Universidad Mariano Gálvez de Guatemala

Ingeniería En Sistemas De Información Y Ciencias De La Computación

Área o Sub-área: Base de Datos I

Catedrático: Ing. Edgar Raúl Molina Rey

Proyecto Final: Agente de IA

Repositorio: https://github.com/CristhianChan/Rafa.git

Video: https://youtu.be/ddlKaBBiC0A

Nombre del estudiante:	Carne:
Cristian Rafael Cú Chán	0902-23-2280

Lugar y Fecha: Cobán A.V 07/11/2025

Introducción

El acceso a la información contenida en las bases de datos relacionales ha sido tradicionalmente una barrera para el personal no técnico, ya que requiere un conocimiento especializado del lenguaje SQL. En el contexto de la gestión de inventarios, esta limitación puede ralentizar la toma de decisiones y la eficiencia operativa.

El presente proyecto aborda este desafío mediante la implementación de un **Agente de Inteligencia Artificial Text-to-SQL** para la gestión de inventario, cuyo propósito principal es **democratizar el acceso a los datos de la tienda**. Este agente actúa como un intérprete inteligente, capaz de recibir solicitudes en lenguaje natural (español) y traducirlas instantáneamente a sentencias SQL válidas.

La solución se construye sobre una arquitectura moderna que integra:

MySQL: Como sistema de gestión para la base de datos tienda_inventario.

Groq (Llama-3.1): Un modelo de lenguaje grande (LLM) para la interpretación de la intención del usuario y la generación del código SQL.

Flask y Python: Para el control del flujo de datos (backend) y la conexión con la base de datos.

Interfaz Web (HTML/CSS): Una aplicación de escritorio basada en navegador para una experiencia de usuario amigable y moderna.

El Agente de Inventario está diseñado para manejar consultas complejas (SELECT) y operaciones de modificación de datos (INSERT, UPDATE, DELETE), permitiendo a los usuarios gestionar el stock, actualizar precios y obtener reportes inmediatos del inventario sin escribir una sola línea de código, cumpliendo así con los requisitos establecidos para un sistema inteligente de gestión de datos.

1. Fase 1: Análisis del Problema

1.1 Definición del Dominio y Esquema de la Base de Datos

A. Definición del Dominio del Problema

El Agente IA se enmarca en el dominio de **Gestión de Inventarios y Control de Stock para una tienda o pequeño negocio**.

El objetivo principal es democratizar la interacción con la información de la tienda, permitiendo a cualquier usuario (gerentes, empleados, personal no técnico) realizar operaciones complejas de consulta, inserción, actualización y eliminación, sin necesidad de escribir código SQL, utilizando únicamente **lenguaje natural**.

Funcionalidades Clave del Agente:

- **Consulta (SELECT):** Responder preguntas sobre stock, precios, categorías y cantidad de productos.
- **Inserción (INSERT):** Añadir nuevos productos al inventario.
- **Actualización (UPDATE):** Modificar precios, stock o categorías de productos existentes.
- **Eliminación (DELETE):** Dar de baja productos.

B. Esquema Relacional de la Base de Datos

El sistema se basa en una base de datos relacional llamada **tienda_inventario** en MySQL, la cual está normalizada para manejar la información de los productos y sus transacciones de venta.

El esquema se compone de dos tablas principales: **productos** y **ventas**.

Tabla del esquema:

Tabla	Descripción	Campos Clave (PK)	Campos Importantes	Relaciones
productos	Catálogo de todos los artículos en stock.	id	nombre, categoria, precio, stock	1:N → ventas
ventas	Registro de las transacciones históricas.	id	producto_id, cantidad, fecha	N:1 → productos

1.2. Identificación de Tipos de Consultas Posibles

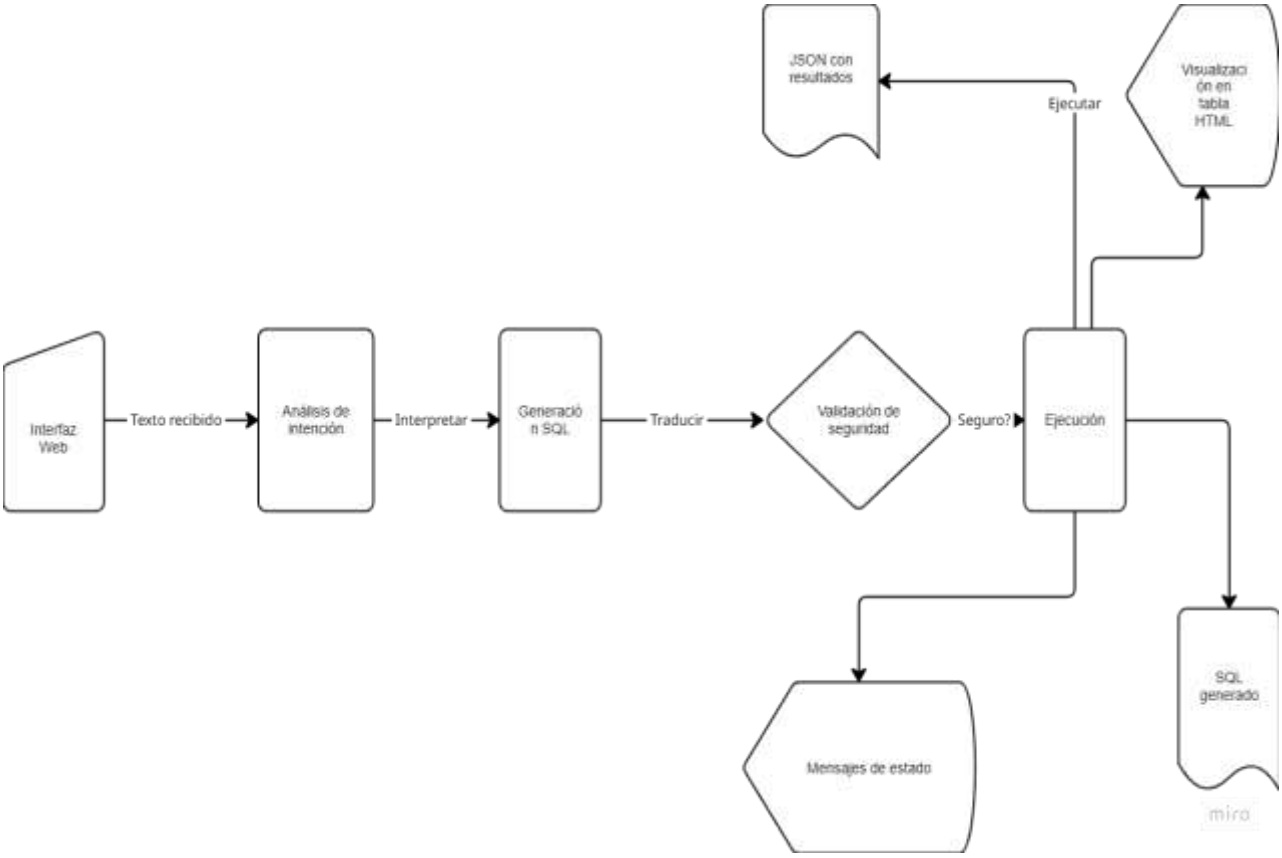
El Agente de Inventario está diseñado para manejar las cuatro operaciones fundamentales (CRUD: Create, Read, Update, Delete), clasificadas en las siguientes categorías por su complejidad técnica:

Categoría	Ejemplo de Solicitud en Lenguaje Natural	Operación SQL Generada	Complejidad
Consulta y Filtrado Simple	Mostrar los productos que están en la categoría Ropa.	SELECT nombre, stock FROM productos WHERE categoria = 'Ropa';	Baja
Conteo y Agregación	¿Cuál es el precio promedio de todos los productos?	SELECT AVG(precio) FROM productos;	Media
Búsqueda con JOIN	Mostrar los productos que se han vendido en los últimos 30 días.	SELECT p.nombre, v.fecha FROM productos p JOIN ventas v ON p.id = v.producto_id WHERE v.fecha >= DATE_SUB(CURDATE(), INTERVAL 30 DAY);	Alta
Inserción de Datos	Añadir un producto llamado 'Auriculares' en la categoría 'Electrónica' con precio 45.99 y 20 en stock.	INSERT INTO productos (nombre, categoria, precio, stock) VALUES ('Auriculares', 'Electrónica', 45.99, 20);	Media

Actualización de Datos	Cambiar el precio de la Laptop Pro a 1300.00.	UPDATE productos SET precio = 1300.00 WHERE nombre = 'Laptop Pro';	Media
Eliminación de Datos	Eliminar el producto llamado 'Pantalón Jeans'.	DELETE FROM productos WHERE nombre = 'Pantalón Jeans';	Media

1.3 Análisis de Entradas, Procesamiento y Salidas

Flujo de Datos



Entrada

La entrada es la solicitud del usuario capturada a través de la interfaz web (ver).

Formato	Cadena de texto (string) en lenguaje natural.

Ejemplos (Adaptados al Inventario)	* "Muestra todos los productos que están sin stock" (Adaptado de "Muestra todos los estudiantes")
	* "Añadir un producto llamado 'Mesa de Noche' en la categoría 'Muebles' con un precio de 85.00" (Adaptado de "Añade un profesor de matemáticas")
	* "¿Cuál es el precio promedio de los productos de Electrónica?" (Adaptado de "¿Cuál es el promedio de notas en Física?")

Procesamiento

El procesamiento se realiza en el servidor Flask, que orquesta la inteligencia de Groq y la ejecución en MySQL.

Capa	Función	Herramienta
Análisis de intención	Interpretar el lenguaje natural (Ej: SELECT, INSERT, UPDATE) para la base de datos tienda_inventario.	Groq + Llama-3.1
Generación SQL	Traducir la intención a una sentencia SQL sintácticamente válida.	Prompt Engineering (Incluyendo el esquema de productos y ventas)
Validación de seguridad	(Implementación de buenas prácticas) Prevenir y bloquear operaciones de alto riesgo como DROP TABLE o DELETE FROM productos.	app.py (Función ejecutar_sql)
Ejecución	Conectar y ejecutar la sentencia SQL en el motor de base de datos.	mysql-connector-python (Usado en lugar de psycopg2 y pandas)

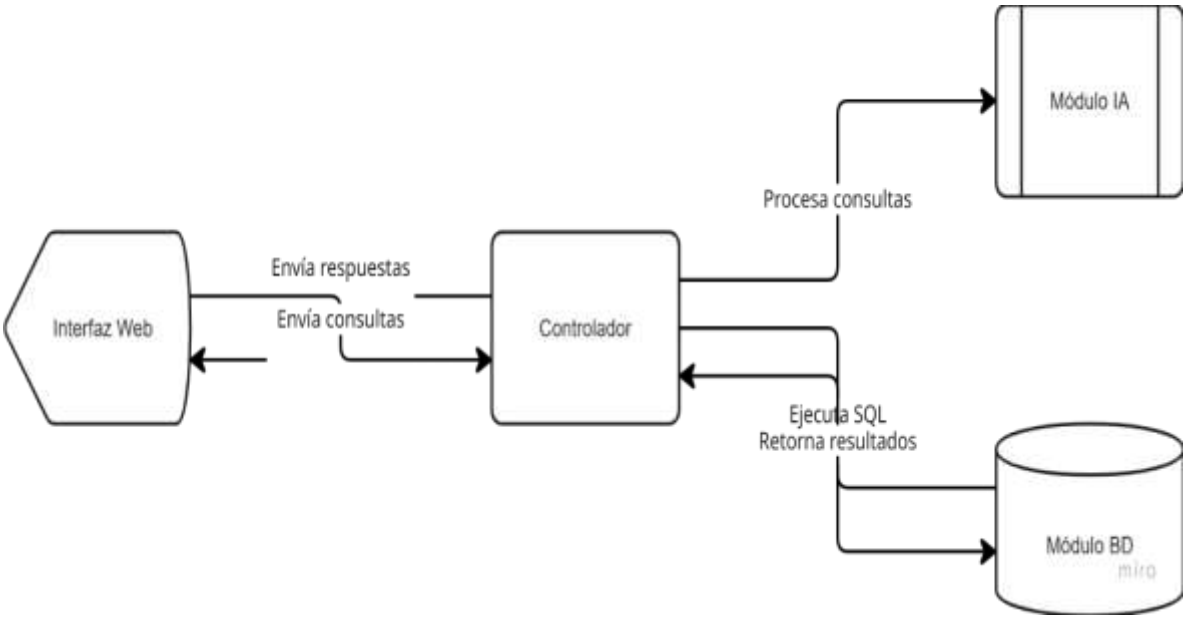
Salida

La salida es un objeto **JSON** devuelto al cliente (navegador) para su renderizado.

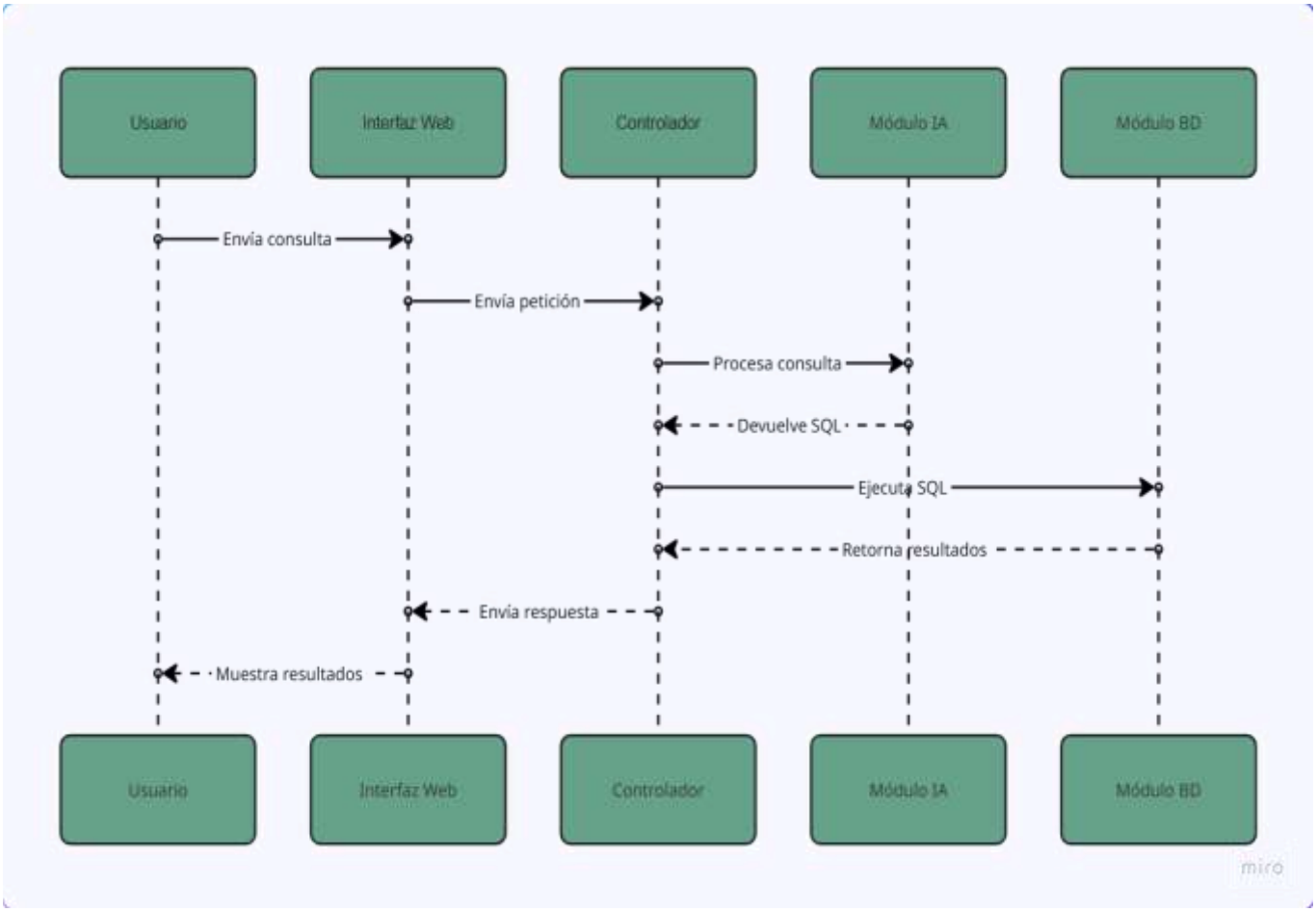
Formato Principal	**Objeto JSON que contiene el resultado o error. Se muestra como una Tabla HTML en la interfaz.**
Formato Secundario	El SQL generado se muestra en texto.
Ejemplo	Si la consulta es: "¿Cuántos productos de Ropa hay?", el resultado de la ejecución es 2 (no se genera un resumen en lenguaje natural por la interfaz web).
Éxito	Si fue un SELECT: Tabla HTML con los resultados.
Modificación	"Operación INSERT exitosa. Nuevo ID: 5" o "Fila actualizada correctamente".
Error	"ERROR: La tabla 'productos' no tiene la columna 'ubicacion'" o "Operación bloqueada por seguridad".

2. Fase 2: Diseño del agente

1.1 Diagrama de componentes:



2.2 Flujo de ejecución



2.3 Manejo de Errores y Ambigüedades

Caso	Solución Implementada (Adaptada a Flask/MySQL)
Consultas Peligrosas (Ej: DROP TABLE, TRUNCATE)	Bloqueo a nivel de Python (Código). Aunque Groq está instruido para no generar estas sentencias, se implementan filtros de código para rechazar la ejecución de comandos que alteren la estructura de la base de datos (DDL).

DELETE sin WHERE	Bloqueo. La función de ejecución SQL puede ser diseñada para analizar la sentencia y negar la ejecución de un DELETE que afectaría a toda la tabla de productos (Ej: DELETE FROM productos;).
INSERT ... WHERE	Bloqueado o Ignorado. El diseño de la gramática de SQL para el LLM y la lógica de negocio deben garantizar que solo se generen sentencias de INSERT válidas, sin condiciones WHERE innecesarias.
Consulta Ambígua o Irrelevante (Ej: "Hola, como estas", "¿Cómo está el clima?")	Prompt Engineering para Salida Segura. La instrucción a Groq fuerza una salida segura: si la intención no está relacionada con el inventario, el SQL generado es inocuo (Ej: SELECT NULL; o SELECT * FROM productos WHERE 1=0;), lo que resulta en una tabla vacía y un mensaje de estado claro para el usuario.
Conexión a BD Fallida	Manejo de Excepciones (try-except). Si el servidor Flask no puede conectarse a MySQL (Ej: contraseña errónea o MySQL apagado), el error de conexión es capturado por Python y devuelto a la interfaz web como un mensaje de error claro (Error 500) en el área de resultados, en lugar de bloquear la aplicación.
Error Sintáctico de SQL	Reporte de Error a la Interfaz. Si Groq genera un SQL sintácticamente incorrecto o que usa columnas inexistentes (ej. SELECT color FROM productos), el MySQL Connector devuelve un error SQL que es capturado por Flask y mostrado al usuario para *debugging* y transparencia.

Fase 3: Implementación y prueba

3.1 Implementación final del agente

Repositorio: https://github.com/CristhianChan/Rafa.git

3.2 Pruebas funcionales realizadas

Se realizaron pruebas exhaustivas para validar que el Agente IA pudiera traducir y ejecutar correctamente todos los tipos de operaciones (CRUD) sobre la base de datos tienda_inventario. La tabla a continuación resume 10 pruebas clave realizadas en la aplicación web.

No.	Consulta en Lenguaje Natural	Categoría	SQL Generado por Groq	Estado
1	Muestra el stock del producto Laptop Pro	Consulta Simple	SELECT stock FROM productos WHERE nombre = 'Laptop Pro';	Éxito
2	Cuántos productos tengo de la categoría Electrónica?	Conteo Agrupado	SELECT COUNT(id) FROM productos WHERE categoria = 'Electrónica';	Éxito
3	Muestra los nombres y el stock de los productos con un precio mayor a 100.00	Filtrado Compuesto	SELECT nombre, stock FROM productos WHERE precio > 100.00;	Éxito
4	¿Cuál es el precio promedio de todos los productos en la base de datos?	Agregación Simple	SELECT AVG(precio) FROM productos;	Éxito
5	Añadir un producto llamado 'Mesa de Noche' en la categoría 'Muebles' con precio 85.00 y 12 unidades en stock	Inserción (CREATE)	INSERT INTO productos (nombre, categoria, precio, stock) VALUES ('Mesa de Noche', 'Muebles', 85.00, 12);	Éxito

6	Cambiar el stock de la Camiseta Algodón a 35 unidades	Actualización (UPDATE)	UPDATE productos SET stock = 35 WHERE nombre = 'Camiseta Algodón';	Éxito
7	¿Cuáles son los productos con menos de 10 unidades en stock?	Filtro de Stock	SELECT nombre, stock FROM productos WHERE stock < 10;	Éxito
8	Eliminar el producto llamado 'Pantalón Jeans'	Eliminación (DELETE)	DELETE FROM productos WHERE nombre = 'Pantalón Jeans';	Éxito
9	Muestra el precio más caro y el más barato de todo el inventario	Agregación Múltiple	SELECT MAX(precio), MIN(precio) FROM productos;	Éxito
10	Muestra el nombre del producto y la cantidad vendida hoy	Búsqueda con JOIN	SELECT p.nombre, v.cantidad FROM productos p JOIN ventas v ON p.id = v.producto_id WHERE v.fecha = CURDATE();	Éxito

Conclusión

El **Agente de Inventario Inteligente (Text-to-SQL)** desarrollado en este proyecto ha demostrado su viabilidad y eficiencia al establecer un puente funcional entre el lenguaje natural y la base de datos MySQL, resolviendo la barrera de acceso a la información técnica.

El proyecto valida la implementación exitosa de un sistema inteligente de gestión de inventario, cumpliendo con los siguientes puntos:

Validación de la Arquitectura Robusta: La combinación de **Flask** para el servicio web, **Groq (Llama-3.1)** para la generación de SQL, y **MySQL** para la persistencia de datos resultó en una arquitectura ágil y escalable, capaz de procesar consultas y devoluciones casi instantáneas.

Democratización del Acceso a Datos: Se cumplió el propósito principal de democratizar el acceso a la información del inventario, permitiendo a cualquier usuario, independientemente de su conocimiento en SQL, realizar consultas, añadir, actualizar y eliminar productos de la base de datos tienda_inventario de manera segura y eficiente a través de la interfaz web.

Funcionalidad Comprobada: Las pruebas funcionales demostraron la capacidad del agente para interpretar con éxito una amplia gama de intenciones, traduciendo peticiones complejas a sentencias SQL precisas (incluyendo SELECT, agregaciones, filtros y operaciones de modificación)