


TALLER Sistema con Base de Datos y Java

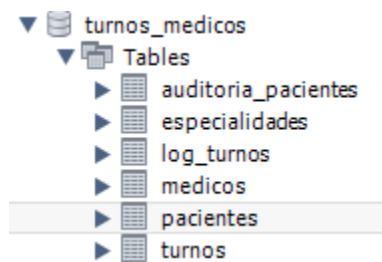
Katherine Sailema

Parte 1: Configuración de la base de datos

1. Descargar el script SQL proporcionado.

 Script base medica	2/7/2025 11:13	SQL Text File	6 KB
--	----------------	---------------	------

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

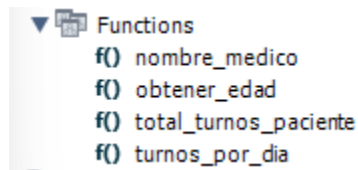


3. Insertar los registros necesarios utilizando INSERT INTO.

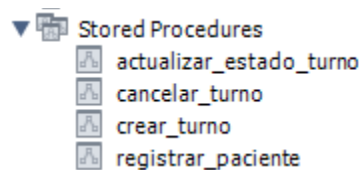
```
1 • use turnos_medicos;
2 • insert into medicos (id, nombre, especialidad_id)
3   values (5, "Dr.Pablo Beltran", 2);
4
5
```

Result Grid			
Filter Rows: <input type="text"/>			
Edit:   			
	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.Pablo Beltran	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;

5 • `show triggers;`

Result Grid

Filter Rows:

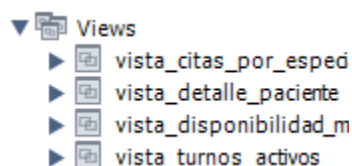
Export:

Wrap Cell Content:

[↗](#)

	Trigger	Event	Table	Statement	Timing	Created	sql_mode
▶	trg_auditar_paciente_nuevo	INSERT	pacientes	BEGIN INSERT INTO auditoria_pacientes(paci...	AFTER	2025-07-02 11:14:47.70	ONLY_FULL_GROI
	trg_validar_fecha_turno	INSERT	turnos	BEGIN IF NEW.fecha < CURDATE() THEN ...	BEFORE	2025-07-02 11:14:47.66	ONLY_FULL_GROI
	trg_auto_estado_turno	INSERT	turnos	BEGIN IF NEW.fecha < CURDATE() THEN ...	AFTER	2025-07-02 11:14:47.72	ONLY_FULL_GROI
	trg_log_cambios_turnos	UPDATE	turnos	BEGIN INSERT INTO log_turnos(turno_id, acc...	AFTER	2025-07-02 11:14:47.68	ONLY_FULL_GROI

7. Crear las vistas definidas.



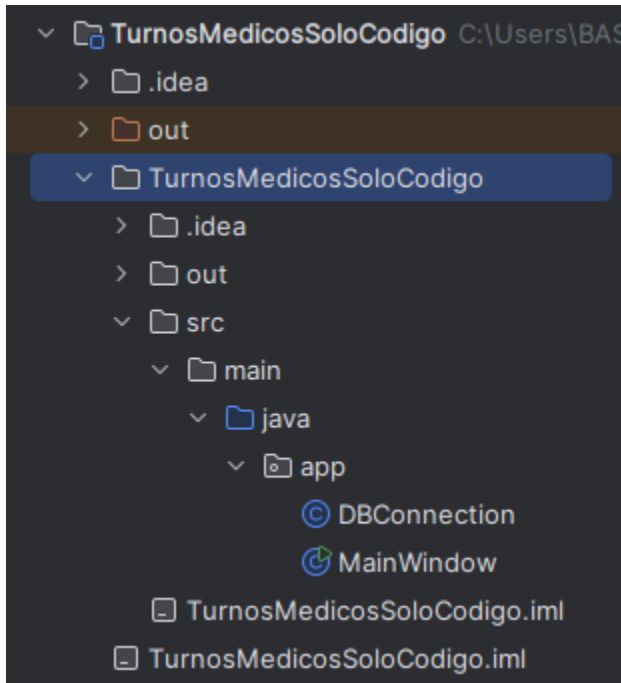
8. Verificar que todos los elementos anteriores (tablas, funciones, procedimientos, triggers y vistas) fueron creados correctamente.

Parte 2: Configuración del proyecto Java en IntelliJ IDEA

1. Abrir IntelliJ IDEA y cargar el proyecto.

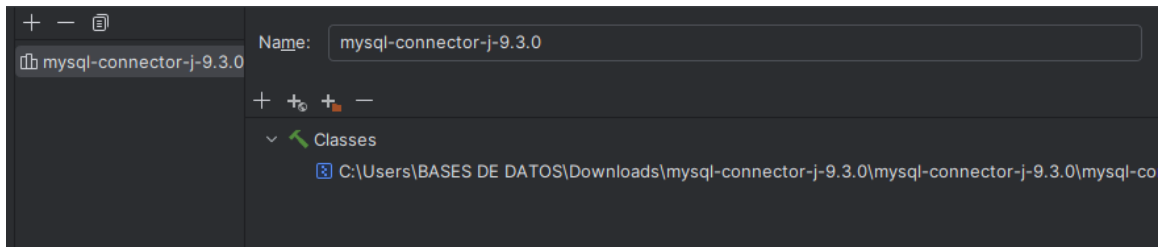
2. Restaurar el código Java del sistema.

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.

Sistema de Turnos Médicos

Nombre: Aleksandry

Cédula: 1725164875

Fecha Nac (YYYY-MM-DD): 2004-05-16

Registrar Paciente

Ver Turnos Activos

Paciente registrado correctamente.

```
1 • SELECT * FROM turnos_medicos.pacientes;
```

Result Grid				
Filter Rows:				
Edit:				
	id	nombre	cedula	fecha_nacimiento
▶	1	María López	1102233445	1990-04-15
	2	Pedro González	1103344556	1985-06-20
	3	Lucía Martínez	1104455667	2002-09-10
	4	Carlos Herrera	1105566778	1978-12-05
	5	Katherine	1727280297	2004-07-25
	6	Aleksandry	1725164875	2004-05-16
✱	NULL	NULL	NULL	NULL

Registrar Paciente

Ver Turnos Activos

ID: 1 - Paciente: María López - Médico: Dra. Ana Torres - Fecha: 2025-07-02
ID: 2 - Paciente: Pedro González - Médico: Dr. Luis Pérez - Fecha: 2025-07-02
ID: 3 - Paciente: Lucía Martínez - Médico: Dra. Carla Gómez - Fecha: 2025-07-02
ID: 4 - Paciente: Carlos Herrera - Médico: Dr. Jorge Lima - Fecha: 2025-07-03

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

- Vistas – Funciones – Procedimientos almacenados – Triggers

```
private void mostrarTurnos() {
    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());
    }
}
```

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

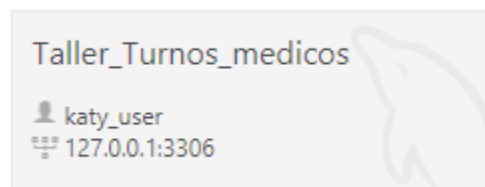
        CallableStatement stmt = conn.prepareCall( sql: "{CALL registrar_paciente(?, ?, ?)}");
        stmt.setString(parameterIndex: 1, nombre);
        stmt.setString(parameterIndex: 2, cedula);
        stmt.setString(parameterIndex: 3, fecha);
        stmt.execute();

        txtResultados.setText("Paciente registrado correctamente.");
    } catch (SQLException ex) {
        txtResultados.setText("Error: " + ex.getMessage());
    }
}
```

7. Crear usuarios para probar las acciones del trigger

- 6 • `create user 'katy_user'@'localhost' identified by 'katy1234';`
- 7 • `grant all privileges on turnos_medicos to 'katy_user'@'localhost';`
- 8 • `flush privileges;`
- 9 • `select user, host from mysql.user;`

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
user	host			
admin	%			
katy_user	localhost			
mysql.infoschema	localhost			
mysql.session	localhost			
mysql.sys	localhost			
root	localhost			



SCHEMAS		1 • <code>SELECT * FROM turnos_medicos.auditoria_pacientes;</code>			
Filter objects		Limit to 1000 rows			
cursoonline					
sakila					
sys					
turnos_medicos					
Tables					
auditoria_pacientes					
especialidades					
log_turnos					
medicos					
pacientes					
turnos					
Views					
Stored Procedures					

Result Grid		Filter Rows:	Edit:	Export/Import:
id	paciente_id	fecha_auditoria	descripcion	
1	5	2025-07-02 11:27:18	Nuevo paciente registrado: Katherine	
2	6	2025-07-02 11:43:58	Nuevo paciente registrado: Aleksandry	
*	NULL	NULL	NULL	

Capturar pantallas.

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

Entender el entorno de la conexión entre intellij y workbench para así crear una interfaz más didáctica y entendible para el usuario así pudiendo ingresar una información como una aplicación (intellij) y guardando esta (workbench) para tener una base de datos ordenada.

También la comprensión de como estas se conectan a través del análisis del código

Git hub

https://github.com/Katherine137/Taller_complementario_Sailema_K.git