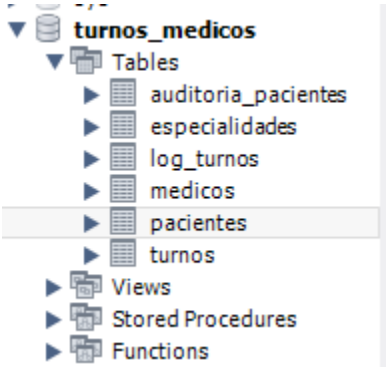
TALLER Sistema con Base de Datos y Java

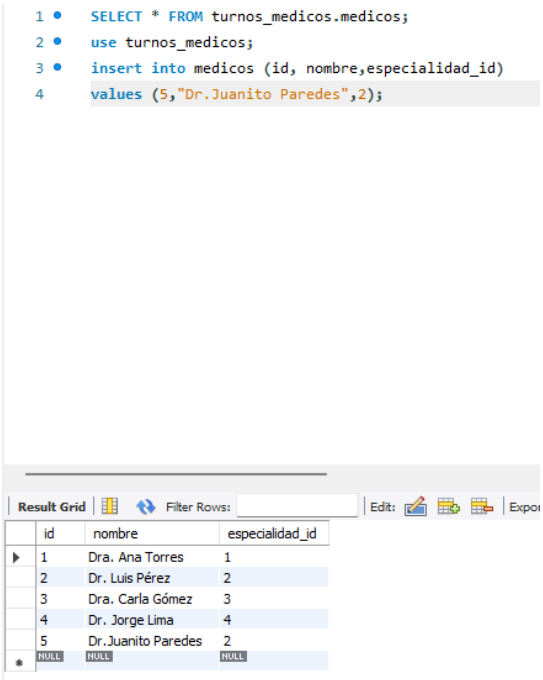
Parte 1: Configuración de la base de datos

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.

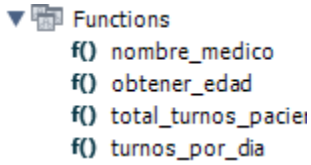


```
SELECT * FROM turnos_medicos.medicos;
use turnos_medicos;
insert into medicos (id, nombre,especialidad_id)
values (5,"Dr.Juanito Paredes",2);
```

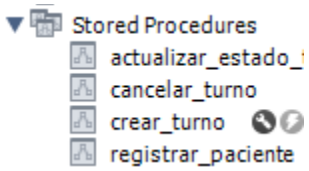
Result Grid | Filter Rows: | Edit: | Expor

id	nombre	especialidad_id
1	Dra. Ana Torres	1
2	Dr. Luis Pérez	2
3	Dra. Carla Gómez	3
4	Dr. Jorge Lima	4
5	Dr.Juanito Paredes	2
NULL	NULL	NULL

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:

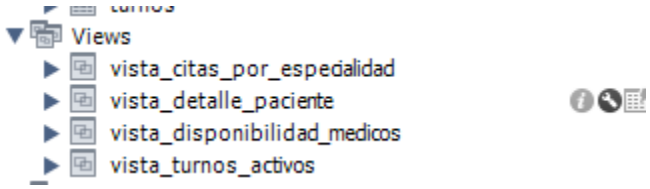
SHOW TRIGGERS;

```
1  SELECT * FROM turnos_medicos.pacientes;
2  use turnos_medicos;
3  show triggers;
4
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Trigger	Event	Table	Statement	Timing	Created
trg_auditar_paciente_nuevo	INSERT	pacientes	BEGIN INSERT INTO auditoria_pacientes(paci...	AFTER	2025-07-02 11:13
trg_validar_fecha_turno	INSERT	turnos	BEGIN IF NEW.fecha < CURDATE() THEN ...	BEFORE	2025-07-02 11:13
trg_auto_estado_turno	INSERT	turnos	BEGIN IF NEW.fecha < CURDATE() THEN ...	AFTER	2025-07-02 11:13
trg_log_cambios_turnos	UPDATE	turnos	BEGIN INSERT INTO log_turnos(turno_id, acc...	AFTER	2025-07-02 11:13

7. Crear las vistas definidas.



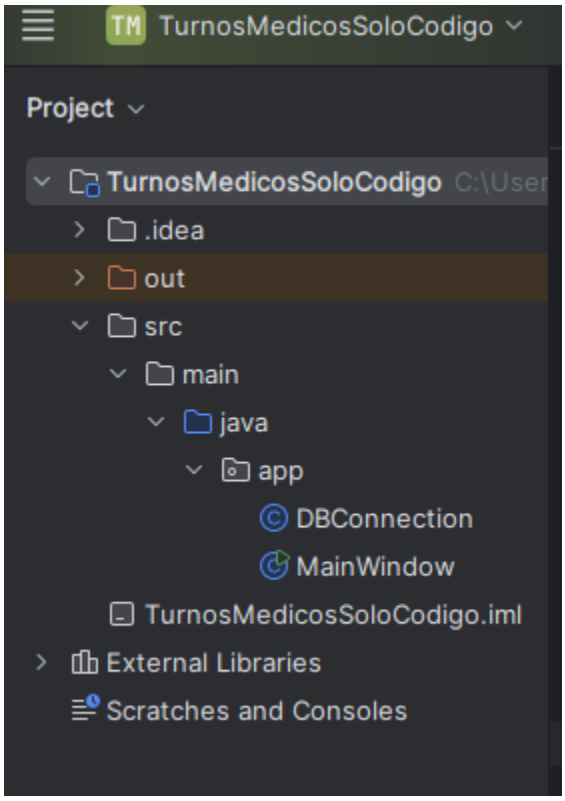
8. Verificar que todos los elementos anteriores (tablas, funciones, procedimientos, triggers y vistas) fueron creados correctamente.

Parte 2: Configuración del proyecto Java en IntelliJ IDEA

1. Abrir IntelliJ IDEA y cargar el proyecto.

2. Restaurar el código Java del sistema.

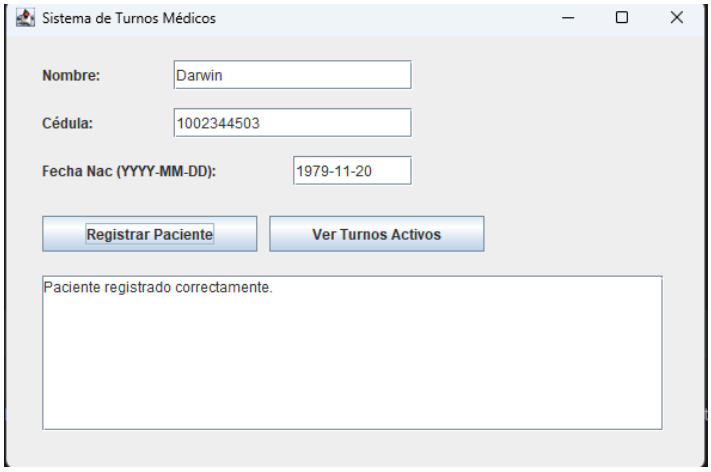
3. Ir a File → Project Structure → Libraries y añadir el conector MySQL (mysql-connector-java-x.x.xx.jar) previamente descargado.

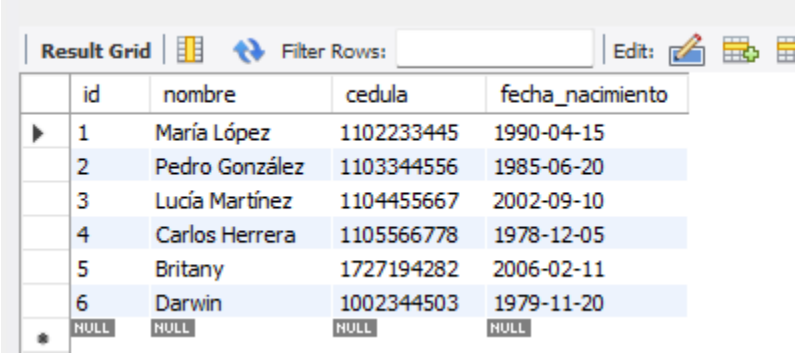


4. Si aún no lo tienes, descargar el conector MySQL desde el sitio oficial:

   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

```
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());
    }
}
```

- Funciones

- Procedimientos almacenados

```
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();
                                                    registrado correctamente.");
    } ca
        txtResultados.setText("Error: " + ex.getMessage());
    }
}
```

Extract Surround

- Triggers

```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();
        Extract  Surround  //  ⋮
        e registrado correctamente.");
    } ca
        txtResultados.setText("Error: " + ex.getMessage());
    }
}
```

8. Crear usuarios para probar las acciones del trigger

```sql
use turnos_medicos;
create user 'Karo_user'@'localhost' identified by '1234';
grant all privileges on turnos_medicos.* to  'Karo_user'@'localhost';
flush privileges;
```

```sql
SELECT * FROM turnos_medicos.pacientes;
use turnos_medicos;
show triggers;
SELECT user, host FROM mysql.user;
```

Result Grid | Filter Rows: | Export:

user	host
admin	%
Karo_user	localhost
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
root	localhost

id	paciente_id	fecha_auditoria	descripcion
1	5	2025-07-02 11:27:12	Nuevo paciente registrado: Britany
2	6	2025-07-02 11:43:43	Nuevo paciente registrado: Darwin

Capturar pantallas.

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

Conclusión:

La práctica realizada nos ayudó a entender cómo funciona la base de datos de mysql con el lenguaje de programación java, También realizamos prácticas con los triggers y nos ayudó a identificar las vistas, procesos almacenados y triggers en java

https://github.com/Britany1727/Taller_Complementario_BD