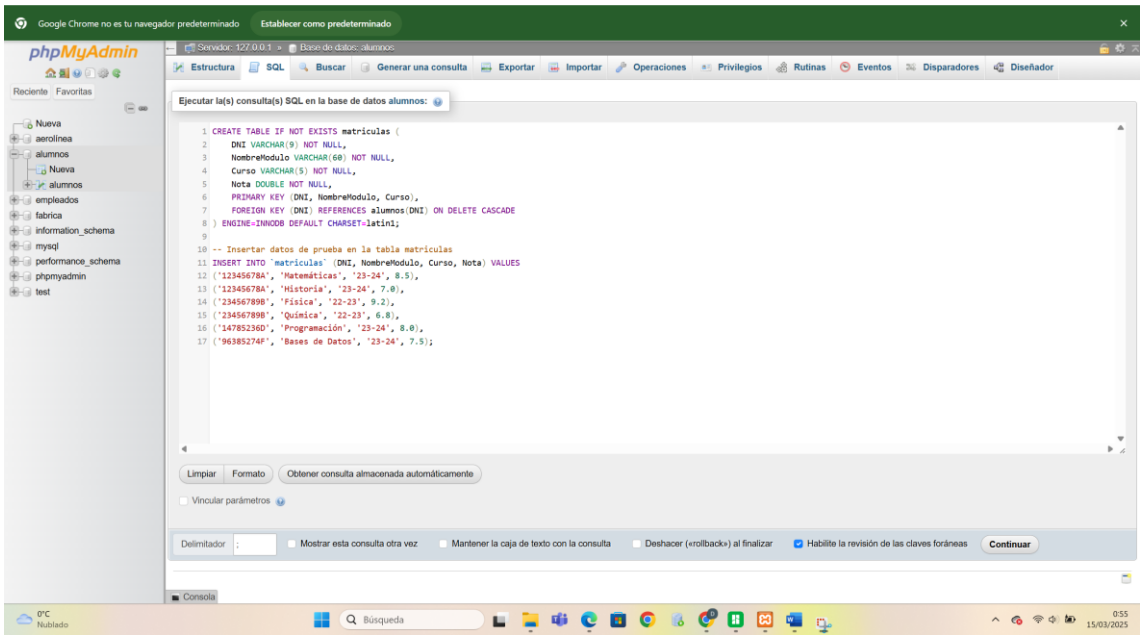
**Enunciado.**

**Los chicos de BK están aprendiendo a hacer componentes de acceso a datos. Están practicando con los datos de la matrícula de los alumnos de la base de datos con la que has estado trabajando durante esta unidad y necesitan que les eches una mano, en concreto te piden que hagas lo siguiente:**

- **Debes añadir una tabla a la base de datos alumnos que represente las matrículas de los alumnos. Consta de los siguientes campos:**
  - **DNI: varchar(9).**
  - **NombreMódulo: varchar(60).**
  - **Curso: varchar(5), el curso se forma con los dos años que lo componen separados por un guión, por ejemplo 11-12.**
  - **Nota: double.**

**Recuerda rellenar la tabla con algunos datos para que puedas hacer pruebas.**



- **Crea un componente nuevo en el proyecto Alumno que para gestionar toda esta información. Además del código necesario para gestionar las propiedades del componente y mantener la información de la base de datos en un vector interno, es preciso que incluyas los siguientes métodos:**
    - **seleccionarFila(i): recupera en las propiedades del componente el registro número i del vector.**
    - **RecargarDNI(): recarga la estructura interna del componente con las matrículas de un DNI en particular.**
    - **AddMatricula(): añade un registro nuevo a la base de datos con la información almacenada en las propiedades del componente.**
    - **Dado que el componente puede funcionar en dos modos diferentes (todos los alumnos o un alumno concreto) se generará un evento cada vez que se cambie de modo, es decir, cuando se carguen todas las matrículas se lanzará un evento que lo señale y cuando se carguen las matrículas para un solo alumno también.**

Antes de nada se incorporará el conector de MySQL en el proyecto que acabamos de crear, y configuramos la conexión con XAMPP para conectar a la base de datos de PHPMyAdmin.

**1. Definición de la Clase MatriculaBean**

Añadimos clase MatriculaBean como un componente JavaBean que gestiona una lista de matrículas y proporciona funcionalidad para agregar, recuperar y filtrar matrículas conectándose a la base de datos alumnos.

Esta clase implementa Serializable lo que permite que este componente pueda ser reutilizado. Creamos un constructor vacío, e implementamos los atributos que permitirán almacenar los datos de matrícula.

Creamos un vector de matrículas que permite recoger una lista de matrículas y un manejador de eventos que lanzará un evento cada vez que se cambie de modo:

```
/**
 *
 * @author David
 */
public class MatriculaBean implements Serializable {

    private PropertyChangeSupport propertySupport;

    protected String DNI;
    protected String NombreModulo;
    protected String Curso;
    protected double Nota;
    protected Vector<Matricula> matriculas = new Vector<>();
    protected MatriculaModoListener receptor;

    public MatriculaBean() {
        propertySupport = new PropertyChangeSupport(sourceBean: this);
    }
```

Incorporamos los métodos getter y setter de los atributos del MatriculaBean, para poder acceder y modificar estas propiedades:

```
        /**
         * Get the value of DNI
         *
         * @return the value of DNI
         */
        public String getDNI() {
            return DNI;
        }

        /**
         * Set the value of DNI
         *
         * @param DNI new value of DNI
         */
        public void setDNI(String DNI) {
            this.DNI = DNI;
        }

        /**
         * Get the value of NombreModulo
         *
         * @return the value of NombreModulo
         */
        public String getNombreModulo() {
            return NombreModulo;
        }

        /**
         * Set the value of NombreModulo
         *
         * @param NombreModulo new value of NombreModulo
         */
        public void setNombreModulo(String NombreModulo) {
            this.NombreModulo = NombreModulo;
        }

        /**
         * Get the value of Curso
         *
         * @return the value of Curso
         */
        public String getCurso() {
            return Curso;
        }

    /**
     * Set the value of Curso
     *
     * @param Curso new value of Curso
     */
    public void setCurso(String Curso) {
        this.Curso = Curso;
    }

    /**
     * Get the value of Nota
     *
     * @return the value of Nota
     */
    public double getNota() {
        return Nota;
    }

    /**
     * Set the value of Nota
     *
     * @param Nota new value of Nota
     */
    public void setNota(double Nota) {
        this.Nota = Nota;
    }

    public Vector<Matricula> getMatriculas() {
        return matriculas;
    }
```

## 2. Clase Interna Matricula

Esta clase representa una matrícula individual con sus atributos y métodos de acceso.

```java
//Clase interna Matricula, representa a cada matricula con sus correspondientes atributos y métodos getter y setter:
public class Matricula {

    String DNI;
    String NombreModulo;
    String Curso;
    double Nota;

    public Matricula(String DNI, String NombreModulo, String Curso, double Nota) {
        this.DNI = DNI;
        this.NombreModulo = NombreModulo;
        this.Curso = Curso;
        this.Nota = Nota;
    }

    public String getDNI() {
        return DNI;
    }

    public void setDNI(String DNI) {
        this.DNI = DNI;
    }

    public String getNombreModulo() {
        return NombreModulo;
    }

    public void setNombreModulo(String NombreModulo) {
        this.NombreModulo = NombreModulo;
    }

    public String getCurso() {
        return Curso;
    }

    public void setCurso(String Curso) {
        this.Curso = Curso;
    }

    public double getNota() {
        return Nota;
    }

    public void setNota(double Nota) {
        this.Nota = Nota;
    }
```

## 3. Método recargarMatriculas()

Este método obtiene todas las matrículas de la base de datos y las almacena en el vector matriculas.

```java
// Método para cargar todas las matriculas
public void recargarMatriculas() throws ClassNotFoundException {
    matriculas.clear();
    try (Connection con = DriverManager.getConnection(url:"jdbc:mysql://localhost:3306/alumnos", user:"root", password:""); Statement st = con.createStatement()) {
        ResultSet rs = st.executeQuery(string: "SELECT * FROM matriculas");
        while (rs.next()) {
            Matricula m = new Matricula(
                    DNI:rs.getString(string: "DNI"),
                    NombreModulo:rs.getString(string: "NombreModulo"),
                    Curso: rs.getString(string: "Curso"),
                    Nota:rs.getDouble(string: "Nota"));
            matriculas.add(e: m);
        }
        // Lanzar el evento de cambio de modo (modo todas las matriculas)
        if (receptor != null) {
            receptor.cambiarModo(modo:"todos");
        }
    } catch (SQLException ex) {
        Logger.getLogger(name:MatriculaBean.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}
```

## 4. Método recargarDNI(String DNI)

Este método carga en el vector matriculas solo las matrículas asociadas a un DNI. Recibe un DNI por parámetro y busca las matrículas para ese DNI:

```java
// Método para recargar matriculas por DNI
public void recargarDNI(String DNI) {

    matriculas.clear();
    String query = "SELECT * FROM matriculas WHERE DNI = ?";
    try (Connection conn = DriverManager.getConnection(url:"jdbc:mysql://localhost:3306/alumnos", user:"root", password:""); PreparedStatement st = conn.prepareStatement(string: query)) {

        // Establecer el parámetro en la consulta (evita SQL injection)
        st.setString(i: 1, string: DNI);

        // Ejecutar la consulta
        try (ResultSet rs = st.executeQuery()) {
            while (rs.next()) {
                // Crear un objeto Matricula y añadirlo al vector
                Matricula m = new Matricula(
                        DNI:rs.getString(string: "DNI"),
                        NombreModulo:rs.getString(string: "NombreModulo"),
                        Curso:rs.getString(string: "Curso"),
                        Nota:rs.getDouble(string: "Nota"));
                matriculas.add(e: m);
            }
        }
        // Lanzar el evento de cambio de modo (modo completo)
        if (receptor != null) {
            receptor.cambiarModo(modo:"DNI");
        }
    } catch (SQLException ex) {
        Logger.getLogger(name:MatriculaBean.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}
```

### 5. Método addMatricula()

Este método inserta una nueva matrícula en la base de datos.

```java
public void addMatricula() throws ClassNotFoundException {

    try {
        Connection conn = DriverManager.getConnection(url:"jdbc:mysql://localhost:3306/alumnos", user:"root", password:"");
        PreparedStatement pdst = conn.prepareStatement(string: "INSERT INTO matriculas (DNI, NombreModulo, Curso, Nota) VALUES (?,?,?,?)");
        pdst.setString(i: 1, string: DNI);
        pdst.setString(i: 2, string: NombreModulo);
        pdst.setString(i: 3, string: Curso);
        pdst.setDouble(i: 4, d: Nota);

        // Ejecutar la inserción y obtener el número de filas afectadas
        int columnasAfectadas = pdst.executeUpdate();

        if (columnasAfectadas > 0) {
            System.out.println(x: "Matricula añadida correctamente.");
        }

    } catch (SQLException ex) {
        Logger.getLogger(name:MatriculaBean.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}
```

### 6. Manejador de Eventos

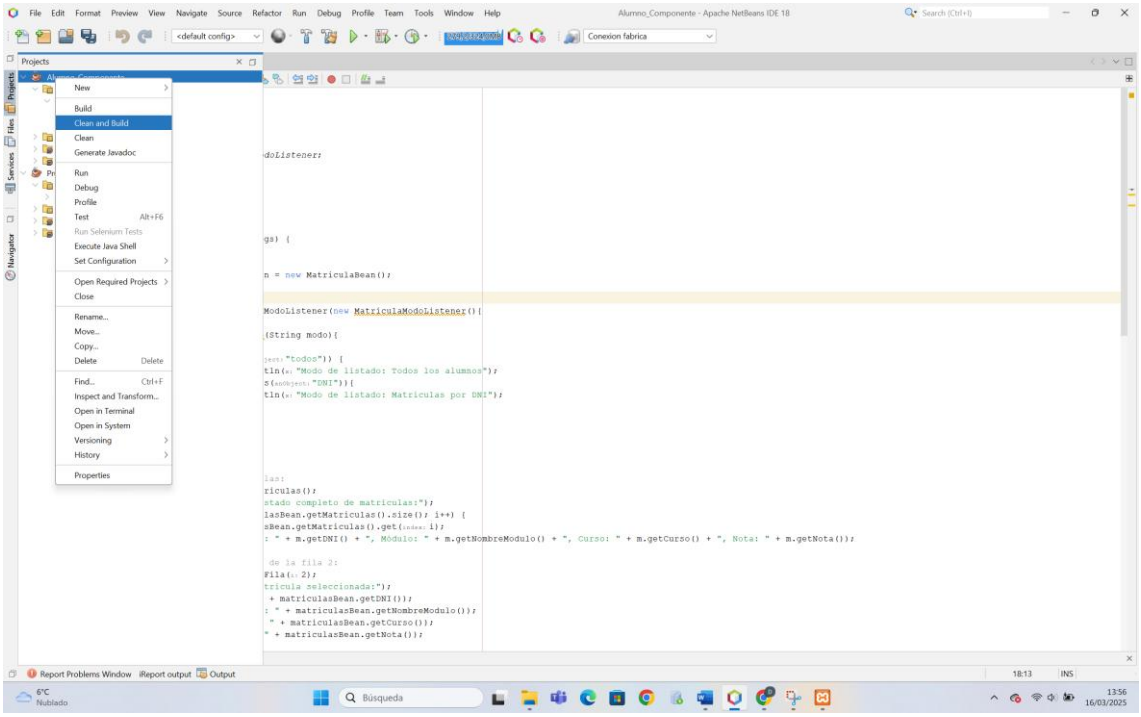
Para manejar eventos, se define una interfaz MatriculaModoListener que permite notificar cambios de modo. En los métodos recargarMatriculas y recargarDNI, los eventos se implementan mediante el manejador MatriculaModoListener. Cada vez que se llama a uno de estos métodos y se actualiza la lista de matrículas, se notifica el cambio de modo llamando a receptor.cambiarModo("todos") o receptor.cambiarModo("DNI").

```java
// Métodos para añadir o quitar un listener
public void addMatriculaModoListener(MatriculaModoListener listener) {
    this.receptor = listener;
}

public void removeMatriculaModoListener(MatriculaModoListener listener) {
    this.receptor = listener;
}

/**
 * Interfaz para manejar eventos de cambio de modo.
 */
public interface MatriculaModoListener {
    void cambiarModo(String modo);
}
```

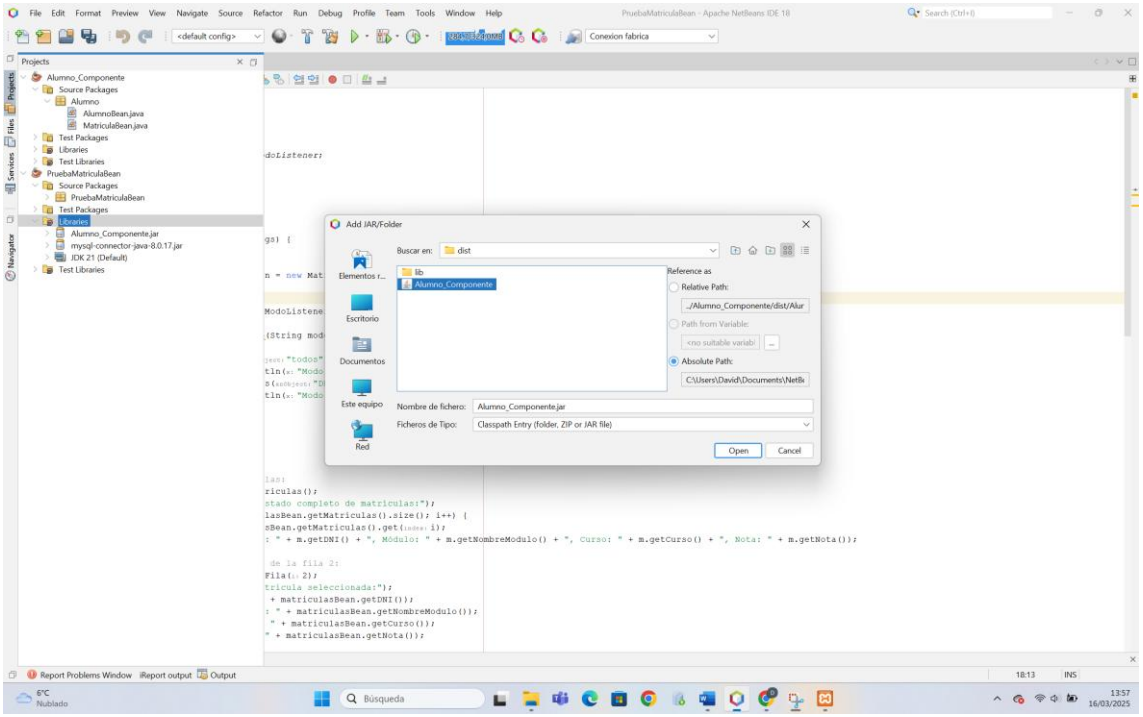
- **Tendrás que crear un proyecto de prueba del componente en el que hagas un listado de todas las matrículas que hay en el sistema, y luego hagas un listado de las matrículas de un alumno concreto.**
- **Cuando cargues la matricula del usuario concreto deberás capturar el evento generado al cambiar de modo.**
- **Añade el código necesario para añadir una matrícula nueva a la base de datos.**

Para verificar su correcto funcionamiento, se creó un nuevo proyecto de prueba con la clase de prueba en Main.java, donde se agregaron eventos y pruebas de las funcionalidades implementadas.

Antes de nada, limpiamos y construimos el proyecto donde se encuentra el MatriculasBean para objener el archivo jar.




Tras esto, incluimos este archivo en las librerías del proyecto de prueba:



La clase Main se utiliza para probar MatriculaBean. Implementa las siguientes pruebas:

- Listar todas las matrículas.
- Seleccionar una matrícula específica.
- Listar matrículas filtradas por DNI.
- Agregar una nueva matrícula y verificar su existencia.

```
try {
    //Listar todas las matriculas:
    matriculasBean.recargarMatriculas();
    System.out.println(x: "\nListado completo de matriculas:");
    for (int i = 0; i < matriculasBean.getMatriculas().size(); i++) {
        Matricula m = matriculasBean.getMatriculas().get(index: i);
        System.out.println("DNI: " + m.getDNI() + ", Módulo: " + m.getNombreModulo() + ", Curso: " + m.getCurso() + ", Nota: " + m.getNota());
    }
    //Seleccionar la matricula de la fila 2:
    matriculasBean.seleccionarFila(i: 2);
    System.out.println(x: "\nMatricula seleccionada:");
    System.out.println("DNI: " + matriculasBean.getDNI());
    System.out.println("Modulo: " + matriculasBean.getNombreModulo());
    System.out.println("Curso: " + matriculasBean.getCurso());
    System.out.println("Nota: " + matriculasBean.getNota());
    System.out.println(x: "");

    //Listar matriculas de un alumno:
   matriculasBean.recargarDNI(DNI:"12345678A");
   System.out.println(x: "\nListado de matriculas del alumno con DNI: 12345678A");
    for (int i = 0; i < matriculasBean.getMatriculas().size(); i++) {
        Matricula m = matriculasBean.getMatriculas().get(index: i);
        System.out.println("DNI: " + m.getDNI() + ", Módulo: " + m.getNombreModulo() + ", Curso: " + m.getCurso() + ", Nota: " + m.getNota());
    }

    //Añadir nueva matricula
    matriculasBean.setDNI(DNI:"90765432A");
    matriculasBean.setNombreModulo(NombreModulo:"Geografia");
    matriculasBean.setCurso(Curso:"24-25");
    matriculasBean.setNota(Nota:9.5);
    matriculasBean.addMatricula();

    //Volver a cargar todas las matriculas para ver si se ha añadido la nueva
    matriculasBean.recargarMatriculas();  // Recarga todas las matriculas

    //Listar todas las matriculas para comprobar que se ha añadido
    matriculasBean.seleccionarFila(matriculasBean.getMatriculas().size()-1);// Seleccionar la última matricula añadida
    System.out.println(x: "\nÚltima matricula añadida:");
    System.out.println("DNI: " + matriculasBean.getDNI());
    System.out.println("Modulo: " + matriculasBean.getNombreModulo());
    System.out.println("Curso: " + matriculasBean.getCurso());
    System.out.println("Nota: " + matriculasBean.getNota());
    System.out.println(x: "");
```

También se han implementado eventos para notificar cambios de modo.

```
public class Main {

    public static void main(String[] args) {


        MatriculaBean matriculasBean = new MatriculaBean();


        matriculasBean.addMatriculaModoListener(new MatriculaModoListener(){

            public void cambiarModo(String modo){

                if (modo.equals(anObject: "todos")) {
                    System.out.println(x: "Modo de listado: Todos los alumnos");
                }else if(modo.equals(anObject: "DNI")){
                    System.out.println(x: "Modo de listado: Matriculas por DNI");
                }
            }
        });
```

**Prueba del funcionamiento del componente:**

```
run:
Modo de listado: Todos los alumnos

Listado completo de matr�culas:
DNI: 12345678A, M�dulo: Historia, Curso: 23-24, Nota: 7.0
DNI: 12345678A, M�dulo: Matem�ticas, Curso: 23-24, Nota: 8.5
DNI: 14785236D, M�dulo: Programaci�n, Curso: 23-24, Nota: 8.0
DNI: 23456789B, M�dulo: F�sica, Curso: 22-23, Nota: 9.2
DNI: 23456789B, M�dulo: Qu�mica, Curso: 22-23, Nota: 6.8
DNI: 96385274F, M�dulo: Bases de Datos, Curso: 23-24, Nota: 7.5

Matricula seleccionada:
DNI: 14785236D
Modulo: Programaci�n
Curso: 23-24
Nota: 8.0

Modo de listado: Matriculas por DNI

Listado de matriculas del alumno con DNI: 12345678A
DNI: 12345678A, M�dulo: Historia, Curso: 23-24, Nota: 7.0
DNI: 12345678A, M�dulo: Matem�ticas, Curso: 23-24, Nota: 8.5
Matr�cula a�adida correctamente.
Modo de listado: Todos los alumnos

�ltima matr�cula a�adida:
DNI: 98765432A
Modulo: Geografia
Curso: 24-25
Nota: 9.5

BUILD SUCCESSFUL (total time: 0 seconds)
```