el modelo para una aplicación de chat, deberá administrar el historial de esa conversación y enviar los mensajes más recientes al modelo.
using OpenAI.Chat;
using Azure;
using Azure.AI.OpenAI;
var endpoint = new Uri("https://orquestador-foundry.cognitiveservices.azure.com/");
var deploymentName = "gpt-5-mini";
var apiKey = "<your-api-key>";
AzureOpenAIClient azureClient = new(
endpoint,
new AzureKeyCredential(apiKey));
ChatClient chatClient = azureClient.GetChatClient(deploymentName);
List<ChatMessage> messages = new List<ChatMessage>()
{
new SystemChatMessage("You are a helpful assistant."),
new UserChatMessage("I am going to Paris, what should I see?"),
};
var response = chatClient.CompleteChat(messages);
System.Console.WriteLine(response.Value.Content[0].Text);
// Append the model response to the chat history.
messages.Add(new AssistantChatMessage(response.Value.Content[0].Text));
// Append new user question.
messages.Add(new UserChatMessage("What is so great about #1?"));
response = chatClient.CompleteChat(messages);
System.Console.WriteLine(response.Value.Content[0].Text);