Universidad Tecnológica de Panamá

Facultad de Sistemas Computacionales

Asignatura: Programación I

Taller Práctico3

Profesor: Napoleón Ibarra

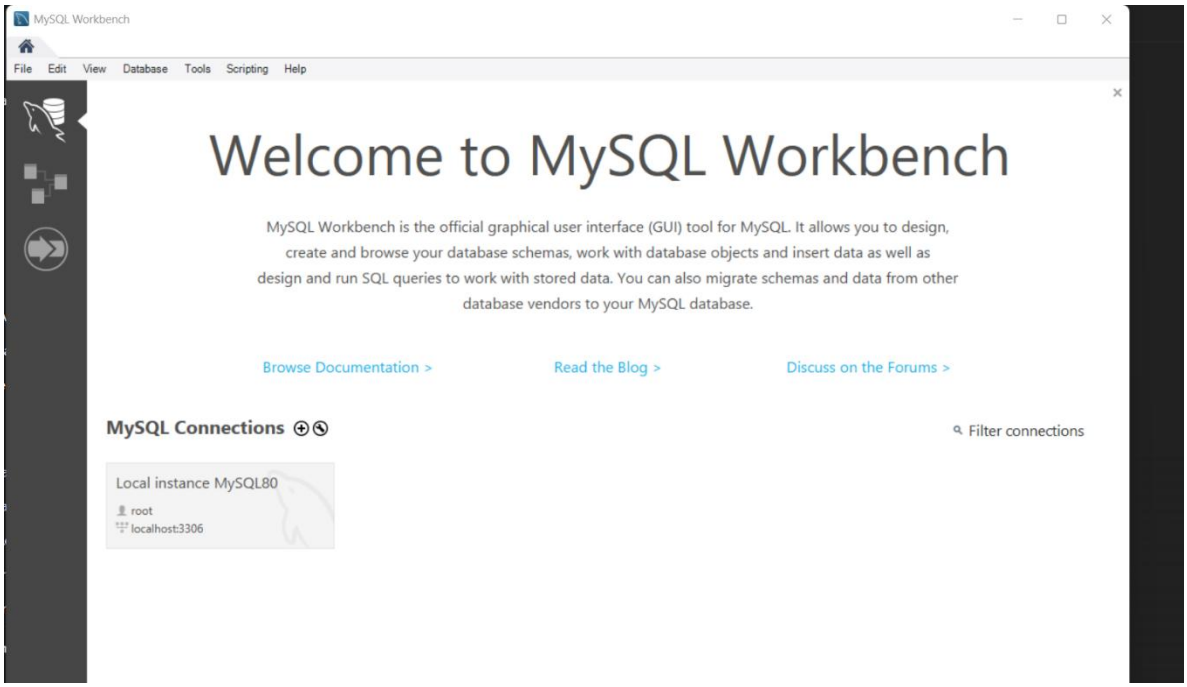
Valor: 100 puntos

Estudiante: Angeline Urriola & Daniela Sanchez

Procedimiento:

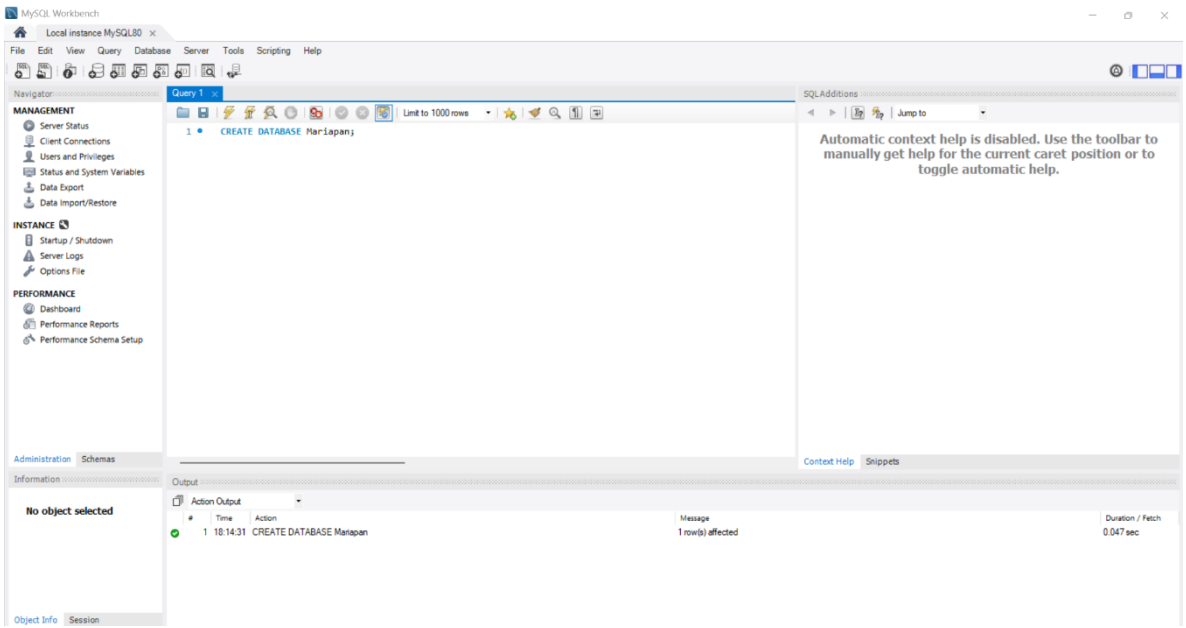
De manera individual o en grupo de 2 personas confeccione la asignación. Utilice la herramienta Internet como apoyo para realizar su proceso de investigación.

Una vez culminada la actividad, subirla a la plataforma Moodle (PDF)y presentar su debida sustentación.



1. Creación de la conexión local en MySQL Workbench

En la imagen se muestra la pantalla principal de MySQL Workbench, donde se configuró la conexión local “Local instance MySQL80” utilizando el usuario root y el puerto 3306. Esta conexión permitió acceder al servidor MySQL instalado en la máquina local para la creación de la base de datos del proyecto.

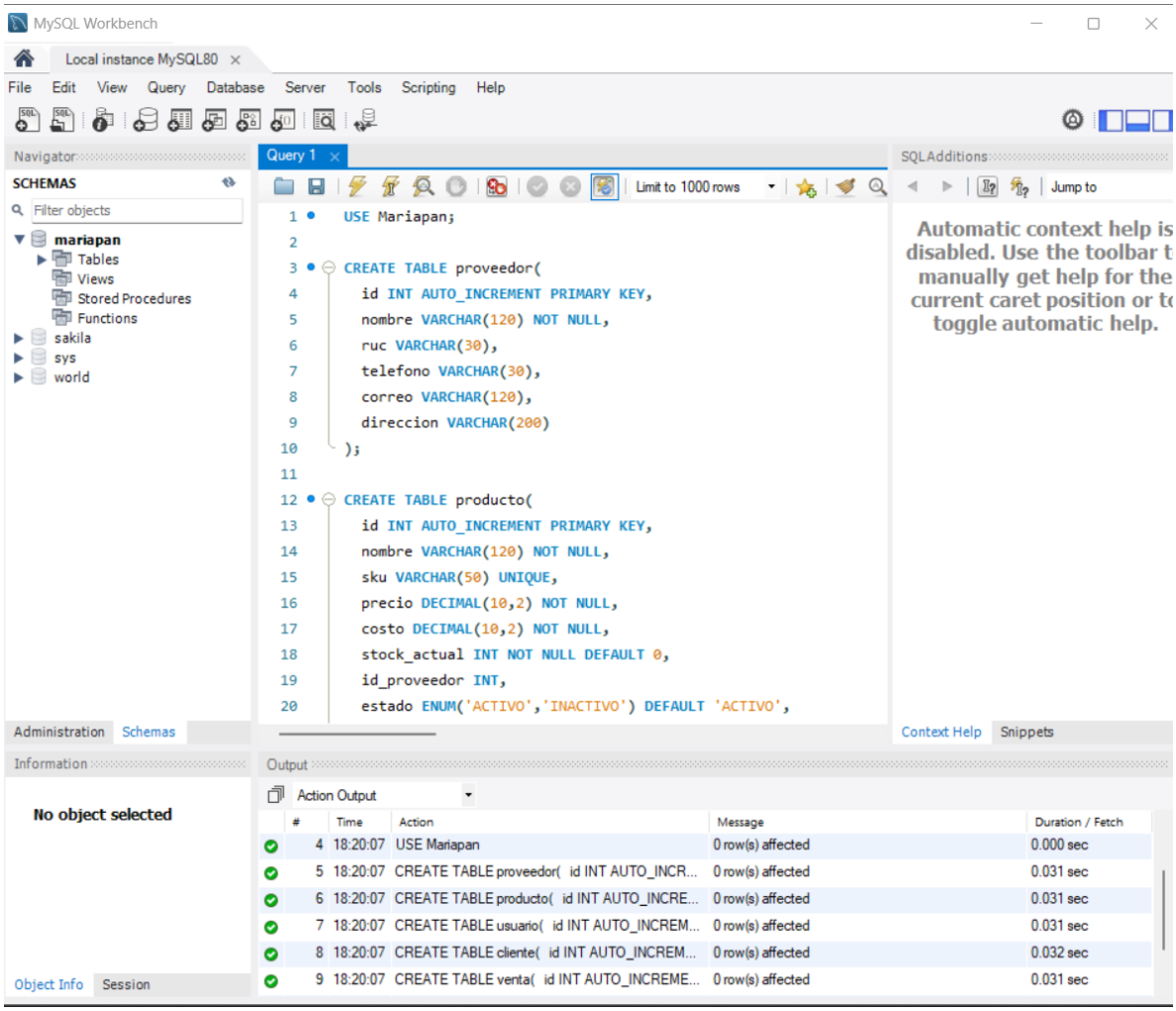


2. Creación de la base de datos “Mariapan”

Una vez establecida la conexión, se ejecutó el comando: CREATE DATABASE mariapan;

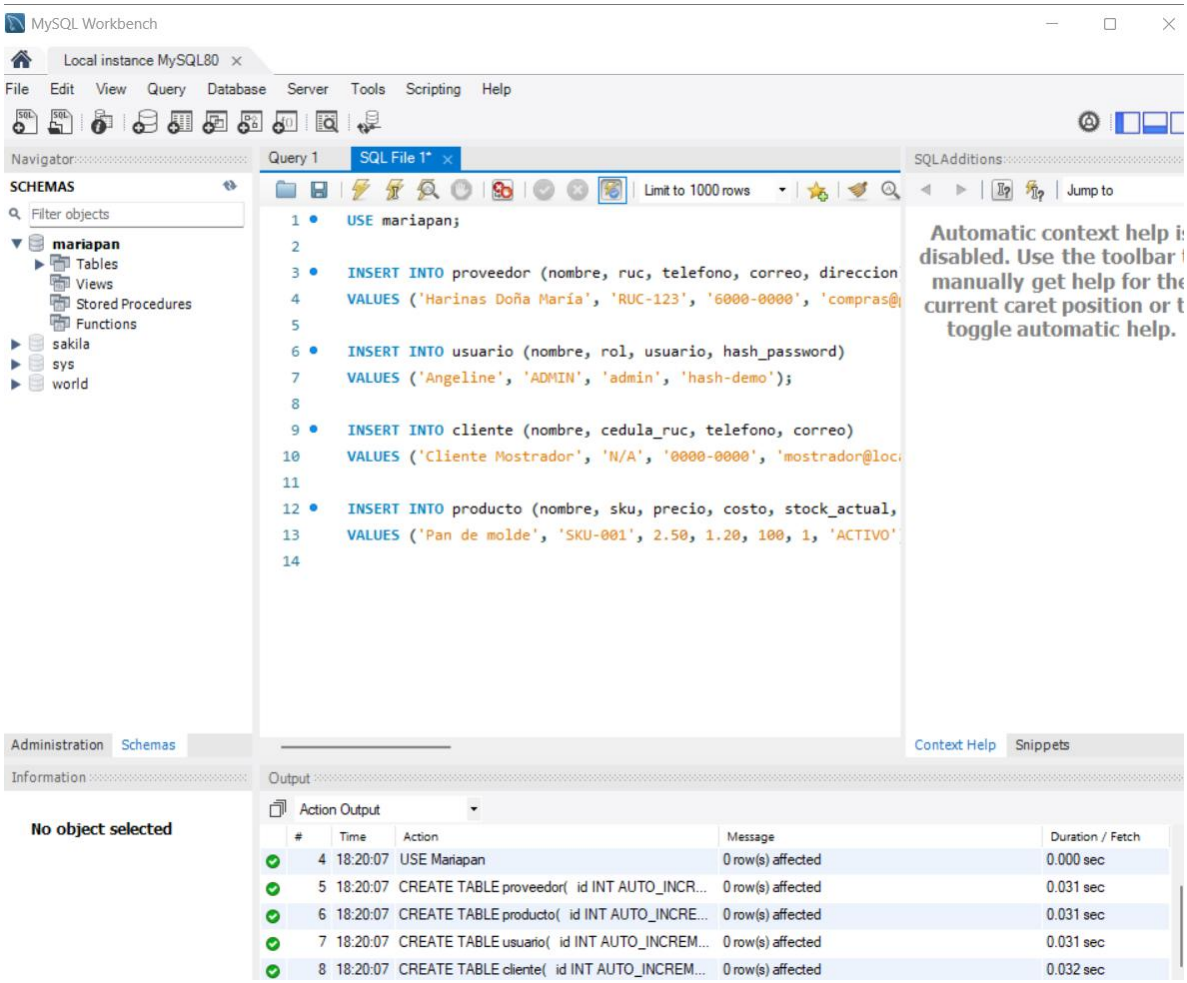
Con este paso se creó la base de datos principal que servirá para almacenar toda la información del sistema de facturación.

En la parte inferior del panel “Output” se confirma que la acción se ejecutó correctamente.



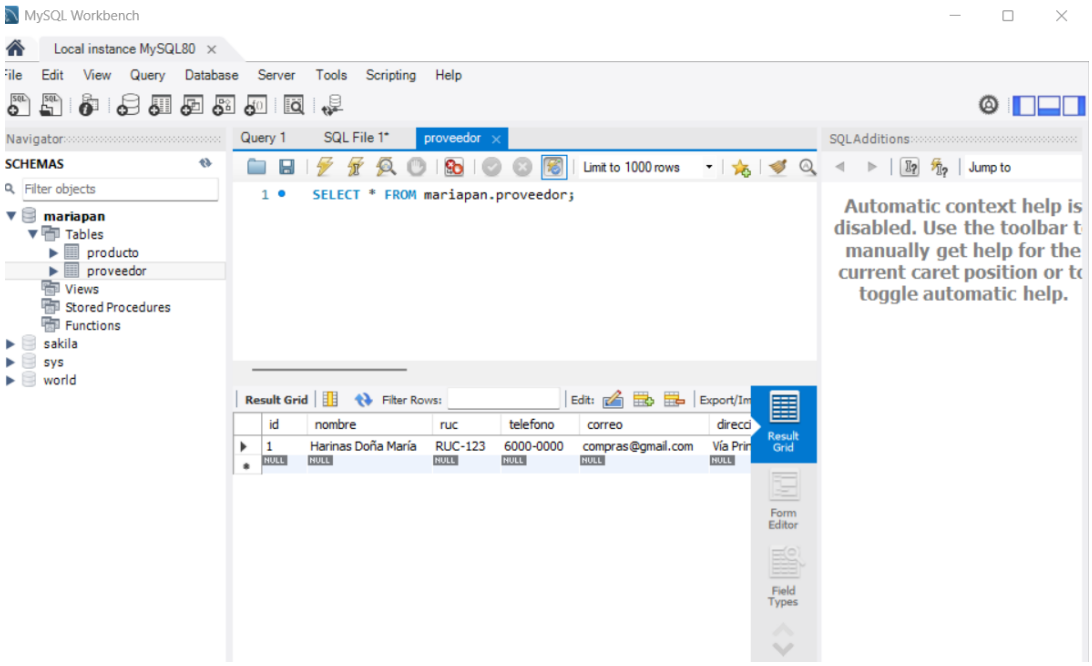
3. Creación de las tablas principales

Posteriormente, se crearon las tablas que conforman el modelo relacional:
proveedor, producto, usuario, cliente, venta, detalle_venta y factura.
Cada tabla fue diseñada con sus claves primarias, tipos de datos adecuados y relaciones con otras tablas mediante llaves foráneas.
En la parte inferior se observa el registro de ejecución correcta de cada instrucción CREATE TABLE, sin errores.



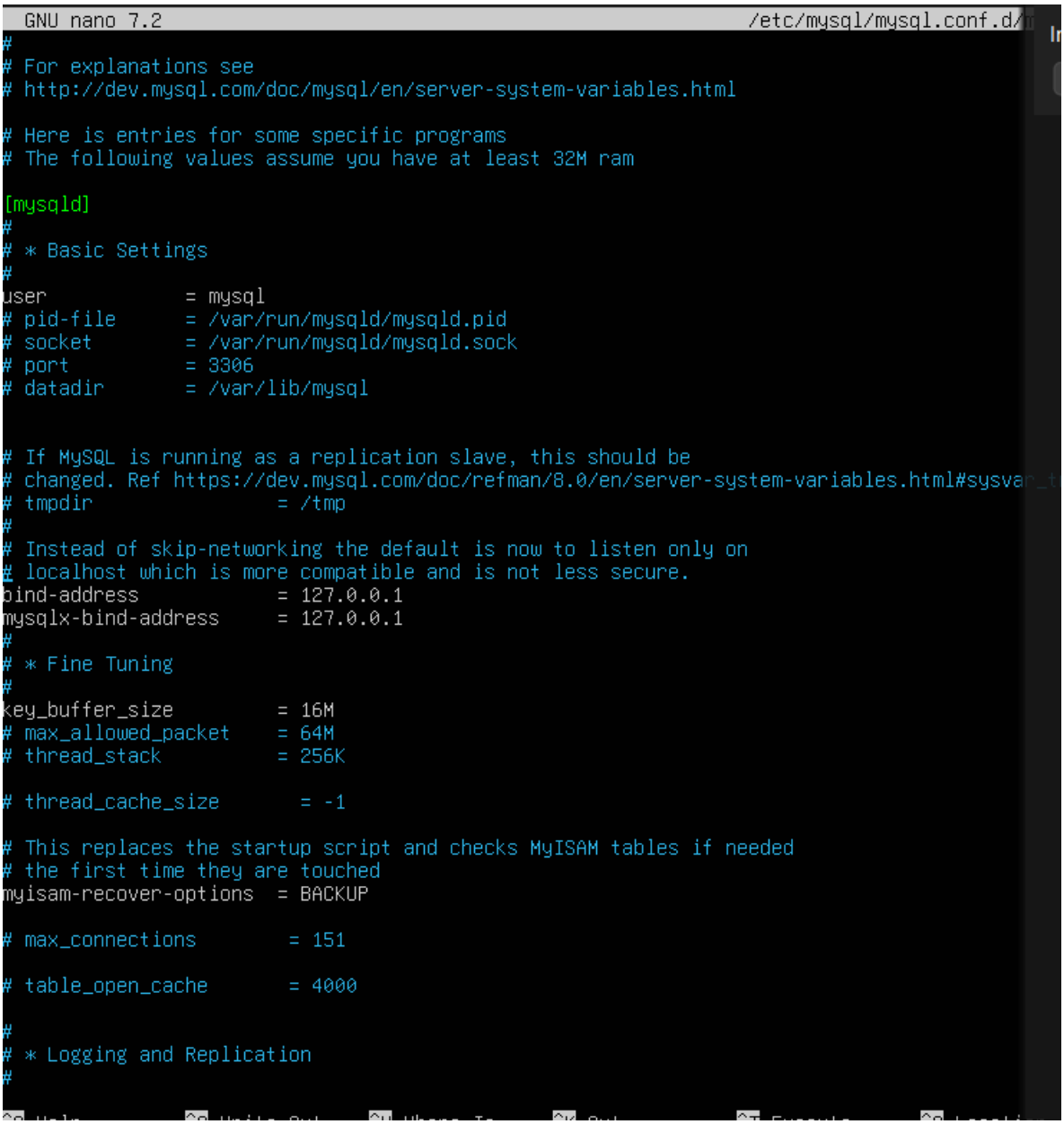
4. Inserción de datos de ejemplo

Luego se insertaron registros de prueba en las tablas para validar su funcionamiento.
Por ejemplo, se añadió un proveedor (“Harinas Doña María”), un usuario administrador (“Angeline”), un cliente genérico (“Cliente Mostrador”) y un producto inicial (“Pan de molde”).
Esto permite verificar la integridad de las relaciones y facilita la demostración práctica del sistema.



5. Consulta de datos
Se ejecutó la instrucción: SELECT * FROM mariapan.proveedor;

La consulta devolvió los registros insertados, confirmando que la tabla almacena y muestra correctamente la información ingresada.

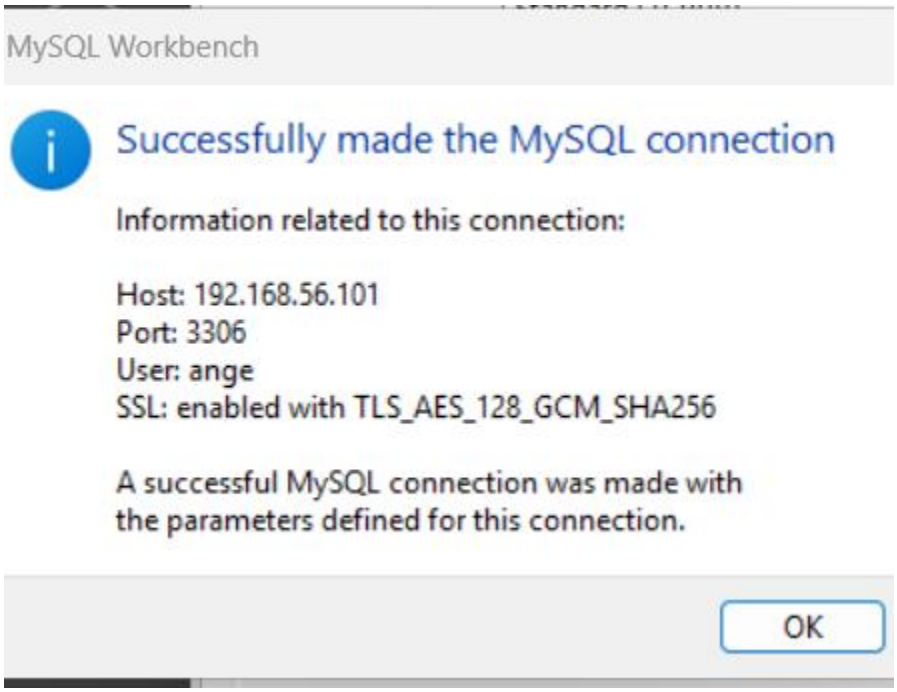


6. Configuración del servidor MySQL en Ubuntu Server
En el entorno del servidor Ubuntu, se configuró el archivo /etc/mysql/mysql.conf.d/mysqld.cnf para definir parámetros de conexión.
Dentro de este archivo se habilitó el puerto 3306 y se verificó la dirección de enlace (bind-address), lo que permite establecer comunicación con el servicio MySQL desde otros equipos o entornos virtuales.

```
ange@ubuntuserver:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:9b:4d:76 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 metric 100 brd 10.0.2.255 scope global dynamic enp0s3
       valid_lft 85647sec preferred_lft 85647sec
    inet6 fd17:625c:f037:2:a00:27ff:fe9b:4d76/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 86160sec preferred_lft 14160sec
    inet6 fe80::a00:27ff:fe9b:4d76/64 scope link
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:e6:e9:52 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.101/24 scope global enp0s8
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fee6:e952/64 scope link
       valid_lft forever preferred_lft forever
ange@ubuntuserver:~$ _
```

7. Verificación de interfaces de red

Mediante el comando ip a se comprobaron las direcciones IP asignadas al servidor Ubuntu.
La interfaz enp0s8 mostró la dirección 192.168.56.101, utilizada para establecer la conexión desde MySQL Workbench y desde las pruebas externas de conexión (cliente–servidor).



8. Prueba de conexión exitosa desde MySQL Workbench (Windows)

Finalmente, se configuró una nueva conexión en Workbench utilizando:

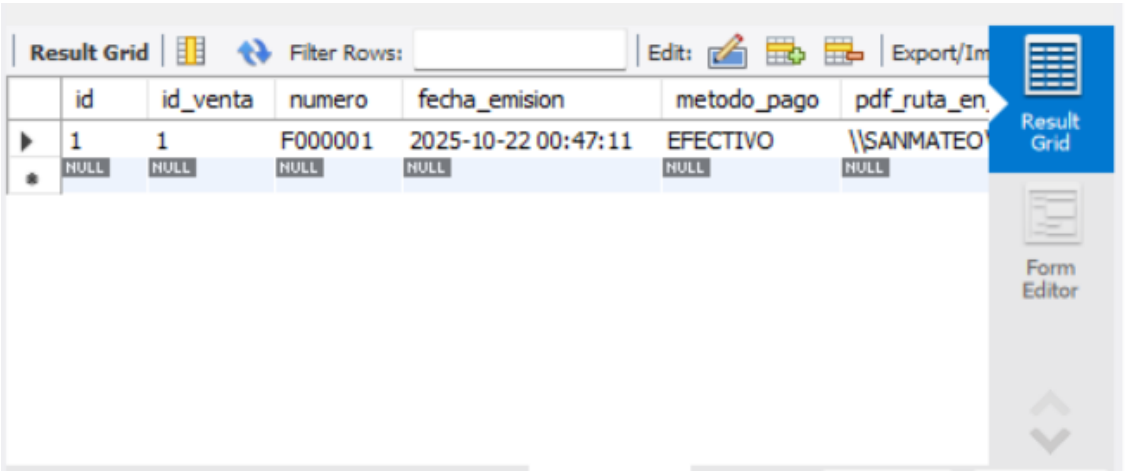
- **Host:** 192.168.56.101
- **Puerto:** 3306

- **Usuario:** ange
  El mensaje “Successfully made the MySQL connection” confirma que la conexión entre el cliente (Windows) y el servidor MySQL (Ubuntu) fue establecida correctamente.

9. Verificación de tablas en el servidor

Desde la consola de Ubuntu se ejecutaron los comandos: SHOW TABLES; SELECT * FROM proveedor;

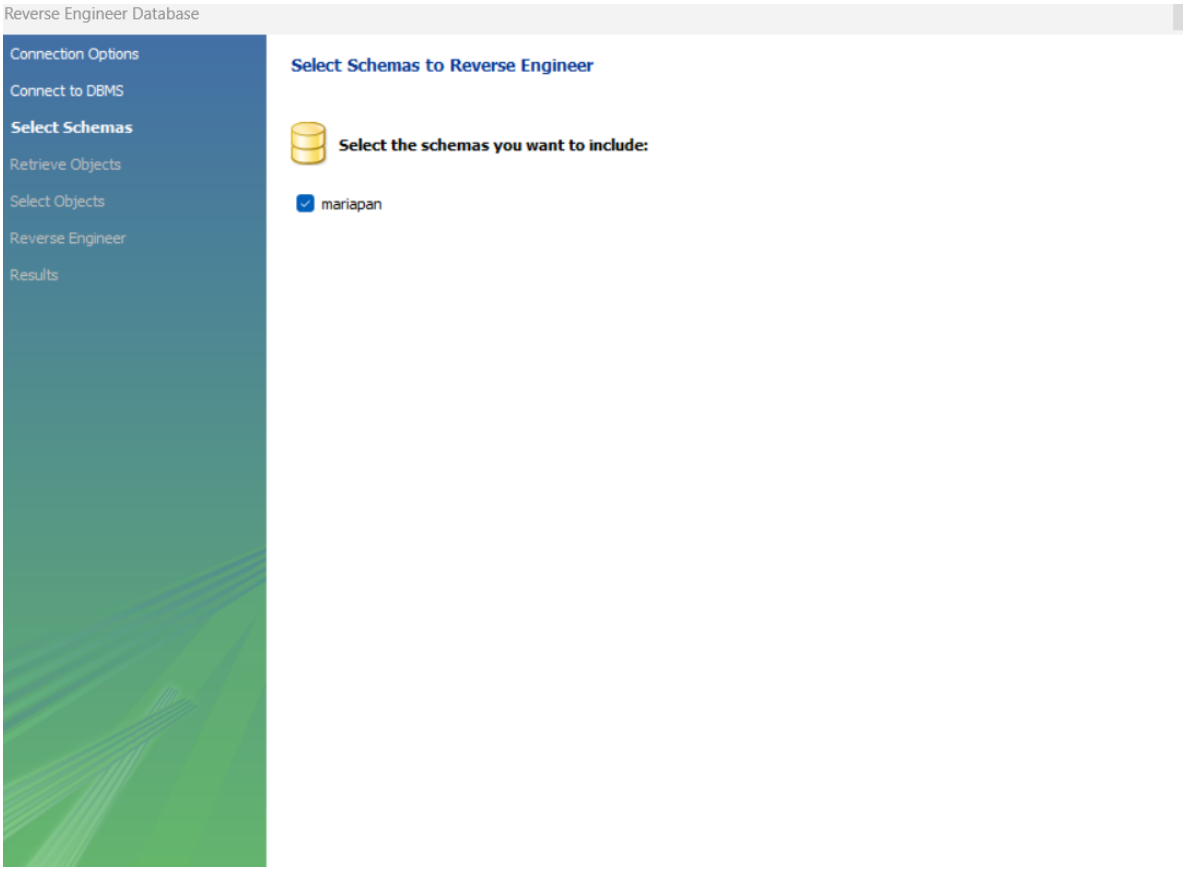
Estos comandos permitieron comprobar que la base de datos “mariapan” y sus tablas existen y están sincronizadas correctamente en el servidor.



10. Inserción de registros en la tabla “factura”

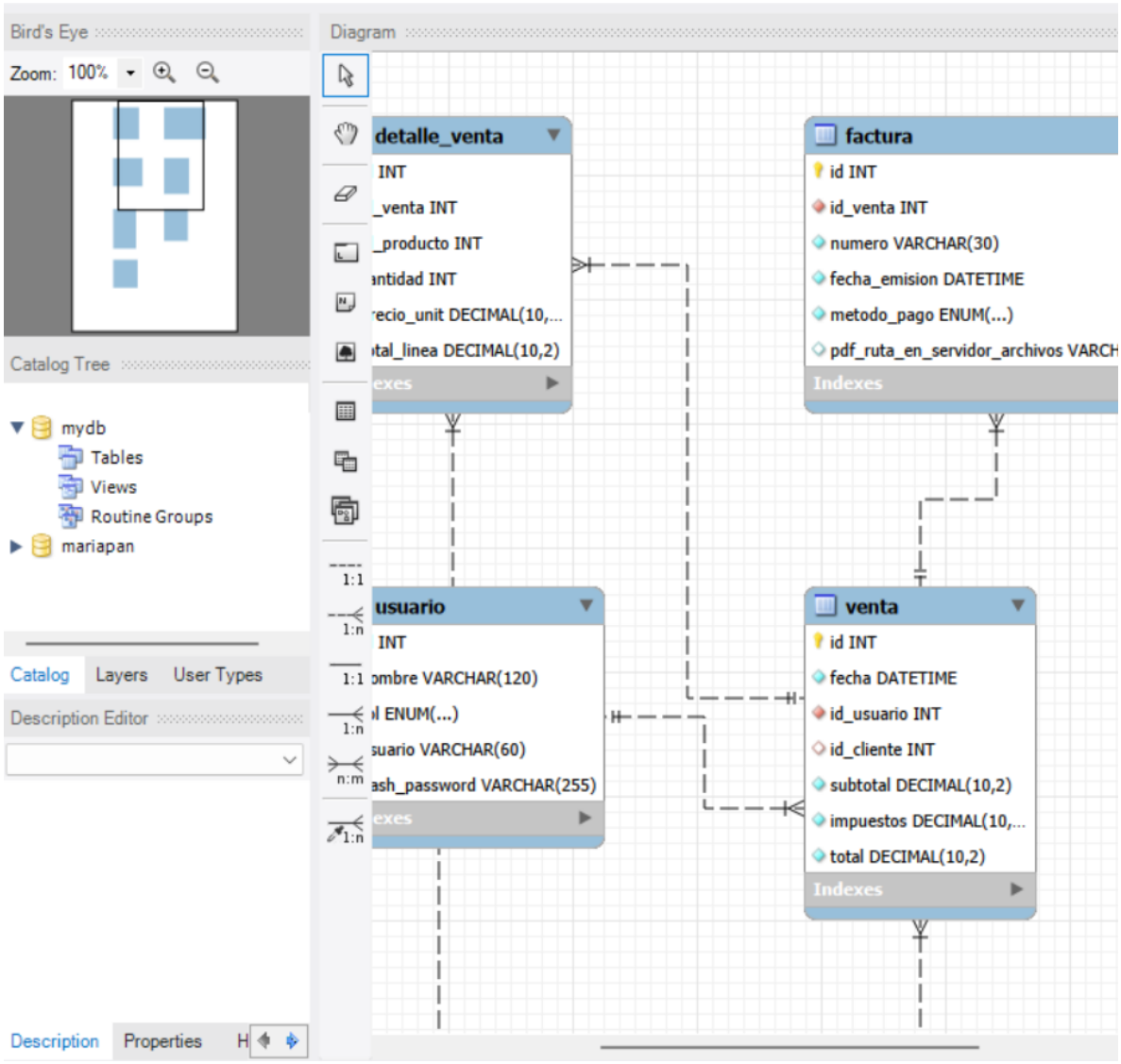
En esta imagen se observa la tabla **factura** con un registro insertado correctamente.

Esta inserción demuestra que la base de datos almacena correctamente la información de las facturas generadas, incluyendo la ruta del archivo PDF en el servidor, simulando el guardado en un recurso compartido.



12. Selección del esquema para Reverse Engineer

En esta captura se aprecia la ventana del asistente de **Reverse Engineer Database**.
El esquema **“mariapan”** aparece seleccionado para ser incluido en el proceso.
Este paso permitió generar automáticamente el **modelo entidad-relación (EER)** a partir de las tablas creadas en la base de datos, facilitando la visualización de las relaciones y estructuras del sistema.



13. Diagrama entidad-relación (EER) final

La última imagen muestra el diagrama EER generado desde el proceso de Reverse Engineer.
En él se representan las tablas principales: **detalle_venta**, **factura**, **usuario** y **venta**, junto con sus relaciones.
Las líneas entre tablas reflejan las claves foráneas establecidas, por ejemplo:

- id_venta conecta la tabla **factura** con **venta**.
- id_usuario y id_cliente conectan **venta** con las tablas correspondientes.
  Este diagrama valida la estructura relacional del proyecto y resume gráficamente la organización de la base de datos **Mariapan**.

Db.java > ...

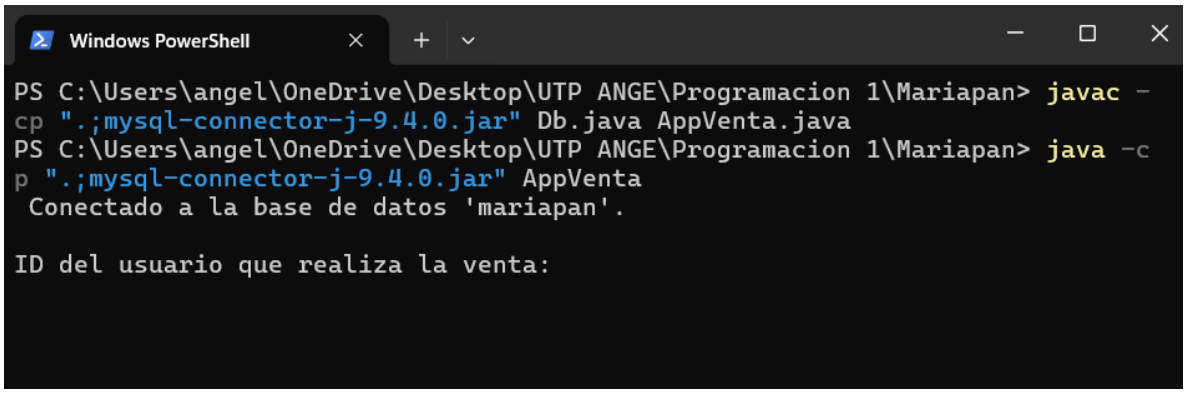
```java
import java.sql.Connection;
import java.sql.DriverManager;

public class Db {
    private static final String URL  = "jdbc:mysql://192.168.56.101:3306/mariapan?useSSL=false&allowPublicKey
    private static final String USER = "ange";
    private static final String PASS = "Angie@2025!";

    public static Connection getConnection() throws Exception {
        return DriverManager.getConnection(URL, USER, PASS);
    }
}

```

En esta imagen se muestra el código fuente de la clase Db.java, encargada de establecer la conexión entre la aplicación Java y el servidor MySQL ubicado en el sistema Ubuntu. Se observa la cadena de conexión JDBC con la dirección IP del servidor (192.168.56.101), el nombre de la base de datos mariapan, y las credenciales de acceso. Este archivo centraliza la configuración de acceso para que la aplicación pueda comunicarse con la base de datos de manera remota y segura.

AppVenta.java > AppVenta > main(String[])

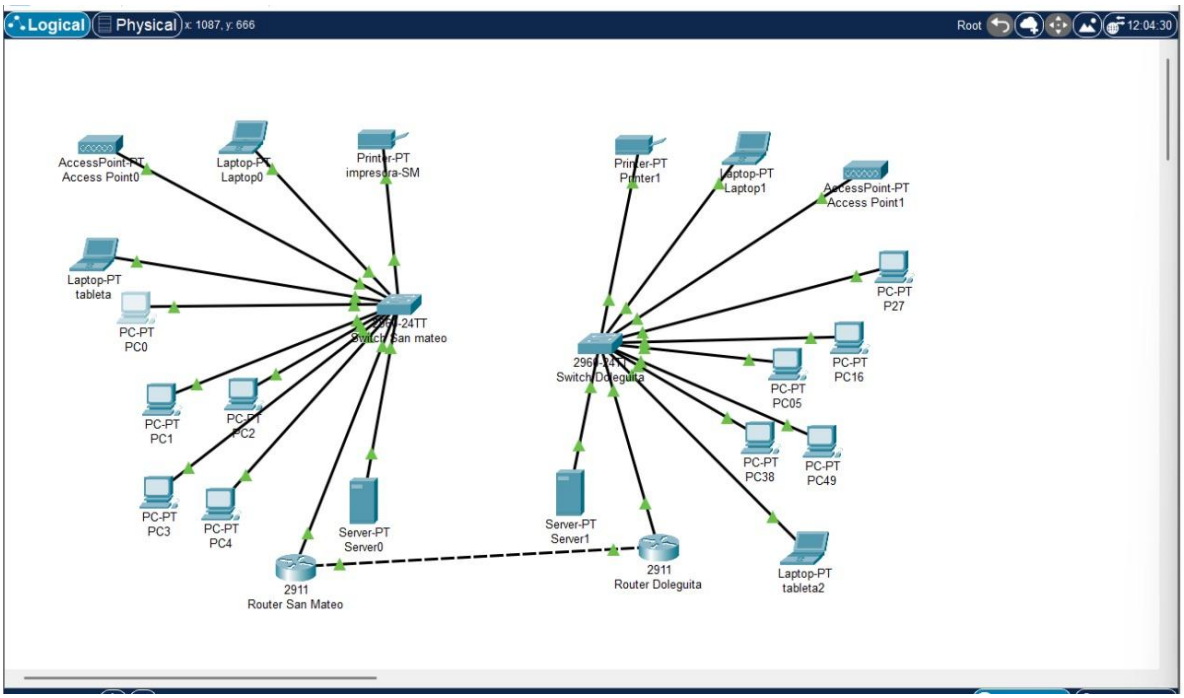
```java
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.math.BigDecimal;
import java.util.Scanner;

public class AppVenta {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        try (Connection cn = Db.getConnection()) {
            System.out.println(x:" Conectado a la base de datos 'mariapan'.\n");

            System.out.print(s:"ID del usuario que realiza la venta: ");
            int idUsuario = Integer.parseInt(sc.nextLine());

            System.out.print(s:"ID del cliente (o presiona Enter si no hay): ");
            String clienteInput = sc.nextLine();
            Integer idCliente = clienteInput.isEmpty() ? null : Integer.parseInt(clienteInput);

            System.out.print(s:"Cantidad de productos en esta venta: ");
            int numProductos = Integer.parseInt(sc.nextLine());

            BigDecimal subtotal = BigDecimal.ZERO;

            String sqlVenta = "INSERT INTO venta (id_usuario, id_cliente, subtotal, impuestos, total) VALUES (?, ?, 0, 0, 0)";
            PreparedStatement psVenta = cn.prepareStatement(sqlVenta, Statement.RETURN_GENERATED_KEYS);
            psVenta.setInt(parameterIndex:1, idUsuario);
            if (idCliente == null) psVenta.setNull(parameterIndex:2, java.sql.Types.INTEGER);
            else psVenta.setInt(parameterIndex:2, idCliente);
            psVenta.executeUpdate();

            ResultSet rs = psVenta.getGeneratedKeys();
            rs.next();
            int idVenta = rs.getInt(columnIndex:1);
```

El código AppProveedor.java conecta Java con la base de datos mariapan y permite registrar proveedores y productos. El programa solicita al usuario datos como nombre del proveedor, producto, cantidad, precio y sucursal, y los guarda automáticamente en la tabla proveedor mediante sentencias SQL preparadas.



El programa AppVenta.java establece la conexión con la base de datos *mariapan* y solicita los datos necesarios para registrar una venta. Inicia mostrando el mensaje de conexión exitosa y luego pide el ID del usuario que realiza la venta, preparando el proceso de ingreso de datos a la tabla correspondiente.

Diseño de red en Cisco Packet Tracer



La imagen muestra el diseño lógico de una red creada en Cisco Packet Tracer, conformada por dos sedes: San Mateo y Doleguita.

Cada sede cuenta con un router, un switch, varios equipos de cómputo, laptops, impresoras, puntos de acceso inalámbrico y un servidor.

Ambas redes están interconectadas mediante un enlace serial entre los routers, simulando la comunicación entre dos sucursales que comparten recursos y servicios dentro de una misma infraestructura.