

Programación

JBCD MySQL

**1º DESARROLLO DE
APLICACIONES
MULTIPLATAFORMA (D.A.M.)
2021-2022**

API (Applications Programming Interface) es un conjunto de clases que trabajan de forma coordinada y conjunta proporcionando ciertos servicios, como por ejemplo interactuar con una base de datos, configurar una red o gestionar los movimientos de un robot.

JDBC: Java DataBase Connectivity. Es una API que permite ejecutar sentencias SQL en un SGBD desde una aplicación Java, que funcionara como un cliente que accede a los servicios del servidor de base de datos.

SGBD (sistema gestor de base de datos) es un conjunto de software que permite almacenar y gestionar información en una base de datos. También proporciona servicios de recuperación e integridad de datos, control de acceso de usuarios, copias de seguridad, etc...

API JDBC

La API de JDBC se encuentra en el paquete `java.sql` y está compuesto por un sinnúmero de clases que trabajan de forma conjunta. No es necesario conocerlas todas, solo es imprescindible saber cuáles son las principales:

- **DriverManager:** permite manipular los distintos drivers. Con cada driver se puede acceder a un SGBD distinto.
- **Connection:** crea una conexión entre la aplicación y la base de datos.
- **Statement:** representa una sentencia SQL que ejecutará el servidor de base de datos.
- **PreparedStatement:** también representa una sentencia SQL, que permite configurar o parametrizar fácilmente valores en la consulta, como por ejemplo la edad de un alumno o su fecha de nacimiento en una condición.
- **ResultSet:** representa una tabla con el resultado que genera el SGBD tras ejecutar una sentencia de consulta de información (SELECT).

Driver

Cada fabricante de un SGBD, al desarrollar su producto, usa mecanismos propios (protocolos, llamadas, API de bajo nivel, etc.) que establecen una conexión con la base de datos y permiten acceder a sus servicios. Las clases que componen la API de JDBC no conocen estos detalles propios de cada producto: MySQL, Oracle Database, PostgreSQL.

Por lo tanto, ¿cómo es posible que las clases de la API de JDBC finalmente lleguen a establecer conexión con el servidor de base de datos? El mecanismo es muy sencillo: entre las clases que componen la API de JDBC existe una especial (que se denomina Driver) que será específica de cada SGBD. De hecho, cuando un fabricante desarrolla un nuevo SGBD deberá desarrollar también el driver que permite su uso con JDBC. La clase Driver o driver de JDBC se añadirá a nuestro programa en función del SGBD que hayamos seleccionado.

API JDBC

Driver

DriverManager

Connection

Statement

PreparedStatement

ResultSet

Oracle

SGBD
Oracle

MySQL

SGBD
MySQL

DB2

SGBD
DB2

...

...

Conexión

Antes de trabajar con la base de datos, hay que crear una conexión entre nuestra aplicación y el SGBD. Esta conexión funciona como un tubo que comunica ambas partes, permitiendo que las sentencias SQL viajen desde la aplicación al SGBD y los resultados de las consultas se muevan en sentido contrario. Mientras la necesitemos, la conexión deberá permanecer abierta; una vez que ya no sea útil, hay que cerrarla. En el caso de que una conexión no se cierre, quedará abierta consumiendo recursos del SGBD.

Para crear una conexión disponemos del método estático de DriverManager:

- `Connection getConnection(String url, String usuario, String password)`

El primer parámetro identifica la base de datos a la que queremos acceder, en general, para MySQL tendrá la forma:

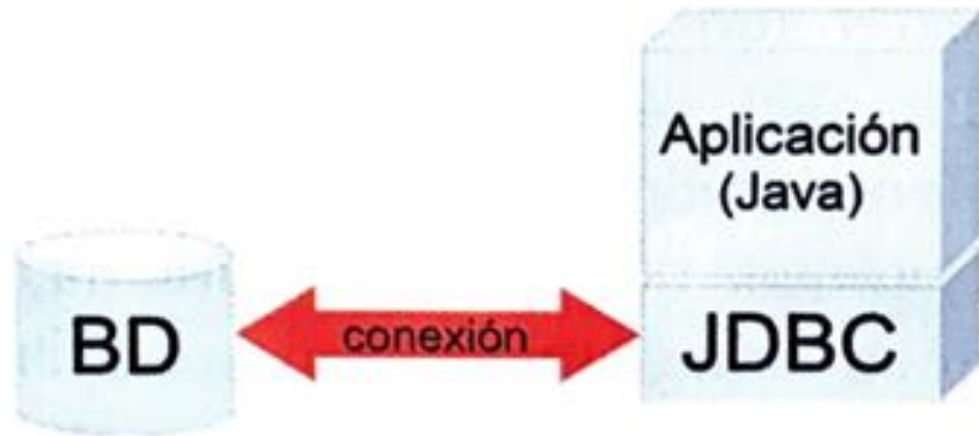
Jdbc:mysql://<servidor>/<base de datos>

Donde:

- <servidor>: es el nombre o la dirección IP de la máquina donde está instalado el servidor de BD. En el caso que el servidor esté instalado en la máquina local, puede usarse localhost.
- <base de datos>: nombre de la BD dentro del SGBD.

Los siguientes dos parámetros son una cadena con el nombre del usuario y su contraseña.

El método devuelve un objeto de tipo **Connection**, que representa la conexión establecida entre nuestra aplicación y la base de datos.



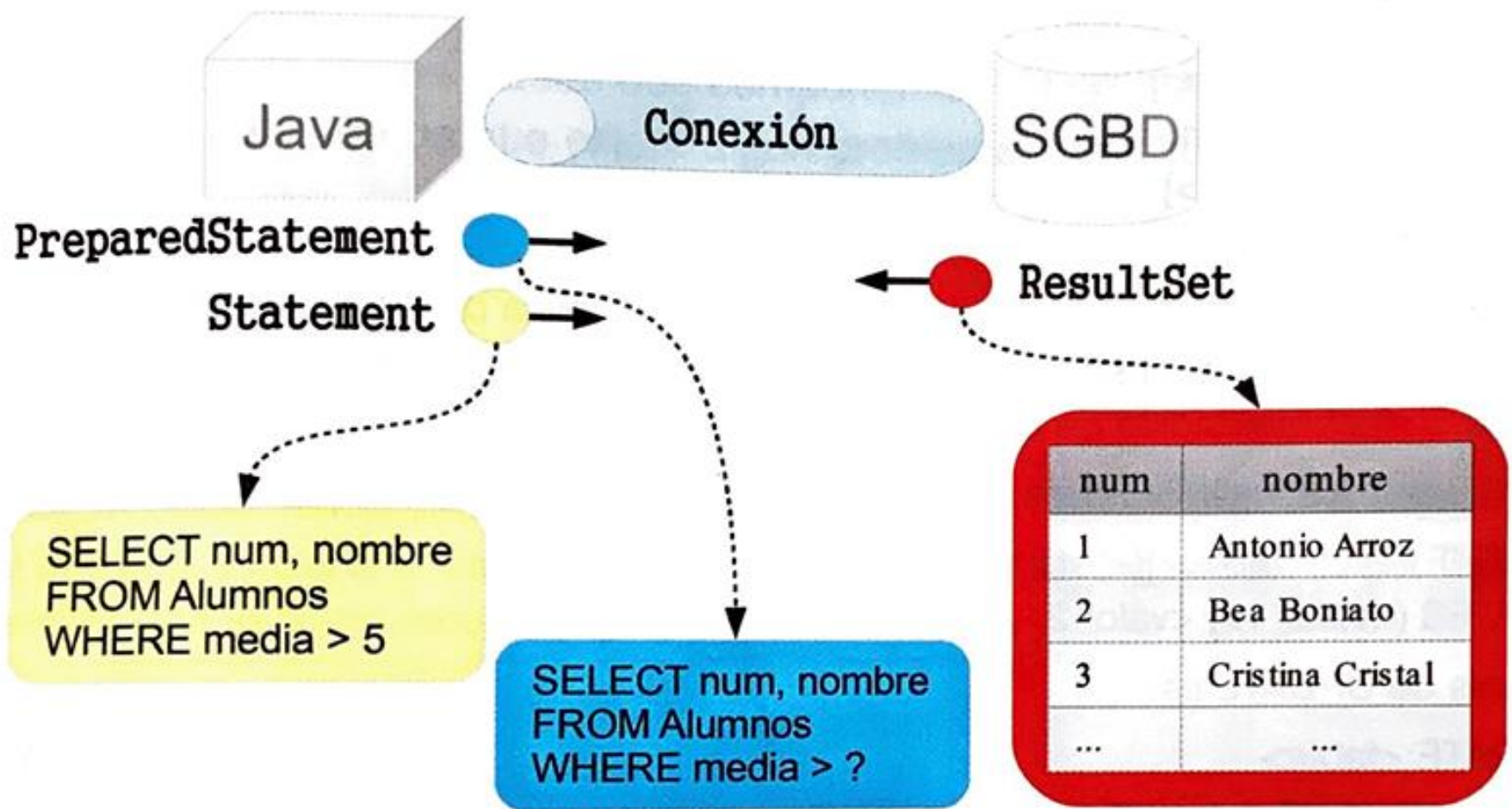
Suponiendo que nuestra base de datos se llama Instituto y que disponemos del usuario «Pepe» con contraseña «12345», la manera de crear una conexión será:

Connection con;

```
String url = "jdbc:mysql://localhost/Instituto"
```

```
con= DriverManager.getConnection(url, "Pepe", "12345");
```

Cada conexión es una tubería que permite que viajen distintos objetos. Cada tipo de objetos representa una información útil para el SGBD o para la aplicación.



El método **getConnection()**, en el caso que se produzca algún error al crear la conexión lanzará alguna de las siguientes excepciones:

- **SQLException**: si ocurre algún error en el acceso a la base de datos o la URL es null.
- **SQLException**: cuando el tiempo que ha transcurrido sin llegar a conectar a la base de datos es excesivo.

59 - JDBC con MySQL

JDBC son las siglas en ingles de **Java Database Connectivity**. Es un conjunto de clases que nos permite acceder a diversos gestores de bases de datos en forma transparente.

Veremos como conectarnos con el motor de base de datos MySQL.

Instalación de **MySQL** integrado en el **WampServer**

Utilizaremos esta herramienta en lugar de descargar solo el MySQL con la finalidad de facilitar la instalación y configuración del motor de base de datos (la instalación de esta herramienta es sumamente sencilla), además utilizaremos otra software que provee el WampServer que es el PhpMyAdmin que nos facilitará la creación de la base de datos.

Procedemos a descargar el WampServer de la siguiente página: [aquí](#).

Luego de descargarlo procedemos a ejecutar el instalador:



DOWNLOAD WAMPSERVER 64

WampServer est disponible gratuitement (sous licence GPL) actualités formation d'Alter Way, société editrice, ainsi que to pas, vous pouvez [you can download it directly.](#)



WampServer

A Windows Web development environment for Apache, MySQL, PHP databases

Brought to you by: [alterway](#), [herveleclerc](#), [otomatic](#)



136 Reviews

Downloads: 53,764 This Week



Download

Get Updates

Share This

Windows

[Summary](#)

[Files](#)

[Reviews](#)

[Support](#)

[Wiki](#)

License Agreement

Please read the following important information before continuing.



Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

** WampServer

Creator : Romain Bourdon

Maintainer/Upgrade to 2.5 : Herve Lederer

Upgrade 2.5 to 3.0.0 : Otomatic (wampserver@otomatic.net)

<http://forum.wampserver.com/index.php>

GNU LESSER GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

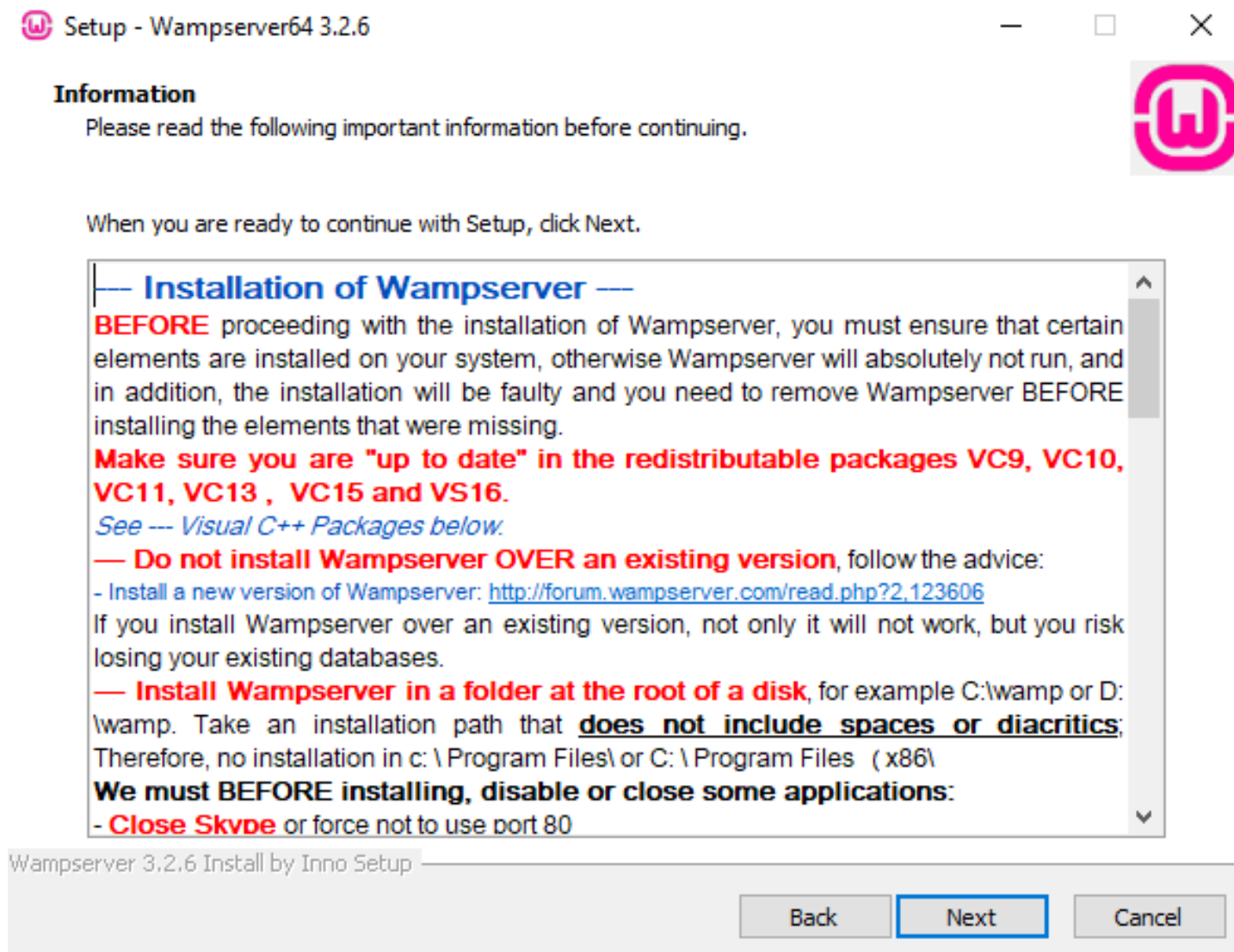
Copyright (C) 2007 Free Software Foundation, Inc. <<http://fsf.org/>>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates

☒ I accept the agreement

☐ I do not accept the agreement

Después de descargarlo procedemos a ejecutar el instalador:



Select Destination Location

Where should Wampserver64 be installed?



Setup will install Wampserver64 into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

c:\wamp64

Browse...

At least 393,9 MB of free disk space is required.

Wampserver 3.2.6 Install by Inno Setup

Back

Next

Cancel



Select Components

Which components should be installed?



Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

Note that you have the possibility, after this installation, to add "addons", i.e. other versions of Apache, PHP, MySQL and MariaDB.

Custom installation	
<input checked="" type="checkbox"/> Wampmanager	
<input checked="" type="checkbox"/> Apache 2.4.51	
<input checked="" type="checkbox"/> PHP 5.6.40	56,1 MB
<input type="checkbox"/> PHP 7.0.33	59,3 MB
<input type="checkbox"/> PHP 7.1.33	58,7 MB
<input type="checkbox"/> PHP 7.2.34	65,1 MB
<input checked="" type="checkbox"/> PHP 7.3.33	66,0 MB
<input checked="" type="checkbox"/> PHP 7.4.26	67,5 MB
<input checked="" type="checkbox"/> PHP 8.0.13	70,3 MB
<input checked="" type="checkbox"/> PHP 8.1.0	78,1 MB
<input checked="" type="checkbox"/> MariaDB	225,0 MB
<input type="checkbox"/> ... MariaDB 10.6.5	225,0 MB

Current selection requires at least 2,45 GB of disk space.



Select Start Menu Folder

Where should Setup place the program's shortcuts?



Setup will create the program's shortcuts in the following Start Menu folder.

To continue, click Next. If you would like to select a different folder, click Browse.

Wampserver64

Browse...

Ready to Install

Setup is now ready to begin installing Wampserver64 on your computer.



Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:

c:\wamp64

Setup type:

Custom installation

Selected components:

Wampmanager

Apache 2.4.51

PHP 5.6.40

PHP 7.3.33

PHP 7.4.26

PHP 8.0.13

PHP 8.1.0

MariaDB

MariaDB 10.6.5

MySQL

MySQL 5.7.36

Applications

OpenMyAdmin 5.1.1

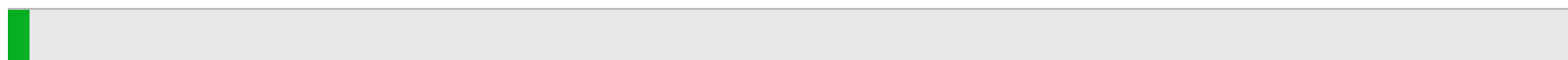
Installing

Please wait while Setup installs Wampserver64 on your computer.



Extracting files...

c:\wamp64\apps\phpmyadmin4.9.7\themes\pmahomme\jquery\images\ui-bg_glass_95_fef1ec_1x400.png



Setup



iexplore.exe (Internet Explorer)
will be used as Browser by Wampserver.

Do you want to choose another Browser installed on your
system?

Sí

No

Information

Please read the following important information before continuing.



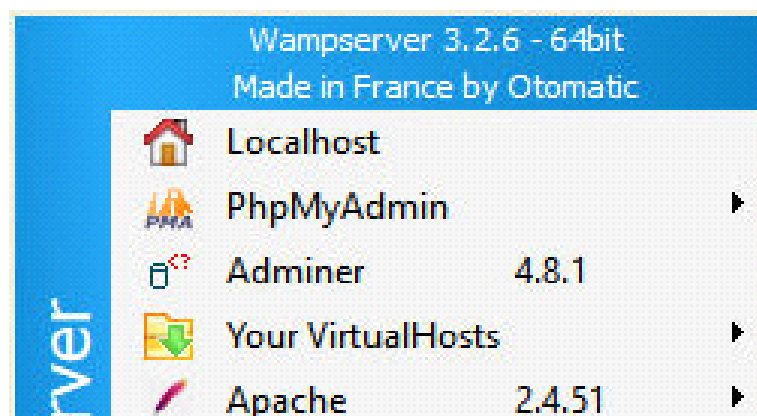
When you are ready to continue with Setup, click Next.

For some explanations on **how Wampserver works**
see the file: wamp(64)\instructions_for_use.pdf

For the use of **MariaDB and MySQL**
 See the file : wamp(64)\mariadb_mysql.txt

Wampmanager icon works with
 Left-Click and Right-Click

Left-Click





WampServer
Version 3.0.0 by
Otomatic

Wampserver

3.2.6

Completing the Wampserver64 Setup Wizard

Setup has finished installing Wampserver64 on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

Back

Finish

Ahora podemos ver el iconos del WampServer en la bandeja del sistema de Windows (si se encuentra en color verde significa que el MySQL está ejecutándose correctamente):



Wampserver

Wampserver 3.2.6 - 64bit
Made in France by Otomatic

 Localhost

 PhpMyAdmin ▶

 Adminer 4.8.1

 Your VirtualHosts ▶

 Apache 2.4.51 ▶

 PHP 7.4.26 ▶

Default DBMS: mysql

 MySQL 5.7.36 ▶

 MariaDB 10.6.5 ▶

 Help -> MariaDB - MySQL

Services

Start All Services

Stop All Services

Restart All Services



El usuario de phpMyAdmin será “root” y la contraseña la dejaremos en blanco.



Bienvenido a phpMyAdmin

Idioma - *Language*

Español - Spanish

Iniciar sesión 

Usuario:

Contraseña:

Server choice: MySQL

Continuar

El PhpMyAdmin es un programa web que nos permite administrar las bases de datos del MySQL:

The screenshot displays the phpMyAdmin web interface in a browser window. The address bar shows 'localhost / MySQL | phpMy...'. The interface includes a top navigation bar with tabs: 'Bases de datos', 'SQL', 'Estado actual', 'Cuentas de usuarios', 'Exportar', 'Importar', 'Configuración', 'Replicación', and 'Más'. The left sidebar shows the 'Servidor actual:' dropdown set to 'MySQL', with buttons for 'Reciente' and 'Favoritas'. Below these are links for 'Nueva', 'information_schema', 'mysql', 'performance_schema', and 'sys'. The main content area is divided into several panels:

- Configuraciones generales**: Includes a 'Cambio de contraseña' link, a 'Server connection collation' dropdown set to 'utf8mb4_unicode_ci', and a 'Más configuraciones' link.
- Configuraciones de apariencia**: Includes an 'Idioma - Language' dropdown set to 'Español - Spanish' and a 'Tema' dropdown set to 'pmahomme'.
- Servidor de base de datos**: Lists server details:
 - Servidor: MySQL (127.0.0.1 via TCP/IP)
 - Tipo de servidor: MySQL
 - Conexión del servidor: No se está utilizando SSL
 - Versión del servidor: 5.7.36 - MySQL Community Server (GPL)
 - Versión del protocolo: 10
 - Usuario: root@localhost
 - Conjunto de caracteres del servidor: cp1252 West European (latin1)
- Servidor web**: Lists web server details:
 - Apache/2.4.51 (Win64) PHP/7.4.26
 - Versión del cliente de base de datos: libmysql - mysqlnd 7.4.26
 - extensión PHP: mysqli, curl, mbstring
 - Versión de PHP: 7.4.26
- phpMyAdmin**: Includes version information:
 - Acerca de esta versión: 5.1.1, versión estable más reciente: 5.2.0
 - [Documentación](#)

A 'Consola' tab is visible at the bottom left of the main content area.

Seleccionamos la pestaña "Base de datos" y donde dice "Crear nueva base de datos" especificamos que nuestra base de datos se llamará "bd1":

localhost / MySQL | phpMy...

phpMyAdmin

Servidor actual: MySQL

Reciente Favoritas

Nueva

- information_schema
- mysql
- performance_schema
- sys

Servidor: MySQL:3306

Bases de datos SQL Estado actual Cuentas de usuario

Bases de datos

Crear base de datos

bd1 latin1_swedish_ci **Crear**

	Base de datos	Cotejamiento	Acción
<input type="checkbox"/>	information_schema	utf8_general_ci	Seleccionar privilegios
<input type="checkbox"/>	mysql	latin1_swedish_ci	Seleccionar privilegios
<input type="checkbox"/>	<u>performance_schema</u>	utf8_general_ci	Seleccionar privilegios
<input type="checkbox"/>	sys	utf8_general_ci	Seleccionar privilegios

Total: 4

Presionamos el botón "crear" y con esto ya tenemos nuestra base de datos creada:

The screenshot displays the phpMyAdmin web interface. On the left sidebar, the 'phpMyAdmin' logo is at the top, followed by navigation icons and a 'Servidor actual:' dropdown menu set to 'MySQL'. Below this are 'Reciente' and 'Favoritas' buttons, and a tree view of databases including 'Nueva', 'bd1', 'information_schema', 'mysql', 'performance_schema', and 'sys'. The main panel has a header bar showing 'Servidor: MySQL:3306' and 'Base de datos: bd1'. A toolbar contains buttons for 'Estructura', 'SQL', 'Buscar', 'Generar una consulta', 'Exportar', and 'Importar'. A message box states 'No se han encontrado tablas en la base de datos.' Below this is the 'Crear tabla' section, which includes a 'Nombre:' text input field, a 'Número de columnas:' label with a value of '4' in its input field, and a 'Continuar' button.

Después de seleccionar la base de datos "bd1" que figura a la izquierda procedemos a crear la primer tabla que contendrá (crearemos una tabla llamada "artículos" y que tendrá tres campos):

The screenshot shows the phpMyAdmin web interface. On the left sidebar, the 'bd1' database is selected. The main panel displays the 'Estructura' (Structure) tab for the 'bd1' database. A message states: 'No se han encontrado tablas en la base de datos.' (No tables found in the database). Below this, the 'Crear tabla' (Create table) form is visible. The 'Nombre' (Name) field contains 'articulos' and the 'Número de columnas' (Number of columns) field contains '3'. A 'Continuar' (Continue) button is at the bottom of the form.

phpMyAdmin

Servidor actual: MySQL

Reciente Favoritas

Nueva

bd1

information_schema

mysql

performance_schema

sys

Servidor: MySQL:3306 » Base de datos: bd1

Estructura SQL Buscar Generar una consulta Exportar

⚠ No se han encontrado tablas en la base de datos.

Crear tabla

Nombre: Número de columnas:

Continuar

En la tabla "articulos" definimos el campo "codigo" de tipo int (este campo será el "primary key" y auto_increment lo tildamos para que el código se genere automáticamente), el segundo campo es la descripción que es de tipo varchar con un máximo de 50 caracteres y por último el campo precio que es de tipo float.

Una vez especificados los tres campos en la parte inferior de la misma ventana aparece un botón llamada "Guardar" para confirmar la estructura de la tabla:

← Servidor: MySQL:3306 » Base de datos: bd1 » Tabla: articulos

Examinar Estructura SQL Buscar Insertar Exportar Importar Privilegios Operaciones Disparadores

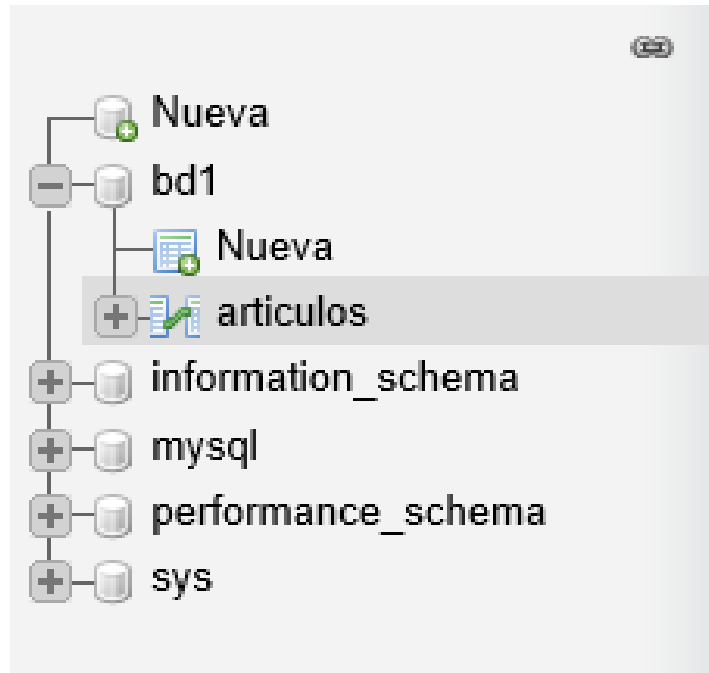
Nombre de la tabla: articulos Agregar 1 columna(s) Continuar

Nombre	Tipo	Longitud/Valores	Predeterminado	Cotejamiento	Atributos	Nulo	Índice
codigo	INT		Ninguno			<input checked="" type="checkbox"/>	UNIQUE
descripcion	VARCHAR	50	Ninguno			<input type="checkbox"/>	---
precio	FLOAT		Ninguno			<input type="checkbox"/>	---

Estructura

Comentarios de la tabla: Cotejamiento: Motor de almacenamiento:

En el lado izquierdo del navegador podemos ver ahora que la base de datos "bd1" tiene una tabla llamada "artículos"



Hasta aquí lo que nos ayuda el PhpMyAdmin (a crear la base de datos y la tabla), de ahora en adelante todas las otras actividades las desarrollaremos desde nuestro programa en java (poblar o insertar datos, listar registros, consultar, modificar y borrar datos)

Descarga del Driver para permitir conectar nuestro programa Java con el MySQL

La última actividad de configuración previa a implementar los programas en Java para acceder a MySQL es la descarga del Driver que nos permita conectarnos con la base de datos.

El Driver lo podemos descargar de la página: [aquí](#)

Podemos descomprimir el archivo `mysql-connector-java-3.1.8.zip` que acabamos de descargar. Luego veremos que en nuestro programa en java haremos referencias al archivo `mysql-connector-java-3.1.8-bin.jar` (que es el Driver propiamente dicho).

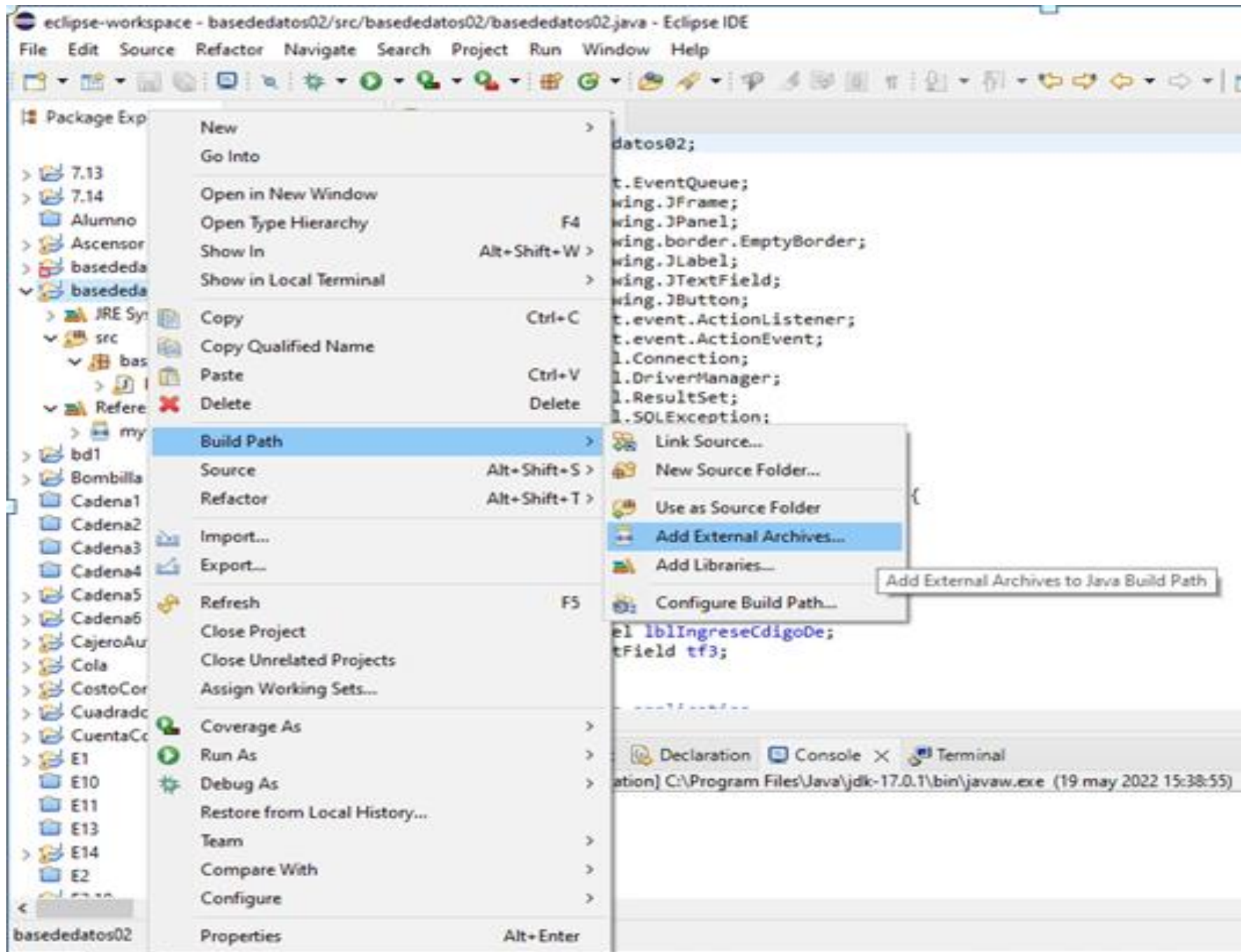
60 - Alta y Consulta de una tabla de MySQL

Problema 1


Ya creamos en el concepto anterior una base de datos llamada bd1 y en la misma creamos una tabla llamada articulos.








Procederemos a implementar en Java un programa que nos permita comunicarnos con la base de datos "bd1" e insertar filas en la tabla "articulos" y posteriormente consultar su contenido.

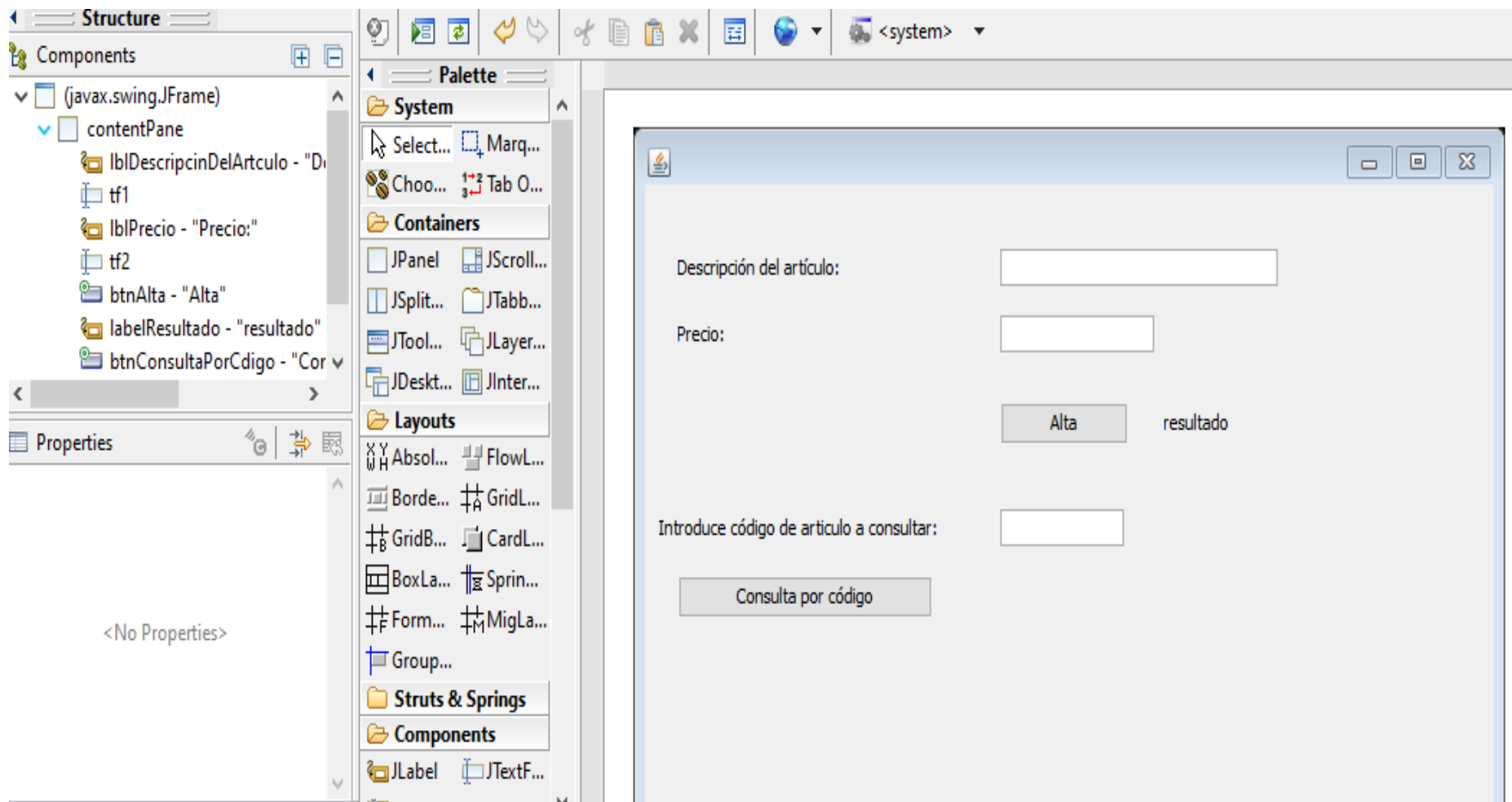
- 1 - Creamos desde Eclipse un proyecto llamado "basededatos01" y seguidamente con el WindowBuilder creamos una clase llamada "Formulario".
- 2 - Primero debemos añadir el driver que descargamos (mysql-connector-java-3.1.8-bin.jar) presionamos el botón derecho del mouse sobre nuestro proyecto, aparece el siguiente diálogo:



Seleccionamos la opción "Build Path", y procedemos a presionar el botón "Add External Archives.", donde procedemos a buscar el archivo mysql-connector-java-3.1.8-bin.jar

Nombre	Fecha de modificación	Tipo	Tamaño
 mysql-connector-java-3.1.8-bin	14/04/2005 22:44	Executable Jar File	400 KB

- ▼  basededatos02
 - >  JRE System Library [JavaSE-17]
 - ▼  src
 - ▼  basededatos02
 - >  basededatos02.java
 - ▼  Referenced Libraries
 - >  mysql-connector-java-3.1.8-b



El código fuente de la clase formulario es:

```
package basededatos01;

import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

```
public class Formulario extends JFrame {
```

```
    private JPanel contentPane;  
    private JTextField tf1;  
    private JTextField tf2;  
    private JLabel labelResultado;  
    private JButton btnConsultaPorCdigo;  
    private JLabel lblIngreseCdigoDe;  
    private JTextField tf3;
```

```
/**
 * Launch the application.
 */
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                Formulario frame = new Formulario();
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}
```

```
/**
 * Create the frame.
 */
public Formulario() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 606, 405);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(null);

    JLabel lblDescripcinDelArticulo = new JLabel("Descripción del artículo:");
    lblDescripcinDelArticulo.setBounds(23, 38, 193, 14);
    contentPane.add(lblDescripcinDelArticulo);

    tf1 = new JTextField();
    tf1.setBounds(247, 35, 193, 20);
    contentPane.add(tf1);
    tf1.setColumns(10);

    JLabel lblPrecio = new JLabel("Precio:");
    lblPrecio.setBounds(23, 74, 95, 14);
    contentPane.add(lblPrecio);
```

```
tf2 = new JTextField();  
tf2.setBounds(247, 71, 107, 20);  
contentPane.add(tf2);  
tf2.setColumns(10);
```

```
JButton btnAlta = new JButton("Alta");  
btnAlta.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent arg0) {  
        labelResultado.setText("");  
        try {  
            Connection  
conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root" ,"");  
            Statement comando=conexion.createStatement();  
            comando.executeUpdate("insert into articulos(descripcion,precio) values  
('"+tf1.getText()+"','"+tf2.getText()+"");  
            conexion.close();  
            labelResultado.setText("se registraron los datos");  
            tf1.setText("");  
            tf2.setText("");  
        } catch(SQLException ex){  
            setTitle(ex.toString());  
        }  
    }  
});
```

```
btnAlta.setBounds(247, 118, 89, 23);  
contentPane.add(btnAlta);
```

```
labelResultado = new JLabel("resultado");  
labelResultado.setBounds(361, 122, 229, 14);  
contentPane.add(labelResultado);
```

```
btnConsultaPorCodigo = new JButton("Consulta por código");  
btnConsultaPorCodigo.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent arg0) {  
        labelResultado.setText("");  
        tf1.setText("");  
        tf2.setText("");  
        try {  
            Connection  
conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root" ,"");  
            Statement comando=comexion.createStatement();  
            ResultSet registro = comando.executeQuery("select descripcion,precio from articulos  
where codigo="+tf3.getText());
```

```
if (registro.next()==true) {  
    tf1.setText(registro.getString("descripcion"));  
    tf2.setText(registro.getString("precio"));  
} else {  
    labelResultado.setText("No existe un artículo con dicho código");  
}  
conexion.close();  
  
} catch(SQLException ex){  
    setTitle(ex.toString());  
}  
}  
});
```



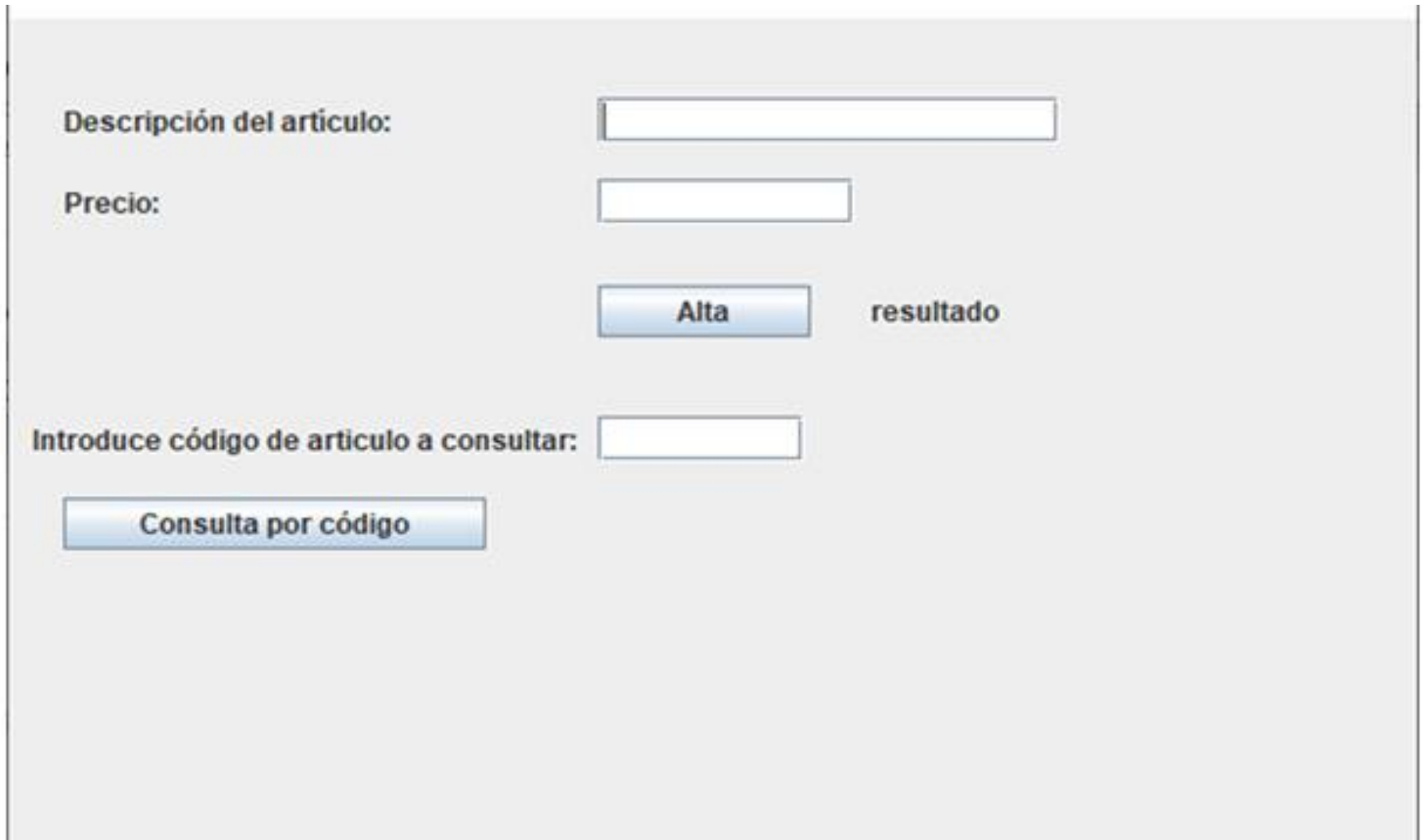
```
btnConsultaPorCdigo.setBounds(23, 212, 177, 23);  
contentPane.add(btnConsultaPorCdigo);
```

```
lblIngreseCdigoDe = new JLabel("Introduce código de articulo a consultar:");  
lblIngreseCdigoDe.setBounds(10, 179, 243, 14);  
contentPane.add(lblIngreseCdigoDe);
```

```
tf3 = new JTextField();  
tf3.setBounds(247, 176, 86, 20);  
contentPane.add(tf3);  
tf3.setColumns(10);  
cargarDriver();  
}
```

```
private void cargarDriver() {  
    try {  
        Class.forName("com.mysql.jdbc.Driver");  
    } catch (Exception ex) {  
        setTitle(ex.toString());  
    }  
}  
}
```

La interfaz visual de la aplicación a implementar es la siguiente (se solicita introducir la descripción del artículo y su precio, cuando se presiona el botón "Alta" se procede a insertar una fila en la tabla articulos de la base de datos bd1):



The image shows a web application interface with a light gray background. It contains several form elements and buttons:

- Descripción del artículo:** A text label followed by a long, empty text input field.
- Precio:** A text label followed by a shorter, empty text input field.
- Alta**: A blue button with a gradient and a shadow, located below the price input field.
- resultado**: A text label positioned to the right of the "Alta" button.
- Introduce código de articulo a consultar:** A text label followed by a text input field.
- Consulta por código**: A blue button with a gradient and a shadow, located below the search input field.

Expliquemos ahora el código fuente de la aplicación:

Primero debemos cargar en memoria el Driver, esto lo hacemos mediante el método `cargarDriver` que es llamado luego desde el constructor de la clase:

```
private void cargarDriver() {  
    try {  
        Class.forName("com.mysql.jdbc.Driver");  
    } catch (Exception ex) {  
        setTitle(ex.toString());  
    }  
}
```

Tenemos una clase llamada "Class" que tiene un método estático llamado `forName`, al mismo hay que pasar el nombre de la clase a importar:

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

`com.mysql.jdbc` es el nombre del paquete donde se encuentra la clase `Driver`.
Esta es la forma en que importamos los driver en Java.

El método `forName` de la clase `Class` genera excepciones de tipo `Exception` que deben ser capturadas obligatoriamente (luego por eso encerramos el código en un bloque `try/catch`).

Si no importamos el driver desde el diálogo Properties del proyecto o indicamos en forma incorrecta el nombre del paquete o clase luego aparece en el título del JFrame un mensaje del error sucedido.

Luego desde el constructor llamamos por única vez al método cargarDriver:

```
.....  
tf3 = new JTextField();  
tf3.setBounds(247, 176, 86, 20);  
contentPane.add(tf3);  
tf3.setColumns(10);  
cargarDriver();  
}
```

Veamos ahora cual es el código a implementar cuando se presiona el botón "Alta":

```
JButton btnAlta = new JButton("Alta");
    btnAlta.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
            labelResultado.setText("");
            try {
                Connection
conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root" ,"");
                Statement comando=conexion.createStatement();
                comando.executeUpdate("insert into articulos(descripcion,precio) values
('"+tf1.getText()+"','"+tf2.getText()+"')");
                conexion.close();
                labelResultado.setText("se registraron los datos");
                tf1.setText("");
                tf2.setText("");
            } catch(SQLException ex){
                setTitle(ex.toString());
            }
        }
    });
```

En el `actionPerformed` procedemos primero a limpiar la label que puede tener un mensaje de ejecuciones anteriores:

```
labelResultado.setText("");
```

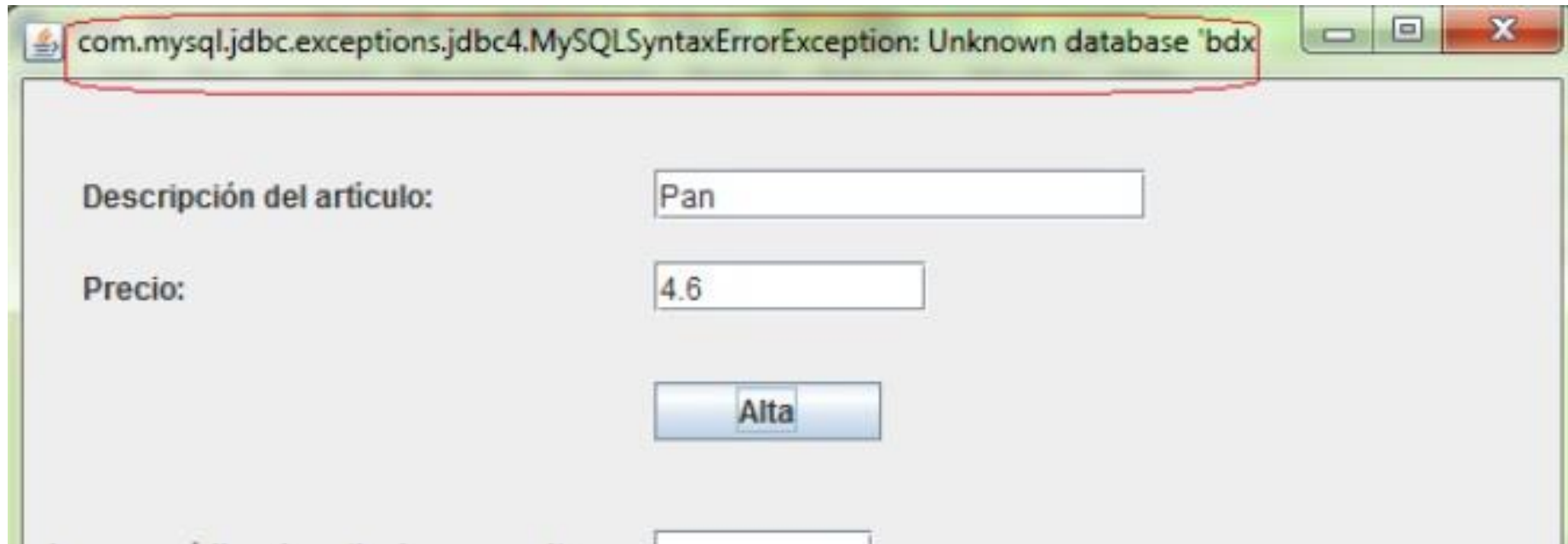
Todas las clases orientadas al acceso a base de datos generan excepciones de tipo `SQLException` y deben ser capturadas obligatoriamente. Lo primero que hacemos es crear un objeto de la clase `Connection`, para esto la clase `DriverManager` tiene un método llamado `getConnection` que retorna un objeto de la clase `Connection`:

```
Connection conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root" , "");
```

El método `getConnection` debemos pasarle tres `String`, el primero indica el nombre de la base de datos que queremos acceder (en este caso "bd1"), el segundo parámetro es el nombre de usuario (recordemos que cuando instalamos el MySQL se crea un usuario por defecto llamado "root") y el último parámetro el de la clave del usuario "root", por defecto esta clave es un `String` vacío.

Como podemos ver también previo a la base de datos tenemos en la cadena de conexión el nombre de nuestro servidor (localhost)

Si nos equivocamos por ejemplo con el nombre de base de datos a comunicarnos (por ejemplo cambiar "bd1" por "bdx") veremos en el título del JFrame el mensaje de error que nos devuelve el MySQL:



Luego creamos un objeto de la clase Statement a partir del objeto de la clase Connection que acabamos de crear:

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

La clase Statement tiene un método llamado executeUpdate que le pasamos el comando SQL insert para agregar una fila a la tabla articulos:

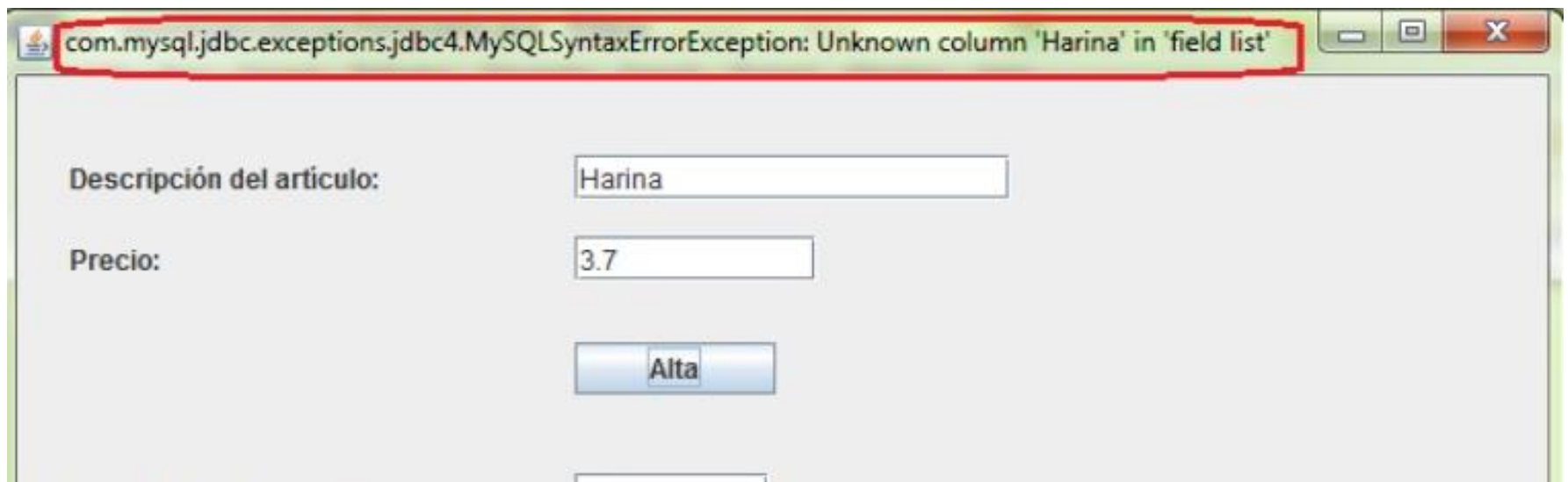
```
comando.executeUpdate("insert into articulos(descripcion,precio) values  
('"+tf1.getText()+"','"+tf2.getText()+"");
```

Como podemos ver generamos el String con el comando insert rescatando los datos de los dos controles de tipo JTextField. Es importante notar que en Java los String están encerrados entre comillas dobles y los concatenamos con el operador +. Las comillas simples son necesarias para los campos de tipo varchar de MySql (como podemos notar el lugar donde se dispondrá el texto de la descripción del artículo deben ir obligatoriamente las comillas simples):

```
..." + tf1.getText() + "..."
```

Si nos olvidamos las comillas simples al generar el String con el comando Insert el MySQL nos devolverá un error que será capturado por el try/catch, por ejemplo si lo ejecutamos con la siguiente sintaxis (sin las comillas simples envolviendo el valor de la descripción):

```
comando.executeUpdate("insert into articulos(descripcion,precio) values  
("+tf1.getText()+", "+tf2.getText()+")");
```



Luego de solicitar la ejecución del comando Insert al MySQL procedemos a llamar al método close de la clase Connection:

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

Con lo visto ya podemos agregar filas a la tabla articulos. Veamos ahora como consultar datos. El código a implementar cuando se presiona el botón "Consulta por código" es el siguiente:

```

btnConsultaPorCodigo = new JButton("Consulta por código");
btnConsultaPorCodigo.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        labelResultado.setText("");
        tf1.setText("");
        tf2.setText("");
        try {
            Connection conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root","");
            Statement comando=conexion.createStatement();
            ResultSet registro = comando.executeQuery("select descripcion,precio from articulos where codigo="+tf3.getText());
            if (registro.next()==true) {
                tf1.setText(registro.getString("descripcion"));
                tf2.setText(registro.getString("precio"));
            } else {
                labelResultado.setText("No existe un artículo con dicho código");
            }
            conexion.close();
        } catch(SQLException ex){
            setTitle(ex.toString());
        }
    }
});

```

De forma similar al Insert procedemos a crear un objeto de la clase Connection y otro objeto de la clase Statement:

```
Connection conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root",
,"");
Statement comando=conexion.createStatement();
```

Seguidamente definimos una variable de la clase ResultSet llamada registro y llamamos al método executeQuery de la clase Statement del objeto que acabamos de crear previamente:

```
ResultSet registro = comando.executeQuery("select descripcion,precio from articulos
where codigo="+tf3.getText());
```

La clase ResultSet lo podemos imaginar como una tabla con todos los datos recuperados del comando SQL select que acaba de ejecutar el MySQL. En este ejemplo puede retornar una fila o ninguna ya que estamos utilizando la cláusula where y preguntando por el campo clave código.

Para acceder al registro devuelto debemos llamar al método next(), si retorna true es que si se recuperó una fila de la tabla articulos (es decir si existe el código de artículo introducido), en caso que retorne false el método next() significa que no hay un artículo con el código que hemos introducido en el control JTextField:

```
if (registro.next()==true) {  
  
    tf1.setText(registro.getString("descripcion"));  
    tf2.setText(registro.getString("precio"));  
} else {  
    labelResultado.setText("No existe un artículo con dicho código");  
}
```


Descripción del artículo:

Precio:

Alta

resultado

Introduce código de artículo a consultar:

Consulta por código

61 - Baja y modificación de datos de una tabla de MySQL

Problema 1

Ya creamos anteriormente una base de datos llamada bd1 y en la misma creamos una tabla llamada articulos.

Procederemos a implementar en Java un programa que nos permita comunicarnos con la base de datos "bd1" y consultar, borrar y modificar filas en la tabla "articulos".

1 - Creamos desde Eclipse un proyecto llamado "basededatos02" y seguidamente con el WindowBuilder creamos una clase llamada "Formulario".

2 - Primero debemos añadir el driver que descargamos (mysql-connector-java-3.1.8-bin.jar) presionamos el botón derecho del mouse sobre nuestro proyecto y seleccionamos la opción "Build path", aparece el siguiente diálogo:

Package Explorer

7.13
7.14
Alumno
Ascensor
basededatos02
basededatos02
JRE System Library
src
basededatos02
Referencias
my
bd1
Bombilla
Cadena1
Cadena2
Cadena3
Cadena4
Cadena5
Cadena6
CajeroAutomatico
Cola
CostoCadena
Cuadrado
CuentaCorriente
E1
E10
E11
E13
E14
E2
basededatos02

New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Show in Local Terminal
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Assign Working Sets...
Coverage As
Run As
Debug As
Restore from Local History...
Team
Compare With
Configure
Properties Alt+Enter

```
datos02;  
  
t.EventQueue;  
wing.JFrame;  
wing.JPanel;  
wing.border.EmptyBorder;  
wing.JLabel;  
wing.JTextField;  
wing.JButton;  
t.event.ActionListener;  
t.event.ActionEvent;  
l.Connection;  
l.DriverManager;  
l.ResultSet;  
l.SQLException;
```

Link Source...
New Source Folder...
Use as Source Folder
Add External Archives...
Add Libraries...
Configure Build Path...

Add External Archives to Java Build Path

```
el lblIngresoCodigo;  
TextField tf3;
```

Declaration Console Terminal
[C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (19 may 2022 15:38:55)]

Seleccionamos la opción "Build Path", de la parte central seleccionamos la pestaña "ADD external Archives..." y procedemos a buscar el archivo mysql-connector-java-3.1.8-bin.jar

El código fuente completo que resuelve este problema es:

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

```
public class Formulario extends JFrame {  
  
    private JPanel contentPane;  
    private JTextField tf1;  
    private JTextField tf2;  
    private JLabel labelResultado;  
    private JButton btnConsultaPorCdigo;  
    private JTextField tf3;
```

```
/**
 * Launch the application.
 */
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                Formulario frame = new Formulario();
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}
```

```
/**
 * Create the frame.
 */
public Formulario() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 606, 405);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(null);

    JLabel lblDescripionDelArticulo = new JLabel("Descripción del artículo:");
    lblDescripionDelArticulo.setBounds(23, 38, 193, 14);
    contentPane.add(lblDescripionDelArticulo);

    tf1 = new JTextField();
    tf1.setBounds(247, 35, 193, 20);
    contentPane.add(tf1);
    tf1.setColumns(10);

    JLabel lblPrecio = new JLabel("Precio:");
    lblPrecio.setBounds(23, 74, 95, 14);
    contentPane.add(lblPrecio);
```



```
tf2 = new JTextField();  
    tf2.setBounds(247, 71, 107, 20);  
    contentPane.add(tf2);  
    tf2.setColumns(10);
```

```
labelResultado = new JLabel("resultado");  
labelResultado.setBounds(361, 122, 229, 14);  
contentPane.add(labelResultado);
```

```
btnConsultaPorCodigo = new JButton("Consulta por código");  
btnConsultaPorCodigo.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent arg0) {  
        labelResultado.setText("");  
        tf1.setText("");  
        tf2.setText("");  
        try {  
            Connection  
conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root" ,"");  
            Statement comando=comexion.createStatement();  
            ResultSet registro = comando.executeQuery("select descripcion,precio from articulos  
where codigo="+tf3.getText());
```

```
if (registro.next()==true) {  
  
    tf1.setText(registro.getString("descripcion"));  
    tf2.setText(registro.getString("precio"));  
    } else {  
        labelResultado.setText("No existe un  
artículo con dicho código");  
    }  
    conexion.close();  
    } catch(SQLException ex){  
        setTitle(ex.toString());  
    }  
    }  
});
```

```
btnConsultaPorCdigo.setBounds(25, 122, 177, 23);  
contentPane.add(btnConsultaPorCdigo);
```

```
tf3 = new JTextField();  
tf3.setBounds(247, 123, 86, 20);  
contentPane.add(tf3);  
tf3.setColumns(10);
```

```
JButton btnNewButton = new JButton("Borrar");  
btnNewButton.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent arg0) {  
        labelResultado.setText("");
```

```
try {  
    Connection  
    conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root" ,"");  
    Statement comando=conexion.createStatement();  
    int cantidad = comando.executeUpdate("delete from articulos where  
codigo="+tf3.getText());  
    if (cantidad==1) {  
        tf1.setText("");  
        tf2.setText("");  
        labelResultado.setText("Se borro el artículo con dicho código");  
    } else {  
        labelResultado.setText("No existe un artículo con dicho código");  
    }  
    conexion.close();  
} catch(SQLException ex){  
    setTitle(ex.toString());  
}  
}  
});
```

```
btnNewButton.setBounds(24, 156, 177, 23);
contentPane.add(btnNewButton);
JButton btnNewButton_1 = new JButton("Modificar");
btnNewButton_1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        labelResultado.setText("");
        try {
            Connection
conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root" ,"");
            Statement comando=comexion.createStatement();
            int cantidad = comando.executeUpdate("update articulos set descripcion=" +
tf1.getText() + "," +
                                "precio=" + tf2.getText() + " where codigo="+tf3.getText());
            if (cantidad==1) {
                labelResultado.setText("Se modifiko la descripcion y el precio del artículo con dicho
código");
            } else {
                labelResultado.setText("No existe un artículo con dicho código");
            }
            conexion.close();
        } catch(SQLException ex){
            setTitle(ex.toString());
        }
    }
});
```

```
btnNewButton_1.setBounds(21, 190, 179, 23);  
    contentPane.add(btnNewButton_1);  
    cargarDriver();  
}
```

```
private void cargarDriver() {  
    try {  
        Class.forName("com.mysql.jdbc.Driver");  
    } catch (Exception ex) {  
        setTitle(ex.toString());  
    }  
}  
}
```



Descripción del artículo:

Precio:

Consulta por código

resultado

Borrar

Modificar

El código a implementar cuando se presiona el botón "Consulta por código" es el visto en el concepto anterior:

```
btnConsultaPorCodigo = new JButton("Consulta por código");
btnConsultaPorCodigo.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        labelResultado.setText("");
        tf1.setText("");
        tf2.setText("");
        try {
            Connection conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root","");
            Statement comando=conexion.createStatement();
            ResultSet registro = comando.executeQuery("select descripcion,precio from articulos where codigo="+tf3.getText());
            if (registro.next()==true) {
                tf1.setText(registro.getString("descripcion"));
                tf2.setText(registro.getString("precio"));
            } else {
                labelResultado.setText("No existe un artículo con dicho código");
            }
            conexion.close();
        } catch(SQLException ex){
            setTitle(ex.toString());
        }
    }
});
```


Veamos el código para efectuar una baja en la tabla artículos:

```
btnNewButton.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent arg0) {  
        labelResultado.setText("");  
        try {  
            Connection conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root" ,"");  
            Statement comando=comexion.createStatement();  
            int cantidad = comando.executeUpdate("delete from articulos where codigo="+tf3.getText());  
            if (cantidad==1) {  
                tf1.setText("");  
                tf2.setText("");  
                labelResultado.setText("Se borro el artículo con dicho código");  
            } else {  
                labelResultado.setText("No existe un artículo con dicho código");  
            }  
            conexion.close();  
        } catch(SQLException ex){  
            setTitle(ex.toString());  
        }  
    }  
});
```

Después de crear un objeto de la clase Statement procedemos a llamar al método executeUpdate con un comando SQL válido (delete from articulos where codigo=código de artículo) El código de artículo lo extraemos del tercer JTextField.

El método executeUpdate retorna un entero que representa la cantidad de registros borrados de la tabla articulos. Luego en caso que retorne un uno procedemos a mostrar en un JLabel el mensaje "Se borro el artículo con dicho código", en caso contrario mostramos el mensaje "No existe un artículo con dicho código".

Para la modificación procedemos de forma muy similar al borrado:

```
btnNewButton_1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        labelResultado.setText("");
        try {
            Connection conexion=DriverManager.getConnection("jdbc:mysql://localhost/bd1","root" ,"");
            Statement comando=conexion.createStatement();
            int cantidad = comando.executeUpdate("update articulos set descripcion='" + tf1.getText() + "'," +
                                                "precio=" + tf2.getText() + " where codigo="+tf3.getText());
            if (cantidad==1) {
                labelResultado.setText("Se modifiko la descripcion y el precio del artículo con dicho código");
            } else {
                labelResultado.setText("No existe un artículo con dicho código");
            }
            conexion.close();
        } catch(SQLException ex){
            setTitle(ex.toString());
        }
    }
});
```

Al método `executeUpdate` le pasamos un comando SQL de tipo `update`. Debemos concatenar los datos fijos del comando `update` con los valores que extraemos de los `TextField`:

```
int cantidad = comando.executeUpdate("update articulos set descripcion='" + tf1.getText() + "'," +  
                                     "precio=" + tf2.getText() + " where codigo="+tf3.getText());  
.....
```

Es importante fijarse en las comillas simples después del carácter `=`, esto debido a que se trata de un campo de tipo `varchar`.

Nuevamente el método `executeUpdate` retorna la cantidad de registros modificados. En caso que retorne un `1` significa que se modificaron los datos correctamente.

Ejercicio 1: Consulta a la tabla departments de la BD employees

```
MySQLCon.java X
1 import java.sql.*;
2
3 class MysqlCon{
4     public static void main(String args[]){
5         try{
6             Class.forName("com.mysql.cj.jdbc.Driver");
7             Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/employees","root","vistaalegre");
8             Statement stmt=con.createStatement();
9             ResultSet rs=stmt.executeQuery("select * from departments order by 1 asc");
10            while(rs.next())
11                //System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
12                System.out.println(rs.getString(1)+" "+rs.getString(2));
13            con.close();
14        }catch(Exception e){
15            System.out.println(e);
16        }
17    }
18 }
```

Problems Javadoc Declaration Console X

<terminated> MysqlCon [Java Application] C:\Users\Fernando\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_18.0.1.v20220515-1614\jre\bin\javaw.exe (24 may 2022)

```
d001 Marketing
d002 Finance
d003 Human Resources
d004 Production
d005 Development
d006 Quality Management
d007 Sales
d008 Research
d009 Customer Service
```

Ejercicio 2: Mostramos todas las empleadas apellidadas Perez del departamento d007 (Ventas) haciendo una consulta con una subconsulta.

```
MySQLCon.java X
1 import java.sql.*;
2
3 class MysqlCon{
4     public static void main(String args[]){
5         try{
6             Class.forName("com.mysql.cj.jdbc.Driver");
7             Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/employees","root","vistaalegre");
8             Statement stmt=con.createStatement();
9             ResultSet rs=stmt.executeQuery("select first_name, last_name "
10                                           + "from employees "
11                                           + "where last_name = 'Perez' "
12                                           + "and gender = 'F' "
13                                           + "and emp_no in (select emp_no "
14                                                         + "from dept_emp "
15                                                         + "where dept_no = 'd007')");
16             while(rs.next()){
17                 System.out.println(rs.getString(1)+" "+rs.getString(2));
18             }
19             con.close();
20         }catch(Exception e){
21             System.out.println(e);
22         }
23     }
```

Problems Javadoc Declar

<terminated> MySqlConnection [Java Applicati

```
Fumiya Perez  
Weiyi Perez  
Masaki Perez  
Sajjad Perez  
Harngdar Perez  
Jungsoon Perez  
Shigeu Perez  
Rasikan Perez  
Patricia Perez  
Paloma Perez
```

Ejercicio 3: Mostramos todos los países del mundo (base de datos world) donde se hable alemán (haciendo una join entre dos tablas):

```
MySQLCon.java X
1 import java.sql.*;
2
3 class MysqlCon{
4     public static void main(String args[]){
5         try{
6             Class.forName("com.mysql.cj.jdbc.Driver");
7             Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/world","root","vistaalegre");
8             Statement stmt=con.createStatement();
9             ResultSet rs=stmt.executeQuery("select a.Name as Pais, a.Continent as Continente "
10                 + "from country a, countrylanguage b "
11                 + "where a.Code = b.CountryCode "
12                 + "and b.Language = 'German' "
13                 + "order by 2, 1");
14             while(rs.next())
15                 System.out.println("Pais: "+rs.getString(1)+" - Continente: "+rs.getString(2));
16             con.close();
17         }catch(Exception e){
18             System.out.println(e);
19         }
20     }
21 }
```


Problems Javadoc Declaration Console X

<terminated> MysqlCon (1) [Java Application] C:\Users\Fernando\.p2\pool\plu

Pais: Kazakstan - Continente: Asia
Pais: Austria - Continente: Europe
Pais: Belgium - Continente: Europe
Pais: Czech Republic - Continente: Europe
Pais: Denmark - Continente: Europe
Pais: Germany - Continente: Europe
Pais: Hungary - Continente: Europe
Pais: Italy - Continente: Europe
Pais: Liechtenstein - Continente: Europe
Pais: Luxembourg - Continente: Europe
Pais: Poland - Continente: Europe
Pais: Romania - Continente: Europe
Pais: Switzerland - Continente: Europe
Pais: Canada - Continente: North America
Pais: United States - Continente: North America
Pais: Namibia - Continente: Africa
Pais: Australia - Continente: Oceania
Pais: Brazil - Continente: South America
Pais: Paraguay - Continente: South America