Universidad Mariano Gálvez de Guatemala, Jutiapa

Docente: **Ruldin Efraín Ayala Ramos**

Curso: **Programación**

Tema:
Proyecto Final #1

Alumnos: **Geofrey Florián**

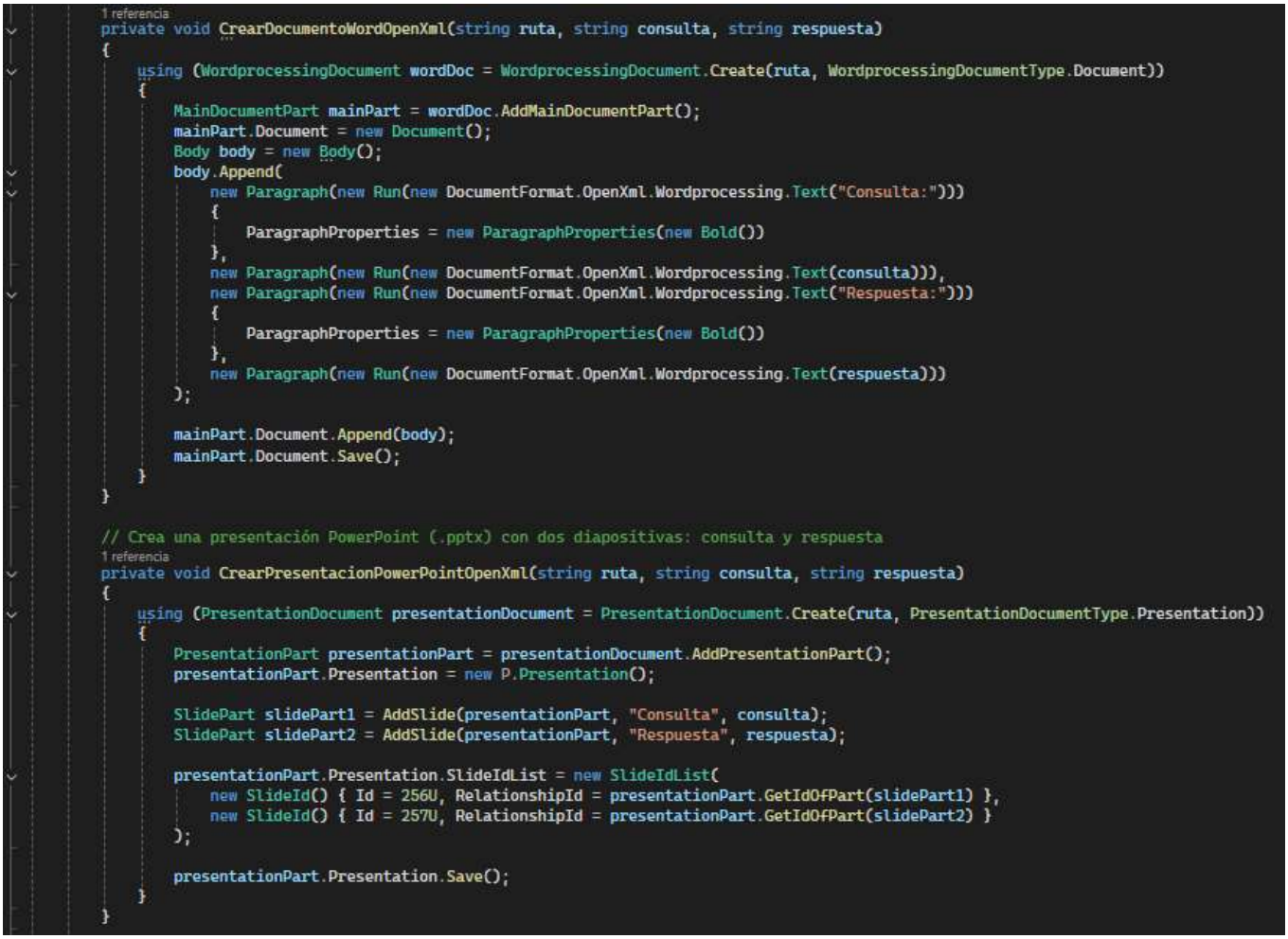
Carné: **0905-24-17570**

Sección: **“A”**

¿Cómo lo elaboré?

Comencé diseñando una interfaz amigable utilizando Windows Forms. Incluí elementos básicos como un `TextBox` para escribir la consulta, un botón para enviar esa consulta, y un campo de texto para mostrar la respuesta generada por la inteligencia artificial. A medida que el proyecto crecía, implementé también la funcionalidad para que, tras cada consulta, los resultados se guardaran automáticamente tanto en SQL Server como en documentos físicos (Word y PowerPoint), organizados en carpetas específicas con base en la consulta realizada.

También dediqué tiempo a implementar un mecanismo de control de errores y reintentos para la API, de manera que el sistema fuera estable, incluso si la API retornaba errores por demasiadas solicitudes.



```
1 referencia
private void CrearDocumentoWordOpenXml(string ruta, string consulta, string respuesta)
{
    using (WordprocessingDocument wordDoc = WordprocessingDocument.Create(ruta, WordprocessingDocumentType.Document))
    {
        MainDocumentPart mainPart = wordDoc.AddMainDocumentPart();
        mainPart.Document = new Document();
        Body body = new Body();
        body.Append(
            new Paragraph(new Run(new DocumentFormat.OpenXml.Wordprocessing.Text("Consulta:")))
            {
                ParagraphProperties = new ParagraphProperties(new Bold())
            },
            new Paragraph(new Run(new DocumentFormat.OpenXml.Wordprocessing.Text(consulta))),
            new Paragraph(new Run(new DocumentFormat.OpenXml.Wordprocessing.Text("Respuesta:")))
            {
                ParagraphProperties = new ParagraphProperties(new Bold())
            },
            new Paragraph(new Run(new DocumentFormat.OpenXml.Wordprocessing.Text(respuesta)))
        );

        mainPart.Document.Append(body);
        mainPart.Document.Save();
    }
}

// Crea una presentación PowerPoint (.pptx) con dos diapositivas: consulta y respuesta
1 referencia
private void CrearPresentacionPowerPointOpenXml(string ruta, string consulta, string respuesta)
{
    using (PresentationDocument presentationDocument = PresentationDocument.Create(ruta, PresentationDocumentType.Presentation))
    {
        PresentationPart presentationPart = presentationDocument.AddPresentationPart();
        presentationPart.Presentation = new P.Presentation();

        SlidePart slidePart1 = AddSlide(presentationPart, "Consulta", consulta);
        SlidePart slidePart2 = AddSlide(presentationPart, "Respuesta", respuesta);

        presentationPart.Presentation.SlideIdList = new SlideIdList(
            new SlideId() { Id = 256U, RelationshipId = presentationPart.GetIdOfPart(slidePart1) },
            new SlideId() { Id = 257U, RelationshipId = presentationPart.GetIdOfPart(slidePart2) }
        );

        presentationPart.Presentation.Save();
    }
}
```

¿Qué APIs utilicé?

Para este proyecto utilicé la API de **Groq**, una plataforma que permite acceder a modelos de lenguaje avanzados, similar a OpenAI. En este caso, utilicé el modelo **llama3-70b-8192**, que ofrece un equilibrio excelente entre comprensión y generación de texto.

La API se accede mediante una solicitud HTTP POST, a la que se le envía un `prompt` (consulta del usuario) junto con el modelo que se quiere usar. Una vez procesado, el modelo devuelve un texto generado como respuesta. Yo simplemente parseo esa respuesta y la presento al usuario en la interfaz.

Además, para generar los documentos, hice uso de la biblioteca **Open XML SDK** de Microsoft, la cual me permitió crear tanto archivos `.docx` (Word) como `.pptx` (PowerPoint) de forma programática y con un buen nivel de personalización.

Conclusión personal

Este proyecto fue una gran experiencia de integración entre diferentes tecnologías. Pude combinar la interacción con una API moderna de IA con herramientas tradicionales como Windows Forms, bases de datos en SQL Server y generación de documentos de Office, todo desde C#. Me permitió consolidar conocimientos de programación, servicios web, manejo de archivos y trabajo con bases de datos.

Quiero seguir mejorando esta herramienta con nuevas funcionalidades, como exportar también a PDF, agregar un historial más completo de consultas y permitir la edición del resultado generado antes de exportarlo.