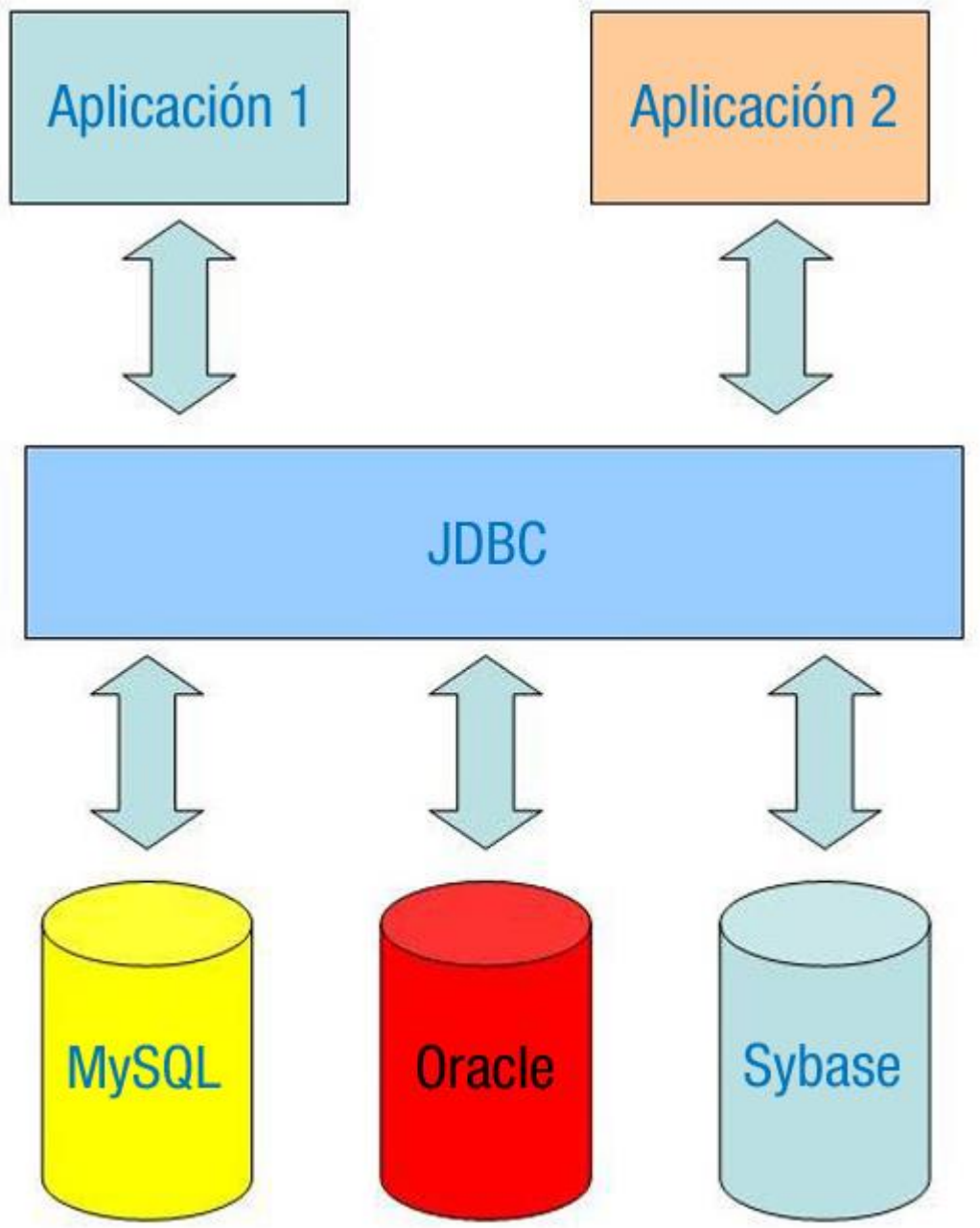
TEMA 13: Gestión de bases de datos.

CICLO GRADO SUPERIOR

DAM: Diseño Aplicaciones Multiplataforma

DAW: Diseño de Aplicaciones Web



INDICE

1. Introducción
2. Conectores
   1. Instalación en NETBEANS. (versiones 5.7 o anteriores)
   2. Instalación en ECLIPSE
3. MySQL 8.0 y NetBeans con dependencias.
4. Ejecutar consultas.

13.1- INTRODUCCIÓN

- Los datos se encuentran almacenados en un SGBD, en bases de datos relacionales.
- Sun Microsystems desarrollo una API para el **acceso a datos** denominada JDBC (Java Database Connectivity):
  - Es una API con soporte SQL
  - Consta de un conjunto de cases e interfaces escritas en JAVA.
  - Simplifica el manejo.
  - Se permite ejecutar en diferentes plataformas.
- ODBC (Open DataBase Connectivity) se utiliza como medio principal de la bases de datos en Windows.

13.2- CONECTORES O DRIVES

- Es un conjunto de clases encargadas de implementar las interfaces del API y acceder a la base de datos.
- Los conectores suelen ser ficheros *.jar
- El API JDBC viene distribuido en dos paquetes:
  - Java.sql dentro de J2SE
  - Javax.sql dentro de J2EE
- Todos los conectores deben de ser compatibles con ANSI SQL-2

Modo antiguo: añadir librería

13.3- INSTALAR CONNECTOR EN NETBEANS

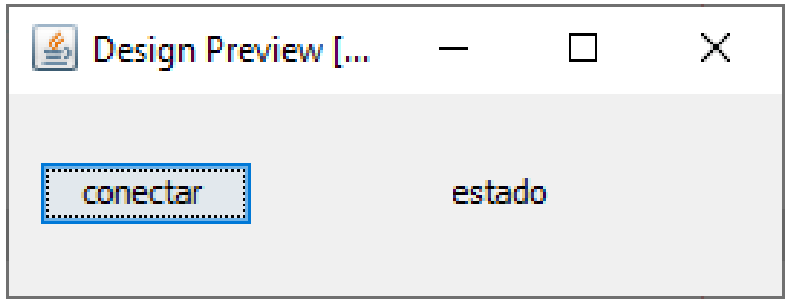
1º) Descargar conector de Java de MySQL:

- mysql-connector-java
- URL1: https://www.mysql.com/products/connector/
- URL2: archivo binario https://github.com/mysql/mysql-connector-j
- https://dev.mysql.com/downloads/connector/j/?os=26 (versión actual)

2º) Añadir Archivo *mysql-connector-java.JAR a la librería

13.3- INSTALAR CONNECTOR EN NETBEANS

3º) Añado un nuevo Package con una interfaz gráfica compuesta de un botón de **conectar** y una etiqueta para describir el **estado**:



4º) En el código fuente se añade la etiqueta de:

private static Connection con;

```
12  public class NewJFrame extends javax.swing.JFrame {
        private static Connection con;
14      /**
15       * Creates new form NewJFrame
16       */
```

import com.mysql.jdbc.Connection;

13.3- INSTALAR CONNECTOR EN NETBEANS

5º) Genero la conexión a la base de datos con el driver, user, password y URL:

```
//Datos de conexión a la Base de datos
   private static final String driver="com.mysql.jdbc.Driver";
   private static final String user="root";
   private static final String pass="root";
   private static final String url="jdbc:mysql://dir_IP:3306/prueba";
```

13.3- INSTALAR CONNECTOR EN NETBEANS

6º) Me creo una función de conectar con llamada a excepción:

```
public void conector() {
    // Reseteamos a null la conexion a la bd
    conex=null;
    try{
       Class.forName(driver);
       // Nos conectamos a la bd
       conex= (Connection) DriverManager.getConnection(url, user, pass);
       // Si la conexion fue exitosa mostramos un mensaje de conexion exitosa
       if (conex!=null){
          jLabel1.setText("Conexion establecida");
       }
    }
    // Si la conexion NO fue exitosa mostramos un mensaje de error
    catch (ClassNotFoundException | SQLException e){
       jLabel1.setText("Error de conexion" + e);
    }
 }
```

13.3- INSTALAR CONNECTOR EN NETBEANS

7º) Error de conexión con el servidor de MySQL:

com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure

The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server

➔El servidor MySQL no tiene activado el acceso remoto:

Configurar el archivo conf de MySQL (/etc/mysql/mysql.conf.d/mysqld.cnf)

```
bin-address = 127.0.0.1

#bin-address = 127.0.0.1
```

13.3- INSTALAR CONNECTOR EN NETBEANS

Opciones de CONFIGURACIÓN:
(/etc/mysql/mysql.conf.d/mysqld.cnf)

a) Poner la dirección IP del servidor

```
[mysqld]bind-address=IP_SERVIDOR_MYSQL
```

b) Poner la dirección o.o.o.o en caso de tener más de una interfaz de red en el servidor.

c) Poner un comentario

```
#bin-address = 127.0.0.1
```

Por último, se reinicia el servicio de MySQL con

```
sudo /etc/init.d/mysql restart
```

13.3- INSTALAR CONNECTOR EN NETBEANS

Opciones de CONFIGURACIÓN:
(/etc/mysql/mysql.conf.d/mysqld.cnf)

a) Poner la dirección IP del servidor

```
[mysqld]bind-address=IP_SERVIDOR_MYSQL
```

b) Poner la dirección o.o.o.o en caso de tener más de una interfaz de red en el servidor.

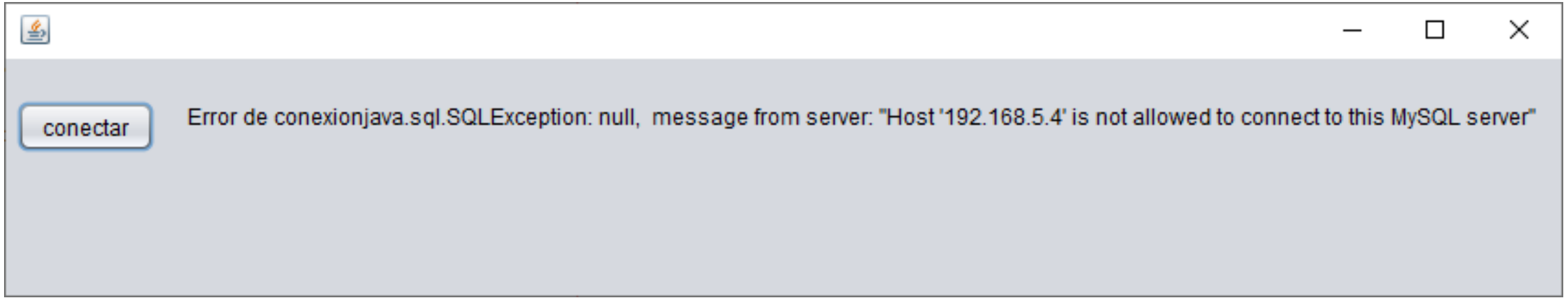
c) Poner un comentario

```
#bin-address = 127.0.0.1
```

Por último, se reinicia el servicio de MySQL con

```
sudo /etc/init.d/mysql restart
```

13.3- INSTALAR CONNECTOR EN NETBEANS



Base de datos MySQL 8.0 con Java NetBEANS 12.0:

a) Proyecto de tipo Maven
b) Se añaden dependencias.

13.3- ¿Qué ES MAVEN?

Herramienta de software para la construcción de proyectos en JAVA con una construcción más simple basada en formato XML.

- Es un proyecto superior de Apache Software Foundation.
- Ventajas de MAVEN:
  - Esta listo para usarse en red.
  - Descarga plugins automáticamente de un repositorio.
  - Puede tener acceso a múltiples versiones de Open Source Java y otros desarrolladores.

13.3- ¿Qué ES MAVEN?

- Utiliza un Project Object Model (POM) para describir el proyecto de software a construir.
- Se integra en diferentes entornos de desarrollo:
  - ECLIPSE
  - NETBEANS
  - INTELLIJ IDEA
  - Jdeveloper 11G (11.1.1.3)

Mas info:wikipedia

13.3- ¿Qué es un fichero JAR?

Se utiliza en entorno de JAVA y puede ser 2 términos:

1º) Una aplicación de Java que puede ser ejecutada.

2º) Una biblioteca de archivos de Java con metadatos que se envían de manera sintetizada y comprimida.

JAR = Java Archive

13.3- Base de datos MySQL 8.0

Pasos Previos:

a) Obtener la IP del servidor y Poner la dirección IP del servidor

```
[mysqld]bind-address=IP_SERVIDOR_MYSQL
```

b) Crear un usuario MySQL con el nuevo método y aplicarle los permisos GRANT.

```
CREATE USER 'nombre'@'dirIP' IDENTIFIED BY
'password';

GRANT ALL PRIVILEGES ON bd_nombre.* TO
'nombre'@'dirIP' WITH GRANT OPTION;

FLUSH PRIVILEGES;
```

13.3- Entorno de desarrollo con NeatBeans

Pasos Previos:

a) Me creo un nuevo proyecto con MAVEN, con la correspondiente clase Main:

```
Project: JDBCejemplo01

Packed:com.JDBCejemplo01

New> Java Main Class> named: Main
```

13.3- Entorno de desarrollo con NeatBeans

Pasos Previos:

b) Añado las dependencias desde el entorno de GitHUB.

https://github.com/mysql/mysql-connector-j

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.23</version>
</dependency>
```

13.3- Entorno de desarrollo con NeatBeans

b) Se añaden las dependencias en el fichero XML del proyecto.

En Neatbeans → Proyject files→pom.xml se añaden las dependencias:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.c
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany</groupId>
    <artifactId>ejemplo13a</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>13</maven.compiler.source>
        <maven.compiler.target>13</maven.compiler.target>
    </properties>
    <dependencies>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.23</version>
        </dependency>
    </dependencies>
</project>
```

13.3- Entorno de desarrollo con NeatBeans

c) Actualizo el proyecto con el nuevo conector.

Proyecto > Botón derecho Clean AND BUILD

```
--- maven-install-plugin:2.4:install (default-install) @ ejemplo13a ---
Installing D:\Programacion\MySQL\ejemplo13a\target\ejemplo13a-1.0-SNAPSHOT.jar to C:\U
Installing D:\Programacion\MySQL\ejemplo13a\pom.xml to C:\Users\JTT\.m2\repository\cor
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  2.655 s
Finished at: 2021-04-03T15:43:27+02:00
------------------------------------------------------------------------
```

13.3- Entorno de desarrollo con NeatBeans

d) Creo la conexión registrando las siguientes variables:

driver: conexión JDBC

User: usuario con el que se va a conectar

Password: password del usuario

url: enlace con el servidor de MySQL

*nuevas conexiones a partir de la versión 8 de MySQL:

- **serverTimezone**
- **useSSL**
- **allowPublicKeyRetrieval**

13.3- Entorno de desarrollo con NeatBeans

d) Creo la conexión registrando las siguientes variables:

```
private static final String driver="com.mysql.jdbc.Driver";
private static final String user="juan";
private static final String pass="juan";
private static final String
url="jdbc:mysql://192.168.5.33:3306/prueba?allowPublicKeyRetrieval=true&useSSL=false&serverTimezone=UTC";
```

13.3- Entorno de desarrollo con NeatBeans

e) Me creo una nueva conexión que la paso a un método de conectarse:

```
private static Connection conexion;
```

f) Método de conectarse:

```
 public void conector() {
        // Reseteamos a null la conexion a la bd
      conexion=null;
      Class.forName(driver);
      conexion= (Connection) DriverManager.getConnection(url, user, pass);
}
```

13.3- Entorno de desarrollo con NeatBeans

g) Aplica las excepciones correspondientes que pueden surgir:

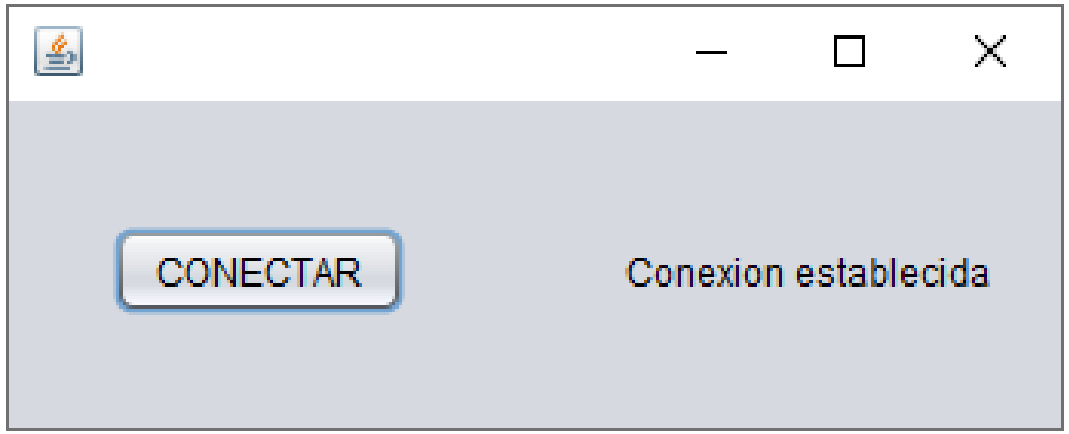
```
catch (ClassNotFoundException | SQLException e)
```

Resultado:

```
try{
      Class.forName(driver);
      conexion= (Connection) DriverManager.getConnection(url, user, pass);
}
catch (ClassNotFoundException | SQLException e){
            labelEstado.setText("Error de conexion" + e);
        }
```

13.3- Entorno de desarrollo con NeatBeans

RESULTADO FINAL



GESTION DE BASES DE DATOS:
Lenguaje DML y DDL
a través de JAVA

PASOS A REALIZAR (resumen)

1. DriverManager. Realiza la conexión con la BD.

Invoca al método .**getConnection** que devuelve un objeto **Connection**.

2. Connection. Construye el **Statement** para enviar sentencias SQL.

Invoca al método **createStatement** que devuelve el objeto **Statement**.

3. Statement. Objeto que permite ejecutar una sentencia SQL sobre la BD. Devuelve un objeto ResultSet.

4. ResultSet. Tabla de datos que representa al conjunto de resultados al ejecutar la sentencia SQL.

Métodos: **first**, **next**, **previous**, **last**, **getInt**, **getDouble**,**getString**, etc…

Métodos a implementear

```
private void conectar() throws java.SQLException{
…}
private void crearSentencia() throws java.SQLException {
…}
private void cerrarConexion() throws java.SQLException {
…}
Public java.sql.ResultSet buscarFilasTelefonos(String
subcad,int tipoBuscqueda,…) throws java.sql.SQLException {
…}
```

13.4- Ejecución de consultas.

Definición de un STATEMENT.

```
Statement <nombre> = conexion.createStatement();
```

Creación de una consulta

```
ResultSet rs = nombre.executeQuery("SELECT dni, edad,FROM PERSONA");
```

Procesar resultados.

```
while (rs.next()){
        String dni= rs.getString("dni");
        int edad =rs.getInt("edad");
}
```

Los muestro

```
Conexion establecida
DNI persona:00000000Anombre:Juan
DNI persona:11111111Bnombre:Sara
DNI persona:22222222Cnombre:Hugo
DNI persona:33333333Dnombre:Pepe
DNI persona:44444444Fnombre:Cristina
DNI persona:55555555Fnombre:Natalia
```

CERRAR CONEXIÓNES(si están abiertas):

```
rs.close(); nombre.close(); conexion.close();
```

13.4- Ejecución de consultas.

Definición de un STATEMENT.

```
Statement <nombre> = conexion.createStatement();
```

Creación de una consulta de actualización.

```
nombre.executeUpdate("UPDATE CLIENTE SET
teléfono='900111222´ WHERE idCLIENTE=3");
```

CERRAR CONEXIÓN:

```
if (conexión!=null) conexion.close();
```

13.4- Ejecución de consultas.

Definición de un STATEMENT.

```
Statement <nombre> = conexion.createStatement();
```

Creación de una consulta de inserción de datos.

```
nombre.executeUpdate("INSERT INTO CLIENTE" + "(idCliente, NIF, NOMBRE, TELEFONO)" + "VALUES(4,'99999999T','Casa PEPE','900123456') ");
```

CERRAR CONEXIÓN:

```
if (conexión!=null) conexion.close();
```

13.4- Ejecución de consultas.

Definición de un STATEMENT.

```
Statement <nombre> = conexion.createStatement();
```

Creación de una consulta de BORRADO

```
ResultSet registro = nombre.executeUpdate("DELETE FROM CLIENTE WHERE NIF='99999999T'");
```

Procesar resultados.

```
System.out.println("\nSe borró " + registro + " registro\n);
```

CERRAR CONEXIÓN:

```
if (conexión!=null) conexion.close();
```

13.5- EXCEPCIONES.

Objeto SQLException.

Presenta dos métodos del error producido y del mensaje informativo.

- getErrorCode( ) → Devuelve el número entero que representa el error.
- getMessage( ) → Imprime el mensaje informativo asociado

PERSISTENCIA DE OBJETOS

13.6- PERSISTENCIA DE OBJETOS.

- Un objeto se dice persistente cuando es almacenado en un archivo u otro medio permanente. Un programa puede grabar objetos persistentes y luego recuperarlos en un tiempo posterior.

- La persistencia de datos posibilita en forma inadvertida la exhibición de información sensible si el medio de almacenaje es dejado en un ambiente sobre el que no se tiene control

BIBLIOGRAFIA

- _(lenguaje_de_programacion)#Historia

WIKI BOOKS:

- https://es.wikibooks.org/wiki/Programacion_en_Java/Caracteristicas_del_lenguaje