

Práctica ACDAT N°2

Unidad 3, Uso de conectores

Alumno: Javier González Rives.

Asignatura: Acceso a Datos

Curso: 2ºDAM

Índice

Índice	1
Introducción.	3
a) Objetivo de la práctica.	3
b) Temática de la base de datos.	3
c) Estructura del programa.	3
1 Inicio del programa	4
2 Ventana Principal	5
d) Estructura del proyecto.	6
1 Creación de tablas.	7
a) Tablas de la base datos instituto.	7
b) creación de las tablas código SQL.	8
1 módulo	8
2 profesor	8
3 alumno	8
4 modulo_alumno	9
c) Implementación en Java.	9
1 almacenaje de sentencias de creacion.	9
2 inicio.	10
3 conexión y creación.	11
2 Inserciones y procedimientos	14
a) Datos de prueba	14
b) diálogos de inserción	15
1 Módulo	15
2 Profesor	17
3 Alumno	18
d) Creación de Procedimientos	20
e) Ejecución Procedimiento	23
3 Cambios En tablas	25
4 Consultas	26
1 Listado de notas	26
2 Profesores de un alumno	28
3 Total de alumnos por modulo	30
5 Actualizaciones y borrado.	32
1 Actualizar nombre y estatus de delegado de un alumno	32
2 Modificación de la nota	33
3 Eliminar Alumno	34
4 borra módulo	35

Introducción.

a) Objetivo de la práctica.

La práctica consiste en el desarrollo de un programa o aplicación que gestionará una base de datos indicada dentro del documento explicativo de la práctica, la aplicación debe una serie de implementaciones que permitieran realizar una serie de acciones indicadas a través de ejercicios. A diferencia de en la anterior práctica en esta era preferente la implantación de una interfaz gráfica la cual en el documento no venía especificada más allá de unas recomendaciones por lo que también se ha realizado una ligera labor de diseño sin ser este en ningún momento el punto central intentando primar en todo momento la funcionalidad. Finalmente los puntos implementados en la aplicación han sido: creación de tablas, inserción de datos, creación de procedimientos almacenados, modificación de las tablas, consultas, actualización de las filas y la eliminación de registros.

b) Temática de la base de datos.

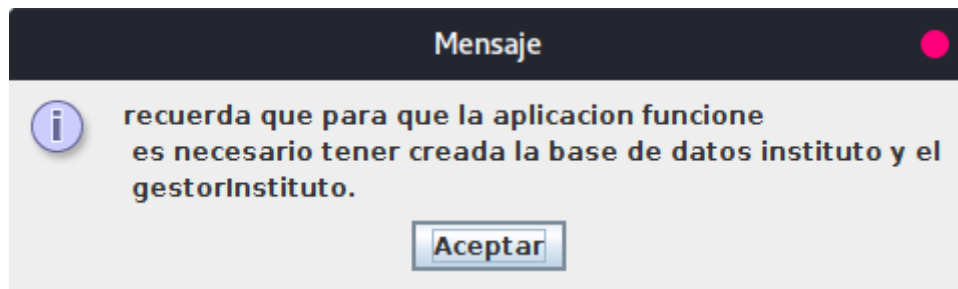
La práctica se ha realizado acerca de una base datos acerca de la gestión de un instituto con capacidad para gestionar los profesores y alumnos estando relacionados con sus respectivos módulos.

c) Estructura del programa.

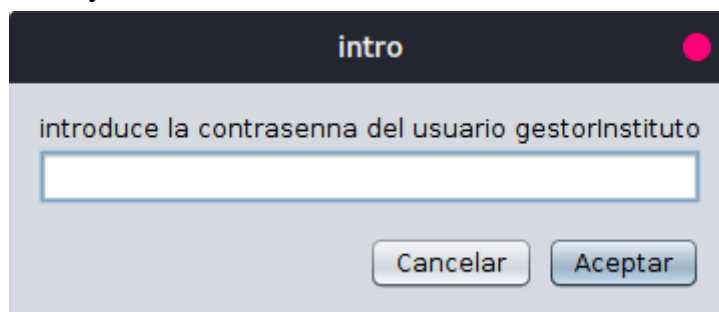
El programa se encuentra dividido en dos fases, en la primera fase se procede a la creación de tablas e inserción , dependiendo si el usuario lo desea, de registros de ejemplo. En la segunda fase se inicia la interfaz central del programa en la que ya sí se puede acceder a todas las funciones implementadas.

1 Inicio del programa

Lo primero que aparecen en el programa es una ventana que indica que es necesario tener creada una base de datos llamada “instituto” y un usuario “gestorInstituto” dentro de nuestro sistema gestor(MySQL), después de esta nos pedirá la contraseña para el usuario gestor y así poder empezar con las gestiones y acciones sobre la base de datos.

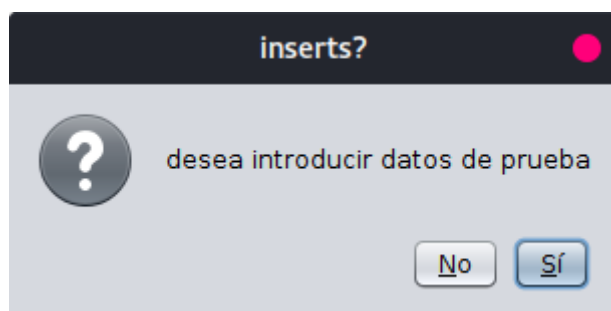


Mensaje de inicio.



Ventana para la contraseña.

Una vez se ha introducido la contraseña el programa procederá a crear las tablas dentro de la base de datos en caso de que estas no existan, acto seguido nos preguntara a través de otro mensaje emergente si deseamos añadir los registros de prueba, este mensaje solo aparece si se han tenido que crear las tablas ya que se dará por hecho que es la primera vez que se accede a la base datos.

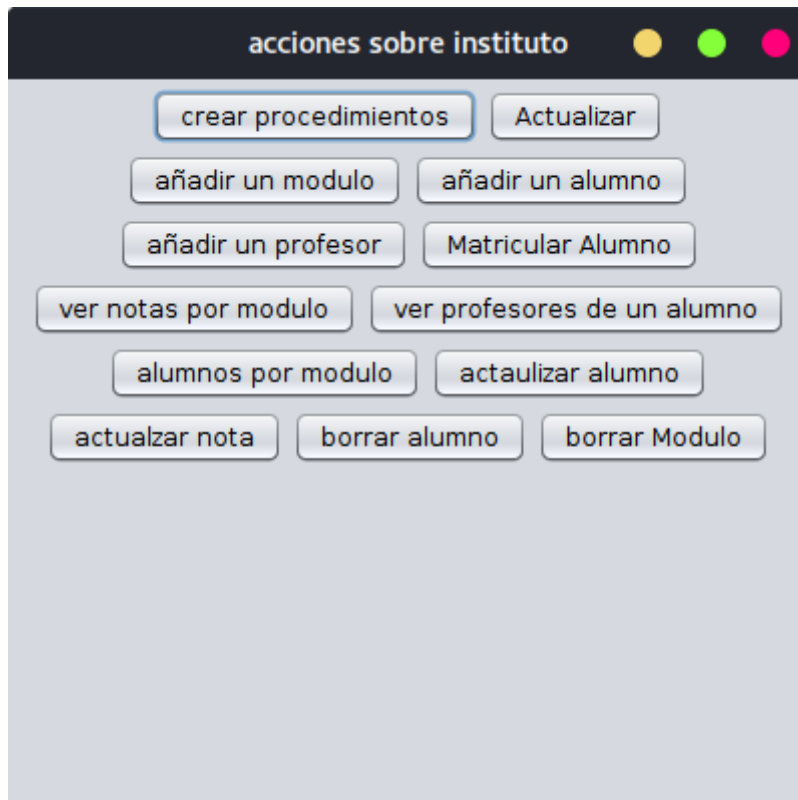


mensaje preguntado por datos de ejemplo.

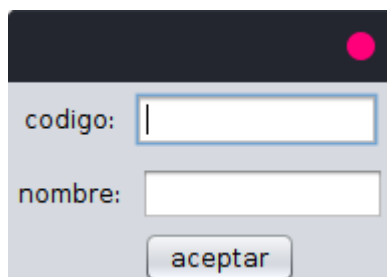
En este punto finaliza la primera parte del programa, una vez que han terminado las gestiones básicas para su funcionamiento pasa al siguiente paso.

2 Ventana Principal

En este punto se llega al centro del programa desde el que se puede acceder a todo el resto de las gestiones que este te permite realizar, Este está formado por una ristra de botones ya que como se ha indicado anteriormente se le ha dado prioridad a la funcionalidad, según vayamos pulsando en cada uno de los botones nos aparecerá un diálogo con los campos necesarios para la acción indicada, para recibir un pequeño extra de información cada boton al dejar el raton posado sobre este nos lanzará un pequeño mensaje de ayuda.



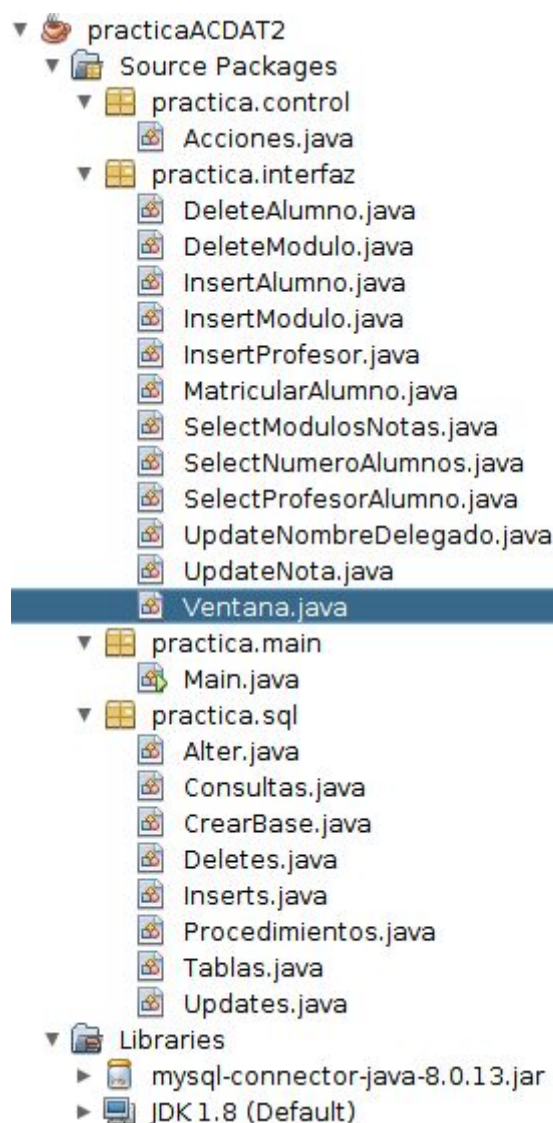
Ventana con los botones para acceder a las opciones del programa.



Ejemplo diálogo.

d) Estructura del proyecto.

El proyecto finalmente se ha dividido en cuatro apartados: Inicio del programa(Main), interfaz, SentenciasSQL y Control. Aunque con el apartado Control pueda parecer que se usa el modelo Vista-Controlador este no se usa, control alberga aquellas acciones que no requieren de interfaz gráfica como la creación de procedimientos o la actualización de la estructura de las tablas, dentro de main está todo aquello que corresponde con el arranque o puesta en marcha del programa como entre otras cosas la creación de tablas e inicio de la interfaz principal, dentro de interfaz las ventanas y diálogos para el funcionamiento del programa y finalmente en “sql” clases almacén con las sentencias divididas según su funcionalidad.



1 Creación de tablas.

Dentro del propio programa aunque posible no tiene sentido la creación de la base de datos y ni mucho menos el usuario para gestionarla por lo que estos apartados serán tarea del encargado de implementar la aplicación.

a) Tablas de la base datos instituto.

La base de datos está formada por un total de cuatro tablas orientadas a la gestión de profesores y alumnos en sus distintos módulos, todo lo relacionado con la seguridad de los datos se ha realizado a través de seguridad referencial en la mayoría de los casos restringir borrados y realizar actualizaciones en cascada por lo que salvo caso contrario no se indicara.

- módulo: formada por “codigo” como clave primaria numérica y “nombre” del módulo.
- Profesor: como clave primaria tiene “RFC” que puede contener tanto caracteres como números, como datos tiene “nombre”, “ApellidoP”, “apellidoM”, “direccion” y “telefono” está relacionado con la asignatura que imparte a través del código de esta asignatura “codigo_modulo”.
- alumno: su clave primaria es el “expediente”, numérica, como datos tiene “nombre”, “apellidoP”, “apellidoM” y “fechaNac” además tiene un campo “delegado” que almacena si tiene este rol.
- modulo_alumno: esta tabla relaciona los alumnos y los módulos a través de su clave primaria, en este caso el borrado de alumnos si es en cascada además tiene clave primaria propia.

b) creación de las tablas código SQL.

1 módulo

```
create table if not exists modulo (  
    codigo int primary key ,  
    nombre varchar(45)  
)  
engine = innodb  
character set UTF8  
collate utf8_spanish_ci;
```

2 profesor

```
create table if not exists profesor(  
    RFC char(15) primary key,  
    nombre varchar(25),  
    apellidoP varchar(25),  
    apellidoM varchar(25),  
    direccion varchar(25),  
    telefono char(10),  
    codigo_modulo int,  
    constraint foreign key(codigo_modulo)references modulo(codigo)  
    on delete restrict  
    on update cascade  
)engine = innodb  
character set UTF8  
collate utf8_spanish_ci;
```

3 alumno

```
create table if not exists alumno(  
    expediente int primary key,  
    nombre varchar(25),  
    apellidoP varchar(25),  
    apellidoM varchar(25),  
    fechaNac date,  
    delegado binary  
)  
engine = innodb  
character set UTF8  
collate utf8_spanish_ci;
```

4 modulo_alumno

```
create table if not exists modulo_alumno(
  idModulo_alumno int primary key,
  codigo_alumno int,
  codigo_modulo int,
  constraint foreign key(codigo_alumno) references alumno(expediente)
  on delete cascade
  on update cascade,
  constraint foreign key(codigo_modulo) references modulo(codigo)
  on delete restrict
  on update cascade
)
engine = innodb
character set UTF8
collate utf8_spanish_ci;
```

c) Implementación en Java.

1 almacenaje de sentencias de creacion.

Como para casi toda la aplicación para en el caso de la aplicación se han almacenado las sentencias en una clase a parte para más tarde llamarlas desde otro paquete, este sistema simplifica altamente la legibilidad del código.

```

1  / **
2   *
3   * @author Javier González Rives
4   */
5  public class Tablas {
6      public static final String MODULO =
7          "create table if not exists modulo (\n" +
8          "    codigo int primary key,\n" +
9          "    nombre varchar(45)\n" +
10         ")\n" +
11         "engine = innodb\n" +
12         "character set UTF8\n" +
13         "collate utf8_spanish_ci;";
14     public static final String PROFESOR =
15         "create table if not exists profesor(\n" +
16         "    RFC char(15) primary key,\n" +
17         "    nombre varchar(25),\n" +
18         "    apellidoP varchar(25),\n" +
19         "    apellidoM varchar(25),\n" +
20         "    direccion varchar(25),\n" +
21         "    telefono char(10),\n" +
22         "    codigo_modulo int,\n" +
23         "    constraint foreign key(codigo_modulo) references modulo(codigo)\n" +
24         "    on delete restrict\n" +
25         "    on update cascade\n" +
26         ")\n" +
27         "engine = innodb\n" +
28         "character set UTF8\n" +
29         "collate utf8_spanish_ci;";
30     public static final String ALUMNO =
31         "create table if not exists alumno(\n" +
32         "    expediente int primary key,\n" +
33         "    nombre varchar(25),\n" +
34         "    apellidoP varchar(25),\n" +
35         "    apellidoM varchar(25),\n" +
36         "    fechaNac date,\n" +
37         "    delegado binary\n" +
38         ")\n" +
39         "engine = innodb\n" +
40         "character set UTF8\n" +
41         "collate utf8_spanish_ci;";
42     public static final String MODULO_ALUMNO =

```

Almacenaje de las sentencias.

2 inicio.

Para la ejecución de la creación se realiza desde el main, los pasos que sigue el programa son: 1 Cargar driver y definir estilo; 2 introducir contraseña y conectar 3 comprobar tablas existentes y crearlas en caso necesario.

La carga del driver y el cambio de estilo se hacen directamente desde el “main”.

main:

```
/**
 * inicio del programa
 * @param args
 */
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {

        @Override
        public void run() {
            try{
funcione"                JOptionPane.showMessageDialog(null, "recuerda que para que la aplicacion
                                + "\n es necesario tener creada la base de datos instituto y el "
                                + "\n gestorInstituto.");
                                Class.forName("com.mysql.cj.jdbc.Driver"); <---- Carga del driver

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"); <--
cambio estilo ventana
                                prepararBase();
                                mostrar();
                                }catch(ClassNotFoundException cl){
                                JOptionPane.showMessageDialog(null,"imposible conectar a la base de
datos",
                                "error", JOptionPane.ERROR_MESSAGE);
                                cl.printStackTrace();
                                System.exit(-1);
                                }catch(Exception ex){
                                }

                                }
                                });
                                }
}
```

3 conexión y creación.

A continuación se llama al método “prepararBase()” que pide la contraseña del usuario gestor, realiza la primera conexión que comprueba y crea las tablas.

```
/**
 * metodo que pide la contrasenna al usuario
 *
 */
public static void introContrasenna(){
    // dialogo para introducir la contrasenna
    pass = JOptionPane.showInputDialog(null, "introduce la contrasenna del usuario
gestorInstituto",
        "intro", JOptionPane.DEFAULT_OPTION);
    // en el caso de cancelar se cierra el programa
    if(pass == null){
        System.exit(0);
    }
}
/**
 * metodo que prepara el programa para funcionar
 * con la base de datos
 */
private static void prepararBase(){
    try{
        introContrasenna();
        // primera conexion de prueba
        Connection conexion = DriverManager.getConnection(conc,usuario,pass);
        DatabaseMetaData dbmt = conexion.getMetaData();
        ResultSet rm = dbmt.getTables(null, null, "%", null);

        int numtablas = 0;
        while(rm.next()){
            String nombreTabla = rm.getString(3);
            // comprobacion de la existencia de las tablas
            if(
                nombreTabla.equals("modulo") ||
                nombreTabla.equals("alumno") ||
                nombreTabla.equals("profesor") ||
                nombreTabla.equals("modulo_alumno")
            ){
                // se summa si se encuentra una de las tablas correspondiente
                numtablas++;
            }
        }
        conexion.close();
        if(numtablas < 4){

            crearTablas();
        }
    }
}
```

```

        int inserts = JOptionPane.showConfirmDialog(null, "desea introducir datos de
prueba", "inserts?", JOptionPane.YES_NO_OPTION);
        if(inserts == 0){
            insertarRegistros();
        }
    }catch(SQLException sql){

        if(!pass.isEmpty()){
            JOptionPane.showMessageDialog(null, "Error la contraseña puede no ser
correcta"
            + "\n o el usuario o la base de datos no existe","ERROR en la
conexion",JOptionPane.ERROR_MESSAGE);
            introContrasenna();
        }else{
            System.exit(0);
        }
        //sql.printStackTrace();
    }
}

/**
 * metodo que crea las tablas dentro de la base de datos
 */
private static void crearTablas(){
    try{
        // conexion con la base de datos instituto
        Connection conexion = DriverManager.getConnection(conc,usuario,pass);
        // creacion de las tablas
        Statement stat = conexion.createStatement();
        // ejecucion de la ceracion de tablas
        stat.execute(Tablas.MODULO);
        stat.execute(Tablas.PROFESOR);
        stat.execute(Tablas.ALUMNO);
        stat.execute(Tablas.MODULO_ALUMNO);
        // cerrado de los canales
        stat.close();
        conexion.close();

    }catch(SQLException sql){
        sql.printStackTrace();
        JOptionPane.showMessageDialog(null, "error en la creacion de tablas", "error",
JOptionPane.INFORMATION_MESSAGE);
        System.exit(-1);
    }
}

```

Desde “IntroContrasenna” se procede a la introducción de la contraseña, en caso que sea errónea el flujo del programa está preparado para que se vuelva a solicitar, en caso de cancelar se finaliza la aplicación, una vez se ha introducido la contraseña se realiza la conexión a través de “DriverManager.getConnection()”, dentro de este se especifica la ruta de la conexión, el usuario y la contraseña que para esta aplicación se almacena en el interior de unas variables en el main, una vez lista la conexión se comprueba la existencia

de las tablas de la base de datos a través de un objeto del tipo “DataBaseMetadata” que nos permite rescatar el nombre de las tablas de la base de datos, en el caso que no existan se crea un objeto de tipo Statement que ejecuta las sentencias almacenadas anteriormente en la clase tablas.

#	Tables_in_instituto
1	alumno
2	modulo
3	modulo_alumno
4	profesor

Tablas en la BBDD

2 Inserciones y procedimientos

a) Datos de prueba

Para experimentar con la aplicación de manera más sencilla se ha implementado la creación de una serie de registros de prueba, estos simplemente están almacenados en un “array de Strings” que los almacena para más tarde ejecutarlos desde un objeto de tipo “Statement”, la utilidad de esta opción está enfocada a un espacio de pruebas .

```
/**
 * metodo que realiza la insercion de las filas de ejemplo
 */
private static void insertarRegistros(){
try{
// conexion
Connection conexion = DriverManager.getConnection(conc,usuario,pass);
// creacion del objeto de ejecucion
Statement stat = conexion.createStatement();
// bucle que ejecuta los inserts
for(int i = 0; i < Inserts.inserts.length;i++){

        stat.executeUpdate(Inserts.inserts[i]);
    }
    stat.close();
    conexion.close();
    JOptionPane.showMessageDialog(null, "datos insertados con exito", "introduccion
existosa", JOptionPane.INFORMATION_MESSAGE);
}catch(SQLException sql){

    }
}
```

Clase Contenedora:

```
public class Inserts {
    public static String[] inserts ={
        "insert into modulo values(1,'ACDAT');",
        "insert into modulo values(2,'PSP');",
        "insert into alumno values(1,'Javier','Gonzalez','Rives','1998-2-9',1);",
        "insert into alumno values(2,'Alicia','Tome','Ortega','1990-6-2',0);",
        "insert into alumno values(3,'Angel','Salas','Calvo','1999-4-5',0);",
        "insert into alumno values(4,'Juan Raul','Panyagua','Casado','1998-3-23',0);",
        "insert into modulo_alumno values(1,1,1);",
        "insert into modulo_alumno values(2,2,1);",
        "insert into modulo_alumno values(3,3,2);",
        "insert into modulo_alumno values(4,4,2);",
        "insert into profesor values(1,'Isabel','nuñez','nuñez','c/ san javier
n32','950034432',2);",
        "insert into profesor values(2,'Manuel','Vazquez','Feijo','c/ Notredame n54
2ºB','950236798',2);",
```

```
"insert into profesor values(3,'Sergismundo','Pico','Puertas','c/ los balcanes nº28 4º','950323323',1);"
```

```
};
```

#	codigo	nombre
1	1	ACDAT
2	2	PSP
*	NULL	NULL

#	RFC	nombre	apellidoP	apellidoM	direccion	telefono	codigo_modulo
1	1	Isabel	nuñez	nuñez	c/ san javier n32	950034432	2
2	2	Manuel	Vazquez	Feijo	c/ Notredame n54 2ºB	950236798	2
3	3	Sergismundo	Pico	Puertas	c/ los balcanes nº28 4º	950323323	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

#	idModulo_alumno	codigo_alumno	codigo_modulo
1	1	1	1
2	2	2	1
3	3	3	2
4	4	4	2
*	NULL	NULL	NULL

#	expediente	nombre	apellidoP	apellidoM	fechaNac	delegad
1	1	Javier	Gonzalez	Rives	1998-02-09	BLOB
2	2	Alicia	Tome	Ortega	1990-06-02	BLOB
3	3	Angel	Salas	Calvo	1999-04-05	BLOB
4	4	Juan Raul	Panyagua	Casado	1998-03-23	BLOB
*	NULL	NULL	NULL	NULL	NULL	NULL

b) diálogos de inserción

En el caso que se desee hacer una inserción se han implementado una serie de diálogos con los campos, la mayoría de texto excepto para selecciones específicas que se han implementado otro tipo de campos como el caso del delegado que se trata de una checkBox, (el código de las partes gráficas no se muestra dado que no es el centro del proyecto) finalmente se encuentra un botón que realiza la acción para las comprobaciones y las inserciones. La conexión con la base de datos se realiza cada vez que se abre un dialogo permaneciendo abierta hasta que este se cierra.

1 Módulo

@Override


```

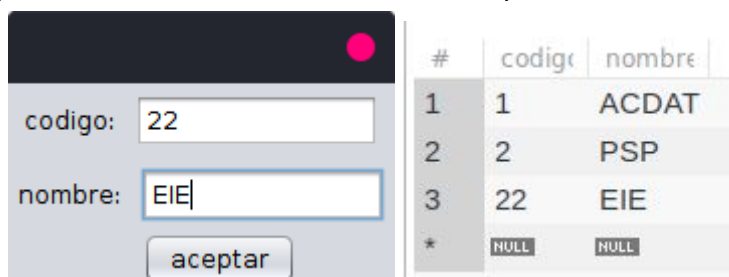
public void actionPerformed(ActionEvent e) {
    try{

        String codigo = txCodigo.getText() ;
        String nombre = txNombre.getText().trim();
        // comprobaciones
        if(nombre.isEmpty() || codigo.isEmpty()){
            JOptionPane.showMessageDialog(this, "faltan campos por rellenar", "error",
JOptionPane.ERROR_MESSAGE);
        }else{
            int numCodigo = Integer.parseInt(codigo);
            PreparedStatement stat = conexion.prepareStatement("insert into "+
nombreTabla +" values(?,?)");
            stat.setInt(1, numCodigo);
            stat.setString(2, nombre);

            if(stat.executeUpdate() > 0){
                JOptionPane.showMessageDialog(this, "insercion
correcta","correcto",JOptionPane.INFORMATION_MESSAGE);
            }
        }
    }catch(SQLException sql){
    }catch(NumberFormatException num){
        JOptionPane.showMessageDialog(this, "codigo solo admite numeros","error de
formato",JOptionPane.ERROR_MESSAGE);
    }
}

```

En primer lugar rescato los valores de los campos en variables listas para ser usadas directamente, los cuadros que únicamente llevan texto se guardan en “String” en el caso de otros campos como el código que son numéricos se transforman en números, para evitar errores he aprovechado la captura de la excepción para mostrar un mensaje de error. Para la ejecución de la sentencia he usado un “preparedStatement” dentro de la sentencia los campos a introducir se indican con “?”, con los metodos set”tipo”(posicion,valor) podemos indicar qué valor queremos permitiéndonos variar las acciones en este caso se sustituyen por los valores introducidos en los campos de texto



#	codigo	nombre
1	1	ACDAT
2	2	PSP
3	22	EIE
*	NULL	NULL

2 Profesor

@Override

```

    public void actionPerformed(ActionEvent e) {
        try{
            // paso a texto
            String RFC = txRFC.getText().trim();
            String nombre = txNombre.getText().trim();
            String ApellidoP = txApellidoP.getText().trim();
            String ApellidoM = txApellidoM.getText().trim();
            String Direccion = txDireccion.getText().trim();
            String Telefono = txTelefono.getText().trim();
            int codigo = idModulo.get(cbAsignaturas.getSelectedIndex());

            // comprobaciones campos vacios
            if(
                nombre.isEmpty() ||
                RFC.isEmpty() ||
                ApellidoM.isEmpty() ||
                ApellidoP.isEmpty() ||
                Direccion.isEmpty() ||
                Telefono.isEmpty()

            ){
                JOptionPane.showMessageDialog(this, "faltan campos por rellenar", "error",
                JOptionPane.ERROR_MESSAGE);
            }else{

                PreparedStatement stat = conexion.prepareStatement("insert into "+
                nombreTabla +" values(?,?,?,?,?,?,?)");
                // annadiendo los valores
                stat.setString(1, RFC);
                stat.setString(2, nombre);
                stat.setString(3, ApellidoP);
                stat.setString(4, ApellidoM);
                stat.setString(5, Direccion);
                stat.setString(6, Telefono);
                stat.setInt(7, codigo);
                if(stat.executeUpdate() > 0){
                    JOptionPane.showMessageDialog(this, "insercion
                    correcta","correcto",JOptionPane.INFORMATION_MESSAGE);
                }
                stat.close();
            }
        }catch(SQLException sql){

```

```

JOptionPane.showMessageDialog(this,"existe un error en alguno de los campos",
    "error",JOptionPane.ERROR_MESSAGE);
}
}

```

#	RFC	nombre	apellidoP	apellidoM	direccion	telefono	codigo_modulo
1	1	Isabel	nuñez	nuñez	c/ san javier n32	950034432	2
2	2	Manuel	Vazquez	Feijo	c/ Notredame n54 2ºB	950236798	2
3	3	Sergismundo	Pico	Puertas	c/ los balcanes nº28 4º	950323323	1
4	sdadqw	Dalias	Gaudi	Emiliano	c/ restrepo	+34-99999	22
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

3 Alumno

@Override

```

public void actionPerformed(ActionEvent e) {
    try{
        // paso a texto
        String codigo = txExpediente.getText().trim();
        String nombre = txNombre.getText().trim();
        String ApellidoP = txApellidoP.getText().trim();
        String ApellidoM = txApellidoM.getText().trim();
        String fechaNac = ftFechaNacimiento.getText().trim();
        byte delegado = (chDelegado.isSelected()?(byte)1:(byte)0);
        // comprobaciones campos vacios
        if(
            nombre.isEmpty() ||
            codigo.isEmpty() ||
            ApellidoM.isEmpty() ||

```

```

        ApellidoP.isEmpty() ||
        fechaNac.isEmpty()
    ){
        JOptionPane.showMessageDialog(this, "faltan campos por rellenar", "error",
JOptionPane.ERROR_MESSAGE);
    }else{
        int numCodigo = Integer.parseInt(codigo);
        PreparedStatement stat = conexion.prepareStatement("insert into "+
nombreTabla +"(expediente,nombre,apellidoP,apellidoM,fechaNac,delegado)
values(?,?,?,?,?,?)");
        // annadiendo los valores
        stat.setInt(1, numCodigo);
        stat.setString(2, nombre);
        stat.setString(3, ApellidoP);
        stat.setString(4, ApellidoM);
        stat.setString(5, fechaNac);
        stat.setByte(6, delegado);
        if(stat.executeUpdate() > 0){
            JOptionPane.showMessageDialog(this, "insercion
correcta","correcto",JOptionPane.INFORMATION_MESSAGE);
        }
        stat.close();
    }
} catch(SQLException sql){
    JOptionPane.showMessageDialog(this,"existe un error en alguno de los campos",
        "error",JOptionPane.ERROR_MESSAGE);
} catch(NumberFormatException num){
    JOptionPane.showMessageDialog(this, "expediente solo admite numeros","error de
formato",JOptionPane.ERROR_MESSAGE);
}
}

```

#	expediente	nombre	apellidoP	apellidoM	fechaNac	delega
1	1	Javier	Gonzalez	Rives	1998-02-09	BLOB
2	2	Alicia	Tome	Ortega	1990-06-02	BLOB
3	3	Angel	Salas	Calvo	1999-04-05	BLOB
4	4	Juan Raul	Panyagua	Casado	1998-03-23	BLOB
5	333	Mariano	Petinto	Eras	1999-12-23	BLOB
*	NULL	NULL	NULL	NULL	NULL	NULL

d) Creación de Procedimientos

La creación de procedimientos está implementada en la clase Acciones en el paquete Control, se inicia a partir de uno de los botones del menú principal, el proceso comprueba si existen los procedimientos y en caso negativo las crea.

```
public static void crearProcedimientos(){
    // control para la conexion
    try{
        // creacion de la conexion con la base de datos
        Connection conexion =
        DriverManager.getConnection(Main.conc,Main.usuario,Main.pass);
        // captura de la meta data
        DatabaseMetaData dbmd = conexion.getMetaData();
        // captura de los procedimientos
        ResultSet result = dbmd.getProcedures(null, null, "%");
        // variables para controlar los procedimientos creados
        boolean altaAl = false;
        boolean matricular = false;
        // recorrido de los alumnos
        while(result.next()){
            String nombre = result.getString(3);
```

```

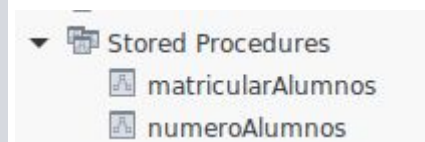
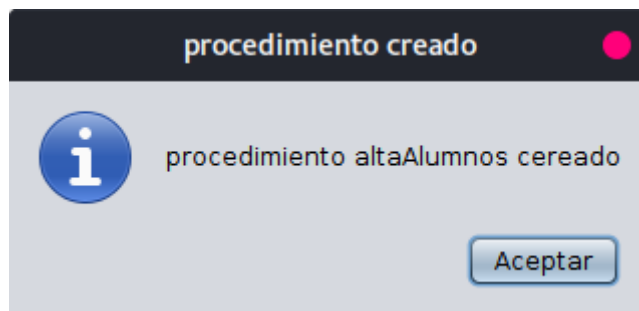
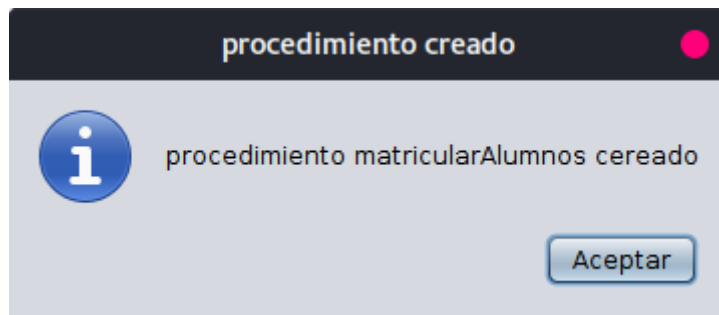
        if(nombre.equals("numeroAlumnos")){
            altaAl = true;
        }
        if(nombre.equals("matricularAlumnos")){
            matricular = true;
        }
        System.out.println(result.getString(3));
    }
    result.close();

    // creacion del procedimiento si no existe
    if(!altaAl){
        Statement stat = conexion.createStatement();
        stat.execute(Procedimientos.altaAlumnos(
creado",
        "procedimiento creado", JOptionPane.INFORMATION_MESSAGE);
    }
    if(!matricular){
        Statement stat = conexion.createStatement();
        stat.execute(Procedimientos.matricularAlumnos(
creado",
        "procedimiento creado", JOptionPane.INFORMATION_MESSAGE);
    }

    conexion.close();
} catch (SQLException sql){
    sql.printStackTrace();
}
}

Procedimientos:

```



altaAlumnos:

```
PROCEDURE `numeroAlumnos`(out numero int)
BEGIN
select count(*) from alumno into numero;
END
```

Matricular Alumno:

```
create procedure `matricularAlumnos`
(in idmodulo int,in idalumno int,out matriculado int)
begin
declare numMod int default 0;
declare existeModulo int default 0;
declare existeAlumno int default 0;
declare ultimo int default 0;
declare existeTupla int default 0 ;
select count(*) from modulo_alumno
where codigo_modulo = idmodulo into numMod;
select count(*) from alumno
where expediente = idalumno into existeAlumno;
select count(*) from modulo
where codigo = idmodulo into existeModulo ;
select count(*) from modulo_alumno
where codigo_alumno = idalumno and codigo_modulo = idmodulo into existeTupla;
if numMod > 30 then
set matriculado = 0;
else
if existeModulo > 0 and existeAlumno > 0 and existeTupla <= 0 then
select max(idModulo_alumno) from modulo_alumno into ultimo;
insert into modulo_alumno (idModulo_alumno,codigo_alumno,codigo_modulo)
values(ultimo+1,idalumno,idmodulo);
```

```

        set matriculado = 1;
    else
        set matriculado = 0;
    end if;
end if;
end

```

e) Ejecución Procedimiento

Para la ejecución de un Procedimiento el proceso es muy parecido con la diferencia que en lugar de un PreparedStatement se crea un CallableStatement la cual permite rescatar los valores que devuelve un procedimiento almacenado

En el caso del procedimiento matricularAlumno el valor devuelto se usa para indicar con un mensaje si el alumno se ha matriculado correctamente.

@Override

```

    public void actionPerformed(ActionEvent e) {
        try{

            int alumno = codAlumno.get(cbAlumno.getSelectedIndex());
            int modulo = codAsignatura.get(cbAsignatura.getSelectedIndex());
            System.out.println(alumno);
            System.out.println(modulo);
            // comprobaciones

            CallableStatement stat = conexion.prepareCall("{call matricularAlumnos(?,?,?)}");

            stat.setInt(1, modulo);
            stat.setInt(2, alumno);
            stat.registerOutParameter(3, Types.INTEGER);

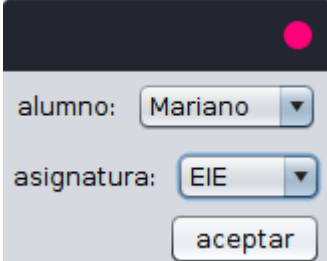
            if(!stat.execute()){
                int resultado = stat.getInt(3);

                if(resultado > 0){
                    JOptionPane.showMessageDialog(this, "matriculacion
correcta","correcto",JOptionPane.INFORMATION_MESSAGE);
                }else{
                    JOptionPane.showMessageDialog(this, "no se ha podido
matricular","error",JOptionPane.ERROR_MESSAGE);
                }
            }
        }catch(SQLException sql){
            System.out.println("error");
        }
    }

```



```
sql.printStackTrace();  
}catch(NumberFormatException num){  
    JOptionPane.showMessageDialog(this, "codigo solo admite numeros","error de  
formato",JOptionPane.ERROR_MESSAGE);  
}  
}
```



#	idModulo_alumno	codigo_alumno	codigo_modulo
1	1	1	1
2	2	2	1
3	3	3	2
4	4	4	2
5	5	333	22

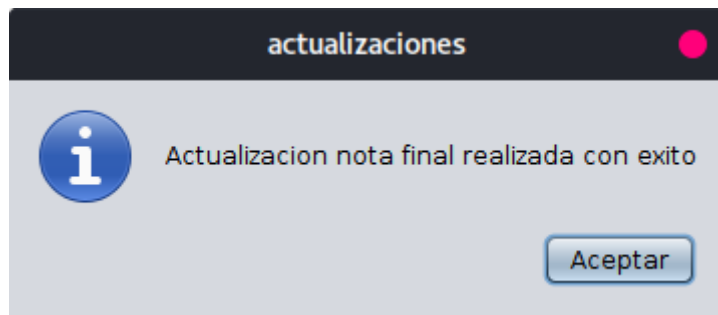
3 Cambios En tablas

Para modificación de las tablas he implementado un botón que llama al método que realiza las actualizaciones dentro de acciones, este realiza el alter table almacenado en su correspondiente clase almacén almacén.

Alter:

Alter table modulo_alumno add column notaFinal int(2);

```
/**
 * metodo que actualzia la base de datos si es necesario
 */
public static void Actualizar(){
    try{
        Connection conexion =
DriverManager.getConnection(Main.conc,Main.usuario,Main.pass);
        // metadata
        Statement stat = conexion.createStatement();
        // captura de las columnas de la tabla
        ResultSet resul = stat.executeQuery("show columns from modulo_alumno");
        boolean act1 = false;
        while(resul.next()){
            String nombre = resul.getString(1);
            if(nombre.equals("notaFinal")){
                act1 = true;
            }
        }
        // ejecucion de modificaciones
        if(!act1){
            stat.executeUpdate(Alter.notaFinal);
            JOptionPane.showMessageDialog(null, "Actualizacion nota final realizada
con exito",
            "actualizaciones", JOptionPane.INFORMATION_MESSAGE);
        }else{
            JOptionPane.showMessageDialog(null, "no hay actualizaciones en la base de
datos",
            "actualizaciones", JOptionPane.INFORMATION_MESSAGE);
        }
        stat.close();
        conexion.close();
    }catch(SQLException sql){
        sql.printStackTrace();
    }
}
```



#	Field	Type	Null	Key	Default	Extra
1	idModulo_alumno	int(11)	NO	PRI	<input type="text" value="NULL"/>	
2	codigo_alumno	int(11)	YES	MUL	<input type="text" value="NULL"/>	
3	codigo_modulo	int(11)	YES	MUL	<input type="text" value="NULL"/>	
4	notaFinal	int(2)	YES		<input type="text" value="NULL"/>	

4 Consultas

Para las consultas se ha usado un resultSet para ir capturando sus resultados e ir añadiendo los a una cadena la cual se pasa a un área de texto que muestra los valores.

1 Listado de notas

Consulta:

```
select mo.nombre, al.nombre , al.apellidoP, al.apellidoM, ma.notaFinal
from modulo mo inner join modulo_alumno ma inner join alumno al on
mo.codigo = ma.codigo_modulo and ma.codigo_alumno = al.expediente
where mo.codigo = 1 order by ma.notaFinal ;
```

@Override

```
public void actionPerformed(ActionEvent e) {
    try{
        int codigoBusqueda = codigos_modulos.get(cbModulo.getSelectedIndex());
        String salida = "modulo\tnombre\tapellido1\tapellido2\tnota final";
        // creacion de la consulta
        PreparedStatement stat = conexion.prepareStatement(Consultas.preModulos);
        // introduccion del valor
        stat.setInt(1, codigoBusqueda);
        ResultSet res = stat.executeQuery();
        while(res.next()){
            String modulo = res.getString(1);
            String nombre = res.getString(2);
```

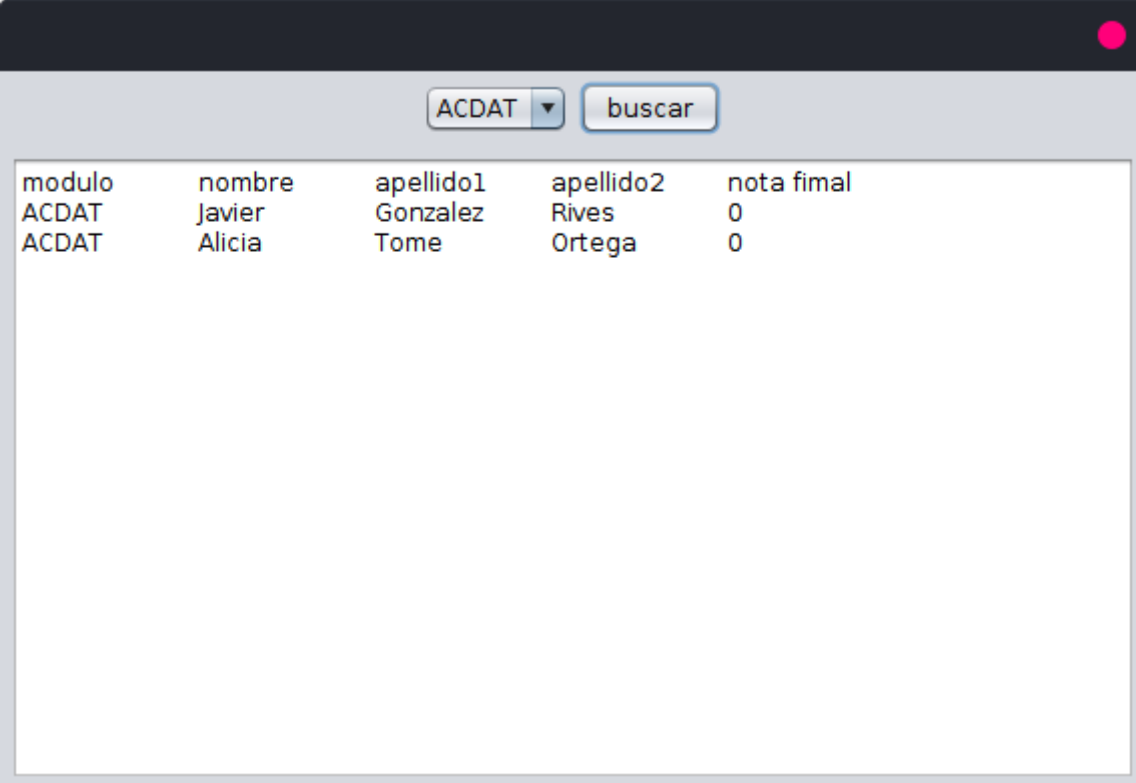
```

        String apellido1 = res.getString(3);
        String apellido2 = res.getString(4);
        int notaFinal = res.getInt(5);
        // control ante nulos
        if(modulo != null){
            salida += '\n' + modulo + '\t';
        }
        else{
            salida += '\n' + "null" + '\t';
        }
        if(nombre != null){
            salida += nombre + '\t';
        }
        else{
            salida += "null" + '\t';
        }
        if(apellido1 != null){
            salida += apellido1 + '\t';
        }
        else{
            salida += "null\t";
        }
        if(apellido2 != null){
            salida += apellido2 + '\t';
        }
        else{
            salida += "null\t";
        }
        salida += notaFinal;
    }
    taResultado.setText(salida);
    stat.close();
}catch(SQLException sql){

}
}catch(Exception ex){
    JOptionPane.showMessageDialog(this, "Ha sucedido un
error","error",JOptionPane.ERROR_MESSAGE);

}
}

```



The screenshot shows a Java Swing window with a dark title bar. Inside, there is a light gray header area containing a dropdown menu set to 'ACDAT' and a 'buscar' button. Below this is a table with five columns: 'modulo', 'nombre', 'apellido1', 'apellido2', and 'nota final'. The table contains two rows of data.

modulo	nombre	apellido1	apellido2	nota final
ACDAT	Javier	Gonzalez	Rives	0
ACDAT	Alicia	Tome	Ortega	0

2 Profesores de un alumno

consulta:

```
select p.nombre, p.apellidoP, p.apellidoM from
alumno al inner join profesor p inner join modulo_alumno ma on
al.expediente = ma.codigo_alumno and ma.codigo_modulo = p.codigo_modulo
where al.expediente = 3;
```

@Override

```
public void actionPerformed(ActionEvent e) {
    try{
        int codigoBusqueda = expedientes_alumnos.get(cbAlumno.getSelectedIndex());
        String salida = "nombre\tapellido1\tapellido2";
        // creacion de la consulta
        PreparedStatement stat = conexion.prepareStatement(Consultas.preProfesor);
        // introduccion del valor
        stat.setInt(1, codigoBusqueda);
        ResultSet res = stat.executeQuery();
        while(res.next()){

            String nombre = res.getString(1);
            String apellido1 = res.getString(2);
            String apellido2 = res.getString(3);
```

```
        if(nombre != null){
            salida += '\n' + nombre + '\t';
        }
        else{
            salida += "null" + '\t';
        }
        if(apellido1 != null){
            salida += apellido1 + '\t';
        }
        else{
            salida += "null\t";
        }
        if(apellido2 != null){
            salida += apellido2 + '\t';
        }
        else{
            salida += "null\t";
        }
    }
    taResultado.setText(salida);
    stat.close();
}catch(SQLException sql){

}
}catch(Exception ex){
    JOptionPane.showMessageDialog(this, "Ha sucedido un
error","error",JOptionPane.ERROR_MESSAGE);

}
}
```

The screenshot shows a Java Swing window with a dark title bar. Inside, there is a search bar with a dropdown menu currently showing 'Alicia' and a 'buscar' button. Below the search bar is a table with three columns: 'nombre', 'apellido1', and 'apellido2'. The table contains one row of data: 'Sergismundo', 'Pico', and 'Puertas'.

nombre	apellido1	apellido2
Sergismundo	Pico	Puertas

3 Total de alumnos por modulo

Consulta:

```
select mo.nombre,count(ma.codigo_alumno) as ord from
modulo mo inner join modulo_alumno ma on mo.codigo = ma.codigo_modulo
group by ma.codigo_modulo order by ord;
```

@Override

```
public void actionPerformed(ActionEvent e) {
    try{

        String salida = "modulo\tnumero de alumnos";
        // creacion de la consulta
        PreparedStatement stat = conexion.prepareStatement(Consultas.numeroAlumnos);
        // introduccion del valor

        ResultSet res = stat.executeQuery();
        while(res.next()){

            String nombre = res.getString(1);
            int cantidad = res.getInt(2);

            if(nombre != null){
```

```

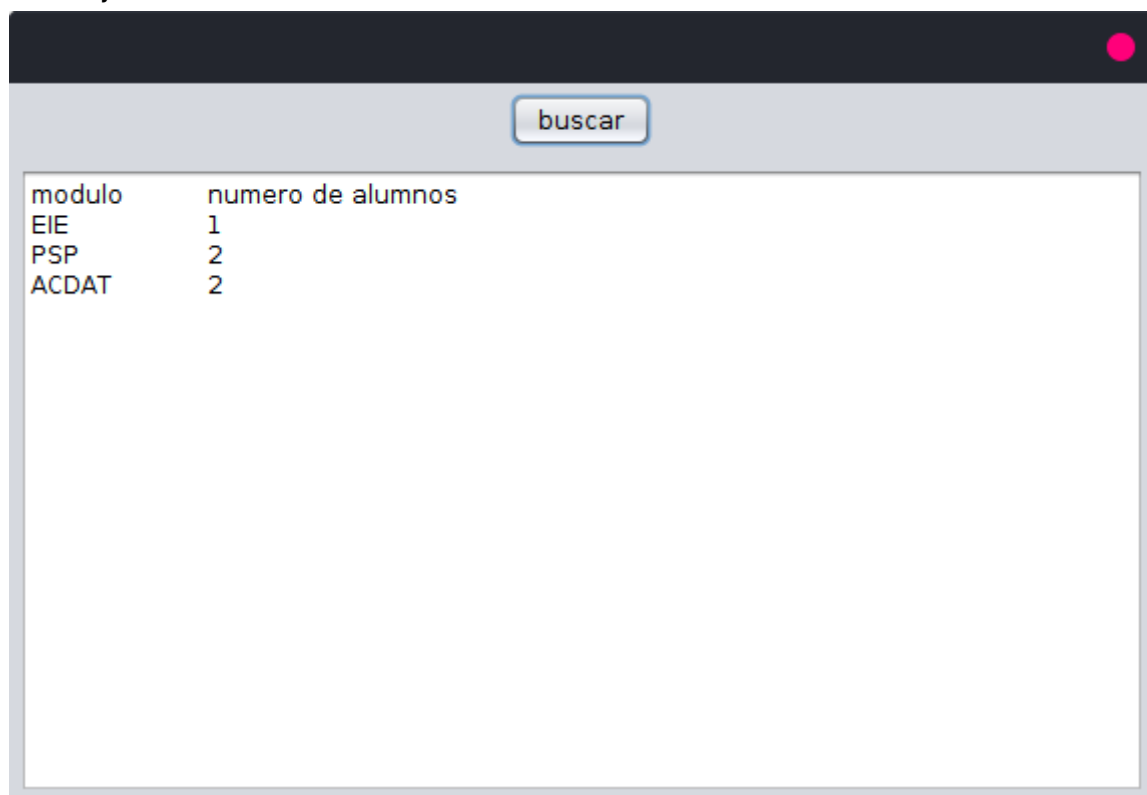
        salida += '\n' + nombre + '\t';
    }
    else{
        salida += "null" + '\t';
    }
    salida += cantidad;

}
taResultado.setText(salida);
stat.close();
}catch(SQLException sql){

}catch(Exception ex){
    JOptionPane.showMessageDialog(this, "Ha sucedido un
error","error",JOptionPane.ERROR_MESSAGE);

}
}

```



5 Actualizaciones y borrado.

Para las actualizaciones y borrados se pide que se indique un identificador para encontrar de forma exacta el registro, una en el caso de las actualizaciones también hay que introducir el nuevo valor del nuevo campo modificado, para este tipo de acciones se usa el método `executeUpdate()` que nos devuelve el número de filas modificadas, si es mayor que 0 quiere decir que la acción ha tenido éxito.

1 Actualizar nombre y estatus de delegado de un alumno

@Override

```

    public void actionPerformed(ActionEvent e) {
        try{
            // captura de los valores
            int codigo = Integer.parseInt(txExpediente.getText().trim());
            String nombre = txNombre.getText().trim();
            int delegado = (chDelegado.isSelected())?1:0;
            if(!nombre.isEmpty()){
                // creacion de sentencia
                PreparedStatement stat =
conexion.prepareStatement(Updates.preNombreAlumno);
                stat.setInt(3, codigo);
                stat.setString(1, nombre);
                stat.setInt(2, delegado);
                // ejecucion
                if(stat.executeUpdate() > 0){
                    // caso correcto
                    JOptionPane.showMessageDialog(this, "actualizacion correcta", "correcto",
JOptionPane.INFORMATION_MESSAGE);
                }else{
                    JOptionPane.showMessageDialog(this, "no se ha realizado ninguna
actualizacion", "fallo", JOptionPane.INFORMATION_MESSAGE);
                }
            }else{
                JOptionPane.showMessageDialog(this, "introduce un
nombre","error",JOptionPane.ERROR_MESSAGE);
            }
        }catch(SQLException sql){

        }catch(NumberFormatException nm){
            JOptionPane.showMessageDialog(this, "Expediente solo puede contener
numeros","error",JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

expediente: nombre: ☐ Delegado:

#	expediente	nombre	apellidoP	apellidoM	fechaNac	delegado
1	1	jose luis	Gonzalez	Rives	1998-02-09	<input type="button" value="BLOB"/>
2	2	Alicia	Tome	Ortega	1990-06-02	<input type="button" value="BLOB"/>
3	3	Angel	Salas	Calvo	1999-04-05	<input type="button" value="BLOB"/>
4	4	Juan Raul	Panyagua	Casado	1998-03-23	<input type="button" value="BLOB"/>
5	333	Mariano	Petinto	Eras	1999-12-23	<input type="button" value="BLOB"/>
*	NULL	NULL	NULL	NULL	NULL	NULL

2 Modificación de la nota

@Override

```

public void actionPerformed(ActionEvent e) {
    try{
        // captura de los valores
        int codigo = Integer.parseInt(txExpediente.getText().trim());
        int modulo = Integer.parseInt(txModulo.getText().trim());
        int nota = (Integer) spNota.getValue();

        // creacion de sentencia
        PreparedStatement stat = conexion.prepareStatement(Updates.preNota);
        stat.setInt(1, nota);
        stat.setInt(2, codigo);
        stat.setInt(3, modulo);
        // ejecucion
        if(stat.executeUpdate() > 0){
            // caso correcto
            JOptionPane.showMessageDialog(this, "actualizacion correcta", "correcto",
JOptionPane.INFORMATION_MESSAGE);
        }else{
            JOptionPane.showMessageDialog(this, "no se ha realizado ninguna
actualizacion", "fallo", JOptionPane.INFORMATION_MESSAGE);
        }

    }catch(SQLException sql){

    }catch(NumberFormatException nm){

```

```

        JOptionPane.showMessageDialog(this, "Expediente y modulo solo puede contener
numeros","error",JOptionPane.ERROR_MESSAGE);
    }
}

```

#	idModulo_alumno	codigo_alumno	codigo_modulo	notaFina
1	1	1	1	6
2	2	2	1	NULL
3	3	3	2	NULL
4	4	4	2	NULL
5	5	333	22	NULL
*	NULL	NULL	NULL	NULL

3 Eliminar Alumno

@Override

```

    public void actionPerformed(ActionEvent e) {
        try{
            // captura de los valores
            int codigo = Integer.parseInt(txExpediente.getText().trim());
            // creacion de sentencia
            PreparedStatement stat = conexion.prepareStatement(Deletes.preAlumno);
            stat.setInt(1, codigo);
            // ejecucion
            if(stat.executeUpdate() > 0){
                // caso correcto
                JOptionPane.showMessageDialog(this, "borrado correcta", "correcto",
JOptionPane.INFORMATION_MESSAGE);
            }else{
                JOptionPane.showMessageDialog(this, "no se ha realizado ningun borrado",
"fallo", JOptionPane.INFORMATION_MESSAGE);
            }

        }catch(SQLException sql){

        }catch(NumberFormatException nm){
            JOptionPane.showMessageDialog(this, "Expediente solo puede contener
numeros","error",JOptionPane.ERROR_MESSAGE);
        }
    }

```

```

}
}

```

#	expediente	nombre	apellidoP	apellidoM	fechaNac	delegar
1	2	Alicia	Tome	Ortega	1990-06-02	BLOE
2	3	Angel	Salas	Calvo	1999-04-05	BLOE
3	4	Juan Raul	Panyagua	Casado	1998-03-23	BLOE
4	333	Mariano	Petinto	Eras	1999-12-23	BLOE
*	NULL	NULL	NULL	NULL	NULL	NULL

4 borra módulo

@Override

```

    public void actionPerformed(ActionEvent e) {
        try{
            // captura de los valores
            int codigo = Integer.parseInt(txModulo.getText().trim());
            // creacion de sentencia para comprobar alumnos
            PreparedStatement compStat = conexion.prepareStatement("select count(*) from
modulo_alumno where codigo_modulo = ?;");
            compStat.setInt(1, codigo);
            ResultSet res = compStat.executeQuery();
            res.first();
            int numeroAl = res.getInt(1);
            compStat.close();
            if(numeroAl <= 0){
                // eliminar el modulo de los profesores que lo dan
                PreparedStatement stat1 =
conexion.prepareStatement(Updates.preProfesorModulo);
                stat1.setInt(1, codigo);
                stat1.execute();
                // BORRADO
                PreparedStatement stat = conexion.prepareStatement(Deletes.preModulo);
                stat.setInt(1, codigo);
                // ejecucion

                if(stat.executeUpdate() > 0){
                    // caso correcto

```

```

        JOptionPane.showMessageDialog(this, "borrado correcto", "correcto",
JOptionPane.INFORMATION_MESSAGE);
    }else{
        JOptionPane.showMessageDialog(this, "no se ha realizado ningun borrado",
"fallo", JOptionPane.INFORMATION_MESSAGE);
    }
    // caso que aun queden alumnos
    stat.close();
}
else{
    JOptionPane.showMessageDialog(this, "quedan alumnos matriculados",
"Error",JOptionPane.ERROR_MESSAGE);
}

}
}

}catch(SQLException sql){
}catch(NumberFormatException nm){
    JOptionPane.showMessageDialog(this, "Expediente solo puede contener
numeros","error",JOptionPane.ERROR_MESSAGE);
}
}

```



#	codigo	nombre
1	2	PSP
2	22	EIE
*	NULL	NULL