Tecnicatura Superior en Telecomunicaciones

Modulo: Programador

PROYECTO INTEGRADOR 2 Cuatrimestre

Entrega N.º 2

Alumnos:

- Zalazar, Joaquin
- Márquez, José
- Durigutti Vittorio

Profesor: Ing. Lanfranco Lisandro

Fecha: 04/10/2024

Entrega N°2: Creación de DB y programa en Python.

A continuación, se presenta un resumen de las tablas necesarias para el funcionamiento del proyecto, junto con una explicación sobre el diagrama de entidad-relación (ER) que podría derivarse de ellas.

Resumen de las Tablas:

1. Tabla **Cliente**

- **Descripción**: Almacena información sobre los clientes.
- **Columnas**:
  - dni_cliente: Clave primaria, identificador único del cliente.
  - nombre: Nombre del cliente.
  - email: Correo electrónico del cliente.
  - telefono: Número de teléfono del cliente.
  - direccion_contacto: Dirección de contacto del cliente.

2. Tabla **Controlador**

- **Descripción**: Almacena información sobre dispositivos controladores asociados a los clientes.
- **Columnas**:
  - id_dispositivo: Clave primaria autoincremental.
  - mac_dispositivo: Dirección MAC única.
  - nombre_dispositivo: Nombre del dispositivo.
  - dni_cliente: Clave foránea que relaciona el controlador con un cliente.
  - ubicacion_dispositivo: Ubicación física del dispositivo.

3. Tabla **Habitacion**

- **Descripción**: Almacena información sobre las habitaciones donde están instalados los dispositivos.
- **Columnas**:
  - id_habitacion: Clave primaria autoincremental.
  - nombre_habitacion: Nombre de la habitación.
  - id_dispositivo: Clave foránea que relaciona la habitación con un controlador.

4. Tabla **Sensor_Actuador**

- **Descripción**: Almacena información sobre sensores y actuadores en las habitaciones.
- **Columnas**:
  - id_sensor_actuador: Clave primaria autoincremental.

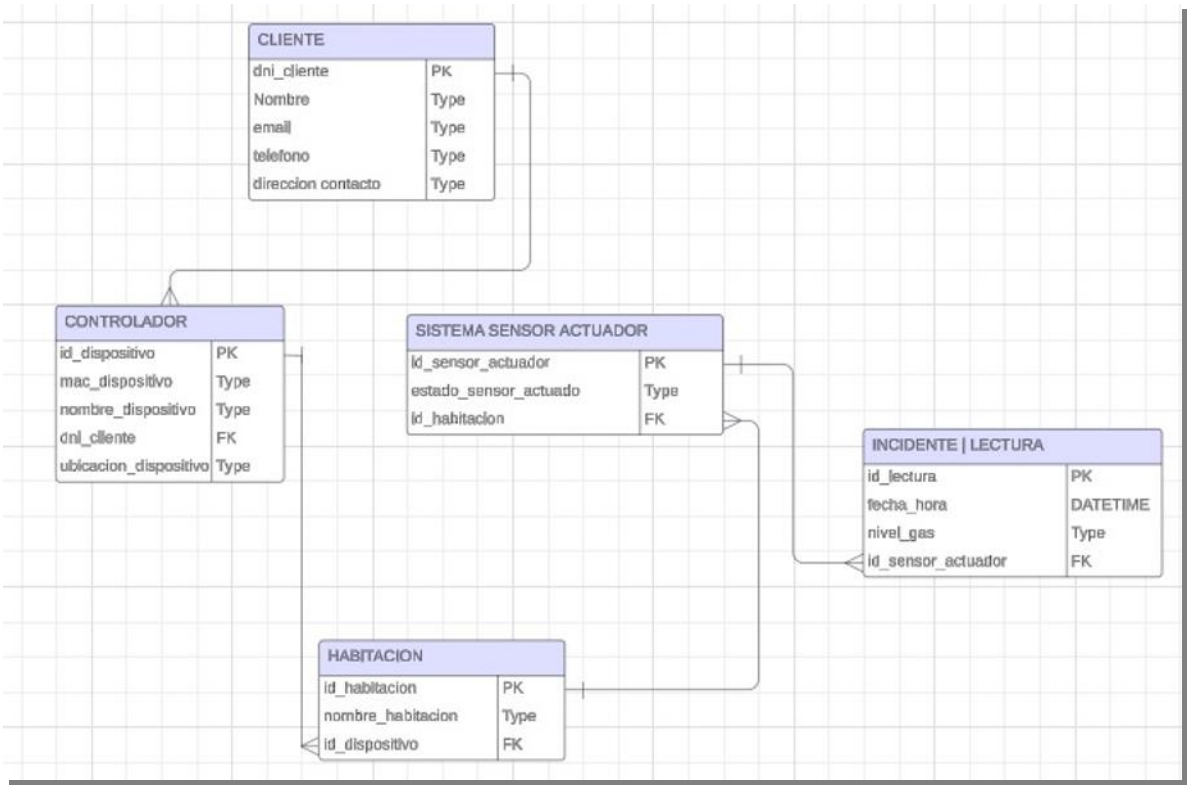
- estado_sensor_actuador: Estado del sensor/actuador (activo, inactivo, en mantenimiento).
- id_habitacion: Clave foránea que relaciona el sensor/actuador con una habitación.

5. Tabla **Lectura**

- **Descripción**: Almacena lecturas de gas realizadas por los sensores/actuadores.
- **Columnas**:
  - id_lectura: Clave primaria autoincremental.
  - fecha_hora: Fecha y hora de la lectura (no nula).
  - nivel_gas: Nivel de gas detectado (no nulo).
  - id_sensor_actuador: Clave foránea que relaciona la lectura con un sensor/actuador.

Diagrama de Entidad/Relación (ER)

A continuación se presenta el diagrama de las entidades:



Relaciones,

1. **Cliente a Controlador**:

   - Relación uno a muchos (1:N): Un cliente puede tener múltiples controladores. Esto

se representa en el modelo ER mediante una línea que conecta la entidad Cliente con Controlador, indicando que un cliente puede estar asociado a varios dispositivos.

2. **Controlador a Habitacion**:

    - Relación uno a muchos (1:N): Un controlador puede estar asociado a múltiples habitaciones. En el modelo ER, se conecta Controlador con Habitacion.

3. **Habitacion a Sensor_Actuador**:

    - Relación uno a muchos (1:N): Una habitación puede tener múltiples sensores/actuadores. Esto se representa conectando Habitacion con Sensor_Actuador.

4. **Sensor_Actuador a Lectura**:

    - Relación uno a muchos (1:N): Un sensor/actuador puede generar múltiples lecturas. En el modelo ER, se conecta Sensor_Actuador con Lectura.

PROGRAMACIÓN:

Se presenta el código fuente del programa elaborado para el proyecto:

```
# Programa en Python
# Programacion -
# Proyecto 2

# EJECUCION DEL PROGRAMA: python menu_mysql.py

'''
El programa consta de 4 partes fundamentales.
- (LINEA 16) Las librerias funcionales para poder conectar con la base de datos.
-  La funcion conectar con la que establecemos la conexion con la base de datos en el entorno local
   - (LINEA 24) Funciones con las que interactuar con la conexion
   - (LINEA 47) Función para cerrar la conexión
- Las funciones con las que interacutar con la base de datos
   - Funciones para agregar datos a la base de datos:
      - (LINEA 70) Función para insertar un cliente en la base de datos
      - (LINEA 97) Funcion para insertar dispositivo
      - (LINEA 116) Funcion para insertar sensor_actuador
   - Funciones para modificar datos a la base de datos:
      - (LINEA 143) Función para modificar un cliente
      - (LINEA 165) Función para cambiar el nombre de la habitación de un actuador
   - Funciones para eliminar datos a la base de datos:
      - (LINEA 188) Función para dar de baja un cliente (y eliminar todos sus dispositivos y sensores)
```

```
    - Funciones para buscar datos a la base de datos:
        - (LINEA 228) Función para buscar un cliente por DNI y mostrar sus datos
        - (LINEA 246) Función para mostrar todos los controladores de un cliente
        - (LINEA 268) Función para mostrar lecturas de un sensor específico
        - (LINEA 289) Función para buscar por DNI del cliente, sensores, ubicación y habitación
        - (LINEA 317) Función para recibir una alarma y mostrar información del sensor
- (LINEA 366) Menu interactivo, mediante la terminal, con el que realizar las solicitudes, asi como la conexion.

'''

import mysql.connector
from mysql.connector import Error
import getpass  # Para solicitar la contraseña de forma segura

#------------------------------------------------------------------#
#------------------------------------------------------------------#

# Funciones con las que interactuar con la conexion

# Función para conectar a la base de datos
def conectar(password):
    try:
        connection = mysql.connector.connect(
            host='localhost',       # Cambia esto si es necesario
            database='GASDETECTOR', # Tu base de datos
            user='root',            # Usuario de MySQL
            password=password       # Contraseña ingresada por el usuario
        )
        if connection.is_connected():
            print("Conexión exitosa a la base de datos")
            return connection
    except Error as e:
        print(f"Error al conectar a la base de datos: {e}")
        return None


# Función para cerrar la conexión
def cerrar_conexion(connection):
    if connection.is_connected():
        connection.close()
        print("Conexión cerrada")

#------------------------------------------------------------------#
#------------------------------------------------------------------#

# Funciones con las que interactuar con la base de datos
```

```
#-------------------------------------------------------------#
#-------------------------------------------------------------#

# A - Funciones para agregar datos a la base de datos:

# 1 - Función para insertar un cliente en la base de datos
def insertar_cliente(connection):
    cursor = connection.cursor()
    try:
        dni = input("Introduce el DNI del cliente: ")
        nombre = input("Introduce el nombre del cliente: ")
        email = input("Introduce el email del cliente: ")
        telefono = input("Introduce el teléfono del cliente: ")
        direccion = input("Introduce la dirección del cliente: ")
        query = """INSERT INTO Cliente (dni_cliente, nombre, email, telefono, direccion_contacto)
                VALUES (%s, %s, %s, %s, %s)"""
        cursor.execute(query, (dni, nombre, email, telefono, direccion))
        connection.commit()
        print("Cliente insertado correctamente")

        # Preguntar si quiere agregar un dispositivo para este cliente
        while input("¿Desea agregar un dispositivo para este cliente? (s/n): ").lower() == 's':
            insertar_dispositivo(connection, dni)

        # Preguntar si quiere agregar sensores para este cliente
        while input("¿Desea agregar un sensor/actuador para este cliente? (s/n): ").lower() == 's':
            insertar_sensor(connection)
    except Error as e:
        print(f"Error al insertar cliente: {e}")
    finally:
        cursor.close()

# 2 - Funcion para insertar dispositivo
def insertar_dispositivo(connection, dni_cliente=None):
    cursor = connection.cursor()
    try:
        if not dni_cliente:
            dni_cliente = input("Introduce el DNI del cliente para asociar el dispositivo: ")
        mac = input("Introduce la MAC del dispositivo: ")
        nombre = input("Introduce el nombre del dispositivo: ")
        ubicacion = input("Introduce la ubicación del dispositivo: ")
        query = """INSERT INTO Controlador (mac_dispositivo, nombre_dispositivo, dni_cliente,
ubicacion_dispositivo)
                VALUES (%s, %s, %s, %s)"""
        cursor.execute(query, (mac, nombre, dni_cliente, ubicacion))
        connection.commit()
        print("Dispositivo insertado correctamente")
    except Error as e:
```

```
        print(f"Error al insertar dispositivo: {e}")
    finally:
        cursor.close()

# 3 - Funcion para insertar sensor_actuador
def insertar_sensor(connection):
    cursor = connection.cursor()
    try:
        id_dispositivo = input("Introduce el ID del dispositivo al que asociar el sensor: ")
        nombre_habitacion = input("Introduce el nombre de la habitación: ")
        query_hab = "INSERT INTO Habitacion (nombre_habitacion, id_dispositivo) VALUES (%s, %s)"
        cursor.execute(query_hab, (nombre_habitacion, id_dispositivo))
        connection.commit()
        id_habitacion = cursor.lastrowid  # Obtener el ID de la habitación recién creada

        query_sensor = "INSERT INTO Sensor_Actuador (estado_sensor_actuador, id_habitacion)
VALUES ('activo', %s)"
        cursor.execute(query_sensor, (id_habitacion,))
        connection.commit()
        print("Sensor/Actuador insertado correctamente")
    except Error as e:
        print(f"Error al insertar sensor: {e}")
    finally:
        cursor.close()

#-------------------------------------------------------------#
#-------------------------------------------------------------#

# B - Funciones para modificar datos a la base de datos:

# 4 - Función para modificar un cliente
def modificar_cliente(connection):
    cursor = connection.cursor()
    try:
        dni = input("Introduce el DNI del cliente a modificar: ")
        telefono = input("Introduce el nuevo teléfono (o deja vacío si no deseas cambiarlo): ")
        email = input("Introduce el nuevo email (o deja vacío si no deseas cambiarlo): ")
        direccion = input("Introduce la nueva dirección (o deja vacío si no deseas cambiarla): ")

        if telefono:
            cursor.execute("UPDATE Cliente SET telefono = %s WHERE dni_cliente = %s", (telefono, dni))
        if email:
            cursor.execute("UPDATE Cliente SET email = %s WHERE dni_cliente = %s", (email, dni))
        if direccion:
            cursor.execute("UPDATE Cliente SET direccion_contacto = %s WHERE dni_cliente = %s",
(direccion, dni))
        connection.commit()
        print("Datos del cliente actualizados correctamente")
```

```
    except Error as e:
        print(f"Error al modificar cliente: {e}")
    finally:
        cursor.close()

# 5- Función para cambiar el nombre de la habitación de un actuador
def modificar_habitacion(connection):
    cursor = connection.cursor()
    try:
        id_sensor = input("Introduce el ID del sensor/actuador a modificar: ")
        nuevo_nombre_habitacion = input("Introduce el nuevo nombre de la habitación: ")
        query = """UPDATE Habitacion
                SET nombre_habitacion = %s
                WHERE id_habitacion = (SELECT id_habitacion FROM Sensor_Actuador WHERE
id_sensor_actuador = %s)"""
        cursor.execute(query, (nuevo_nombre_habitacion, id_sensor))
        connection.commit()
        print("Nombre de la habitación actualizado correctamente")
    except Error as e:
        print(f"Error al modificar habitación: {e}")
    finally:
        cursor.close()

#-------------------------------------------------------------#
#-------------------------------------------------------------#

# C - Funciones para eliminar datos a la base de datos:

# 6 - Función para dar de baja un cliente (y eliminar todos sus dispositivos y sensores)
def eliminar_cliente(connection):
    cursor = connection.cursor()
    try:
        dni = input("Introduce el DNI del cliente a eliminar: ")

        # Eliminar sensores/actuadores asociados
        cursor.execute("""
            DELETE FROM Sensor_Actuador WHERE id_habitacion IN
            (SELECT id_habitacion FROM Habitacion WHERE id_dispositivo IN
            (SELECT id_dispositivo FROM Controlador WHERE dni_cliente = %s))""", (dni,))

        # Eliminar habitaciones asociadas
        cursor.execute("""
            DELETE FROM Habitacion WHERE id_dispositivo IN
            (SELECT id_dispositivo FROM Controlador WHERE dni_cliente = %s)""", (dni,))

        # Eliminar controladores asociados
        cursor.execute("DELETE FROM Controlador WHERE dni_cliente = %s", (dni,))
```

```
        # Eliminar el cliente
        cursor.execute("DELETE FROM Cliente WHERE dni_cliente = %s", (dni,))

        connection.commit()
        print("Cliente y todos sus dispositivos y sensores eliminados correctamente")
    except Error as e:
        print(f"Error al eliminar cliente: {e}")
    finally:
        cursor.close()

#-------------------------------------------------------------#
#-------------------------------------------------------------#

# D - Funciones para buscar datos a la base de datos:

# 7 - Función para buscar un cliente por DNI y mostrar sus datos
def buscar_cliente(connection):
    cursor = connection.cursor()
    try:
        dni = input("Introduce el DNI del cliente a buscar: ")
        query = "SELECT * FROM Cliente WHERE dni_cliente = %s"
        cursor.execute(query, (dni,))
        result = cursor.fetchone()
        if result:
            print("Cliente encontrado:", result)
        else:
            print("Cliente no encontrado.")
    except Error as e:
        print(f"Error al buscar cliente: {e}")
    finally:
        cursor.close()

# 8 - Función para mostrar todos los controladores de un cliente
def mostrar_controladores_cliente(connection):
    cursor = connection.cursor()
    try:
        dni = input("Introduce el DNI del cliente: ")
        query = """SELECT Controlador.id_dispositivo, Controlador.nombre_dispositivo,
Controlador.ubicacion_dispositivo
                FROM Controlador
                INNER JOIN Cliente ON Controlador.dni_cliente = Cliente.dni_cliente
                WHERE Cliente.dni_cliente = %s"""
        cursor.execute(query, (dni,))
        controladores = cursor.fetchall()
        if controladores:
            print("Controladores del cliente:")
            for controlador in controladores:
                print(controlador)
```

```
        else:
            print("No se encontraron controladores para este cliente.")
    except Error as e:
        print(f"Error al mostrar controladores: {e}")
    finally:
        cursor.close()

# 9 - Función para mostrar lecturas de un sensor específico
def mostrar_lecturas_sensor(connection):
    cursor = connection.cursor()
    try:
        id_sensor = input("Introduce el ID del sensor: ")
        query = """SELECT fecha_hora, nivel_gas
                FROM Lectura
                WHERE id_sensor_actuador = %s"""
        cursor.execute(query, (id_sensor,))
        lecturas = cursor.fetchall()
        if lecturas:
            print(f"Lecturas del sensor {id_sensor}:")
            for lectura in lecturas:
                print(lectura)
        else:
            print("No se encontraron lecturas para este sensor.")
    except Error as e:
        print(f"Error al mostrar lecturas: {e}")
    finally:
        cursor.close()

# 10 - Función para buscar por DNI del cliente, sensores, ubicación y habitación
def buscar_sensores_por_cliente(connection):
    cursor = connection.cursor()
    try:
        dni = input("Introduce el DNI del cliente: ")
        query = """
            SELECT Cliente.nombre, Controlador.ubicacion_dispositivo, Habitacion.nombre_habitacion,
Sensor_Actuador.id_sensor_actuador
            FROM Sensor_Actuador
            INNER JOIN Habitacion ON Sensor_Actuador.id_habitacion = Habitacion.id_habitacion
            INNER JOIN Controlador ON Habitacion.id_dispositivo = Controlador.id_dispositivo
            INNER JOIN Cliente ON Controlador.dni_cliente = Cliente.dni_cliente
            WHERE Cliente.dni_cliente = %s
        """
        cursor.execute(query, (dni,))
        sensores = cursor.fetchall()

        if sensores:
            print(f"Sensores asociados al cliente {dni}:")
            for sensor in sensores:
```

```
            print(f"Nombre Cliente: {sensor[0]}, Dirección: {sensor[1]}, Habitación: {sensor[2]}, ID
Sensor: {sensor[3]}")
        else:
            print("No se encontraron sensores para este cliente.")
    except Error as e:
        print(f"Error al buscar sensores por cliente: {e}")
    finally:
        cursor.close()

# 11 - Función para recibir una alarma y mostrar información del sensor
def recibir_alarma(connection):
    cursor = connection.cursor()
    try:
        # Recibir datos de la medición
        id_sensor = input("Introduce el ID del sensor: ")
        nivel_gas = float(input("Introduce el nivel de gas detectado: "))

        # Obtener información del sensor, cliente, ubicación y habitación
        query = """
            SELECT Cliente.nombre, Cliente.direccion_contacto, Controlador.ubicacion_dispositivo,
Habitacion.nombre_habitacion
            FROM Sensor_Actuador
            INNER JOIN Habitacion ON Sensor_Actuador.id_habitacion = Habitacion.id_habitacion
            INNER JOIN Controlador ON Habitacion.id_dispositivo = Controlador.id_dispositivo
            INNER JOIN Cliente ON Controlador.dni_cliente = Cliente.dni_cliente
            WHERE Sensor_Actuador.id_sensor_actuador = %s
        """
        cursor.execute(query, (id_sensor,))
        resultado = cursor.fetchone()

        if resultado:
            # Registrar la medición en la tabla de Lectura
            query_insert = """
                INSERT INTO Lectura (fecha_hora, nivel_gas, id_sensor_actuador)
                VALUES (NOW(), %s, %s)
            """
            cursor.execute(query_insert, (nivel_gas, id_sensor))
            connection.commit()

            print("\n--- Alarma de Gas ---")
            print(f"Cliente: {resultado[0]}")
            print(f"Domicilio: {resultado[1]}")
            print(f"Ubicación Dispositivo: {resultado[2]}")
            print(f"Habitación: {resultado[3]}")
            print(f"Nivel de Gas Detectado: {nivel_gas}")
        else:
            print("No se encontró información para este sensor.")
    except Error as e:
```

```
        print(f"Error al recibir alarma: {e}")
    finally:
        cursor.close()


#---------------------------------------------------------------#
#---------------------------------------------------------------#

# Funciones relacionadas con el menu interactivo

# Menú interactivo
def menu():
    # Solicitar la contraseña al usuario de forma segura
    password = getpass.getpass("Introduce la contraseña de la base de datos: ")
    connection = conectar(password)

    if connection:
        while True:
            print("\n--- Menú de Opciones ---")
            print("1. Insertar cliente")
            print("2. Insertar dispositivo")
            print("3. Insertar sensor/actuador")
            print("4. Modificar datos de un cliente")
            print("5. Cambiar nombre de la habitación de un actuador")
            print("6. Dar de baja un cliente")
            print("7. Buscar cliente por DNI")
            print("8. Mostrar controladores de un cliente")
            print("9. Mostrar lecturas de un sensor")
            print("10. Salir")

            opcion = input("Selecciona una opción: ")

            if opcion == '1':
                insertar_cliente(connection)
            elif opcion == '2':
                insertar_dispositivo(connection)
            elif opcion == '3':
                insertar_sensor(connection)
            elif opcion == '4':
                modificar_cliente(connection)
            elif opcion == '5':
                modificar_habitacion(connection)
            elif opcion == '6':
                eliminar_cliente(connection)
            elif opcion == '7':
                buscar_cliente(connection)
            elif opcion == '8':
                mostrar_controladores_cliente(connection)
```

```
            elif opcion == '9':
                mostrar_lecturas_sensor(connection)
            elif opcion == '10':
                cerrar_conexion(connection)
                break
            else:
                print("Opción no válida. Inténtalo de nuevo.")
        else:
            print("No se pudo conectar a la base de datos.")

# Ejecutar el menú
if __name__ == "__main__":
    menu()
```