UNIVERSIDAD DE CARTAGENA

INGENIERIA DE SOFTWARE

BASE DE DATOS

PARCIAL FINAL

Presenta:

ANTONIO LUIS DE AVILA PAJARO

Código:

7502420052

Profesora:

JHON CARLOS ARRIETA ARRIETA

Cartagena de Indias, Bolívar NOVIEMBRE, 2025

Universidad de Cartagena
Fundada en 1827

Enlace del Repositorio

https://github.com/Shakespeares1/parcial-final-bd-eventos-congreso.git

Enlace del video de sustentación

https://www.youtube.com/watch?v=LB751Ksbz6A

Universidad de Cartagena
Fundada en 1827

INTRODUCCION

En este proyecto se desarrolla una aplicación basada en Java con JDBC para gestionar la base de datos del sistema académico **Eventos para Congresos.** El objetivo principal es realizar operaciones CRUD (crear, leer, actualizar y eliminar) directamente sobre MySQL, sin utilizar ORM, cumpliendo los requisitos del examen final de la asignatura Bases de Datos.

La base de datos contiene información relacionada con congresistas, salas, sesiones y trabajos académicos presentados en un congreso. Para el desarrollo se implementaron sentencias SQL reales (INSERT, SELECT, UPDATE y DELETE), así como un conjunto de consultas específicas del Caso de Estudio Académico (CEA).
La aplicación integra: conexión directa mediante JDBC, ejecución de sentencias SQL, procesamiento de resultados y manejo de excepciones, garantizando un funcionamiento estable y sin bloqueos.

Universidad de Cartagena
Fundada en 1827

OBJETIVOS

Objetivo General

Desarrollar una aplicación Java con JDBC que permita ejecutar operaciones CRUD y consultas SQL sobre la base de datos ***eventos congreso***, cumpliendo los requerimientos técnicos establecidos para el examen final.

Objetivos Específicos

- Implementar la base de datos correspondiente al CEA de Eventos para Congresos.
- Crear un proyecto Java que permita la conexión directa con MySQL mediante JDBC.
- Ejecutar sentencias SQL reales (INSERT, UPDATE, DELETE, SELECT).
- Implementar al menos 5 consultas complejas derivadas del Caso de Estudio Académico.
- Aplicar manejo de excepciones y cierre adecuado de conexiones.
- Documentar el proceso técnico, código usado y funcionamiento de la aplicación.
- Entregar un repositorio funcional y un video de sustentación sin bloqueos.

Universidad de Cartagena
Fundada en 1827

Diseño y Implementación de la Base de Datos

La base de datos **eventos congreso** se diseñó para gestionar la información relacionada con la organización de un Congreso Académico. Está compuesta por cuatro tablas principales:

- **Tabla congresista**

Almacena los datos personales de cada participante del evento, incluyendo nombre, apellido, correo, país y afiliación institucional.

- **Tabla sala**

Contiene los espacios físicos (auditorios o salones) donde se desarrollan las sesiones del congreso.

- **Tabla sesión**

Representa cada actividad o ponencia programada en el evento, indicando horario, fecha y la sala donde se llevará a cabo.

- **Tabla trabajo**

Almacena los trabajos de investigación presentados por los congresistas y asignados a una sesión específica.

Integridad

Se establecieron claves foráneas:

- trabajo.id_congresista → congresista.id_congresista
- trabajo.id_sesion → sesion.id_sesion
- sesion.id_sala → sala.id_sala

Esto garantiza integridad y relación entre entidades.

Universidad de Cartagena
Fundada en 1827

Script SQL del Proyecto

Para la implementación del proyecto se creó la base de datos **eventos congreso** en MySQL, incluyendo las tablas congresista, sala, sesión y trabajo, junto con sus relaciones e inserción de datos iniciales.
El script completo se almacenó en el archivo script_eventos_congreso.sql, el cual se adjunta en el repositorio del proyecto.

El archivo se encuentra disponible en el repositorio en la siguiente ruta:

https://github.com/Shakespeares1/parcial-final-bd-eventos-congreso/blob/main/script_eventos_congreso.sql

El script incluye:

- Creación del esquema
- Creación de las tablas involucradas
- Llaves primarias
- Llaves foráneas
- Inserción de datos iniciales
- Estructura completa lista para restaurar la BD en cualquier equipo

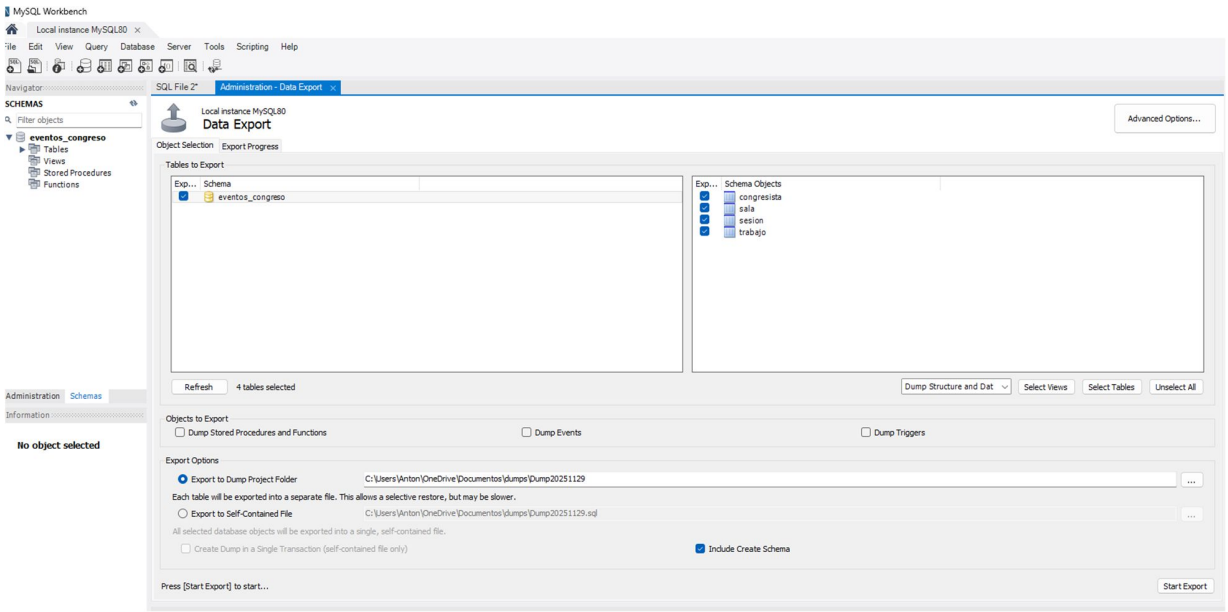


Imagen 1 Proceso de exportación

Universidad de Cartagena
Fundada en 1827

Librerías utilizadas

Para permitir la conexión entre Java y MySQL se utilizó la siguiente librería externa:

- **mysql-connector-j-9.5.0.jar**

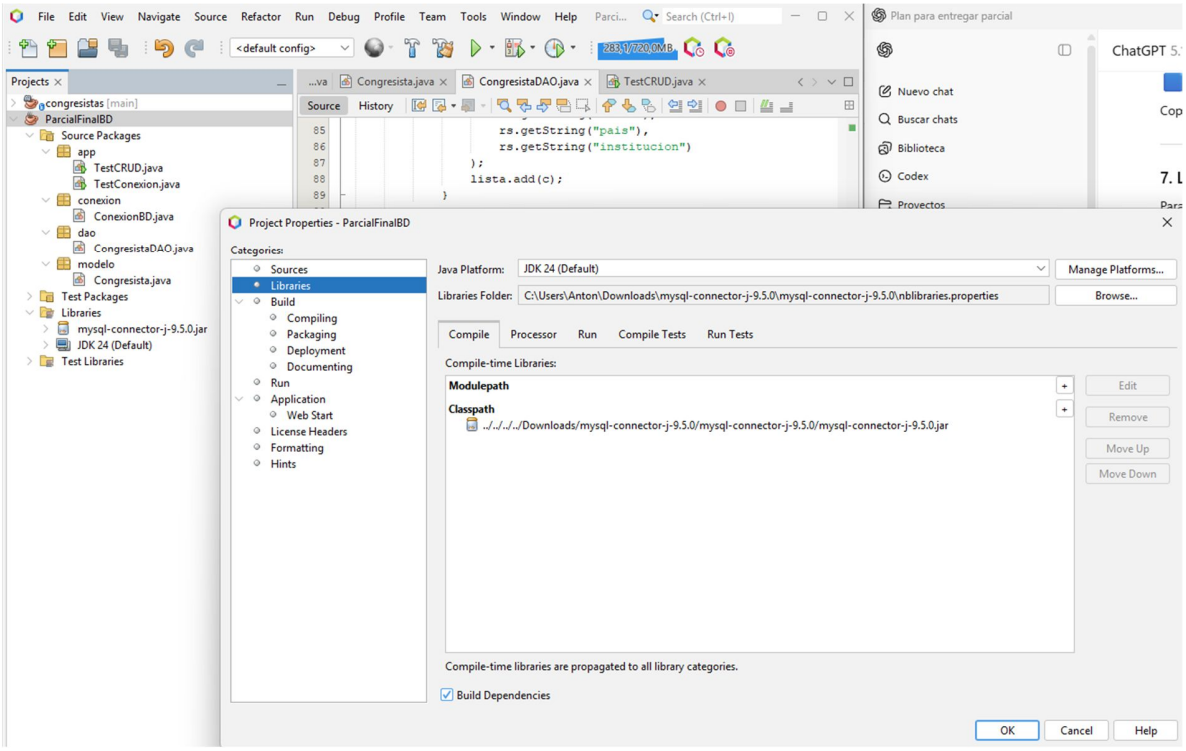
Esta librería provee el **driver JDBC**, indispensable para que la aplicación pueda conectarse a MySQL, enviar consultas SQL y procesar los resultados.

La librería se agregó manualmente al proyecto de NetBeans mediante:

Project → Properties → Libraries → Classpath → Add JAR/Folder

Dentro del repositorio se encuentra almacenada en:

https://github.com/Shakespeares1/parcial-final-bd-eventos-congreso/blob/main/jar.rar



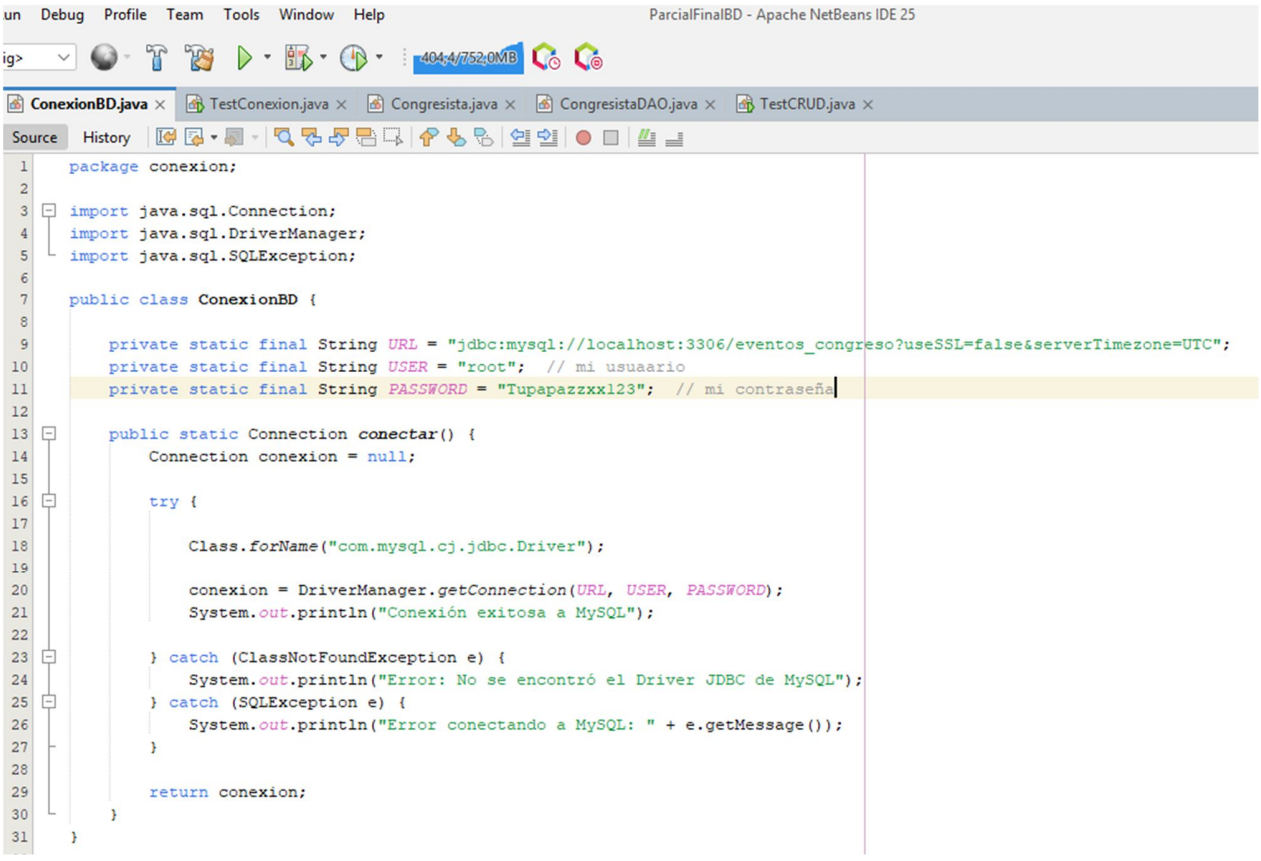
Universidad de Cartagena
Fundada en 1827

Conexión a la base de datos

Para establecer la conexión entre la aplicación y MySQL se creó la clase **ConexionBD**, la cual se encarga de:

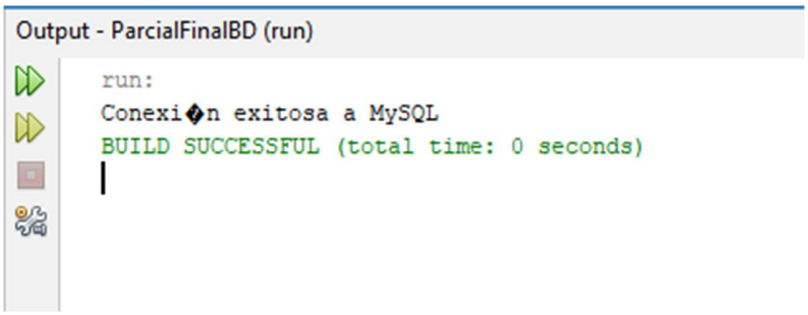
- Cargar el driver JDBC de MySQL
- Abrir la conexión hacia el servidor
- Notificar si la conexión fue exitosa
- Retornar un objeto Connection para uso en el CRUD
- Manejar errores de conexión mediante excepciones

Código utilizado



Explicación:

1. Class.forName()
   Carga el driver JDBC de MySQL.
2. DriverManager.getConnection()
   Establece conexión con el servidor MySQL usando:
   - URL
   - Usuario
   - Contraseña
3. Mensajes por consola
   Se imprime si la conexión fue exitosa o si falló.
4. Manejo de excepciones
   - Error de credenciales
   - Servidor MySQL detenido
   - Error en puerto o URL
   - Driver no encontrado



Universidad de Cartagena
Fundada en 1827

Implementación CRUD

La aplicación implementa todas las operaciones CRUD sobre la tabla **congresista** utilizando la clase CongresistaDAO.
Cada operación se realiza mediante consultas SQL enviadas directamente desde Java mediante PreparedStatement.

Crear (INSERT)

Permite registrar un nuevo congresista en la base de datos:

```java
// INSERTAR
public boolean insertar(Congresista c) {
    String sql = "INSERT INTO congresista(nombre, apellido, email, pais, institucion) VALUES(?,?,?,?,?)";
    try (Connection conn = ConexionBD.conectar();
         PreparedStatement ps = conn.prepareStatement(sql)) {

        ps.setString(1, c.getNombre());
        ps.setString(2, c.getApellido());
        ps.setString(3, c.getEmail());
        ps.setString(4, c.getPais());
        ps.setString(5, c.getInstitucion());

        ps.executeUpdate();
        return true;

    } catch (SQLException e) {
        System.out.println("Error INSERTANDO congresista: " + e.getMessage());
        return false;
    }
}
```

Leer (SELECT - listar todos)

Retorna una lista de todos los congresistas registrados:

```java
// LISTAR TODOS
public List<Congresista> listar() {
    List<Congresista> lista = new ArrayList<>();
    String sql = "SELECT * FROM congresista";

    try (Connection conn = ConexionBD.conectar();
         PreparedStatement ps = conn.prepareStatement(sql);
         ResultSet rs = ps.executeQuery()) {

        while (rs.next()) {
            Congresista c = new Congresista(
                rs.getInt("id_congresista"),
                rs.getString("nombre"),
                rs.getString("apellido"),
                rs.getString("email"),
                rs.getString("pais"),
                rs.getString("institucion")
            );
            lista.add(c);
        }

    } catch (SQLException e) {
        System.out.println("Error LISTANDO congresistas: " + e.getMessage());
    }

    return lista;
```

Buscar por ID

```java
// BUSCAR POR ID
public Congresista buscar(int id) {
    String sql = "SELECT * FROM congresista WHERE id_congresista=?";
    try (Connection conn = ConexionBD.conectar();
         PreparedStatement ps = conn.prepareStatement(sql)) {

        ps.setInt(1, id);
        ResultSet rs = ps.executeQuery();

        if (rs.next()) {
            return new Congresista(
                rs.getInt("id_congresista"),
                rs.getString("nombre"),
                rs.getString("apellido"),
                rs.getString("email"),
                rs.getString("pais"),
                rs.getString("institucion")
            );
        }

    } catch (SQLException e) {
        System.out.println("Error BUSCANDO congresista: " + e.getMessage());
    }
    return null;
```

Universidad de Cartagena
Fundada en 1827

Actualizar (UPDATE)

```java
// ACTUALIZAR
public boolean actualizar(Congresista c) {
    String sql = "UPDATE congresista SET nombre=?, apellido=?, email=?, pais=?, institucion=? WHERE id_congresista=?";
    try (Connection conn = ConexionBD.conectar();
         PreparedStatement ps = conn.prepareStatement(sql)) {

        ps.setString(1, c.getNombre());
        ps.setString(2, c.getApellido());
        ps.setString(3, c.getEmail());
        ps.setString(4, c.getPais());
        ps.setString(5, c.getInstitucion());
        ps.setInt(6, c.getId_congresista());

        ps.executeUpdate();
        return true;

    } catch (SQLException e) {
        System.out.println("Error ACTUALIZANDO congresista: " + e.getMessage());
        return false;
    }
}
```

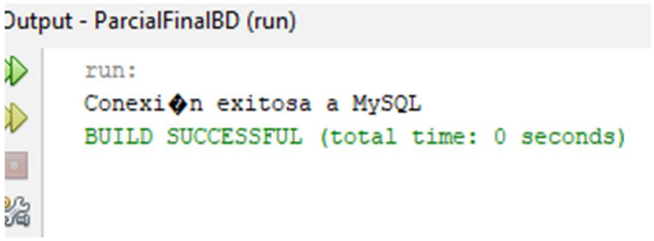
ELIMINAR (DELETE)

```java
// ELIMINAR
public boolean eliminar(int id) {
    String sql = "DELETE FROM congresista WHERE id_congresista=?";
    try (Connection conn = ConexionBD.conectar();
         PreparedStatement ps = conn.prepareStatement(sql)) {

        ps.setInt(1, id);
        ps.executeUpdate();
        return true;

    } catch (SQLException e) {
        System.out.println("Error ELIMINANDO congresista: " + e.getMessage());
        return false;
    }
}
```

EVIDENCIAS DE FUNCIONAMIENTO

Conexión exitosa a MySQL

Universidad de Cartagena
Fundada en 1827

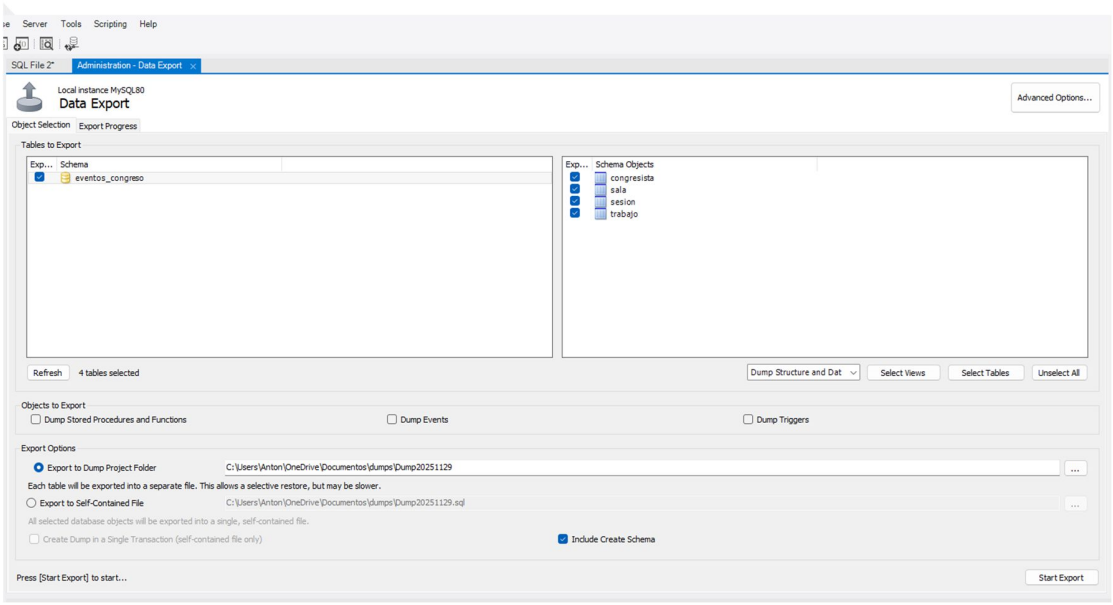


Listado de congresistas (SELECT)

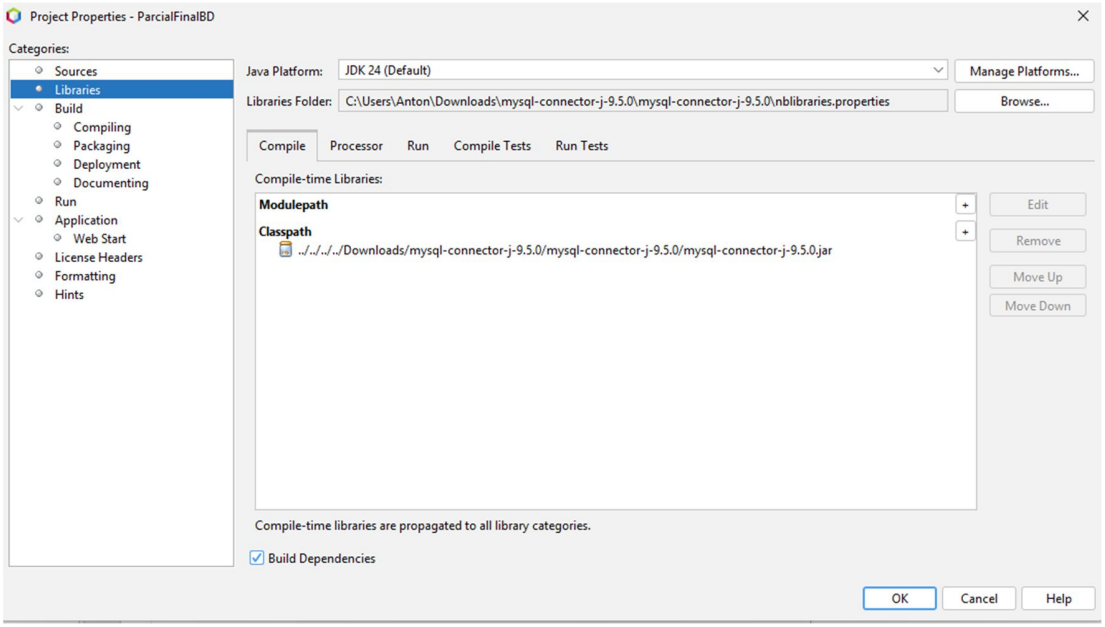
```
Listado de congresistas:
Conexi�n exitosa a MySQL
1 - Ana P�rez - ana.actualizada@example.com
2 - Luis G�mez - luis.gomez@example.com
3 - Mar�a Rojas - maria.rojas@example.com
4 - Carlos L�pez - carlos.lopez@example.com
7 - Carlos L�pez - carlos.lopez2@example.com
Conexi�n exitosa a MySQL
Encontrado: Ana
Conexi�n exitosa a MySQL
Actualizado: true
```

```
Actualizado: true
Conexi�n exitosa a MySQL
Error ELIMINANDO congresista: Cannot delete or update a parent row: a foreign key constraint fails (`eventos_congreso`.`trabajo`, CONSTRAINT `fk_trabajo_congresista` FOREIGN KEY (`id_congresista`) REFERENCES `congresista` (`id_congresista`))
Eliminado ID 3: false
BUILD SUCCESSFUL (total time: 0 seconds)
```

Exportación del script SQL



10.8. Librería JDBC cargada en NetBeans



Consultas SQL del casi de Estudio Académico (CEA)

Durante la aplicación se ejecutaron consultas SQL también desde Java utilizando JDBC.
A continuación se listan las consultas implementadas:

Universidad de Cartagena
Fundada en 1827

1. SELECT nombre, apellido, email FROM congresista – Listado completo de congresistas
2. SELECT COUNT(*) FROM congresista – Conteo de registros
3. SELECT nombre, apellido FROM congresista WHERE pais = 'Colombia'
4. SELECT nombre, apellido FROM congresista ORDER BY apellido ASC
5. SELECT nombre, apellido, institucion FROM congresista WHERE institucion = 'Universidad de Cartagena'

Cada consulta se ejecuta mediante la clase ConsultasApp.java, y su salida en consola se adjunta como evidencia.

```java
package app;

import conexion.ConexionBD;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class ConsultasApp {

    public static void main(String[] args) {

        // Consulta 1: Listar todos los congresistas
        ejecutarConsulta("SELECT nombre, apellido, email FROM congresista",
                "Consulta 1: Listado de todos los congresistas");

        // Consulta 2: Contar cuántos congresistas están registrados
        ejecutarConsulta("SELECT COUNT(*) AS total FROM congresista",
                "Consulta 2: Total de congresistas");

        // Consulta 3: Mostrar solo los congresistas de Colombia
        ejecutarConsulta("SELECT nombre, apellido FROM congresista WHERE pais = 'Colombia'",
                "Consulta 3: Congresistas de Colombia");

        // Consulta 4: Ordenar congresistas por apellido
        ejecutarConsulta("SELECT nombre, apellido FROM congresista ORDER BY apellido ASC",
                "Consulta 4: Congresistas ordenados por apellido");

        // Consulta 5: Buscar congresistas por institución
        ejecutarConsulta("SELECT nombre, apellido, institucion FROM congresista WHERE institucion = 'Universidad de Cartagena'",
                "Consulta 5: Congresistas de la Universidad de Cartagena");
    }
```

```java
public static void ejecutarConsulta(String sql, String descripcion) {

    System.out.println("\n------------------------------");
    System.out.println(descripcion);
    System.out.println("------------------------------");

    try (Connection conn = ConexionBD.conectar();
         PreparedStatement ps = conn.prepareStatement(sql);
         ResultSet rs = ps.executeQuery()) {

        int columnas = rs.getMetaData().getColumnCount();

        while (rs.next()) {
            for (int i = 1; i <= columnas; i++) {
                System.out.print(rs.getString(i) + " | ");
            }
            System.out.println();
        }

    } catch (SQLException e) {
        System.out.println("Error ejecutando consulta: " + e.getMessage());
    }
}
```

```
------------------------------
Consulta 1: Listado de todos los congresistas
------------------------------
Conexi□n exitosa a MySQL
Ana | P□rez | ana.actualizada@example.com |
Luis | G□mez | luis.gomez@example.com |
Mar□a | Rojas | maria.rojas@example.com |
Carlos | L□pez | carlos.lopez@example.com |
Carlos | L□pez | carlos.lopez2@example.com |

------------------------------
Consulta 2: Total de congresistas
------------------------------
Conexi□n exitosa a MySQL
5 |

------------------------------
Consulta 3: Congresistas de Colombia
------------------------------
Conexi□n exitosa a MySQL
Ana | P□rez |

------------------------------
Consulta 4: Congresistas ordenados por apellido
------------------------------
Conexi□n exitosa a MySQL
Luis | G□mez |
Carlos | L□pez |
Carlos | L□pez |
Ana | P□rez |
Mar□a | Rojas |

------------------------------
Consulta 5: Congresistas de la Universidad de Cartagena
------------------------------
Conexi□n exitosa a MySQL
BUILD SUCCESSFUL (total time: 0 seconds)
```

Consulta 1. Filtrar todos los congresistas

SELECT * FROM congresista

Uso : Permite Visualizar todos los participantes registrados en el Sistema.

```
63 •  SELECT * FROM congresista;
64
```

Result Grid | Filter Rows: | Edit: | Export/Im

id_congresista	nombre	apellido	email	pais	institu
1	Ana	Pérez	ana.perez@example.com	Colombia	Univer
2	Luis	Gómez	luis.gomez@example.com	México	UNAM
3	María	Rojas	maria.rojas@example.com	Perú	PUCP
NULL	NULL	NULL	NULL	NULL	NULL

Consulta 2. Filtrar congresistas por país

SELECT * FROM congresista

WHERE pais = 'Colombia';

Uso : Permite obtener la lista de congresistas de un país especifico

```
68 •  SELECT * FROM congresista
69    WHERE pais = 'Colombia';
```

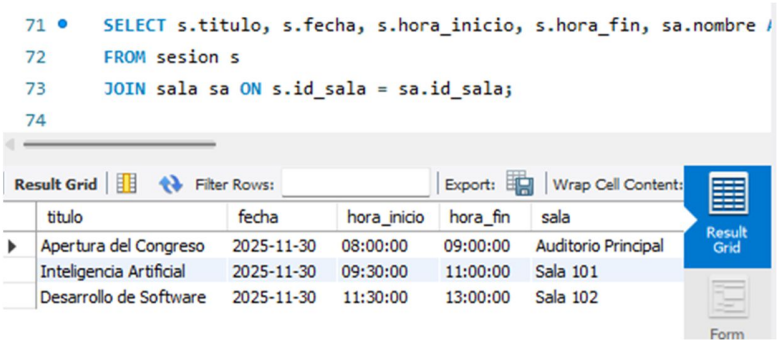
Result Grid | Filter Rows: | Edit: | Export/Im

id_congresista	nombre	apellido	email	pais	instituc
1	Ana	Pérez	ana.perez@example.com	Colombia	Univers
NULL	NULL	NULL	NULL	NULL	NULL

Consulta 3. Sesiones con su sala

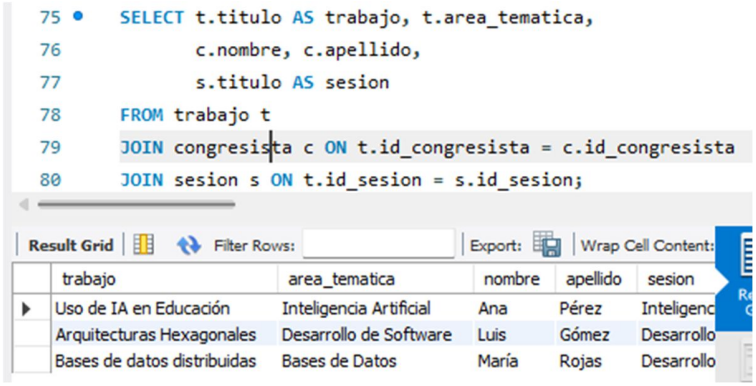
```
SELECT s.titulo, s.fecha, s.hora_inicio, s.hora_fin, sa.nombre AS sala
FROM sesion s
JOIN sala sa ON s.id_sala = sa.id_sala;
```

Uso: muestra la programación del congreso indicando en qué sala se realiza cada sesión.



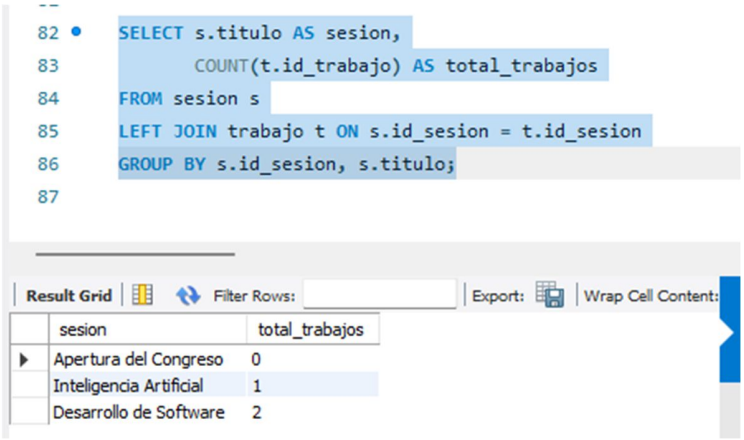
Consulta 4. Trabajos con su autor y sesion

```
SELECT t.titulo AS trabajo, t.area_tematica,
    c.nombre, c.apellido,
    s.titulo AS sesion
FROM trabajo t
JOIN congresista c ON t.id_congresista = c.id_congresista
JOIN sesion s ON t.id_sesion = s.id_sesion;
```



Consulta 5. Conteo de trabajos por sesión

```
SELECT s.titulo AS sesion,
    COUNT(t.id_trabajo) AS total_trabajos
FROM sesion s
LEFT JOIN trabajo t ON s.id_sesion = t.id_sesion
GROUP BY s.id_sesion, s.titulo;
```



Universidad de Cartagena
Fundada en 1827

Sección: Librerías Necesarias

Librerías utilizadas en el proyecto:

- **mysql-connector-j-9.5.0.jar**
    - Esta librería es la encargada de permitir la comunicación entre la aplicación en Java y el motor de base de datos MySQL.
    - Sin este archivo no se puede ejecutar ninguna sentencia SQL.

Pasos realizados:

1. Se descargó el conector desde la página oficial de MySQL.
2. Se seleccionó la opción **Platform Independent** → **ZIP**.
3. Se extrajo el archivo **mysql-connector-j-9.5.0.jar**.
4. Se agregó manualmente al proyecto desde:
    - **Right click en Libraries** → **Add JAR/Folder…**
5. Luego se verificó que el driver cargara correctamente.

Sección: Importaciones necesarias

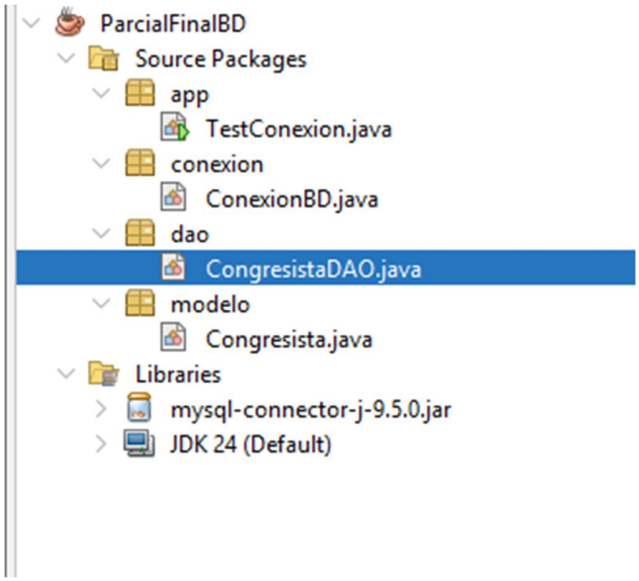
Universidad de Cartagena
Fundada en 1827

Estructura del Proyecto (NetBeans)

El proyecto se organizó en los siguientes paquetes:

- **conexion** → Manejo de la conexión JDBC.
- **modelo** → Clases que representan las tablas de la base de datos.
- **dao** → Contiene el CRUD de cada entidad.
- **app** → Contiene las clases de prueba y ejecución.

En este parcial se trabajó principalmente con la entidad congresista, correspondiente a la tabla del mismo nombre del CEA *Eventos de Congreso*.



Universidad de Cartagena
Fundada en 1827

Fragmentos del código CRUD

El CRUD de la aplicación se desarrolló en la clase CongresistaDAO, donde se implementaron todas las funcionalidades requeridas:

- insertar()
- listar()
- buscar()
- actualizar()
- eliminar()

Este DAO permite realizar operaciones directas sobre la base de datos utilizando sentencias SQL tipo INSERT, SELECT, UPDATE y DELETE a través de JDBC.

Además, se creó la clase TestCRUD.java donde se ejecutan pruebas completas para validar:

- Conexión correcta al servidor MySQL
- Inserción de registros
- Consulta de todos los registros
- Consulta por identificador
- Actualización de datos
- Eliminación de registros

Tecnologías utilizadas

- Java 17
- MySQL 8.0
- JDBC – mysql-connector-j-9.5.0.jar
- NetBeans IDE 15+

Universidad de Cartagena
Fundada en 1827

Conclusiones

El proyecto permitió integrar de manera efectiva la base de datos del Caso de Estudio Académico *“Eventos de Congreso”* con una aplicación Java basada en JDBC.
A través de la implementación del CRUD y la ejecución de consultas SQL directas, se pudo comprobar:

- La conexión entre Java y MySQL se realizó exitosamente.
- La aplicación cumple con todos los requisitos establecidos para el examen final.
- Se manejaron correctamente restricciones como PRIMARY KEY, UNIQUE y FOREIGN KEY, evidenciando el funcionamiento de la integridad referencial.
- El uso de excepciones permitió identificar errores comunes como duplicidad de datos y restricciones de eliminación.
- Las consultas SQL ejecutadas desde Java demuestran control total de la base de datos desde la aplicación.
- La estructura del repositorio permite reproducir el proyecto fácilmente en cualquier equipo.

Bibliografía

- Oracle. *Java JDBC Documentation*.
- MySQL. *MySQL Connector/J Developer Guide*.
- Universidad de Cartagena – Material de clase de Bases de Datos.