

Acceso a Datos

Curso 2023 - 2024

1. Acceso a Datos

Acceso a Datos:

- Base de Datos MySQL
- Driver JDBC
- Conexión JDBC
- Sentencias SQL
 - ♦ Inserción
 - ♦ Modificación
 - ♦ Eliminación
 - ♦ Consulta

2. Base de Datos MySQL

Base de Datos

- Es el repositorio donde están almacenados los datos.
- Instalar el motor de MySQL

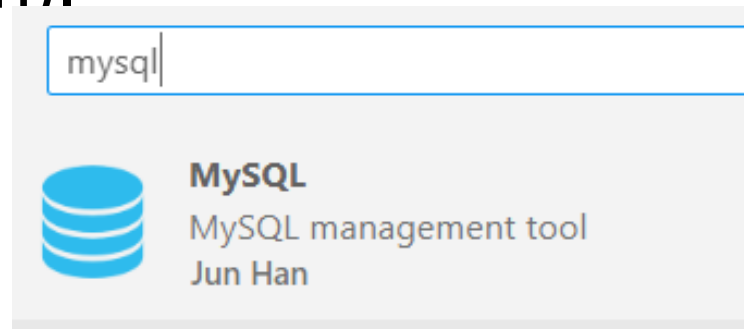
```
$ sudo apt install mysql-server  
$ sudo mysql  
> alter user root@localhost identified with mysql_native_password by 'root';  
> flush privileges;  
> exit;
```

Probamos que podemos acceder con el password

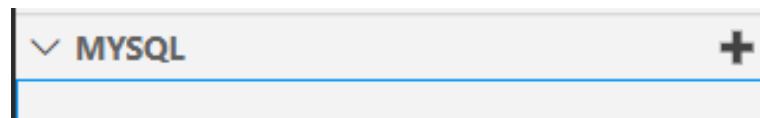
```
$ mysql -u root -p
```

2. Base de Datos MySQL

- Podemos crear una conexión en VisualStudio Code para base de datos y comprobar resultados.
- Instalamos la extensión MySQL para Visual Studio Code (de Jun Han).

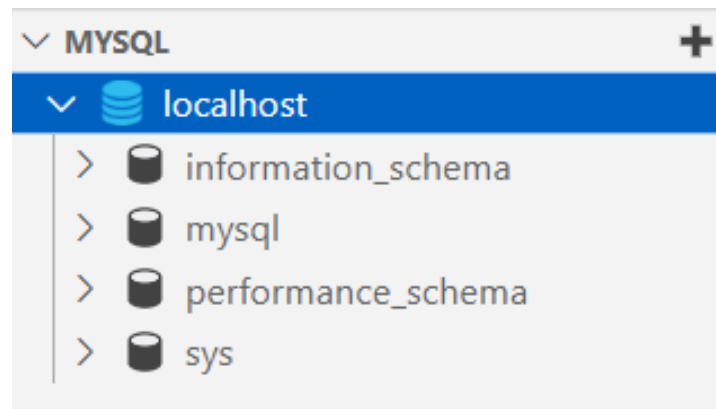


- Crearemos una conexión con MySQL:



2. Base de Datos MySQL

- Añadimos el **acceso**:
 - Ponemos el host (localhost)
 - usuario (root)
 - password (root)
 - puerto 3306
 - Referencia a SSL en blanco



2. Base de Datos MySQL

Pasos para que una aplicación se comuniqué con una BD:

- 1) **Cargar el driver** necesario para enlazar con una BD concreta.
- 2) Establecer una **conexión** con la base de datos.
- 3) Enviar **sentencias SQL**.
- 4) **Procesar el resultado** obtenido
- 5) **Liberar los recursos** al terminar
- 6) **Manejar los errores** que se puedan producir

3. Driver JDBC

JDBC (Java Database Connectivity)

- Es la API para acceder a BD desde Java
- El **driver** o **conector** es un fichero JAR que se añade a la aplicación como una librería (sin instalación adicional).
- La mayoría de las bases de datos incorporan un driver JDBC.
- Para MySQL: `mysql-connector-java-8.0.32.jar`
- Agregar el connector a una carpeta **lib** en nuestro proyecto.
- JDBC proporciona el paquete **java.sql** con el que podremos realizar las operaciones de gestión de la BD.

3. Driver JDBC

Clases para operaciones con las BD:

clase	Descripción
DriverManager	Carga un driver
Connection	Establece las conexiones con BBDD
Statement	Para ejecutar sentencias SQL y enviarlas
PreparedStatement	La ruta de ejecución está predeterminada en el servidor de base de datos que le permite ser ejecutado varias veces
CallableStatement	Ejecutar sentencias SQL de Procedimientos Almacenados.
ResultSet	Almacena el resultado de la consulta

4. Conexión JDBC

La conexión se establece mediante la clase **DriverManager**:

```
DriverManager.getConnection(String url, String user, String password)
```

Ejemplo:

```
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/contactos", "root", "root");
```

Establecida la conexión, crearemos un objeto **Statement** para poder enviar sentencias SQL a la BD

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

El resultado se recoger en un ResultSet

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

4. Conexión JDBC. Ejemplo

Ejemplo de conexión con BD y listado de una tabla:

```
try{
    //Conexión con la BD
    Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/contactos",
    "root", "root");

    // Creamos un objeto para enviar sentencias SQL a la BD
    Statement st = con.createStatement();
    // Ejecutamos la consulta SQL y obtenemos el resultado en ResultSet
    ResultSet rs = st.executeQuery("SELECT * FROM contacto");
    // Recorremos los resultados obtenidos y mostramos sus campos
    while (rs.next()) {
        String nombre = rs.getString("nombre");
        int edad = rs.getInt("edad");
        System.out.println(nombre + ": " + edad);
    }
    // Cerramos la conexión
    con.close();
} catch (SQLException e) {
    System.out.println("Error en la bd: " + e.getErrorCode() + "-" + e.getMessage());
}
```

Salida en el terminal:

```
Beatriz García Martín: 32
Sandra Flores Jorge: 17
Carlos López Carvajal: 24
Vanessa Rodríguez Recio: 37
Ismael Pazos Rincón: 53
Esther Zamora Castillo: 12
```

5. Sentencias SQL

La clase Statement permite ejecutar sentencias SQL en la BD. Dos tipos de sentencias SQL:

- **ExecuteQuery(strSQL):** Consultas que devuelven un conjunto de datos, normalmente SELECT. El resultado es un ResultSet con todos los datos obtenidos de la consulta.

```
ResultSet rs = st.executeQuery("SELECT * FROM contacto");
```
- **ExecuteUpdate(strSQL):** Consultas que se ejecutan pero no devuelven datos: sentencias DML (INSERT, UPDATE y DELETE) y sentencias DDL (CREATE, ALTER, DROP, etc). Devuelve la cantidad de filas afectadas.

```
numfilas=st.executeUpdate("INSERT INTO contactos.contacto (DNI, NOMBRE, EDAD) VALUES ('07834658F', 'Alberto Velez Rodrigo', 51)");
```

5. Sentencias SQL

Conexión con BD contactos y listado de la tabla contactos:

```
try {
    // Establecemos la conexión
    Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/contactos",
"root", "root"           // Creamos un objeto para enviar sentencias SQL a la BD
    Statement st = con.createStatement();
    // Ejecutamos la consulta SQL y obtenemos el resultado en ResultSet
    ResultSet rs = st.executeQuery("SELECT * FROM contacto");
    // Recorremos los resultados obtenidos y mostramos sus campos
    while (rs.next()) {
        String nombre = rs.getString("nombre");
        int edad = rs.getInt("edad");
        System.out.println(nombre + ": " + edad);
    }
    //insertamos un nuevo contacto
    numfilas=st.executeUpdate("INSERT INTO contactos.contacto (DNI, NOMBRE, EDAD) VALUES
('07834658F', 'Alberto Velez Rodrigo', 51)");
    System.out.println("Se han insertado un total de "+numfilas+" filas en la tabla");
    // Cerramos la conexión
    con.close();
} catch (SQLException e) {
    System.out.println("Error en la bd: " + e.getErrorCode() + "-" + e.getMessage());
}
```

5. Sentencias SQL

Consulta parametrizada en una BD

```
String sentenciaSql = "SELECT nombre, precio FROM productos WHERE precio = ?";
PreparedStatement sentencia = null;
ResultSet resultado = null;

try {
    sentencia = conexion.prepareStatement(sentenciaSql);
    sentencia.setFloat(1, filtroPrecio);
    resultado = sentencia.executeQuery();
    while (resultado.next()) {
        System.out.println("nombre: " + resultado.getString(1));
        System.out.println("precio: " + resultado.getFloat(2));
    }
} catch (SQLException sqle) {
    sqle.printStackTrace();
} finally {
    if (sentencia != null)
        try {
            sentencia.close();
            resultado.close();
        } catch (SQLException sqle) {
            sqle.printStackTrace();
        }
}
```

5. Sentencias SQL

Insertar datos en una BD

```
String sentenciaSql = "INSERT INTO productos (nombre, precio) VALUES (?, ?)";  
PreparedStatement sentencia = null;
```

```
try {  
    sentencia = conexion.prepareStatement(sentenciaSql);  
    sentencia.setString(1, nombreProducto);  
    sentencia.setFloat(2, precioProducto);  
    sentencia.executeUpdate();  
} catch (SQLException sqle) {  
    sqle.printStackTrace();  
} finally {  
    if (sentencia != null)  
        try {  
            sentencia.close();  
        } catch (SQLException sqle) {  
            sqle.printStackTrace();  
        }  
}
```

5. Sentencias SQL

Actualizar datos en una BD

```
String sentenciaSql = "UPDATE productos SET nombre = ?, precio = ? WHERE nombre = ?";
PreparedStatement sentencia = null;

try {
    sentencia = conexion.prepareStatement(sentenciaSql);
    sentencia.setString(1, nuevoNombreProducto);
    sentencia.setFloat(2, precioProducto);
    sentencia.setString(3, nombreProducto);
    sentencia.executeUpdate();
} catch (SQLException sqle) {
    sqle.printStackTrace();
} finally {
    if (sentencia != null)
        try {
            sentencia.close();
        } catch (SQLException sqle) {
            sqle.printStackTrace();
        }
}
```

5. Sentencias SQL

Borrado de datos en una BD

```
String sentenciaSql = "DELETE productos WHERE nombre = ?";
PreparedStatement sentencia = null;

try {
    sentencia = conexion.prepareStatement(sentenciaSql);
    sentencia.setString(1, nombreProducto);
    sentencia.executeUpdate();
} catch (SQLException sqle) {
    sqle.printStackTrace();
} finally {
    if (sentencia != null)
        try {
            sentencia.close();
        } catch (SQLException sqle) {
            sqle.printStackTrace();
        }
}
```