MYSQL

1. Introducción

Kivy es un framework de Python que permite desarrollar aplicaciones multiplataforma con interfaces gráficas atractivas.
Por su parte, MySQL es un sistema de gestión de bases de datos relacional ampliamente usado para almacenar y manejar información estructurada.

Conectar ambos permite crear aplicaciones que no solo muestren interfaces visuales, sino que también almacenen, consulten y gestionen datos de manera persistente.

2. Requisitos previos

- **Python 3.x**
- **Kivy**
- **Conector MySQL para Python**
- a**Servidor MySQL** en funcionamiento, con una base de datos creada.

3. Estructura general del proyecto

kivy_mysql_app/

|— main.py

└— database.py

4. Código del módulo de conexión (database.py)

```
import mysql.connector
from mysql.connector import Error
class Database:
  def __init__(self):
    try:
      self.connection = mysql.connector.connect(
        host='localhost',
        user='root',
        password='tu_contraseña',
        database='ejemplo_kivy'
      )
      if self.connection.is_connected():
        print("☑ Conexión exitosa a MySQL")
    except Error as e:
      print(f"❌ Error al conectar a MySQL: {e}")
  def insertar_usuario(self, nombre, correo):
    try:
      cursor = self.connection.cursor()
      query = "INSERT INTO usuarios (nombre, correo) VALUES (%s,
%s)"
      cursor.execute(query, (nombre, correo))
      self.connection.commit()
      print("☑ Registro insertado correctamente.")
    except Error as e:
      print(f"❌ Error al insertar datos: {e}")
  def obtener_usuarios(self):
    try:
      cursor = self.connection.cursor()
      cursor.execute("SELECT * FROM usuarios")
      return cursor.fetchall()
    except Error as e:  print(f"❌ Error al obtener datos: {e}") return []
```

5. Base de datos de ejemplo

Ejecuta en MySQL:

```
CREATE DATABASE ejemplo_kivy;
USE ejemplo_kivy;

CREATE TABLE usuarios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100),
  correo VARCHAR(100)
);
```

6. Código principal de la app (main.py)

```
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.label import Label
from kivy.uix.textinput import TextInput
from kivy.uix.button import Button
from database import Database

class MyApp(BoxLayout):
  def __init__(self, **kwargs):
    super().__init__(orientation='vertical', **kwargs)
    self.db = Database()

    self.add_widget(Label(text="Nombre:"))
    self.nombre_input = TextInput()
    self.add_widget(self.nombre_input)

    self.add_widget(Label(text="Correo:"))
    self.correo_input = TextInput()
    self.add_widget(self.correo_input)
```

```
        self.boton_insertar = Button(text="Guardar en BD")
        self.boton_insertar.bind(on_press=self.guardar_usuario)
        self.add_widget(self.boton_insertar)

        self.boton_mostrar = Button(text="Mostrar usuarios")
        self.boton_mostrar.bind(on_press=self.mostrar_usuarios)
        self.add_widget(self.boton_mostrar)

        self.resultado = Label(text="")
        self.add_widget(self.resultado)

    def guardar_usuario(self, instance):
        nombre = self.nombre_input.text
        correo = self.correo_input.text
        self.db.insertar_usuario(nombre, correo)
        self.resultado.text = f"Usuario {nombre} agregado."

    def mostrar_usuarios(self, instance):
        usuarios = self.db.obtener_usuarios()
        texto = "\n".join([f"{u[1]} - {u[2]}" for u in usuarios])
        self.resultado.text = texto if texto else "No hay usuarios registrados."

class KivyMySQLApp(App):
    def build(self):
        return MyApp()

if __name__ == '__main__':
    KivyMySQLApp().run()
```

7. Explicación del funcionamiento

1. **database.py:**
   Contiene una clase Database que maneja la conexión y las operaciones básicas (insertar y consultar).
2. **main.py:**
   Crea una interfaz con BoxLayout, campos de texto para nombre y correo, y botones para insertar o mostrar registros.
3. **Cuando el usuario pulsa "Guardar en BD", los datos se envían al método insertar_usuario() del módulo Database.**
4. **Al pulsar "Mostrar usuarios", se consultan todos los registros y se muestran en la etiqueta inferior.**

8. Conclusión

Conectar Kivy con MySQL permite crear aplicaciones gráficas robustas y dinámicas que gestionan información en tiempo real. Este tipo de conexión es ideal para proyectos que requieren persistencia de datos, como sistemas de registro, control de inventario o aplicaciones administrativas.