Fase 2:
Script SQL para la creación de la base de datos y sus tablas
Este script debe ejecutarse dentro de MySQL para generar la estructura de la base de datos:

```
-- Crear la base de datos
CREATE DATABASE IF NOT EXISTS Biblioteca;
USE Biblioteca;

-- Tabla de Autores
CREATE TABLE IF NOT EXISTS autores (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(255) NOT NULL,
    apellido VARCHAR(255) NOT NULL,
    nacionalidad VARCHAR(50) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_spanish_ci;

-- Tabla de Editoriales
CREATE TABLE IF NOT EXISTS editoriales (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(255) NOT NULL,
    direccion VARCHAR(50) DEFAULT NULL,
    telefono VARCHAR(9) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_spanish_ci;

-- Tabla de Libros
CREATE TABLE IF NOT EXISTS libros (
    id INT AUTO_INCREMENT PRIMARY KEY,
    titulo VARCHAR(255) NOT NULL,
    anio DATE DEFAULT NULL,
    id_editorial INT,
    FOREIGN KEY (id_editorial) REFERENCES editoriales(id) ON DELETE SET NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_spanish_ci;

-- Tabla intermedia Autor_Libro (Relación muchos a muchos)
CREATE TABLE IF NOT EXISTS autor_libro (
    id_autor INT,
    id_libro INT,
    PRIMARY KEY (id_autor, id_libro),
    FOREIGN KEY (id_autor) REFERENCES autores(id) ON DELETE CASCADE,
    FOREIGN KEY (id_libro) REFERENCES libros(id) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_spanish_ci;
```

2. Levantar MySQL en Docker
Para garantizar que MySQL se ejecute en un entorno controlado, utilizaremos un contenedor Docker. Esto evita conflictos con instalaciones locales y facilita la configuración.

2.1. Crear un archivo docker-compose.yml
Se recomienda usar docker-compose para definir y administrar el servicio MySQL de manera sencilla.

```
services:
 mysql:
  image: mysql:8.0
  container_name: mysql_db
  environment:
   MYSQL_ROOT_PASSWORD: root
   MYSQL_DATABASE: Biblioteca
   MYSQL_USER: usuario
   MYSQL_PASSWORD: contrasena
  ports:
   - "3306:3306"
  volumes:
   - db_data:/var/lib/mysql

 phpmyadmin:
  image: phpmyadmin/phpmyadmin
  container_name: phpmyadmin
  environment:
   PMA_HOST: mysql_db
   PMA_USER: root
   PMA_PASSWORD: root
  ports:
   - "8085:80"

volumes:
 db_data:
```

Nota:
• Se usa la imagen oficial de MySQL 8.0.
• Se define el usuario usuario con la contraseña clave_segura.
• Se expone el puerto 3306 para conexiones externas.
• Se monta un volumen para persistir los datos.
• Se incluye un archivo init.sql para ejecutar el script de creación de la base de datos automáticamente.

2.2. Crear el archivo init.sql
Este archivo contendrá el script SQL definido anteriormente. Para que el contenedor ejecute el script al iniciar, guárdalo con el mismo contenido del apartado 1.

2.3. Ejecutar Docker Compose
Para iniciar el servicio, se ejecuta:
docker-compose up -d
Esto descargará la imagen de MySQL si no está disponible y creará el contenedor con la configuración indicada.

2.4. Verificar que MySQL está corriendo

Para asegurarse de que el contenedor está activo:

docker ps
Debe mostrar un contenedor con la imagen mysql:8.0 en ejecución.
Para acceder a la base de datos dentro del contenedor:
docker exec -it mysql_biblioteca mysql -u root -p
Luego, se puede listar las bases de datos para verificar que biblioteca existe:
SHOW DATABASES;

3. Conexión desde Python con mysql-connector-python

Ahora que MySQL está en funcionamiento, se puede conectar desde Python para realizar operaciones sobre la base de datos.

3.1. Instalar el conector MySQL para Python

Si aún no está instalado, se debe instalar el paquete:
pip install mysql-connector-python

3.2. Crear una función para conectar a la base de datos

En Python, la conexión se establece con la siguiente función:

```
import mysql.connector
def conectar_bd():
try:
conexion = mysql.connector.connect(
host="localhost",
user="usuario",
password="clave_segura",
database="biblioteca"
)
print("Conexión exitosa a la base de datos")
return conexion
except mysql.connector.Error as err:
print(f"Error: {err}")
return None
```

3.3. Probar la conexión

Para asegurarse de que la conexión es exitosa, se puede ejecutar:

```
if __name__ == "__main__":
conn = conectar_bd()
if conn:
conn.close()
```

Si la conexión es exitosa, se imprimirá:

Implementación de MySQL en Docker con docker-compose.
Verificación de la base de datos mediante acceso al contenedor.
Conexión desde Python usando mysql-connector-python.
Esto proporciona la base de datos funcional para la siguiente fase del desarrollo del proyecto.