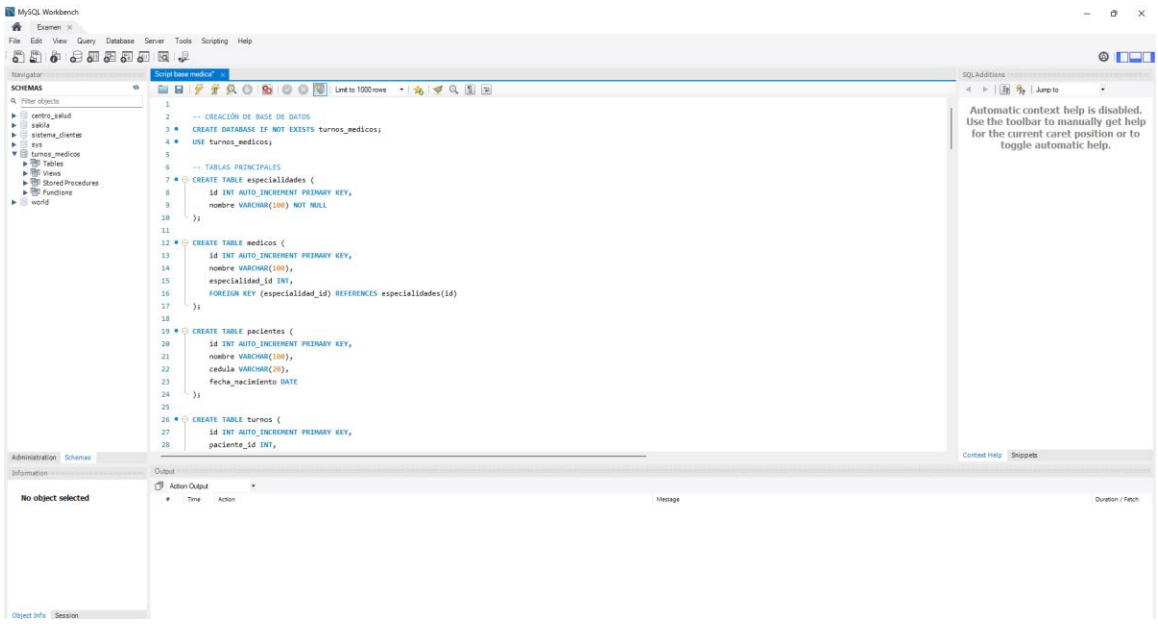
TALLER Sistema con Base de Datos y Java

Antony Cisneros

Parte 1: Configuración de la base de datos

1. Descargar el script SQL proporcionado.



2. Crear la base de datos y las tablas ejecutando el script.

3. Insertar los registros necesarios utilizando INSERT INTO.

4. Crear únicamente las funciones definidas en el script.

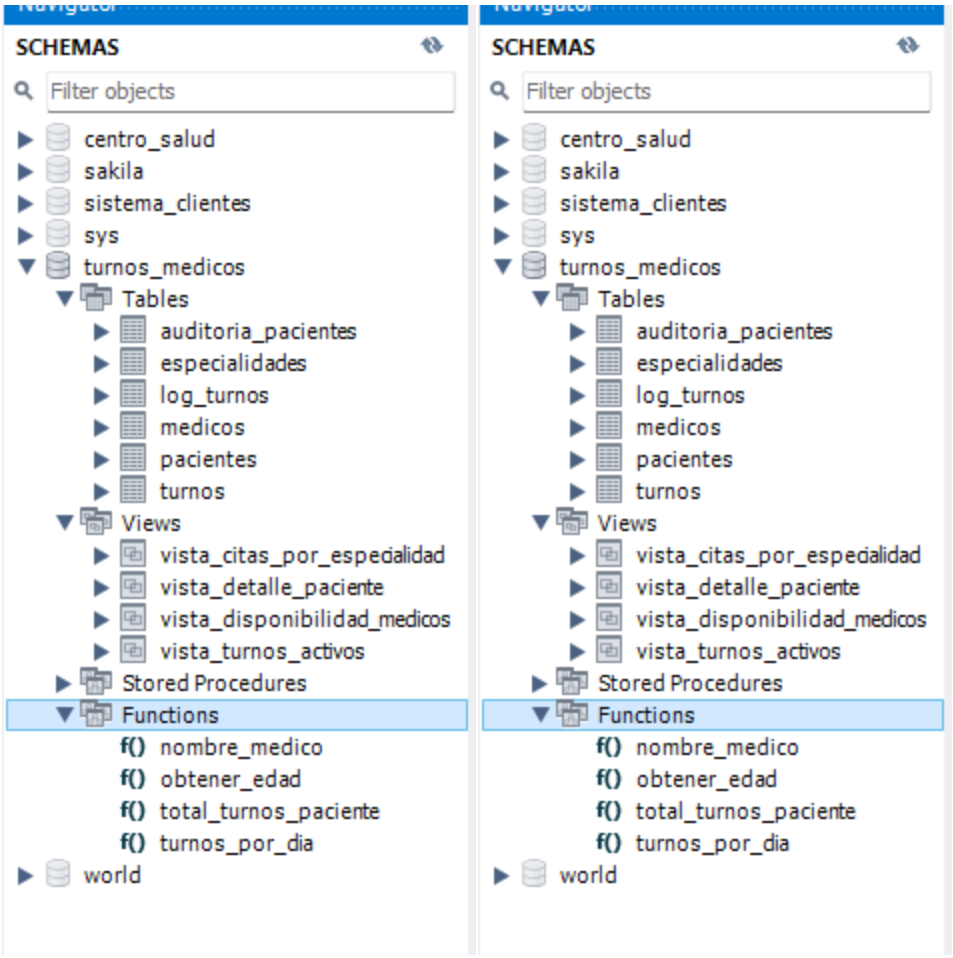
5. Crear únicamente los procedimientos almacenados.

6. Crear únicamente los triggers.

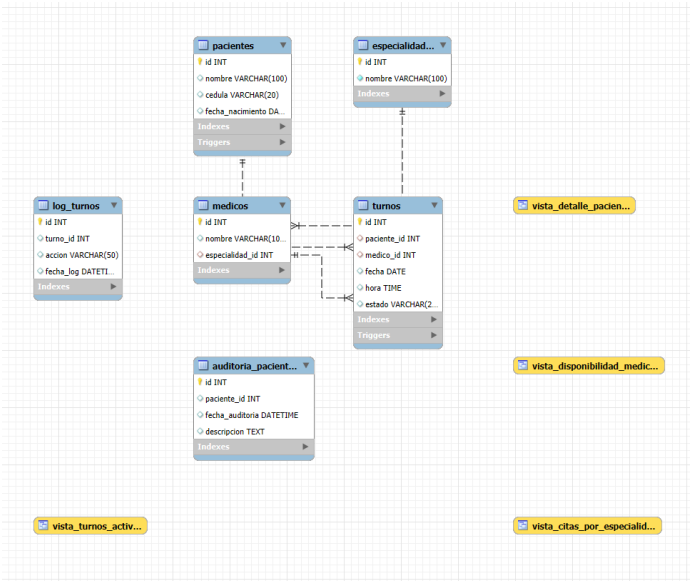
6.1 Verificar que los triggers fueron creados correctamente usando la instrucción:

SHOW TRIGGERS;

7. Crear las vistas definidas.



8. Verificar que todos los elementos anteriores (tablas, funciones, procedimientos, triggers y vistas) fueron creados correctamente.



Parte 2: Configuración del proyecto Java en IntelliJ IDEA

1. Abrir IntelliJ IDEA y cargar el proyecto.

2. Restaurar el código Java del sistema.

3. Ir a File → Project Structure → Libraries y añadir el conector MySQL (mysql-connector-java-x.x.xx.jar) previamente descargado.

4. Si aún no lo tienes, descargar el conector MySQL desde el sitio oficial:

   https://dev.mysql.com/downloads/connector/j/

5. Ejecutar el código Java y comprobar que la conexión con la base de datos funcione.

6. Verificar que el código Java utilice correctamente los elementos de base de datos:

   - Vistas

   - Funciones

   - Procedimientos almacenados

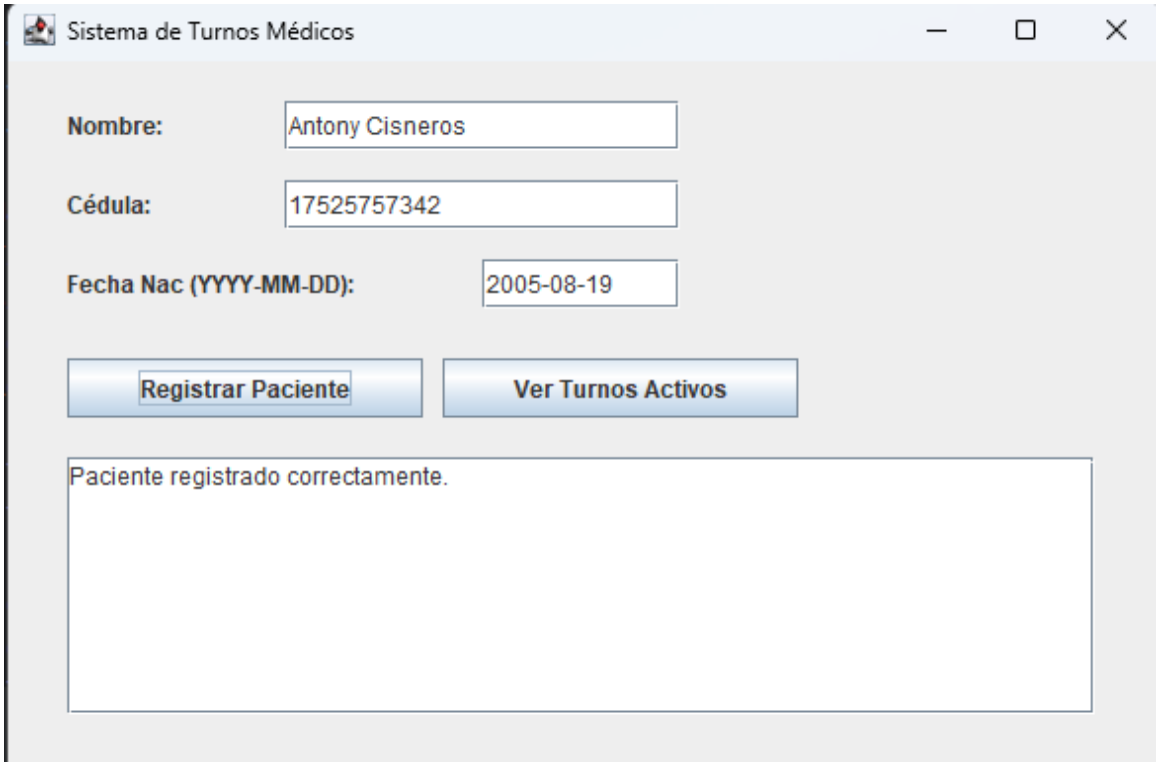
   - Triggers

7. Crear usuarios para probar las acciones del trigger

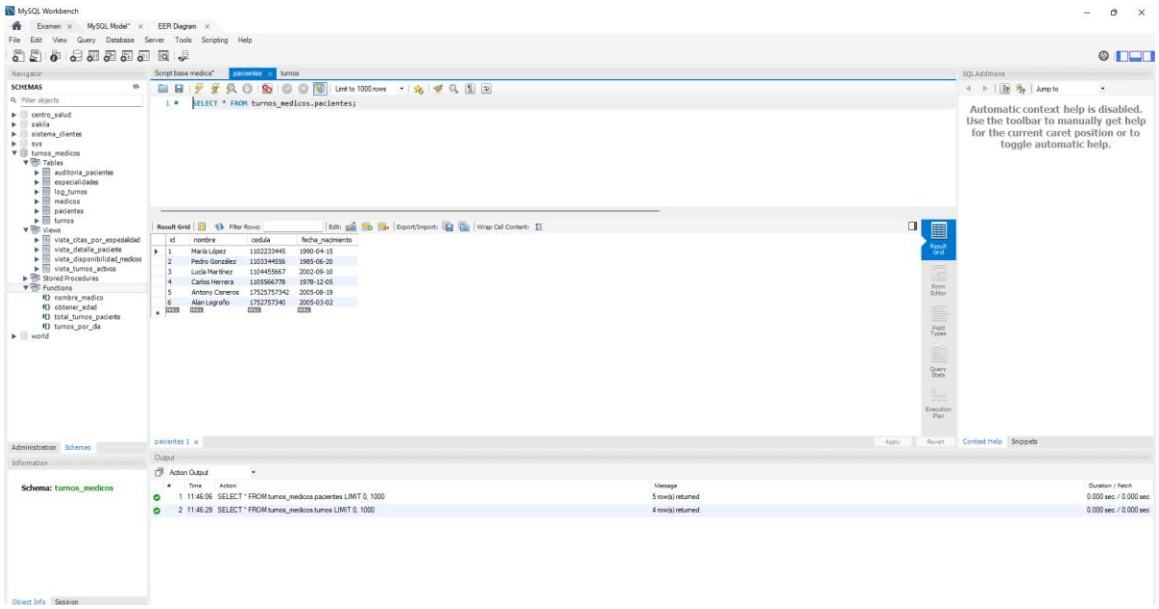
Capturar pantallas.

Subir información a git HUB y poner conclusión de la práctica realizada

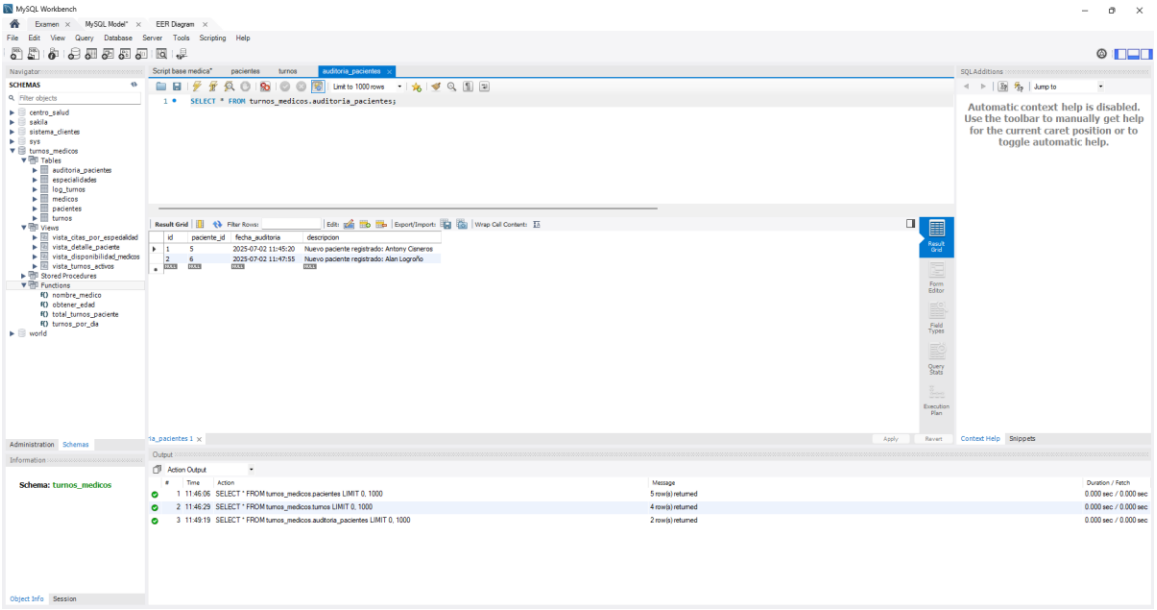
REGISTRO DE PACIENTES



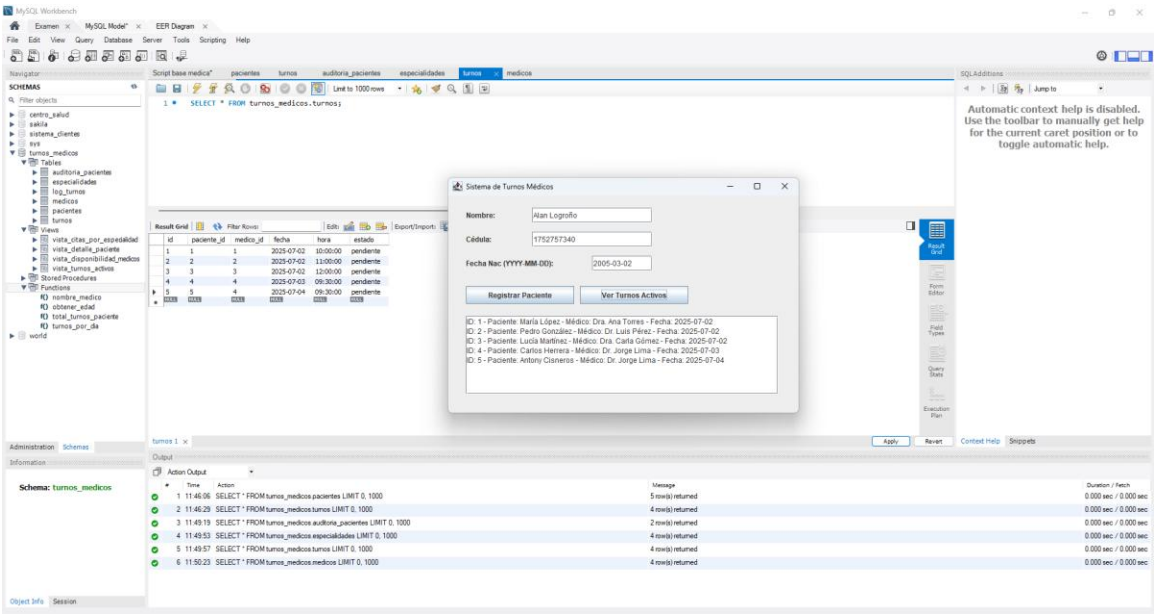
INGRESO EN LA BASE DE DATOS MYSQL



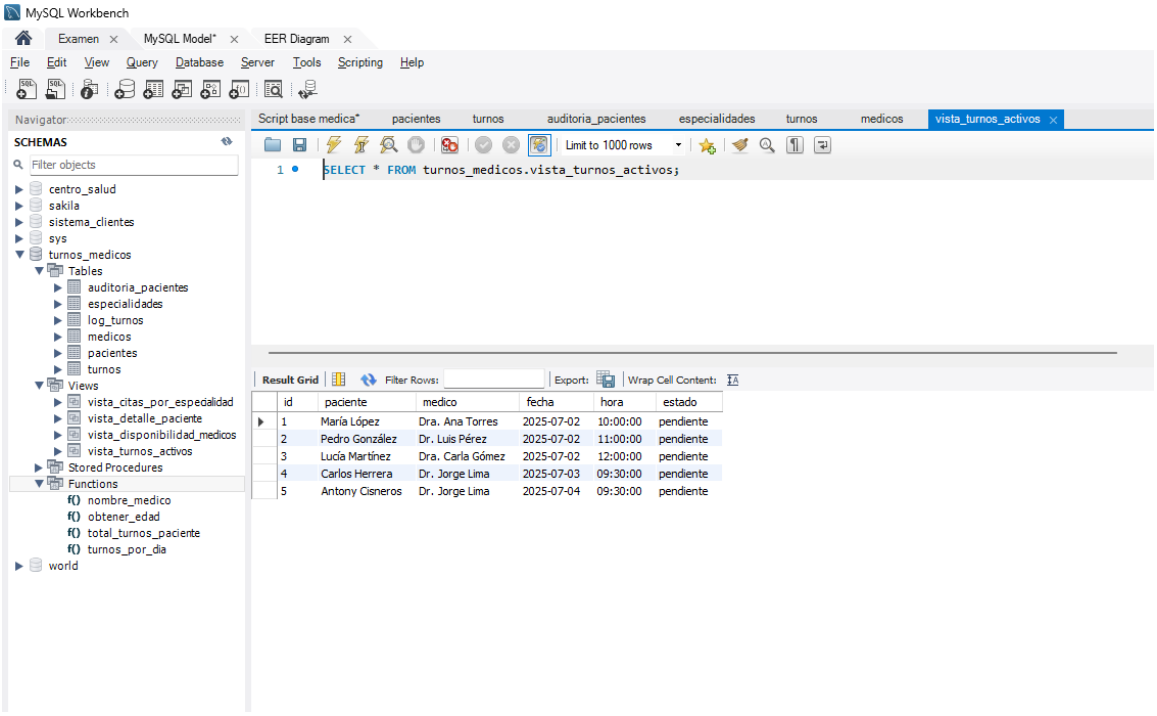
TRIGGER DE INGRESO DE PACIENTES



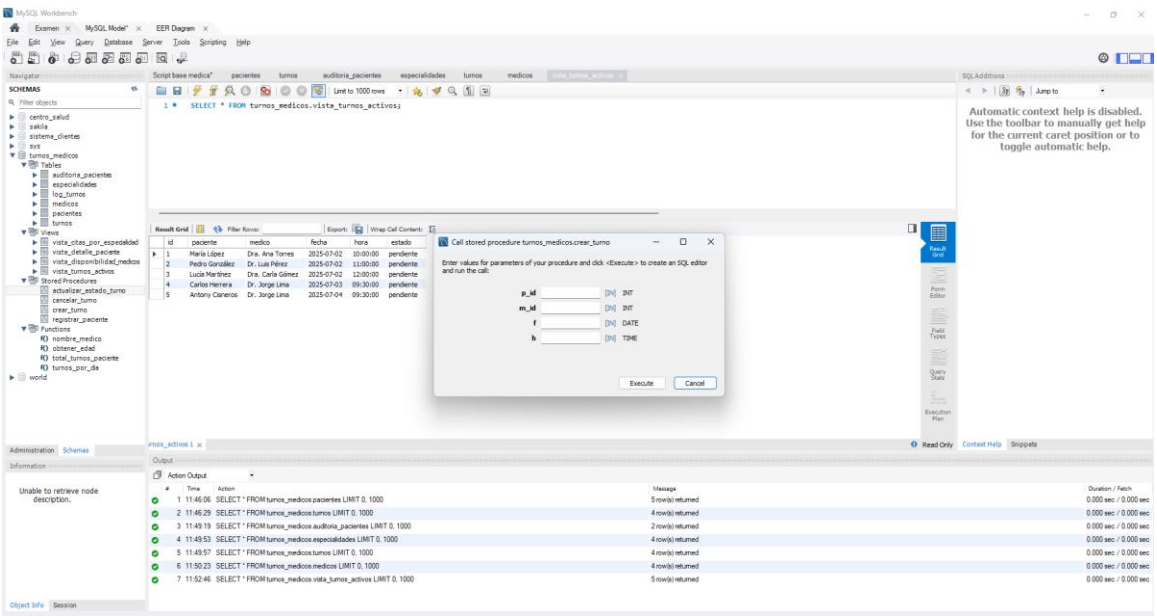
AGENDAMIENTO DEL TURNO EN LA TABLA TURNOS Y SU ACTUALIZACIÓN EN EL SISTEMA

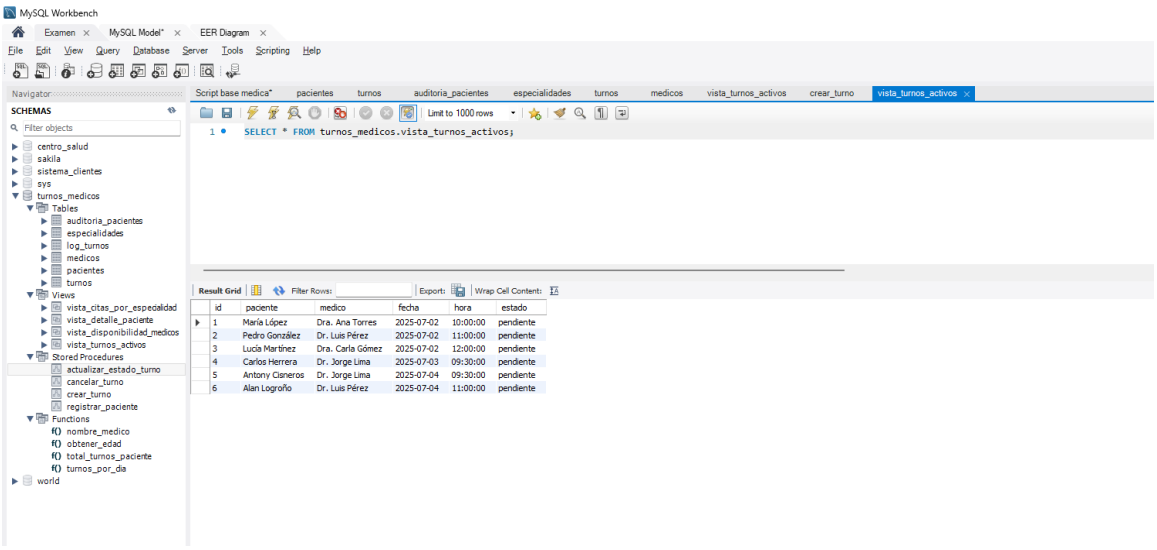


VISTAS



PROCEDIMIENTOS ALMACENADOS





CONCLUSION

En la ejecución de la interfaz gráfica de java, se puede observar diferentes funciones e vistas aplicadas. Como el trigger de Registrar Paciente, y la vista de Ver turnos activos.

```java
private void registrarPaciente() {  1 usage
    try (Connection conn = DBConnection.getConnection()) {
        String nombre = txtNombre.getText();
        String cedula = txtCedula.getText();
        String fecha = txtFechaNacimiento.getText();

        CallableStatement stmt = conn.prepareCall( sql: "{CALL registrar_paciente(?, ?, ?)}");
        stmt.setString( parameterIndex: 1, nombre);
        stmt.setString( parameterIndex: 2, cedula);
        stmt.setString( parameterIndex: 3, fecha);
        stmt.execute();
        txtResultados.setText("Paciente registrado correctamente.");
    } catch (SQLException ex) {
        txtResultados.setText("Error: " + ex.getMessage());
    }
}
```

```java
private void mostrarTurnos() {  1 usage
    try (Connection conn = DBConnection.getConnection()) {
        String query = "SELECT * FROM vista_turnos_activos";
        PreparedStatement stmt = conn.prepareStatement(query);
        ResultSet rs = stmt.executeQuery();
        StringBuilder sb = new StringBuilder();
        while (rs.next()) {
            sb.append("ID: ").append(rs.getInt( columnLabel: "id")).append(" - ")
              .append("Paciente: ").append(rs.getString( columnLabel: "paciente")).append(" - ")
              .append("Médico: ").append(rs.getString( columnLabel: "medico")).append(" - ")
              .append("Fecha: ").append(rs.getDate( columnLabel: "fecha")).append("\n");
        }
        txtResultados.setText(sb.toString());
    } catch (SQLException ex) {
        txtResultados.setText("Error: " + ex.getMessage());
    }
}
```

De esta manera, podemos simplificar mucho las funciones de MYSQL con una interfaz grafica que facilita las acciones, tan solo instalando una librería de MYSQL Connector.