TEMA 11. CONTROL DE ACCESO Y MANTENIMIENTO DE BASES DE DATOS RELACIONALES

INDICE

BASES DE DATOS RELACIONALES

Muchas de las aplicaciones que se desarrollan hoy en día incluyen el trabajo con bases de datos.

Las bases de datos más usadas en la actualidad son las bases de datos relacionales.

Uno de los Sistemas Gestores de Bases de Datos más usados en la actualidad es MySQL. Es por ello que lo vamos a usar para el desarrollo de aplicaciones Java con acceso a bases de datos.

CREACIÓN DE UNA BASE DE DATOS RELACIONAL SENCILLA

Para poder crear una base de datos MySQL debemos tener instalado un servidor HTTP configurado para usar MySQL. El modo más sencillo de instalar un servidor HTTP configurado para usar MySQL consiste en instalar XAMPP.

INSTALAR XAMPP EN WINDOWS

En primer lugar debemos descargar la última versión portable disponible de XAMPP desde http://sourceforge.net/projects/xampp/files/XAMPP%20Windows/

Después debemos descomprimir la versión portable comprimida de XAMPP (xampp-portable-win32-1.8.3-5-VC11.7z en mi caso) y copiar la carpeta xampp a c:\

La ruta de XAMPP debe de ser **C:\xampp**

Las páginas web deben estar en **c:\xampp\htdocs** que es el directorio por defecto.

Una vez instalado XAMPP debemos ir al **panel de control de XAMPP** e **iniciar** el servidor **Apache**.

Si hay algún problema con el puerto por defecto de **HTTP** (**puerto 80**) debemos editar el fichero de configuración **C:\xampp\apache\conf\httpd.conf** y sustituir el puerto de escucha (linea **Listen 80**) por otro que esté disponible (por ejemplo reemplazar la línea por **Listen 8088**)

Si hay algún problema con el puerto por defecto de **HTTPS** (**puerto 443**) debemos editar el fichero de configuración **C:\xampp\apache\conf\extra\httpd-ssl.conf** y sustituir el puerto de escucha (linea **Listen 443**) por otro que esté disponible (por ejemplo reemplazar la línea por **Listen 4433**)

El archivo ejecutable del **panel de control de XAMPP** es **C:\xampp\xampp-control.exe**. Podemos crear un **Acceso Directo** en el Escritorio para facilitar la ejecución.

Una vez que hemos comprobado que el servidor Apache está en ejecución vamos al navegador e introducimos la dirección **http://localhost/** para comprobar que funciona correctamente. Esto es válido si el **puerto** de **HTTP** es el **80**, si hemos cambiado el **puerto** de HTTP por el **8088** la dirección sería la **http://localhost:8088/**

Para modificar el sitio web por defecto debemos de ir a la carpeta C:\xampp\htdocs y copiar allí los documentos de nuestro sitio web.

Nota: Si XAMPP no funciona correctamente debemos borrar la carpeta C:\xampp y descargar e instalar la versión instalable de XAMPP.

MySQL - CONFIGURACIÓN

Una vez que ya hemos instalado XAMPP e iniciado Apache (con HTTP en el puerto 8088) y MySQL sin problemas, podemos acceder a la configuración de MySQL.

Para ello vamos al enlace http://localhost:8088/xampp/index.php y en el menú de la izquierda hacemos clic sobre la opción **phpMyAdmin** para ir a la página de configuración.

Una vez allí hacemos clic sobre la opción **Nueva** para crear una nueva base de datos de nombre **bdalumnos** con cotejamiento **utf8_spanish_ci**.

Una vez creada la base de datos bdalumnos vamos a crear tres tablas de nombre alumnos, asignaturas, y calificaciones que tienen las siguientes características

alumnos
dni. Campo de tipo texto de 9 caracteres. Clave primaria.
nombre. Campo de tipo texto de 30 caracteres.
apellidos. Campo de tipo texto de 50 caracteres.
grupo. Campo de tipo texto de 4 caracteres.

asignaturas
codasignatura. Campo de tipo texto de 4 caracteres. Clave primaria
nombreasignatura. Campo de tipo texto de 30 caracteres.
descripcion. Campo de tipo texto de 200 caracteres.

calificaciones
dni. Campo de tipo texto de 9 caracteres. Clave ajena.
codasignatura. Campo de tipo texto de 4 caracteres. Clave ajena.
nota. Campo de tipo entero sin signo de tamaño 2.

Podemos crear un **archivo de texto** en formato **UTF-8** con las instrucciones SQL necesarias para crear las tablas con datos de prueba e importar el archivo desde la opción **importar**.

Yo he creado un documento de texto de nombre bdalumnos.txt con el siguiente contenido:

```
drop table calificaciones;
drop table alumnos;
drop table asignaturas;

create table alumnos(
 dni varchar(9) primary key,
 nombre varchar(30) not null,
 apellidos varchar(50) not null,
 grupo varchar(4) not null
);

insert into alumnos values ('11111111A','N1','A1','1AS3');
insert into alumnos values ('22222222B','N2','A2','1DW3');
insert into alumnos values ('33333333C','N3','A3','2AS3');
insert into alumnos values ('44444444D','N4','A4','1DW3');
insert into alumnos values ('55555555E','N5','A5','1AS3');
insert into alumnos values ('66666666F','N6','A6','1AS3');
insert into alumnos values ('77777777G','N7','A7','2DW3');
insert into alumnos values ('88888888H','N8','A8','1AS3');

create  table asignaturas(
 codasignatura varchar(4) primary key,
 nombreasignatura varchar(30) not null,
```

```
 descripcion varchar(200)
);

insert into asignaturas values ('BD','Bases de Datos','');
insert into asignaturas values ('ED','Entornos de Desarrollo','');
insert into asignaturas values ('FOL','Formación y Orientación Laboral','');
insert into asignaturas values ('ISO','Implantación de Sistemas Operativos','');
insert into asignaturas values ('PROG','Programación','');


create table calificaciones(
 dni varchar(9) not null,
 codasignatura varchar(4) not null,
 nota int(2) unsigned not null,
 foreign key (dni) references alumnos(dni) on delete cascade,
 foreign key (codasignatura) references asignaturas(codasignatura) on delete cascade,
 unique (dni,codasignatura)
);

insert into calificaciones values ('11111111A','BD',5);
insert into calificaciones values ('11111111A','ED',4);
insert into calificaciones values ('11111111A','FOL',7);
insert into calificaciones values ('11111111A','PROG',8);
insert into calificaciones values ('22222222B','BD',5);
insert into calificaciones values ('22222222B','ED',4);
insert into calificaciones values ('33333333C','FOL',7);
insert into calificaciones values ('77777777G','PROG',8);
```

CONEXIONES CON BASES DE DATOS RELACIONALES

Para que una aplicación Java se pueda comunicar con una base de datos MySQL debemos instalar un driver. El driver JDBC para permitir conexiones a bases de datos MySQL desde aplicaciones lo podemos descargar en el siguiente enlace http://dev.mysql.com/downloads/connector/j/5.0.html

Una vez descargado, descomprimimos el archivo. De todos los ficheros descomprimidos solo nos interesa el archivo .jar (mysql-connector-java-5.0.8-bin.jar en mi caso) que es el archivo que contiene la clase Driver que necesitamos.

Para poderlo usar dentro de nuestro proyecto en Eclipse debemos importar el archivo .jar a nuestro proyecto.

Vamos a crear un nuevo paquete de nombre **basesdedatos** dentro de nuestro proyecto. Una vez creado vamos a crear dentro la clase Java **BDConexion** que se intentará conectar a la base de datos MySQL de nombre **bdalumnos** y mostrará un mensaje indicando si se ha realizado la conexión correctamente o no.

Copiamos el archivo .jar con el **Driver** dentro del paquete de nombre **basesdedatos**. Hacemos clic sobre él con el botón derecho del ratón y en el menú desplegable seleccionamos la opción **Build Path** -> **Add to Build Path**

A partir de este momento ya podemos usar el Driver en nuestro proyecto.

Para poder usar las clases que nos permiten conectarnos a una base de datos MySQL debemos importar las clases **java.sql.Connection** y **java.sql.DriverManager**.

Además, para registrar el uso del Driver y poderlo usar debemos escribir la siguiente línea dentro del código

```
Class.forName("com.mysql.jdbc.Driver");
```

Lo anterior es necesario ya que si no registramos el Driver nos dará un error porque no lo encuentra. El hecho de usar **Class.forName** es para no tener que importar Drivers de distintos tipos de bases de datos en aplicaciones en las que sólo se realiza la conexión a un tipo de base de datos. Al hacerlo de este modo evitamos problemas al importar los Drivers ya que si al importar un Driver de un tipo de base de datos se produce un error la aplicación se verá afectada, mientras que si los drivers los registramos mediante Class.forName podemos realizar este registro sólo si se va a usar dicho Driver.

```
/* No se hacen los imports */
...
if (ha_elegido_mysql)
  Class.forName("driver_de_mysql");
if (ha_elegido_sqlserver)
  Class.forName("driver_de_ sqlserver");
if (ha_elegido_oracle)
  Class.forName("driver_de_oracle");
```

Una vez realizado el registro del Driver del tipo de la base de datos que vamos a usar, podemos proceder la conexión mediante **DriverManager**.

Por ejemplo, para conectarnos a la base de datos MySQL de nombre bdalumnos que se encuentra en el equipo, con el usuario root y contraseña "", escribimos

```
Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost/bdalumnos", "root", "");
```

Si todo ha ido bien la conexión se habrá realizado correctamente.

Como las operaciones con bases de datos son propensas a errores se deben controlar las excepciones. Las excepciones típicas de bases de datos son las SQLException.

Para controlar errores en la aplicación anterior escribimos

```
try{
 Class.forName("com.mysql.jdbc.Driver");
 Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost/bdalumnos", "root", "");
 // si se ha conectado correctamente
 System.out.println("Conexión Correcta.");
 // cierro la conexion
 conexion.close();
}
catch (SQLException | ClassNotFoundException sqle){
 // si NO se ha conectado correctamente
 sqle.printStackTrace();
 System.out.println("Error de Conexión");
}
```

HERRAMIENTAS DE MANIPULACIÓN DE DATOS SQL. RESULSET

Una vez que nos hemos conectado a una base de datos, para enviar comandos SQL a la base de datos se usa la clase Java **Statement**. Para obtener un objeto de esta clase a partir de la conexión escribimos

```
Statement st = conexion.createStatement();
```

Una vez que ya tenemos un objeto Statement asociado a una conexión, podemos ejecutar consultas. Los objetos Statement tienen el método **executeUpdate**() para ejecutar sentencias SQL que impliquen modificaciones en la base de datos (INSERT, UPDATE, DELETE, etc).y el método **executeQuery**() para consultas (SELECT y similares).

El método **executeQuery**() devuelve un objeto de tipo ResultSet que contiene todos los datos devueltos por la consulta.

Por ejemplo, si queremos realizar una consulta que obtenga los datos de todos los alumnos de la tabla alumnos escribimos.

```
ResultSet rs = st.executeQuery("SELECT * FROM alumnos");
```

Para mostrar por pantalla los datos de los alumnos que hay en el ResultSet escribimos

```
while (rs.next()){
 System.out.println("DNI: " + rs.getObject("dni") + ", nombre: " + rs.getObject("nombre") + ", Apellidos: " + rs.getObject("apellidos") + ", Grupo: " + rs.getObject("grupo"));
}
// cierro el ResultSet
rs.close();
// cierro el Statement despues de realizar la consulta
st.close();
```

Para actualizar los datos del alumno con dni '11111111A' y cambiar el valor del grupo a '1DW3' debemos usar el método **executeUpdate**() de la siguiente manera

```
// abro una nueva instancia del Statement
st = conexion.createStatement();
// actualizo el valor del grupo del alumno '11111111A' a '1DW3'
st.executeUpdate("UPDATE alumnos SET grupo='1DW3' WHERE dni='11111111A'");
// cierro el Statement despues de realizar la consulta
st.close();
```

Es muy importante que después de ejecutar una consulta **cerremos el Statement** para evitar problemas a la hora de realizar consultas posteriores.

También es importante que cuando dejemos de trabajar con la base de datos cerremos la conexión mediante el comando

```
conexion.close();
```

Los objetos ResultSet son muy útiles para obtener datos. Sin embargo, el hecho de necesitar estar **conectado** el **ResultSet** al origen de datos para funcionar correctamente hace que trabajar con ResulSet no sea adecuado en muchas aplicaciones cliente / servidor. En estos casos es mejor trabajar con **RowSets**.

HERRAMIENTAS DE MANIPULACIÓN DE DATOS SQL. CONJUNTOS DE FILAS. ROWSETS

Java a partir de la versión JDBC 2.0 introduce la clase **RowSet** que puede almacenar los resultados de las consultas y desconectarse de la base de datos una vez tenga los datos.
Un objeto RowSet puede actualizar sus filas mientras está desconectado del origen de datos, y su implementación puede incluir un elemento de escritura que escriba estas actualizaciones en el origen de datos, pero **la mayor parte del tiempo no tendrá abierta la conexión**.

Java proporciona tres implementaciones diferentes de RowSet:

CachedRowSet. Un RowSet desconectado que mantiene sus datos en memoria. Es navegable y serializable, lo que lo hace ideal para enviar datos a un cliente sencillo.

JDBCRowSet. Un RowSet que mantiene una conexión con el origen de datos mientras se está utilizando. Se utiliza para aprovechar algunas de las mejoras que incorpora respecto a un ResultSet.

WebRowSet. Un RowSet que puede generar una representación de sus contenidos en formato XML. Es un excelente medio para proporcionarle datos a un cliente mediante HTTP/XML.

En un RowSet navegable podemos utilizar los métodos:
first() : Se coloca en la primera fila del RowSet o devuelve false si está vacío.
last(): Se coloca en la última fila del RowSet o devuelve false si está vacío.
previous(): Se coloca en la fila anterior del RowSet o devuelve false si se terminan las filas.
absolute(int x): Se coloca en la fila x del RowSet o devuelve false si no existe la fila.
relative(int x): Se coloca en una fila relativa a la fila actual, hacia delante si el número es positivo o hacía atrás si el número es negativo, devuelve false si no existe la fila.

En cualquier RowSet podemos utilizar el método:
next(): Se coloca en la fila siguiente del ResultSet o devuelve false si se terminan las filas.

Dado que un **CachedRowSet** copia todos los datos a la memoria, **no** es **adecuado** para las consultas que devuelvan un **gran número de filas**.

Al igual que un CachedRowSet, un **JdbcRowSet** se puede utilizar para realizar consultas, leer y actualizar tablas de una base de datos. La principal diferencia entre estas dos clases es que un JdbcRowSet **permanece conectado** a la base de datos, mientras que un CachedRowSet no. Realmente un JdbcRowSet actúa como un envoltorio de un ResultSet haciendo que parezca un JavaBean. Esto aporta ciertas ventajas, por ejemplo el poder utilizarlo en los entornos de diseño gráfico, y sobre todo lo que a nosotros más nos interesa: un objeto puede registrarse como oyente de eventos de un RowSet. Cuando varíe el valor de una columna, éste notificará al oyente (Listener), que tomará la acción apropiada. La interfaz que debe implementar el oyente de eventos es **RowSetEventListener** que especifica que hay que implementar los métodos:

void cursorMoved(RowSetEvent ev)
void rowChanged(RowSetEvent ev)
void rowSetChanged(RowSetEvent ev)

Para cargar los datos desde una base de datos en un CachedRowSet escribimos

```
// me conecto usando una conexion
Class.forName("com.mysql.jdbc.Driver");
Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost/bdalumnos", "root", "");
// desactivo la actualizacion automatica de datos
conexion.setAutoCommit(false);
// creo el CachedRowSet
RowSetFactory myRowSetFactory = null;
myRowSetFactory = RowSetProvider.newFactory();
```

```
crs = myRowSetFactory.createCachedRowSet();
// selecciono todos los alumnos
// usando la conexion anterior
crs.setCommand("SELECT * FROM alumnos");
crs.execute(conexion);
// cierro la conexion con la base de datos
conexion.close();
```

EJERCICIOS DE MANIPULACIÓN DE DATOS SQL EN BASES DE DATOS RELACIONALES

1. Crea la clase Java BDUpdateAlumno que modifica en la tabla alumnos de la base de datos MySQL bdalumnos los datos del alumno de dni '11111111A', para que el grupo sea '1DW3'. Si todo ha ido bien muestra todo el contenido de la tabla alumnos por pantalla. Si se ha producido algún error muestra un mensaje informativo.

2. Crea la clase Java BDInsertAlumno que inserta en la tabla alumnos de la base de datos MySQL bdalumnos un alumno de dni '12345678A', nombre 'Nuevo', apellidos 'Alumno', y grupo '1AS3'. Si todo ha ido bien muestra todo el contenido de la tabla alumnos por pantalla. Si se ha producido algún error muestra un mensaje informativo.

3. Crea la clase Java BDDeleteAlumno que borra de la tabla alumnos de la base de datos MySQL bdalumnos el alumno de dni '12345678A'. Si todo ha ido bien muestra todo el contenido de la tabla alumnos por pantalla. Si se ha producido algún error muestra un mensaje informativo.

COMPONENTES GRÁFICOS PARA TRABAJAR CON BASES DE DATOS

Normalmente la manipulación de los elementos de una base de datos se realiza a través de componentes gráficos.

Uno de los componentes que más se usa es JTable.

Para especificar las cabeceras de las columnas de una tabla podemos hacerlo de forma manual escribiendo nosotros los datos. Por ejemplo, para la tabla que muestra los datos de los alumnos vamos a crear un Vector de datos de tipo String de nombre columnas que va a almacenar el valor de la cabecera de cada una de las columnas.

```
// cabeceras de las columnas
columnas = new Vector<String>();
columnas.add("DNI");
columnas.add("Nombre");
columnas.add("Apellidos");
columnas.add("Grupo");
```

También podemos usar un **ResultSetMetaData** de nombre **metaDatos** que guarde todos los metadatos del ResulSet y después sacar de él los nombres que tienen las columnas en la base de datos. Para ello escribimos.

```
// cabeceras de las columnas
ResultSetMetaData metaDatos = rs.getMetaData();
// Se obtiene el número de columnas.
int numeroColumnas = metaDatos.getColumnCount();

columnas = new Vector<String>();
```

```
// Se obtiene cada una de las etiquetas para cada columna
for (int i = 0; i < numeroColumnas; i++){
 // cojo el valor de la etiqueta de la columna
 // los datos del rs empiezan en 1
 columnas.add(metaDatos.getColumnLabel(i + 1));
}
```

Para cargar datos en una JTable podemos usar la clase Java Vector. Lo que haremos será cargar los datos de cada fila en un **Vector** de nombre **fila** y añadirlo a un **Vector de Vectores** de nombre **datosTabla**.

```
// creo el vector para los datos de la tabla
datosTabla = new Vector<Vector<String>>();
// añado uno a uno los alumnos al vector de datos
while (rs.next()) {
 Vector<String> fila = new Vector<String>();
 fila.add(rs.getString("dni"));
 fila.add(rs.getString("nombre"));
 fila.add(rs.getString("apellidos"));
 fila.add(rs.getString("grupo"));
 fila.add("\n\n\n\n\n\n");
 datosTabla.add(fila);
}
```

Una vez que tengamos los datos los podemos crear un objeto de la clase **DefaultTableModel** de nombre **modelo** que almacene los datos de la JTable.

```
// creo la JTable
DefaultTableModel modelo = new DefaultTableModel(datosTabla, columnas);
```

Una vez que ya tenemos el modelo de la tabla creado podemos crear la JTable. Nosotros vamos a crear una **JTable** de nombre **tabla**.

```
tabla = new JTable(modelo);
```

Podemos crear un componente de tipo **JScrollPane** de nombre **scrollPane** para meter en él la tabla y que aparezcan barras de desplazamiento si los datos no entran en el espacio destinado a la tabla.

```
// creo un scroll pane y le añado la tabla
JScrollPane scrollPane = new JScrollPane(tabla);
```

Una vez terminado el diseño de la tabla la añadimos al panel principal de la aplicación.

```
// añado el scroll pane al panel principal
contentPane.add(scrollPane, BorderLayout.CENTER);
```

Podemos ordenar el contenido de la tabla de varias formas. Para permitir que los datos de una tabla se puedan ordenar al hacer clic sobre la cabecera de una columna escribimos

```
tabla.setAutoCreateRowSorter(true);
```

Si lo que queremos es ordenar por los datos de una determinada columna mediante código escribimos

```
// ordeno los datos de la tabla
TableRowSorter<DefaultTableModel> sorter = new TableRowSorter<>(modelo);
tabla.setRowSorter(sorter);
List<RowSorter.SortKey> sortKeys = new ArrayList<>();

// para que ordene por la primera columna (dni en este caso) en Ascendente
int columnIndexToSort = 0;
sortKeys.add(new RowSorter.SortKey(columnIndexToSort, SortOrder.ASCENDING));

sorter.setSortKeys(sortKeys);
sorter.sort();
```

Es importante resaltar que al reordenar los datos de la tabla cambia la posición en qué se encuentran los mismos.

Otra de las operaciones que se suelen hacer con los registros de una tabla es filtrarlos para que sólo aparezcan los que cumplen un determina condición.

Por ejemplo, para que solo aparezcan en una tabla de calificaciones las calificaciones del alumno que esté seleccionado en ese momento escribimos

```
// Defino el método de ordenación
private TableRowSorter<TableModel> metodoOrdenacion;
...
// Después de crear la tabla
// Instanciamos el TableRowSorter y lo añadimos al JTable
metodoOrdenacion = new TableRowSorter<TableModel>(modeloCalificaciones);
tablaCalificaciones.setRowSorter(metodoOrdenacion);
...
// A partir de aquí podemos definir un filtro basado en metodoOrdenacion
// cambio el filtro de la tabla de calificaciones
metodoOrdenacion.setRowFilter(RowFilter.regexFilter(this.txtDNI.getText(), 0));
```

En RowFilter.regexFilter el primer parámetro indica el valor del filtro, y el segundo la columna de la tabla con la que se compara el valor del filtro. Por ejemplo, si el DNI vale '11111111A' y está en la primera columna de la tabla, el ejemplo anterior aplicaría un filtro para que sólo salieran las calificaciones del alumno que tiene ese DNI.

EJERCICIOS COMPONENTES GRÁFICOS PARA TRABAJAR CON BASES DE DATOS

4. Crea la clase Java VentanaJTable que deriva de JFrame y que contiene un panel de nombre contentPane en el que hay una etiqueta de nombre lblTexto con el valor "Datos de los Alumnos" en la parte superior, una tabla de nombre tabla con los datos de los alumnos de la base de datos MySQL bdalumnos y una cabecera con los valores "DNI", "Nombre","Apellidos","Grupo", en el centro, y un botón de nombre btnSalir en la parte inferior. Al pulsar el botón Salir se saldrá de la aplicación.

5. Crea la clase Java VentanaJTableMetadata que modifica VentanaJTable para que la cabecera de la tabla muestre los valores que tienen las columnas en la base de datos.

6. Crea la clase Java VentanaJTableCalificaciones que modifica VentanaJTableMetadata para que en la tabla aparezcan los datos de las calificaciones del alumno con dni '11111111A'.

7. Crea la clase Java VentanaJTableCalificacionesRowSet que modifica VentanaJTableCalificaciones para que en la tabla aparezcan los datos de las calificaciones del alumno con dni '11111111A' usando un CachedRowSet.

8. Crea la clase Java VentanaJTableAlumnosRowSet que modifica VentanaJTableCalificacionesRowSet para que en la tabla aparezcan los datos de los alumnos.

9. Crea la clase Java VentanaJTableAlumnosRowSetEventosInsertar que modifica VentanaJTableAlumnosRowSet. Añade un nuevo registro al RowSet y controla los eventos que se producen. Los datos de la tabla se deben de actualizar para reflejar los cambios.

10. Crea la clase Java VentanaJTableAlumnosRowSetEventosBorrar que modifica VentanaJTableAlumnosRowSetInsertar. Borra el nuevo registro creado antes del RowSet y controla los eventos que se producen. Los datos de la tabla se deben de actualizar para reflejar los cambios.

11. Crea la clase Java VentanaJTableAlumnosRowSetEventosActualizar que modifica VentanaJTableAlumnosRowSetInsertar. Modifica el valor del campo grupo del registro con DNI '11111111A' con el valor 'ADW3' y controla los eventos que se producen. Los datos de la tabla se deben de actualizar para reflejar los cambios.

12. Crea la clase Java VentanaJTableAlumnosRowSetEventosCampos que modifica VentanaJTableCalificacionesRowSet. Añade un campo de texto por cada campo de la tabla Alumnos con su correspondiente etiqueta. Además, añade los botones
    Primero. Permite ir al primer registro de datos (si es que hay).
    Siguiente. Permite ir al registro siguiente (si es que hay).
    Anterior. Permite ir al registro anterior (si es que hay).
    Ultimo. Permite ir al último registro (si es que hay).
    Nuevo. Inicializa los campos de texto a unos valores por defecto.
    Insertar. Añade un nuevo registro a la base de datos y a la tabla con los datos de los campos de texto. Debe controlar todos los errores que se produzcan.
    Actualizar. Modifica los datos del registro cuyo dni coincide con el campo de texto DNI en la base de datos y en la tabla con los datos de los otros campos de texto. Debe controlar todos los errores que se produzcan.
    Borrar. Borra el registro cuyo dni coincide con el campo de texto DNI de la base de datos y de la tabla. Debe controlar todos los errores que se produzcan.

13. Crea la clase Java VentanaJTableAsignaturasRowSetEventosCampos que modifica VentanaJTableAlumnosRowSetEventosCampos para que ahora trabaje con los datos de la tabla Asignaturas.

14. Crea la clase Java VentanaJTableAsignaturasListas que modifica VentanaJTableAsignaturasRowSetEventosCampos de la siguiente manera. Elimina los campos de texto y la tabla y añade dos combo box de nombre cmbCodAsignatura y cmbNombreAsignatura en las que carga al inicio los datos de codasignatura y nombreasignatura desde la tabla Asignaturas de la base de datos. Añade un botón de nombre btnInsertarAsignatura que al hacer clic sobre él añade a una lista de nombre lstAsignaturas el nombre de la asignatura seleccionada en ese momento en cmbNombreAsignatura. Al cambiar el elemento seleccionado en cmbCodAsignatura se seleccionará su elemento correspondiente en cmbNombreAsignatura. La lista estará dentro de un panel desplegable.

15. Crea la clase Java VentanaJTableAsignaturasCalificaciones que modifica VentanaJTableAlumnosRowSetEventosCampos para que ahora en la tabla aparezcan los datos de las Calificaciones del alumno que esté seleccionado en ese momento.

16. Crea la clase Java VentanaBDAlumnosCompleto que modifica VentanaJTableAlumnosRowSetEventosCampos. En la tabla aparecerán ahora sólo las calificaciones del alumno que esté seleccionado. Debemos añadir una lista desplegable de asignaturas para seleccionar la calificación de qué asignatura queremos añadir a la tabla de calificaciones y un botón que permita añadir la calificación de esa asignatura para ese alumno (por defecto la nota será 5.0). Las calificaciones se pueden borrar de la tabla al hacer doble clic sobre ellas.