GUIA DOCUMENTO DE REQUISITOS

INFORME PROYECTO:

LINA LIZETH LONDOÑO MARÍN: 2226650
LEANDRO RIVERA RÍOS: 2226651
BALMER VALENCIA BANGUERO: 2227097

CAPÍTULO 2: ELICITACIÓN DE REQUISITOS

DOCENTE:
SANDRA LUCIA GUAÑARITA FERNANDEZ

02/17/2025

TABLA DE CONTENIDO

1 Definir el Contexto:

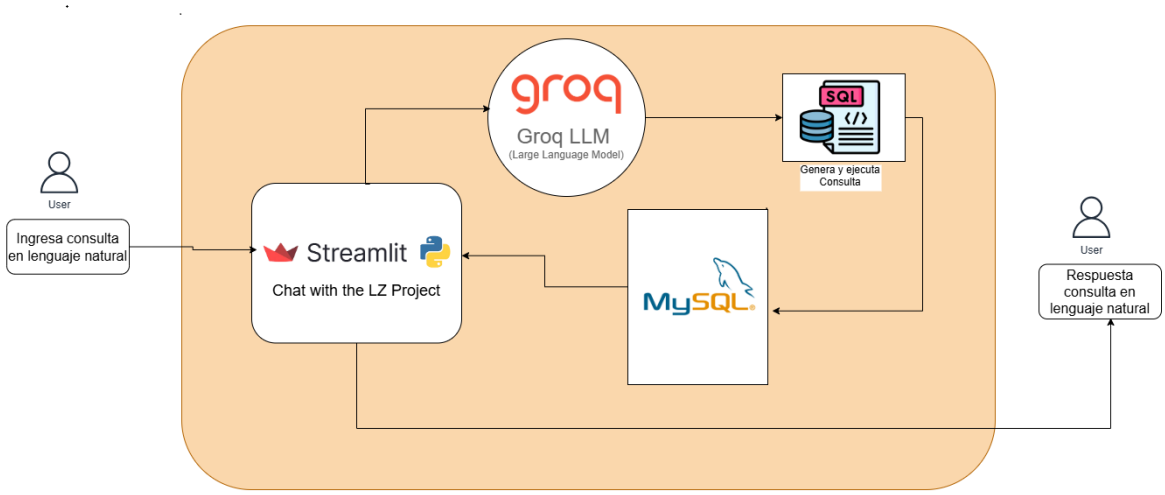
1.1 Usuario: Ingresa consulta en lenguaje natural.

1.2 Streamlit chat con LZ Project: Interactúa con el usuario y envía consultas al modelo de lenguaje y a la base de datos.

1.3 Groq Large Language Model: Procesa las consultas en lenguaje natural y genera consultas SQL.

1. 4 MySQL: Almacena y proporciona datos según las consultas.

Diagrama de Contexto



2 Definir los Contenedores

2.1 Aplicación Web (Streamlit chat con LZ Project):

- Streamlit: GUI interactiva para que el usuario ingrese consultas y vea los resultados.
- Python 3.9+: Lenguaje de programación utilizado para desarrollar la aplicación y manejar la lógica.

2.2 Modelo de Lenguaje (Groq Large Language Model):

- Groq: Motor que entiende las consultas en lenguaje natural y las convierte en consultas SQL.
- Pandas: Librería utilizada para el procesamiento y formato de los datos recibidos.

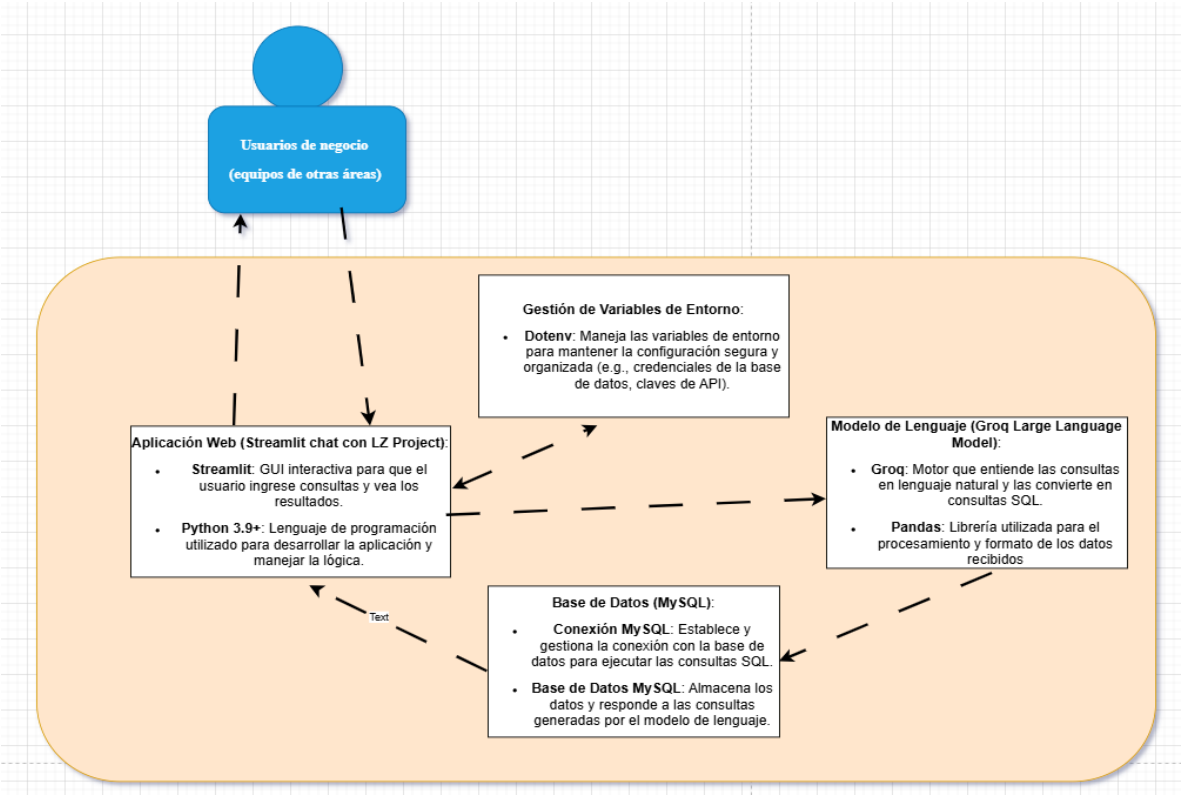
2.3 Base de Datos (MySQL):

- Conexión MySQL: Establece y gestiona la conexión con la base de datos para ejecutar las consultas SQL.
- Base de Datos MySQL: Almacena los datos y responde a las consultas generadas por el modelo de lenguaje.

2.4 Gestión de Variables de Entorno:

- Dotenv: Maneja las variables de entorno para mantener la configuración segura y organizada (e.g., credenciales de la base de datos, claves de API).

Diagrama Contenedores



3 Diagrama de Componentes

3.1 Aplicación Web (Streamlit chat con LZ Project):

- Formulario de Consulta: Interfaz para que el usuario ingrese consultas en lenguaje natural.
- Controlador de Consultas: Lógica que maneja el envío de consultas al modelo de lenguaje y la recepción de respuestas.
- Interfaz de Resultados: Muestra las respuestas de las consultas en lenguaje natural.

3.2 Modelo de Lenguaje (Groq Large Language Model):

- Procesador de Consultas: Componente que transforma las consultas en lenguaje natural en consultas SQL.
- Ejecutor de Consultas: Componente que envía las consultas a MySQL y recibe los resultados.

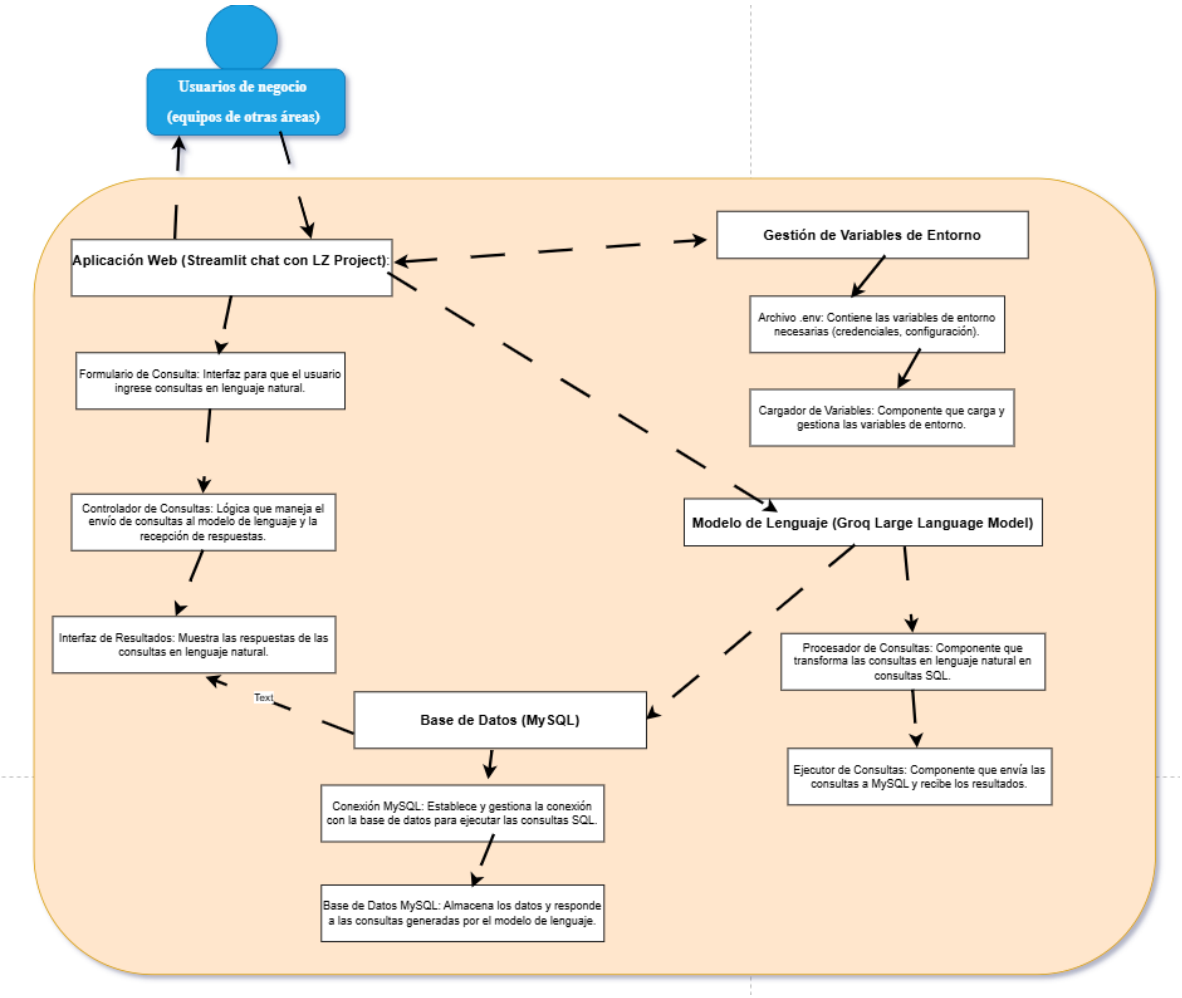
3.3 Base de Datos (MySQL):

- Tablas de Datos: Estructura de almacenamiento que contiene los datos necesarios.
- Manejador de Consultas: Ejecuta las consultas SQL y devuelve los resultados.

3.4 Gestión de Variables de Entorno (Dotenv):

- Archivo .env: Contiene las variables de entorno necesarias (credenciales, configuración).
- Cargador de Variables: Componente que carga y gestiona las variables de entorno.

Diagrama Componentes



4 Detallar el Código

+-------------------------+
| Aplicación Web |
| (Streamlit chat with LZ)|
+-------------------------+
|
| Ingreso de consultas en lenguaje natural
V
	`# Revisamos si la respuesta contiene el campo esperado`
	`content = response.get("choices", [{}])[0].get("messag`
	`print("Contenido original de Groq:", content)  # Para`

+-------------------------+
| Controlador de |
| Consultas |
| |
| +-------------------+ |
| | Formulario de | |
| | Consulta | |
| +-------------------+ |
| +-------------------+ |
	`sql_query = ""`
	`if "SELECT" in content:  # Aseguramos que la respuesta contiene una consulta SQL`
	`    # Encontramos donde comienza la consulta SQL (SELECT, INSERT, etc.)`
	`    start_index = content.find("SELECT")  # Podrías añadir más palabras clave com`
	`    sql_query = content[start_index:].strip()`

| | Interfaz de | |
| | Resultados | |
| +----------+ |
| |
+----------+
| Envía consultas
V

+----------+
| Modelo de Lenguaje
|
| (Groq Large Language
|
| Model) |
+----------+
| Envía consultas
SQL generadas
V

+----------+
| Base de Datos (MySQL)
|
| |
| +----------+ |
| | Conexión MySQL |
|
| +----------+ |
| +----------+ |
| | Base de Datos | |
| +----------+ |
+----------+
| Envía resultados
de la consulta
V

```
with st.sidebar:
    st.subheader("Configuración de la base de datos")
    host = st.text_input("Host", value="localhost")
    port = st.text_input("Puerto", value="3306")
    user = st.text_input("Usuario", value="root")
    password = st.text_input("Contraseña", type="password", value="")
    database = st.text_input("Base de datos", value="chat")
```

+----------+
| Aplicación Web |
| (Streamlit chat with LZ)|
+----------+

*Gestión de
Variables de Entorno*
+----------+
| Archivo .env
|
| +----------+ |
| | Contiene variables de
entorno | |
| | (credenciales,
configuración) | |
| +----------+ |
| +----------+ |
| | Cargador de Variables
de Entorno | |
| | (Dotenv)
| |

```
# Cargar variables de entorno
load_dotenv()

# Obtener la clave de la API de Groq
groq_api_key = os.getenv("GROQ_API_KEY")
print(f"loading graq api key --> {groq_api_key}" )

# URL de la API de Groq
groq_url = "https://api.groq.com/openai/v1/chat/completions"
```

| +---------- ----------+ | +---------- ----------+	

5 de Datos y Tablas

```
CREATE DATABASE  IF NOT EXISTS `chat` /*!40100 DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */;
USE `chat`;

--
-- Table structure for table `album`
--

DROP TABLE IF EXISTS `album`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `album` (
  `AlbumId` int NOT NULL,
  `Title` varchar(160) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,
  `ArtistId` int NOT NULL,
  PRIMARY KEY (`AlbumId`),
  KEY `ArtistId` (`ArtistId`),
  CONSTRAINT `album_ibfk_1` FOREIGN KEY (`ArtistId`) REFERENCES `artist` (`ArtistId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

DROP TABLE IF EXISTS `artist`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `artist` (
  `ArtistId` int NOT NULL,
  `Name` varchar(120) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  PRIMARY KEY (`ArtistId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

DROP TABLE IF EXISTS `customer`;
```

```
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `customer` (
  `CustomerId` int NOT NULL,
  `FirstName` varchar(40) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,
  `LastName` varchar(20) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,
  `Company` varchar(80) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  `Address` varchar(70) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  `City` varchar(40) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  `State` varchar(40) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  `Country` varchar(40) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  `PostalCode` varchar(10) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  `Phone` varchar(24) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  `Fax` varchar(24) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  `Email` varchar(60) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,
  `SupportRepId` int DEFAULT NULL,
  PRIMARY KEY (`CustomerId`),
  KEY `SupportRepId` (`SupportRepId`),
  CONSTRAINT `customer_ibfk_1` FOREIGN KEY (`SupportRepId`) REFERENCES `employee`
(`EmployeeId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```