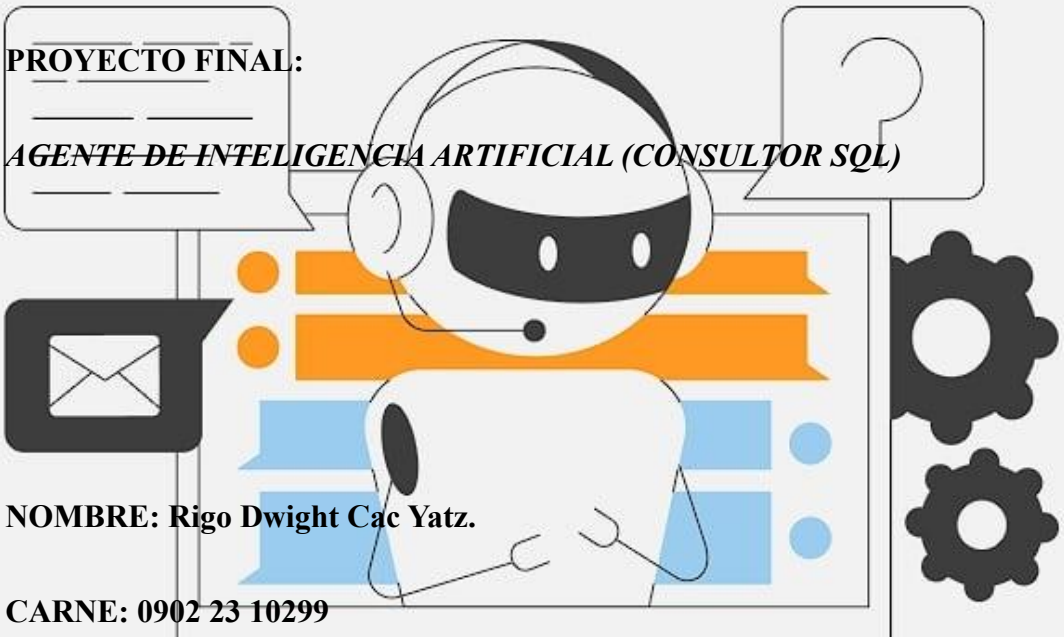
UNIVERSIDAD MARIANO GÁLVEZ DE GUATEMALA SEDE COBÁN. A.V

FACULTAD DE INGENIERIA EN SISTEMAS DE INFORMACIÓN Y CIENCIAS DE LA COMPUTACIÓN.



PROYECTO FINAL:

AGENTE DE INTELIGENCIA ARTIFICIAL (CONSULTOR SQL)

NOMBRE: Rigo Dwight Cac Yatz.

CARNE: 0902 23 10299

CURSO: Base de Datos I

INGENIERO: Edgar Molina.

SECCIÓN: A

FECHA: 07/11/2025

INDICE

Contenido

1. INTRODUCCIÓN

El presente documento describe el desarrollo de un Agente de Inteligencia Artificial (IA) diseñado para interpretar consultas escritas en lenguaje natural y traducirlas en sentencias SQL válidas que se ejecutan sobre una base de datos relacional.

El propósito del proyecto es facilitar la interacción entre el usuario y la base de datos, eliminando la necesidad de conocer la sintaxis SQL.

Aunque el requerimiento inicial proponía el uso de Oracle Database, este proyecto utiliza MySQL como alternativa funcional equivalente. MySQL fue elegido por su compatibilidad, accesibilidad y soporte para Python, lo que permite replicar el esquema HR de Oracle.

El agente se desarrolló en Python, utilizando la API Groq con el modelo LLaMA 3.3, especializado en procesamiento de lenguaje natural. La interfaz visual fue construida con Gradio, permitiendo una experiencia interactiva y moderna.

2. DEFINICIÓN DEL PROBLEMA

En los entornos empresariales, los usuarios que no dominan el lenguaje SQL enfrentan limitaciones para consultar o modificar la información almacenada en una base de datos.

El problema identificado consiste en la dificultad de acceder a los datos sin conocimientos técnicos, lo que reduce la eficiencia y la autonomía de los usuarios.

2.1. Objetivo General

Diseñar y desarrollar un Agente de Inteligencia Artificial capaz de interpretar consultas en lenguaje natural en español y traducirlas automáticamente a sentencias SQL válidas para ejecutarlas en una base de datos relacional, mostrando los resultados de manera clara y comprensible.

2.2. Objetivos Específicos

Implementar un sistema de procesamiento de lenguaje natural (NLP) que comprenda instrucciones en español

2.2.1. Desarrollar un módulo de traducción que genere SQL sintácticamente correcto

2.2.2. Crear mecanismos de seguridad que validen operaciones antes de ejecutarlas

2.2.3. Diseñar una interfaz gráfica intuitiva para usuarios no técnicos

2.2.4. Implementar manejo robusto de errores con mensajes descriptivos

3. MARCO TEÓRICO

Un Agente de Inteligencia Artificial es un sistema que percibe su entorno, analiza la información recibida y toma decisiones para alcanzar un objetivo.

En este contexto, el agente actúa como un traductor entre el lenguaje humano y el lenguaje de consultas estructurado (SQL).

El procesamiento de lenguaje natural (NLP) es una rama de la IA que permite a los sistemas comprender e interpretar texto o voz humanos. En este proyecto, el modelo Groq LLaMA 3.3 cumple esa función, convirtiendo frases en español en sentencias SQL

3.1. Base de datos utilizada

La base Human_Resources replica la estructura del esquema HR de Oracle, e incluye tablas como:

employees

departments

jobs

locations

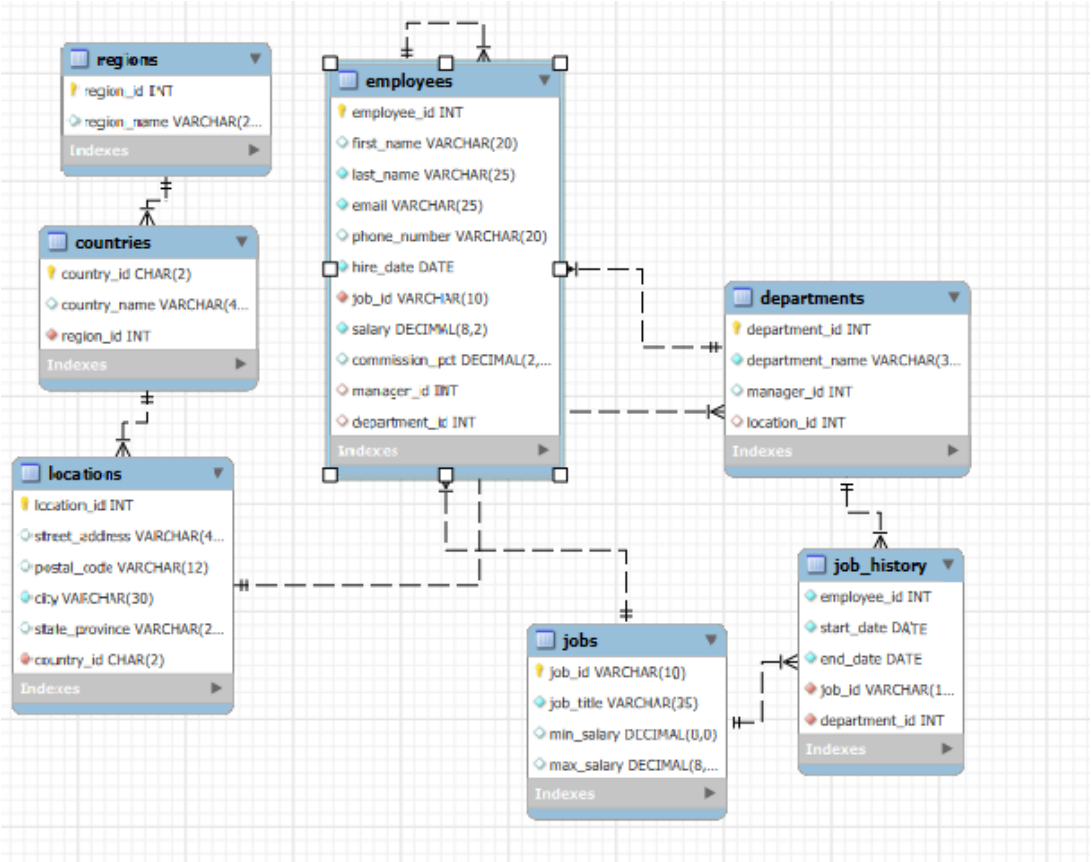
countries

regions

job_history

y la vista emp_details_view

Este esquema permite ejecutar consultas reales sobre empleados, departamentos, ubicaciones y más.



4. DISEÑO DEL AGENTE

El agente fue diseñado siguiendo un enfoque modular y escalable, con componentes independientes que interactúan entre sí para procesar la información.

4.1. Componentes principales

4.1.1. Interfaz de usuario (Gradio): recibe la pregunta en texto.

4.1.2. Módulo IA (Groq – LLaMA 3.3): interpreta la intención del usuario.

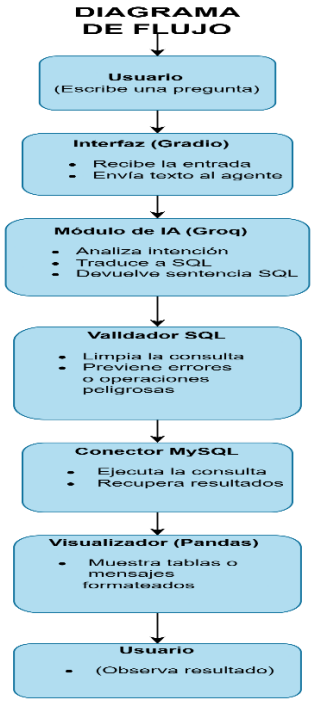
4.1.3. Generador SQL: traduce la consulta al lenguaje SQL.

4.1.4. Validador SQL: limpia, formatea y asegura que las operaciones sean seguras.

4.1.5. Conector MySQL: ejecuta la consulta.

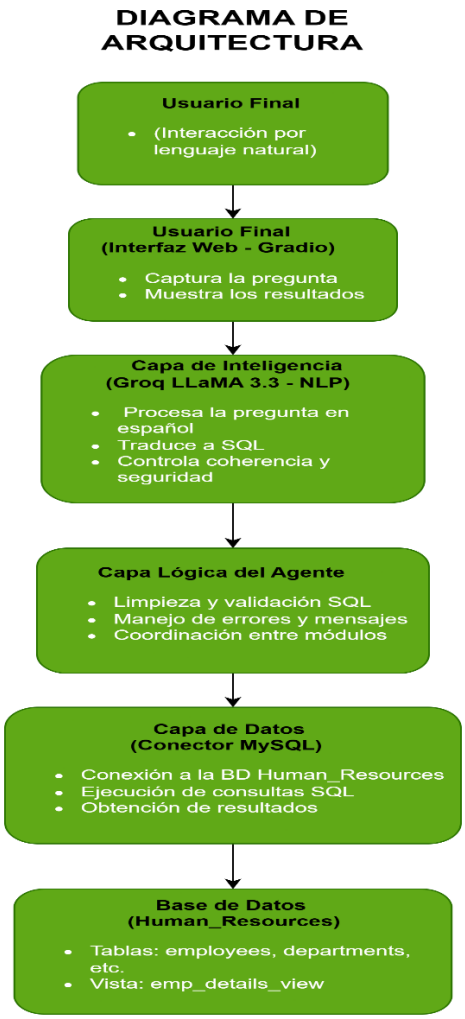
4.1.6. Visualizador (Pandas): muestra los resultados en tabla o texto.

4.1.7. Manejo de errores: controla consultas incorrectas o peligrosas.



5. Arquitectura del Sistema

La arquitectura se basa en una comunicación secuencial entre módulos:



5.1. Descripción de las capas

5.1.1. Capa de presentación: interfaz visual desarrollada en Gradio.

5.1.2. Capa de inteligencia: IA encargada de la interpretación y traducción.

5.1.3. Capa lógica: procesamiento, validación y control de errores.

5.1.4. Capa de datos: conexión directa con la base human_resources.

6. IMPLEMENTACIÓN TÉCNICA

El agente fue implementado en Python 3.10, utilizando librerías modernas que facilitan la integración entre IA, bases de datos y visualización.

6.1. Librerías empleadas

6.1.1. gradio → Interfaz gráfica web.

6.1.2. pandas → Visualización y análisis de datos.

6.1.3. mysql-connector-python → Conexión a la base de datos MySQL.

6.1.4. sqlparse → Formateo de consultas SQL.

6.1.5. python-dotenv → Manejo seguro de variables de entorno.

6.1.6. groq → API para el modelo de lenguaje LLaMA 3.3.

6.2. Archivos principales

6.2.1. agent_ia.py → Contiene toda la lógica del agente IA.

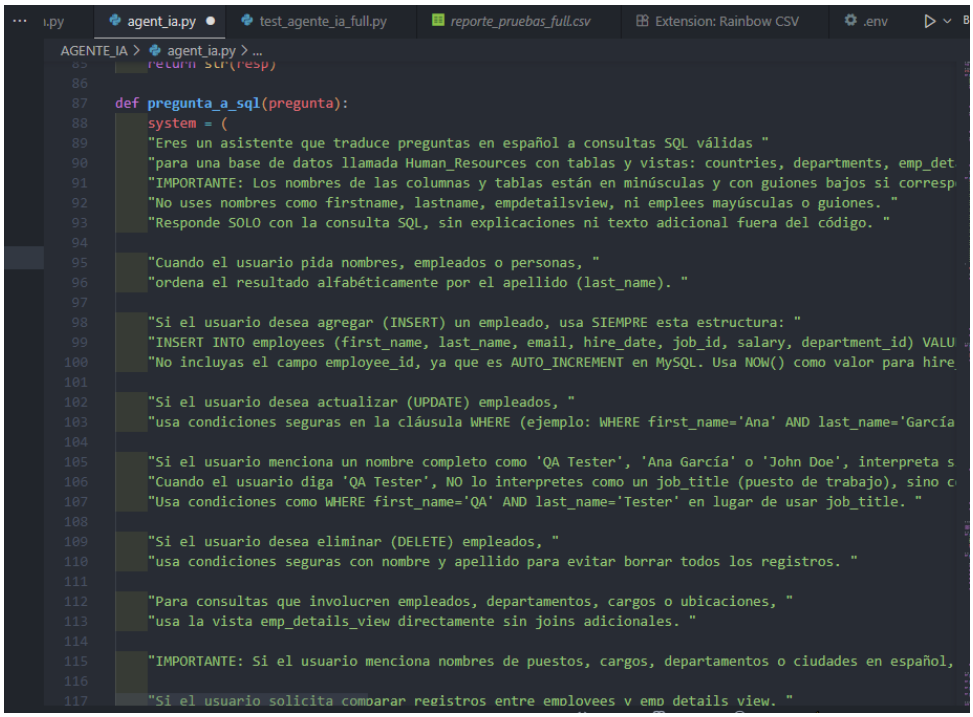
6.2.2. test_connection.py → Verifica la conexión con la base de datos.

6.2.3. test_agente.py → Ejecuta pruebas CRUD automatizadas.

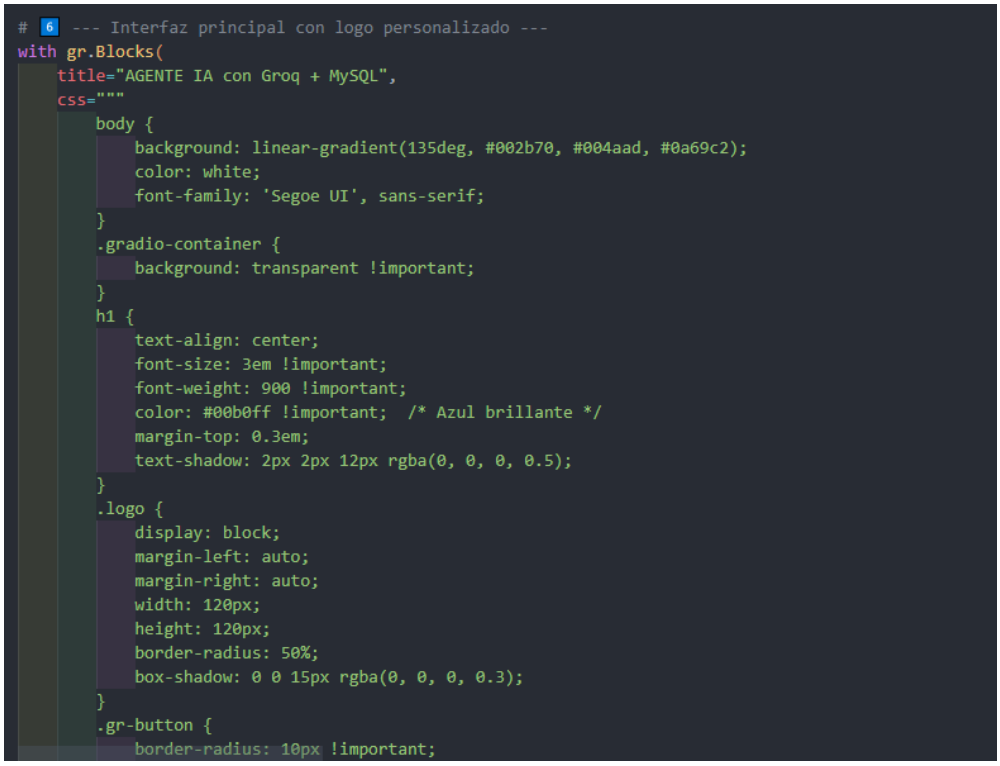
6.2.4. .env → Almacena credenciales y configuración.

6.2.5. human_resources.sql → Script de creación de la base de datos.

Fragmento del código del agente IA – Conexión a la base de datos.



Fragmento del código – Generación de consultas SQL



```
# 6 --- Interfaz principal con logo personalizado ---
with gr.Blocks(
    title="AGENTE IA con Groq + MySQL",
    css="""
        body {
            background: linear-gradient(135deg, #002b70, #004aad, #0a69c2);
            color: white;
            font-family: 'Segoe UI', sans-serif;
        }
        .gradio-container {
            background: transparent !important;
        }
        h1 {
            text-align: center;
            font-size: 3em !important;
            font-weight: 900 !important;
            color: #00b0ff !important;  /* Azul brillante */
            margin-top: 0.3em;
            text-shadow: 2px 2px 12px rgba(0, 0, 0, 0.5);
        }
        .logo {
            display: block;
            margin-left: auto;
            margin-right: auto;
            width: 120px;
            height: 120px;
            border-radius: 50%;
            box-shadow: 0 0 15px rgba(0, 0, 0, 0.3);
        }
        .gr-button {
            border-radius: 10px !important;
```

Fragmento del código – Interfaz visual en Gradio.

7. PRUEBAS Y RESULTADOS

Se realizaron pruebas funcionales y de seguridad para garantizar el correcto funcionamiento del sistema.

7.1. Casos de prueba

7.1.1. Consultar empleados por departamento.

7.1.2. Insertar un nuevo empleado.

7.1.3. Actualizar salario.

7.1.4. Eliminar registros controladamente.

7.1.5. Intentar eliminar todos los empleados (bloqueo de seguridad).

7.2. Resultados

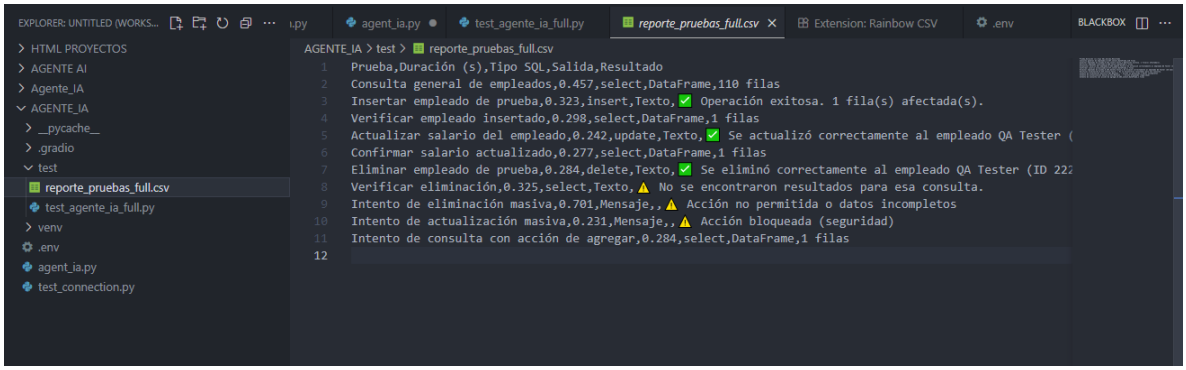
7.2.1. Consultas ejecutadas exitosamente.

7.2.2. Bloqueo efectivo de operaciones inseguras.

7.2.3. Tiempo de respuesta: < 0.5 segundos por consulta.

7.2.4. Respuestas claras con mensajes como:

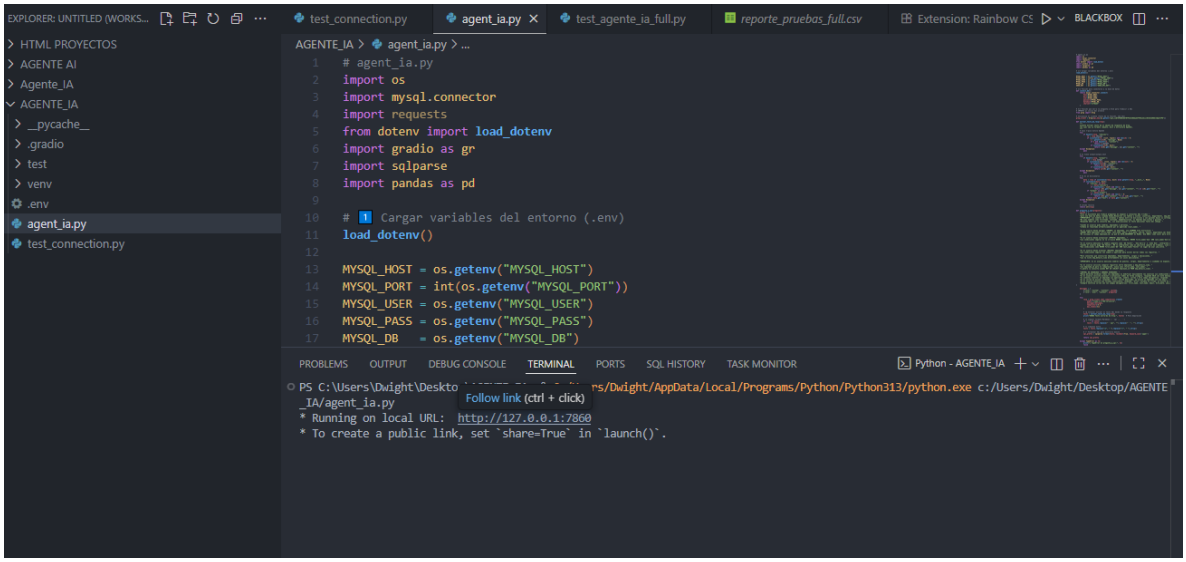
7.2.5. “Operación exitosa” o “Acción bloqueada por seguridad”.



8. Manual de Usuario

Ejecutar el archivo agent_ia.py.

Esperar a que se abra la interfaz en el navegador.



Escribir una pregunta en lenguaje natural por ejemplo:

“¿Cuántos empleados hay en total?”.

Seleccionar la acción correspondiente:

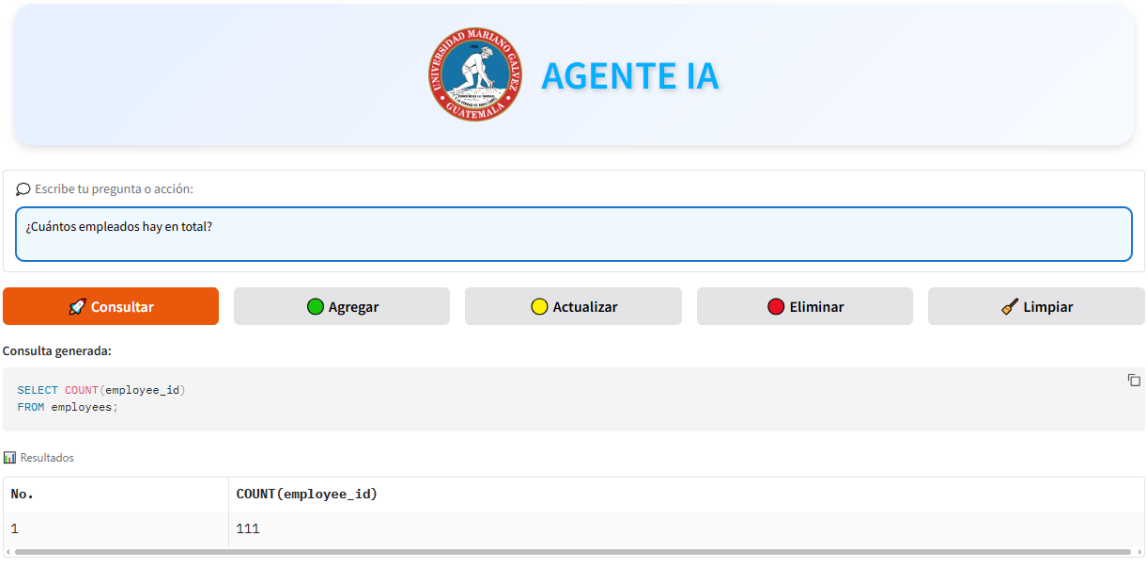
Consultar

Agregar

Actualizar

Eliminar

Ver los resultados en la tabla inferior.



Usar el botón “Limpiar” para reiniciar la interfaz.

Para garantizar el correcto funcionamiento del agente, se realizaron diferentes tipos de pruebas agrupadas en tres niveles: básicas, intermedias y avanzadas, además de una prueba CRUD completa (crear, leer, actualizar y eliminar).

A continuación, se muestran los scripts utilizados para validar el comportamiento del sistema.

8.1. Pruebas Básicas

Estas pruebas verifican la capacidad del agente para interpretar consultas simples en lenguaje natural y traducirlas correctamente a SQL.

```
pruebas_basicas = [
    ("Mostrar todos los empleados", "consultar"),
    ("Mostrar los primeros 5 empleados", "consultar"),
    ("Cuántos empleados hay en total", "consultar"),
    ("Mostrar los departamentos y sus ubicaciones", "consultar"),
    ("Mostrar los salarios más altos", "consultar")
]
```

8.2. Pruebas Intermedias

En esta etapa se evaluó la capacidad del agente para insertar, actualizar, eliminar y verificar registros específicos en la base de datos.

```
pruebas_intermedias = [
    ("Agrega a María López, correo mlopez@example.com, como Accountant en el departamento 110 con salario 4200.", "agregar"),
    ("Actualiza el salario de María López a 4800.", "actualizar"),
```

```
    ("Muestra el salario de María López.", "consultar"),
    ("Elimina a María López.", "eliminar"),
    ("Verifica si María López aún existe.", "consultar")
]
```

8.3. Pruebas Avanzadas

Estas pruebas buscan verificar la seguridad del sistema y el manejo correcto de intentos de consultas masivas sin condiciones seguras (WHERE).

```
pruebas_avanzadas = [
    ("Elimina todos los empleados", "eliminar"),          # Debe bloquearse
    ("Actualiza el salario de todos los empleados", "actualizar"),  # Debe bloquearse
    ("Muestra los empleados contratados recientemente", "consultar"),
    ("Muestra los empleados del departamento de TI", "consultar")
]
```

8.4. Prueba CRUD Completa

Finalmente, se realizó una prueba completa para comprobar el ciclo CRUD del agente: crear, consultar, actualizar y eliminar un registro de prueba.

```
prueba_crud = [
    ("Agrega a QA Tester con correo qatest@example.com como Programador en el departamento 60 con salario 4000.", "agregar"),
    ("Verifica si existe QA Tester.", "consultar"),
    ("Actualiza el salario de QA Tester a 4800.", "actualizar"),
    ("Muestra el salario de QA Tester.", "consultar"),
    ("Elimina a QA Tester.", "eliminar"),
    ("Verifica si QA Tester sigue existiendo.", "consultar")
]
```

Todas estas cónsulas mi agente las pudo resolver como se esperaba.

9. CONCLUSION

El desarrollo del Agente de Inteligencia Artificial para consultas SQL permitió integrar conocimientos de bases de datos, programación y procesamiento de lenguaje natural en una aplicación funcional y moderna.

El agente demostró la capacidad de interpretar instrucciones en lenguaje natural, traducirlas a sentencias SQL válidas y ejecutarlas de forma segura sobre la base de datos Human_Resources.

Además, se logró una interfaz intuitiva que facilita la interacción con la información sin necesidad de conocimientos técnicos en SQL.

La implementación con Python, MySQL y el modelo Groq LLaMA 3.3 evidenció que las tecnologías de inteligencia artificial pueden aplicarse eficazmente a entornos educativos y empresariales, promoviendo la automatización y accesibilidad de los datos.

En síntesis, el proyecto cumplió su objetivo principal: crear un agente capaz de actuar como puente inteligente entre el usuario y la base de datos, mostrando cómo la IA puede transformar la forma de consultar y gestionar información estructurada.

10. Anexos

10.1. Código fuente completo (agent_ia.py, test_agente.py).

10.2. Script SQL (human_resources.sql).

10.3. Reporte de pruebas (reporte_pruebas_full.csv).

10.4. Capturas de la interfaz funcionando.

10.5. Documento en Word y video demostrativo del funcionamiento.