Universidad Mariano Gálvez Guatemala, Campus Jutiapa

Facultad de Ingeniería en Sistemas

Curso: Programación 1

Catedrático: Ing. Ruldin Ayala

Nombre del estudiante: Lester David Payes Méndez

Carné: 0905-24.22750

Tarea: Agente de Automatización de Investigación y Elaboración de Informes:

Introducción

Este proyecto es una aplicación de escritorio desarrollada en C# (.NET 8, WinForms) que permite a los usuarios generar investigaciones académicas, títulos y resúmenes utilizando la API de OpenAI. Además, la aplicación guarda un historial de consultas en una base de datos SQL Server y genera documentos Word y presentaciones PowerPoint de manera automática y profesional.

APIs y Tecnologías Utilizadas

- ## OpenAI API:

¿Por qué elegí la API de OpenAI?

La elección de la API de OpenAI se fundamenta en varios factores clave:

A. **Calidad y potencia del modelo:**
   OpenAI ofrece modelos de lenguaje de última generación, como GPT-3.5-turbo, que han demostrado ser líderes en tareas de generación de texto, comprensión semántica y respuesta contextualizada. Esto asegura que los resultados generados sean relevantes, coherentes y de alta calidad, lo cual es esencial en el ámbito académico.

B. **Facilidad de integración:**
   La API de OpenAI está diseñada para ser fácilmente integrable en aplicaciones de cualquier lenguaje de programación moderno. Su comunicación basada en HTTP y el uso de JSON como formato de intercambio de datos simplifican la implementación y el mantenimiento.

C. **Flexibilidad y personalización:**
   La API permite ajustar parámetros como la temperatura, el número máximo de tokens, la penalización por frecuencia y presencia, y el modelo a utilizar. Esto brinda un control preciso sobre la creatividad, longitud y estilo de las respuestas, adaptándose a diferentes necesidades y contextos.

D. **Documentación y soporte:**
   OpenAI proporciona una documentación extensa, ejemplos prácticos y una comunidad activa, lo que facilita la resolución de dudas y la implementación de nuevas funcionalidades.

Microsoft Office Interop:

Para la generación de documentos Word y presentaciones PowerPoint, utilicé las librerías de Interop de Microsoft Office, lo que permite crear y modificar archivos de Office desde C#.

- **SQL Server:**

La aplicación almacena el historial de consultas en una base de datos SQL Server, permitiendo así llevar un registro de los temas consultados, resultados y fechas.

Esquema de Base de Datos

CONSULTAS		
int	Id	PK
nvarchar(MAX)	Prompt	
nvarchar(MAX)	Resultado	
nvarchar(255)	Titulo	
datetime	Fecha	

- **.NET 8 y WinForms:**

El proyecto está desarrollado sobre la plataforma .NET 8, usando Windows Forms para la interfaz gráfica.

Proceso de Desarrollo

Diseño de la Base de Datos:

Lo primero que hice fue definir la estructura de la base de datos. Decidí que la tabla principal (Consultas) debía almacenar:

- **Prompt:** El texto de la consulta o el tema.
- **Resultado:** La respuesta generada por la IA.
- **Titulo:** El título académico generado.
- **Fecha:** La fecha de la consulta.

Esto permite llevar un historial completo y consultar investigaciones previas.

Estructura del Código y Carpetas

El proyecto está organizado en varias carpetas, cada una con una responsabilidad clara:

Carpeta; Config

Contiene las clases para mapear la configuración (appsettings.json):

- **AppSettings:** Clase principal de configuración.
- **OpenAISettings:** Almacena la API Key de OpenAI.
- **ConnectionStrings:** Almacena la cadena de conexión a la base de datos.

Carpeta; Models

Contiene los modelos de datos utilizados en la comunicación con la API de OpenAI y para la lógica interna:

- **OpenAIConsultaParams:** Define los parámetros que se envían a la API de OpenAI (prompt, temperatura, modelo, etc.).
- **OpenAIResponse, Choice, Message, Usage:** Modelos para deserializar la respuesta de OpenAI.

Carpeta; Services

Contiene la lógica de negocio y acceso a recursos externos:

- **DatabaseService:** Encapsula la lógica de conexión y operaciones con la base de datos SQL Server.
- **DocumentoService:** Encapsula la lógica para crear documentos Word y presentaciones PowerPoint usando plantillas y marcadores.
- **OpenAiService:** Encapsula la lógica para interactuar con la API de OpenAI, incluyendo la generación de textos, títulos y resúmenes.

Carpeta; Plantillas

Contiene los archivos de plantilla de Word y PowerPoint que se usan como base para los documentos generados.

Form1.cs (UI Principal)

Contiene la lógica de la interfaz gráfica, donde el usuario puede:

- Ingresar un tema o un prompt personalizado.
- Consultar a la IA y ver el resultado.
- Editar el resultado (solo si es prompt libre).
- Guardar el resultado, lo que genera los documentos y almacena la consulta en la base de datos.

Appsettings

El archivo appsettings.json es fundamental en aplicaciones .NET modernas, ya que permite centralizar la configuración de la aplicación en un solo lugar. En este archivo se almacenan datos sensibles y variables, como:

- La API Key de OpenAI (clave privada para acceder a la IA).
- La cadena de conexión a la base de datos SQL Server.

.gitignore

Lo utilice para que mi api key no se suba al repo para evitar problemas de bloque de API

Funcionamiento General

Flujo de la Aplicación

Inicio:

La aplicación verifica la conexión a la base de datos antes de iniciar.

Consulta:

El usuario ingresa un tema o un prompt personalizado y pulsa el botón de consulta.

- Si es un tema, la aplicación estructura el prompt para obtener una respuesta académica dividida en secciones.
- Si es un prompt libre, se envía tal cual a la IA.

Generación de Respuesta:

La aplicación llama a la API de OpenAI y muestra la respuesta en pantalla.

- Si el prompt es libre, el usuario puede editar la respuesta antes de guardarla.

Generación de Documentos:

Al pulsar "Guardar", la aplicación:

- Genera un documento Word y una presentación PowerPoint usando las plantillas y la información obtenida.
- Guarda la consulta en la base de datos.

Estructura y Uso de Namespaces

Proyecto1LesterFinalPrograma.Config:

Para la configuración de la aplicación.

Proyecto1LesterFinalPrograma.Models:

Para los modelos de datos y las clases que representan la estructura de las peticiones y respuestas de la API.

Proyecto1LesterFinalPrograma.Services:

Para los servicios que interactúan con recursos externos (base de datos, OpenAI, Office).

Proyecto1LesterFinalPrograma:

Para la lógica principal de la aplicación y la interfaz de usuario.

Decisiones de Diseño

Separación de responsabilidades:

Cada carpeta y clase tiene una responsabilidad única, siguiendo buenas prácticas de diseño.

Uso de plantillas:
Para facilitar la generación de documentos word profesionales y mantener un formato consistente.

Edición de resultados:
Se permite la edición del resultado solo cuando el usuario utiliza un prompt libre, para mantener la coherencia en los documentos estructurados.

Persistencia de datos:
Se almacena el historial de consultas para referencia futura y trazabilidad.

Uso de JSON, async/await, Task y Deserialización

JSON

- Se utiliza para la configuración de la aplicación (appsettings.json).
- Todas las comunicaciones con la API de OpenAI se realizan en formato JSON, tanto para enviar los parámetros como para recibir las respuestas.

async/await y Task

- Los métodos que interactúan con la API de OpenAI y la base de datos son asíncronos (async Task), lo que permite que la aplicación siga siendo responsiva y no se bloquee durante operaciones largas.
- **Ejemplo:**

  public async Task GuardarConsultaAsync(...)

  public async Task<(string respuesta, int promptTokens, int totalTokens)> HacerConsultaAsync(...)

Conclusión

El desarrollo de este proyecto demuestra el valor de integrar inteligencia artificial de última generación en aplicaciones de escritorio para resolver necesidades reales en el ámbito académico y profesional.
La API de OpenAI ha sido una elección estratégica por su potencia, flexibilidad y facilidad de integración, permitiendo generar textos de alta calidad y adaptados a las necesidades del usuario. El manejo eficiente de los tokens y la personalización de los parámetros de consulta han sido claves para optimizar los resultados y el costo del servicio.
La arquitectura modular del proyecto, basada en la separación de responsabilidades y el uso de buenas prácticas de desarrollo, facilita el mantenimiento, la escalabilidad y la incorporación de nuevas funcionalidades en el futuro.
En resumen, este sistema no solo automatiza la generación de documentos académicos, sino que también sienta las bases para futuras aplicaciones que requieran procesamiento avanzado de lenguaje natural, demostrando el impacto positivo de la IA en la productividad y la innovación.