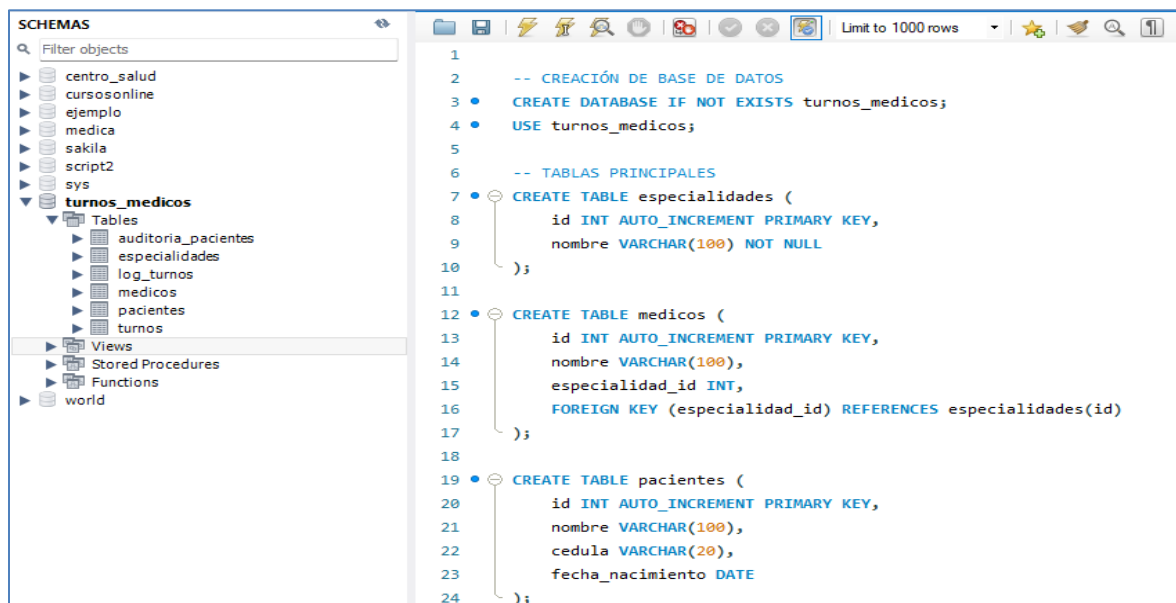


# TALLER Sistema con Base de Datos y Java

## 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.



The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' panel displays a tree view of the database structure, including a 'turnos\_medicos' database with tables like 'auditoria\_pacientes', 'especialidades', 'log\_turnos', 'medicos', 'pacientes', and 'turnos'. The main editor on the right contains SQL code for creating the database and tables.

```
1
2  -- CREACIÓN DE BASE DE DATOS
3  • CREATE DATABASE IF NOT EXISTS turnos_medicos;
4  • USE turnos_medicos;
5
6  -- TABLAS PRINCIPALES
7  • CREATE TABLE especialidades (
8      id INT AUTO_INCREMENT PRIMARY KEY,
9      nombre VARCHAR(100) NOT NULL
10 );
11
12 • CREATE TABLE medicos (
13     id INT AUTO_INCREMENT PRIMARY KEY,
14     nombre VARCHAR(100),
15     especialidad_id INT,
16     FOREIGN KEY (especialidad_id) REFERENCES especialidades(id)
17 );
18
19 • CREATE TABLE pacientes (
20     id INT AUTO_INCREMENT PRIMARY KEY,
21     nombre VARCHAR(100),
22     cedula VARCHAR(20),
23     fecha_nacimiento DATE
24 );
```

3. Insertar los registros necesarios utilizando INSERT INTO.

```
51 • USE turnos_medicos;
52 • INSERT INTO especialidades (nombre) VALUES
53     ('Pediatria'), ('Cardiología'), ('Dermatología'), ('Neurología');
54
55 -- REGISTROS PARA MÉDICOS
56 • INSERT INTO medicos (nombre, especialidad_id) VALUES
57     ('Dra. Ana Torres', 1),
58     ('Dr. Luis Pérez', 2),
59     ('Dra. Carla Gómez', 3),
60     ('Dr. Jorge Lima', 4);
61
62 -- REGISTROS PARA PACIENTES
63 • INSERT INTO pacientes (nombre, cedula, fecha_nacimiento) VALUES
64     ('María López', '1102233445', '1990-04-15'),
65     ('Pedro González', '1103344556', '1985-06-20'),
66     ('Lucía Martínez', '1104455667', '2002-09-10'),
67     ('Carlos Herrera', '1105566778', '1978-12-05');
68
69 -- REGISTROS PARA TURNOS
70 • INSERT INTO turnos (paciente_id, medico_id, fecha, hora) VALUES
71     (1, 1, CURDATE(), '10:00:00'),
72     (2, 2, CURDATE(), '11:00:00'),
73     (3, 3, CURDATE(), '12:00:00'),
74     (4, 4, CURDATE() + INTERVAL 1 DAY, '09:30:00');
75
```

#### 4. Crear únicamente las funciones definidas en el script002E

```
-- FUNCIONES
DELIMITER //

• CREATE FUNCTION obtener_edad(fecha_nac DATE)
  RETURNS INT
  DETERMINISTIC
  BEGIN
    RETURN TIMESTAMPDIFF(YEAR, fecha_nac, CURDATE());
  END;
//

• CREATE FUNCTION total_turnos_paciente(p_id INT)
  RETURNS INT
  DETERMINISTIC
  BEGIN
    DECLARE total INT;
    SELECT COUNT(*) INTO total FROM turnos WHERE paciente_id = p_id;
    RETURN total;
  END;
//

• CREATE FUNCTION nombre_medico(m_id INT)
  RETURNS VARCHAR(100)
  DETERMINISTIC
  BEGIN
    DECLARE nombre VARCHAR(100);
    SELECT nombre INTO nombre FROM medicos WHERE id = m_id;
    RETURN nombre;
  END;
//
```

Functions

- f() nombre\_medico
- f() obtener\_edad
- f() total\_turnos\_paciente
- f() turnos\_por\_dia

SQL File 6\*    pacientes    medicos    **obtener\_edad** ×

Limit to 1000 rows

```
1 • select turnos_medicos.obtener_edad('2006-02-23');
2
```

Result Grid    Filter Rows:    Export:    Wrap Cell Content:

turnos_medicos.obtener_edad('2006-02-23')
19

## 5. Crear únicamente los procedimientos almacenados.

```
-- PROCEDIMIENTOS
DELIMITER //

CREATE PROCEDURE registrar_paciente(IN nom VARCHAR(100), IN ced VARCHAR(20), IN fnac DATE)
BEGIN
    INSERT INTO pacientes(nombre, cedula, fecha_nacimiento)
    VALUES (nom, ced, fnac);
END;
//

CREATE PROCEDURE crear_turno(IN p_id INT, IN m_id INT, IN f DATE, IN h TIME)
BEGIN
    INSERT INTO turnos(paciente_id, medico_id, fecha, hora) VALUES (p_id, m_id, f, h);
END;
//

CREATE PROCEDURE cancelar_turno(IN t_id INT)
BEGIN
    DELETE FROM turnos WHERE id = t_id;
END;
//

CREATE PROCEDURE actualizar_estado_turno(IN t_id INT, IN nuevo_estado VARCHAR(20))
BEGIN
    UPDATE turnos SET estado = nuevo_estado WHERE id = t_id;
END;
//
DELIMITER ;
```

Stored Procedures

- actualizar\_estado\_turno
- cancelar\_turno
- crear\_turno
- registrar\_paciente

Call stored procedure turnos\_medicos.registrar\_paciente

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

<b>nom</b>	<input type="text" value="Raúl Pérez"/>	[IN] VARCHAR(100)
<b>ced</b>	<input type="text" value="0912723061"/>	[IN] VARCHAR(20)
<b>fnac</b>	<input type="text" value="1967-12-21"/>	[IN] DATE

	id	nombre	cedula	fecha_nacimiento
1	1	María López	1102233445	1990-04-15
2	2	Pedro González	1103344556	1985-06-20
3	3	Lucía Martínez	1104455667	2002-09-10
4	4	Carlos Herrera	1105566778	1978-12-05
5	5	Alessia Pérez	1750048942	2006-02-23
6	6	Raúl Pérez	1750048942	1976-12-21
	NULL	NULL	NULL	NULL

6. Crear únicamente los triggers.

```
-- TRIGGERS
DELIMITER //
CREATE TRIGGER trg_validar_fecha_turno
BEFORE INSERT ON turnos
FOR EACH ROW
BEGIN
    IF NEW.fecha < CURDATE() THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se pueden agendar turnos en fechas pasadas';
    END IF;
END;
//

CREATE TRIGGER trg_log_cambios_turnos
AFTER UPDATE ON turnos
FOR EACH ROW
BEGIN
    INSERT INTO log_turnos(turno_id, accion) VALUES (OLD.id, 'Actualización');
END;
//

CREATE TRIGGER trg_auditar_paciente_nuevo
AFTER INSERT ON pacientes
FOR EACH ROW
BEGIN
    INSERT INTO auditoria_pacientes(paciente_id, descripcion)
    VALUES (NEW.id, CONCAT('Nuevo paciente registrado: ', NEW.nombre));
END;
//
```

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

SHOW TRIGGERS;

1	•	use turnos_medicos;
2	•	SHOW TRIGGERS;

Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer	character_set_client	collation_connection	Database Collation
trg_auditar_paciente_nuevo	INSERT	pacientes	BEGIN INSERT INTO auditoria_pacientes(paci...	AFTER	2025-07-02 11:25:24.34	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
trg_validar_fecha_turno	INSERT	turnos	BEGIN IF NEW.fecha < CURDATE() THEN ...	BEFORE	2025-07-02 11:25:24.33	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
trg_auto_estado_turno	INSERT	turnos	BEGIN IF NEW.fecha < CURDATE() THEN ...	AFTER	2025-07-02 11:25:24.35	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
trg_log_cambios_turnos	UPDATE	turnos	BEGIN INSERT INTO log_turnos(turno_id, acc...	AFTER	2025-07-02 11:25:24.34	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci

## 7. Crear las vistas definidas.

```
-- VISTAS
CREATE VIEW vista_turnos_activos AS
SELECT t.id, p.nombre AS paciente, m.nombre AS medico, t.fecha, t.hora, t.estado
FROM turnos t
JOIN pacientes p ON t.paciente_id = p.id
JOIN medicos m ON t.medico_id = m.id
WHERE t.estado = 'pendiente';

CREATE VIEW vista_detalle_paciente AS
SELECT p.id, p.nombre, obtener_edad(p.fecha_nacimiento) AS edad,
       total_turnos_paciente(p.id) AS total_turnos
FROM pacientes p;

CREATE VIEW vista_disponibilidad_medicos AS
SELECT m.id, m.nombre, e.nombre AS especialidad
FROM medicos m
JOIN especialidades e ON m.especialidad_id = e.id
WHERE m.id NOT IN (
    SELECT medico_id FROM turnos WHERE fecha = CURDATE()
);

CREATE VIEW vista_citas_por_especialidad AS
SELECT e.nombre AS especialidad, COUNT(*) AS total_citas
FROM turnos t
JOIN medicos m ON t.medico_id = m.id
JOIN especialidades e ON m.especialidad_id = e.id
GROUP BY e.nombre;
```

Views

- vista\_citas\_por\_especialidad
- vista\_detalle\_paciente
- vista\_disponibilidad\_medicos
- vista\_turnos\_activos

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

result Grid		Filter Rows:	Export:	Wrap Cell Content:
especialidad	total_citas			
Pediatría	1			
Cardiología	1			
Dermatología	1			
Neurología	1			

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

turnos_medicos	
Tables	
auditoria_pacientes	
especialidades	
log_turnos	
medicos	
pacientes	
turnos	
Views	
vista_citas_por_especialidad	
vista_detalle_paciente	
vista_disponibilidad_medicos	
vista_turnos_activos	
Stored Procedures	
actualizar_estado_turno	
cancelar_turno	
crear_turno	
registrar_paciente	
Functions	
nombre_medico	
obtener_edad	
total_turnos_paciente	
turnos_por_dia	

## Parte 2: Configuración del proyecto Java en IntelliJ IDEA

1. Abrir IntelliJ IDEA y cargar el proyecto.
2. Restaurar el código Java del sistema.

```
package app;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class MainWindow extends JFrame {
    private JPanel panel; 12 usages
    private JButton btnRegistrarPaciente; 4 usages
    private JButton btnVerTurnos; 4 usages
    private JTextField txtNombre; 4 usages
    private JTextField txtCedula; 4 usages
    private JTextField txtFechaNacimiento; 4 usages
    private JTextArea txtResultados; 7 usages

    public MainWindow() { 1 usage
        setTitle("Sistema de Turnos Médicos");
        setSize( width: 600, height: 400);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        panel = new JPanel();
        panel.setLayout(null);

        JLabel lblNombre = new JLabel( text: "Nombre:");
        lblNombre.setBounds( x: 30, y: 20, width: 100, height: 25);
        panel.add(lblNombre);

        txtNombre = new JTextField();
        txtNombre.setBounds( x: 140, y: 20, width: 200, height: 25);
        panel.add(txtNombre);

        JLabel lblCedula = new JLabel( text: "Cédula:");
        lblCedula.setBounds( x: 30, y: 60, width: 100, height: 25);
        panel.add(lblCedula);

        txtCedula = new JTextField();
        txtCedula.setBounds( x: 140, y: 60, width: 200, height: 25);
        panel.add(txtCedula);
    }
}
```

```
package app;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnection { no usages
    private static final String URL = "jdbc:mysql://localhost:3306/turnos_medicos";
    private static final String USER = "root"; 1 usage
    private static final String PASSWORD = "1234"; 1 usage

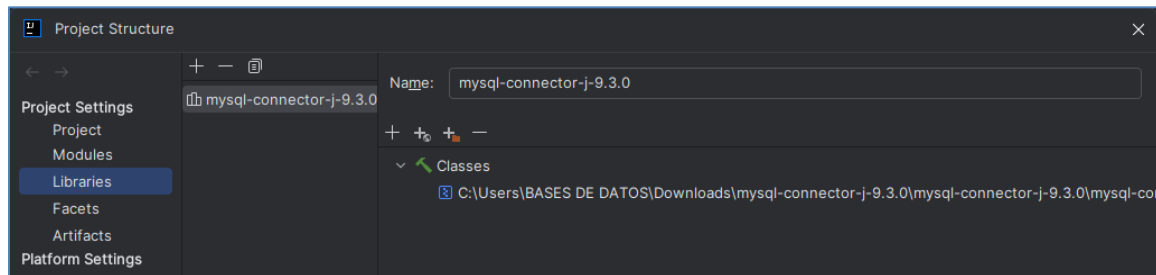
    public static Connection getConnection() throws SQLException { no usages
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

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 = app.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

Como tal no está presente en el código, solo está en el MySql.

```
▼ Functions
f() nombre_medico
f() obtener_edad
f() total_turnos_paciente
f() turnos_por_dia
```

## - Procedimientos almacenados

```
private void registrarPaciente() { 1 usage
    try (Connection conn = app.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());
    }
}
```

## - Triggers

Como tal no está presente en el código, solo está en el MySQL.

Result Grid   Filter Rows:   Edit:   Export/Import:				
	id	paciente_id	fecha_auditoria	descripcion
▶	1	5	2025-07-02 11:41:10	Nuevo paciente registrado: Alessia Pérez
	2	6	2025-07-02 11:53:36	Nuevo paciente registrado: Raúl Pérez
	3	7	2025-07-02 12:24:24	Nuevo paciente registrado: Isabel Palacios
*	NULL	NULL	NULL	NULL

## 7. Crear usuarios para probar las acciones del trigger

```
-- TRIGGERS
DELIMITER //
CREATE TRIGGER trg_validar_fecha_turno
BEFORE INSERT ON turnos
FOR EACH ROW
BEGIN
    IF NEW.fecha < CURDATE() THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se pueden agendar turnos en fechas pasadas';
    END IF;
END;
//
```



- `CREATE USER 'Alessi'@'localhost' IDENTIFIED BY '1234';`
- `GRANT ALL PRIVILEGES ON turnos_medicos.* TO 'Alessi'@'localhost';`
- `FLUSH PRIVILEGES;`

```
public class DBConnection { 2 usages
    private static final String URL = "jdbc:mysql://localhost:3306/turnos_medicos"; 1 usage
    private static final String USER = "Alessi"; 1 usage
    private static final String PASSWORD = "1234"; 1 usage

    public static Connection getConnection() throws SQLException { 2 usages
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

**Sistema de Turnos Médicos**

Nombre:

Cédula:

Fecha Nac (YYYY-MM-DD):

Paciente registrado correctamente.

Result Grid				
		Filter Rows:		
		Edit:		
		Export/Import:		
	id	paciente_id	fecha_auditoria	descripcion
▶	1	5	2025-07-02 11:41:10	Nuevo paciente registrado: Alessia Pérez
	2	6	2025-07-02 11:53:36	Nuevo paciente registrado: Raúl Pérez
	3	7	2025-07-02 12:24:24	Nuevo paciente registrado: Isabel Palacios
*	NULL	NULL	NULL	NULL

**Capturar pantallas.**

## **CONCLUSIONES**

Mediante esta práctica se pudo aplicar una correcta conexión de una base de datos en este caso de MySQL a Java, y los cambios que se realicen en la base de datos pueden ser auditados, mediante un trigger, así mismo se aplicó lo que son funciones, procedimientos almacenados y demás, es así como se pueden combinar base de datos y proyectos.

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