	<p align="center">UNIVERSIDAD DON BOSCO ESCUELA DE COMPUTACIÓN</p>
<p align="center">CICLO I</p>	<p align="center">GUIA DE LABORATORIO #4 Programación Orientada a Objetos JDBC</p>

I.OBJETIVOS

- Que el alumno sea capaz de conocer las clases específicas que posee Java para el manejo de base de datos y el manejo de las API de JDBC.
- Que al alumno sea capaz de aprender el uso de las API JDBC para la creación de consultas.

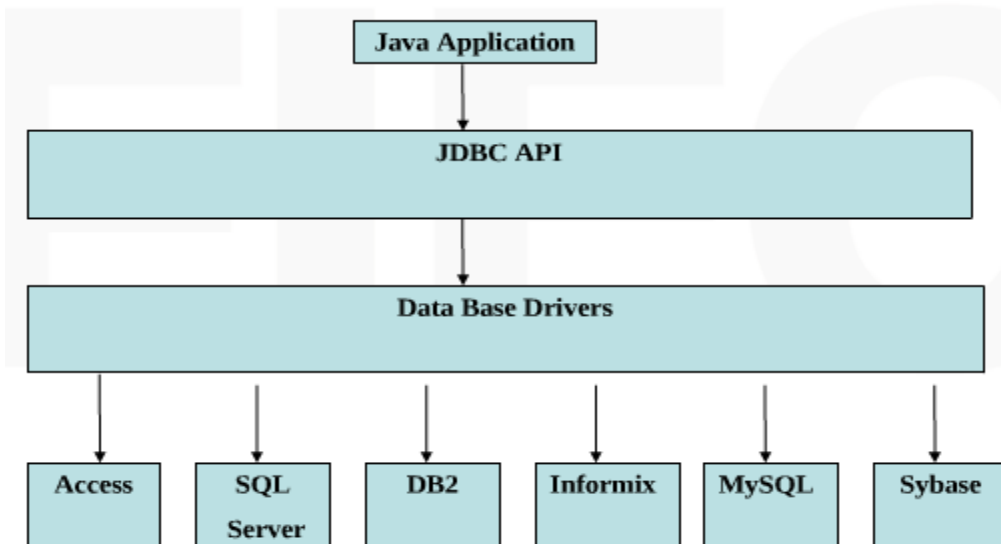
II. INTRODUCCIÓN

Conexión a Bases de Datos con IntelliJ IDEA IDE 2020 o superior

JDBC (Java DataBase Connectivity) es la tecnología Java que permite a las aplicaciones interactuar directamente con motores de base de datos relacionales.

La API JDBC es una parte integral de la plataforma Java, por lo tanto no es necesario descargar ningún paquete adicional para usarla. JDBC es una interface única que independiza a las aplicaciones del motor de la base de datos.

Un driver JDBC es usado por la JVM para introducir las invocaciones JDBC en invocaciones que la base de datos entiende.



Conectar a la base de datos

Para que una aplicación en Java se comunice con una base de datos usando la API JDBC, se requiere de un conector que realice la conexión de la aplicación a la Base de Datos.

Ese conector es específico para el manejador de base de datos y viene en la forma de un archivo “.jar” o “.zip”.

Por ejemplo:

MySQL, está en el archivo: mysql-connector-java-8.0.27.zip (mysql-connector-java-8.0.27)

SQL Server, está en el archivo: sqljdbc_1.2.2828.100_esn.zip (sqljdbc.jar)

Nota: es necesario aclarar que para las versiones más recientes de IntelliJ IDEA se cuenta con la versión actualizada del mysql connector, la cual ya no cuenta con el directorio javax, si no con el directorio jakarta.

IntelliJ Idea nos permite realizar algunas tareas relacionadas con bases de datos:

1. Conectar una aplicación a una base de datos.
2. Conectar a IntelliJ IDEA directamente a una base de datos para crear, eliminar, modificar tablas, agregar, eliminar, modificar registros y hacer consultas de datos.
3. Generar aplicaciones a partir de ingeniería inversa.
4. Crear persistencia con la Base de Datos.
5. Otras innumerables.

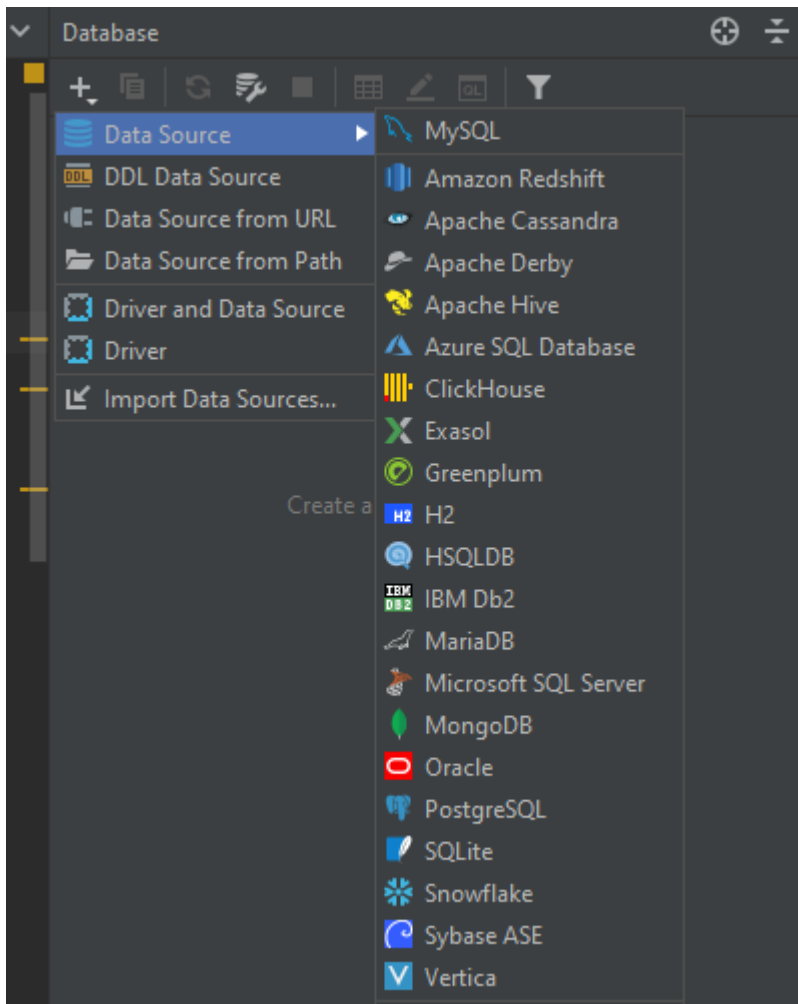
Conexión de una aplicación a una base de datos

Para conectar a una aplicación a una base de datos, se requiere:

1. Agregar a IntelliJ IDEA el conector como una biblioteca o librería. Esto permite que el conector esté disponible para los proyectos.
2. Agregar a un proyecto el conector. Esto permite que la aplicación se pueda conectar a la base de datos.

Agregar a IntelliJ IDEA el conector para utilizar una base de datos en MySQL.

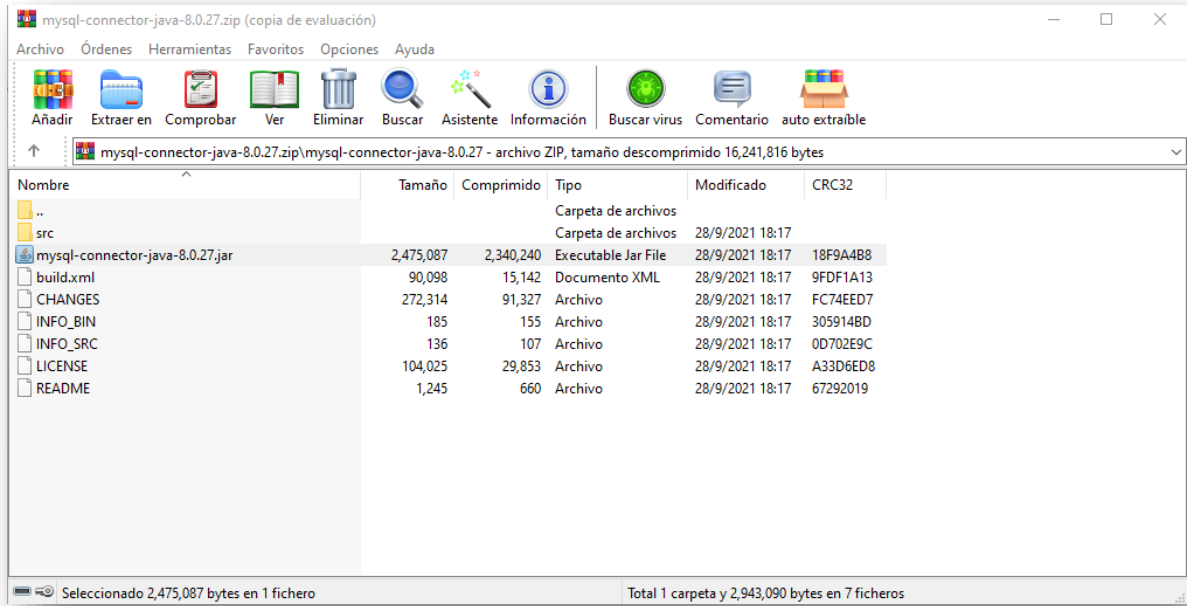
IntelliJ IDEA, cuenta ya con las librerías para la mayoría de gestores de bases de datos, o al menos, los más importantes. Con esto usted tiene una base muy completa y servicios de ayuda con las dependencias necesarias para poder conectarse desde su aplicación a la base de datos que tenga o desee crear.



¿Cómo obtener el mysql Connector más actualizado?

Para este desarrollo práctico se debe contar con un recurso indispensable para poder conectarse a su proveedor de bases de datos; particularmente centraremos nuestra atención en el driver de mysql para Java, el cual es una dependencia del lenguaje muy necesaria y quien será la que permita establecer una conexión exitosa.

Para obtener este recurso puede hacerlo entrando al siguiente link <https://dev.mysql.com/downloads/connector/j/>; una vez dentro se debe seleccionar en el cuadro "Select Operating System:", la opción "platform independent", con lo cual tendrá la opción de obtener archivos comprimidos con la dependencia. Se le recomienda descargar el archivo.zip, el cual en su interior tiene el mysql connector más actualizado como se muestra en la siguiente figura.



Para esta guía se le debe proporcionar dicho recurso adjunto, por lo cual es necesario tener en cuenta que la versión más actualizada por el momento, es la que tiene por nombre **mysql-connector-java-8.0.27.jar**.

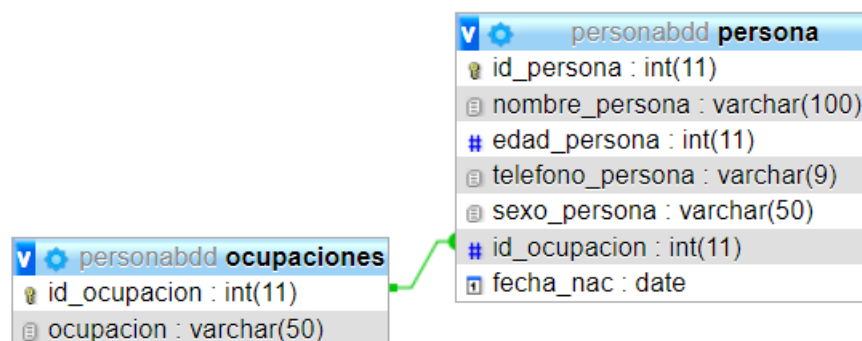
III. PROCEDIMIENTO

En esta guía trabajaremos la conexión a **MySQL** desde una aplicación de Java creada en IntelliJ IDEA 2020.

Ejercicio 1. Crear un proyecto en IntelliJ IDEA con el nombre **Guia4**.

¿Lógica del negocio?

Comenzaremos entendiendo un poco de la lógica estructural de la base de datos que se le presentará en el siguiente diagrama relacional.



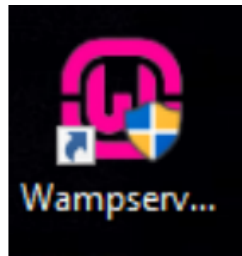
Como se puede ver, dicha estructura cuenta con 2 tablas relacionadas, la cual es de uno a muchos y de la que podemos interpretar lo siguiente: Muchas personas tienen una ocupación; obviamente en la realidad esta afirmación no es 100% real, pero para fines prácticos trataremos de utilizar esa lógica. Es decir entonces que en este caso **Emerson Torres** es un profesor, **Alejandro Pineda** es un doctor y **Fernando Calderón** es un emprendedor.

Con la lógica de negocio clara, podemos dirigirnos a nuestro cliente de base de datos WampServer o Xampp.

¿Tiene instalado en su Equipo WampServer?

Realizar los siguientes pasos:

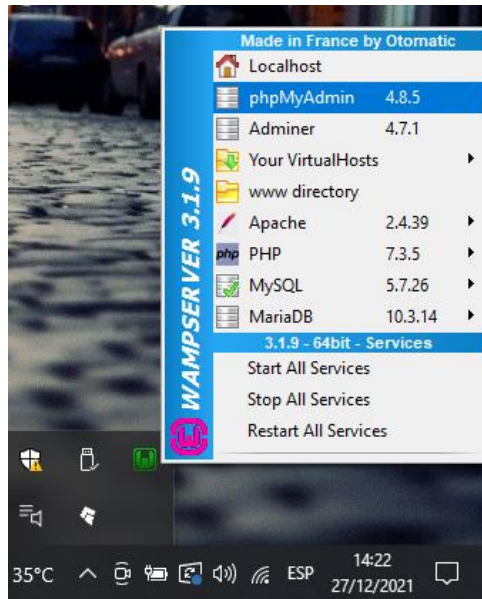
1. Ejecutar como administrador, mediante el icono creado en el escritorio.



2. Ir a la barra de tareas y esperar a que el icono de wampserver se torne de color verde.



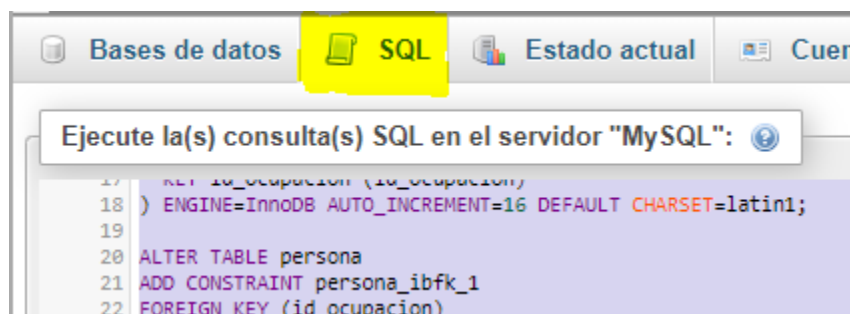
3. Ingresar al cliente de “phpMyAdmin” mediante la siguiente ruta (<http://localhost/phpmyadmin/>) o como se muestra en la imagen.



4. Ingresar mediante el usuario “root” el cual por defecto no cuenta con una contraseña definida, como se presenta en la siguiente imagen.



5. Nos dirigiremos a la pestaña que dice **SQL** para poder correr el siguiente script.



```
CREATE DATABASE personabdd;  
USE personabdd;
```

```
CREATE TABLE ocupaciones(  
id_ocupacion int(11) NOT NULL AUTO_INCREMENT,  
ocupacion varchar(50) NOT NULL,  
PRIMARY KEY (`id_ocupacion`)  
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT  
CHARSET=latin1;
```

```
CREATE TABLE persona (  
id_persona int(11) NOT NULL AUTO_INCREMENT,  
nombre_persona varchar(100) NOT NULL,  
edad_persona int(11) NOT NULL,  
telefono_persona varchar(9) NOT NULL,  
sexo_persona varchar(50) NOT NULL,  
id_ocupacion int(11) NOT NULL,  
fecha_nac date NOT NULL,  
PRIMARY KEY (`id_persona`),  
KEY id_ocupacion (id_ocupacion)  
) ENGINE=InnoDB AUTO_INCREMENT=16 DEFAULT  
CHARSET=latin1;
```

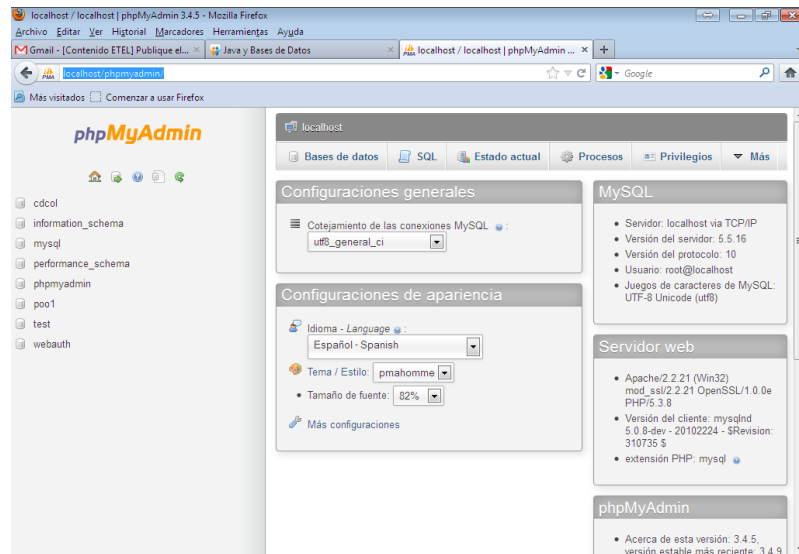
```
ALTER TABLE persona  
ADD CONSTRAINT persona_ibfk_1  
FOREIGN KEY (id_ocupacion)  
REFERENCES ocupaciones (id_ocupacion);
```

```
INSERT INTO `ocupaciones` (`id_ocupacion`, `ocupacion`)  
VALUES(1, 'Doctor'), (2, 'Emprendedor'), (3, 'Profesor');
```

```
INSERT INTO persona (id_persona, nombre_persona, edad_persona,
sexo_persona, id_ocupacion, fecha_nac) VALUES
(2, 'Alejandro Pineda', 45, '7722-4455' , 'Masculino', 1, '1999-01-05'),
(12, 'Fernando Calderón', 21, '7667-7890' , 'Masculino', 2, '2001-05-
07'),
(15, 'Emerson Torres', 22, '7123-9800' , 'Masculino', 3, '1999-08-03');
```

¿Tiene otro MySQL instalado en su equipo en otro servicio como XAMP?

XAMP es un aplicativo que incluye otros servicios como Apache,MySQL y PHP. La administración de MySQL puede realizarse desde una interfase de aplicación PHP incluida (<http://localhost/phpmyadmin/>)



Luego continuar desde el paso 4 de este ejercicio, debe examinar las opciones de la aplicación para lograr crear la base de datos y las tablas.

¿Se conectará a un servidor MySQL instalado en otro equipo?

Debe conocer la IP y el puerto del servidor MySQL. Este será el caso más habitual en un entorno de trabajo real. Consulte a su instructor para más detalles.

Luego continuar desde el paso 4 de este ejercicio.

Ejercicio 2. Conexión de IntelliJ IDEA IDE a una base de datos.

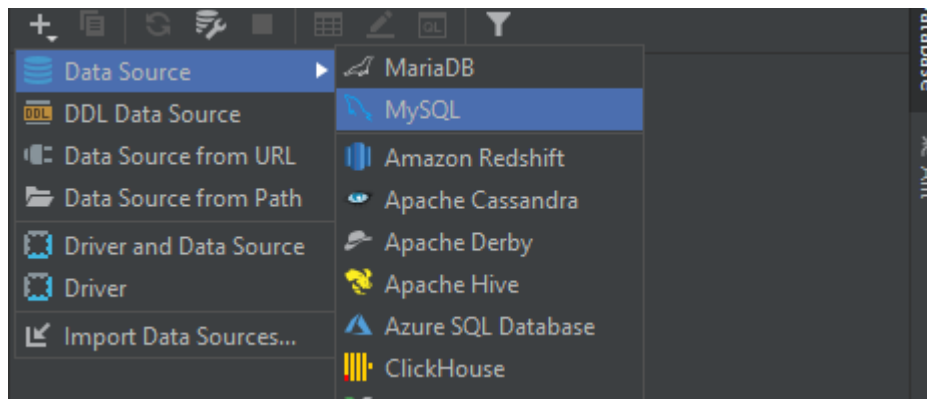
Para conectar una aplicación a una base de datos, se requiere:

- Instalar en IntelliJ IDEA IDE el conector de la base de datos **SOLO SI NO LO TIENE INSTALADO**
- Establecer la conexión entre IntelliJ IDEA IDE y la base de datos.

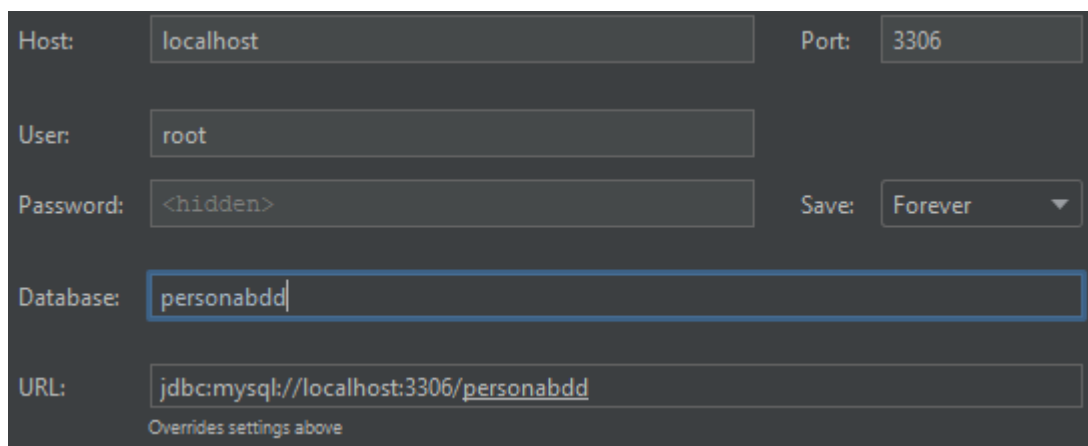
Instalación en IntelliJ IDEA IDE del conector de la Base de Datos (MySQL).

El procedimiento a seguir es el siguiente:

1. Ir a la pestaña **Database** que se encuentra al costado derecho de su pantalla > seleccionar la pestaña más (+) > luego Data Source > MySQL.



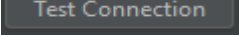
2. Ver el siguiente cuadro en el cual ingresara los datos necesarios para una correcta conexión a su base de datos MySQL (Notese que la url se crea automáticamente).

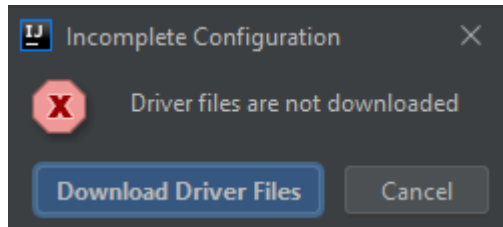
A screenshot of the MySQL connection configuration dialog. The fields are filled with the following values: Host: localhost, Port: 3306, User: root, Password: <hidden>, Save: Forever, Database: personabdd, and URL: jdbc:mysql://localhost:3306/personabdd. The URL field is highlighted with a blue border. At the bottom, it says 'Overrides settings above'.

Para este caso en particular, las credenciales por defecto son:

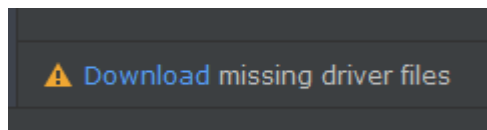
- Usuario: root.
- Password: (no hay un password definido).

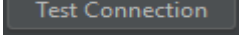
- Nombre de la bdd: personabdd.

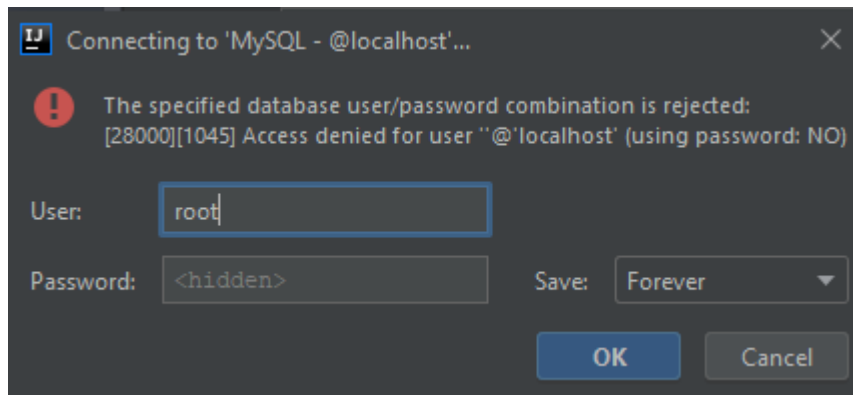
3. Si da click en  se dará cuenta que lanza el error marcado en la siguiente figura. Lo cual indica que hace falta el driver de MySQL el cual trae todas las dependencias necesarias para conectarse.



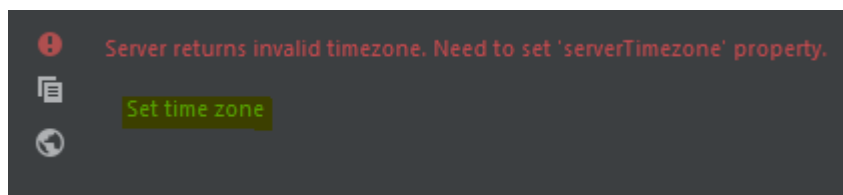
4. Para mayor facilidad se puede dar click al enlace que se muestra al pie del formulario, para descargar de forma automática la última actualización estable en el mercado para ese driver.



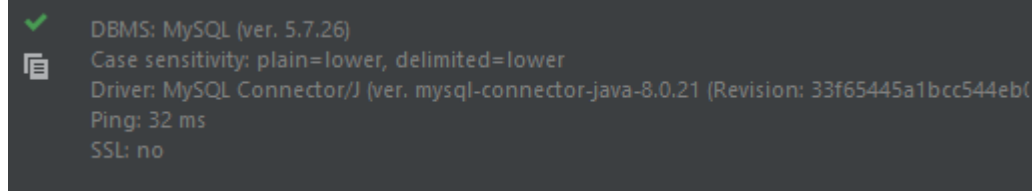
5. Si lo anterior ha funcionado correctamente y da click en , aparecerá el siguiente formulario para confirmar sus credenciales como súper usuario de la bdd.



6. Si al final de su formulario aparece el siguiente error, podrá solventarlo dando click en “set time zone”.

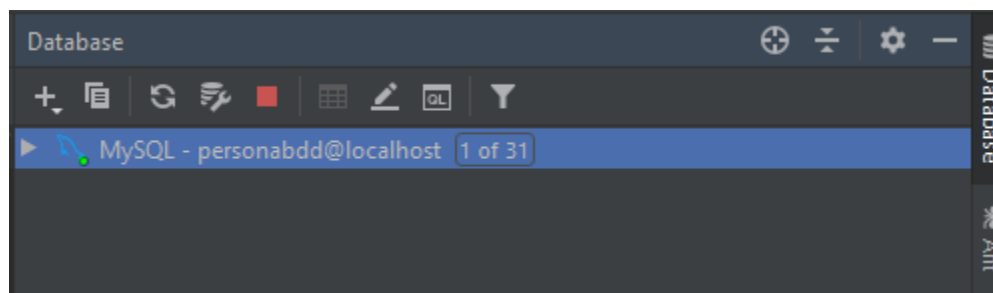


7. Vuelva a verificar su conexión. Deberá notar el siguiente mensaje el cual muestra una conexión exitosa.




Acceso a la Base de Datos desde IntelliJ IDEA IDE.

Ahora podemos ver que la conexión con la base de datos **personabdd** ha sido exitosa. La idea de esta herramienta proporcionada por el IDE, es brindar un cliente para la gestión, administración y manipulación para el servicio de MySQL, sin necesidad de salir de nuestro entorno de desarrollo. Básicamente funciona como el cliente **phpMyAdmin** que pueda venga incluido con los servicios de MySQL o no, de cualquier forma el IDE se adapta a nuestras necesidades, brindándonos su propio cliente para gestión de Bases de datos.



Para ejecutar cualquier comando de Lenguaje de **Definición de datos (DDL)** o **Lenguaje de manipulación de datos (DML)**, se cuenta con una consola; para este caso en particular ejecutaremos algunos comandos para obtener información de la tabla persona y ocupación.

1. **SELECT * FROM persona**

Con este comando se busca obtener toda la data que se encuentra dentro de la tabla persona, para lo cual nos dirigiremos a la consola proporcionada por el IDE para ejecutar dicho comando y posteriormente daremos click en el botón  para ejecutarlo y ver el resultado.

The screenshot shows a database console interface. The top panel displays the SQL query `select * from persona`. The bottom panel shows the output of the query as a table with 7 columns: `id_persona`, `nombre_persona`, `edad_persona`, `telefono_persona`, `sexo_persona`, `id_ocupacion`, and `fecha_nac`. The results are as follows:

id_persona	nombre_persona	edad_persona	telefono_persona	sexo_persona	id_ocupacion	fecha_nac
2	Alejandro Pineda	45	7722-4455	Hombre	1	1999-01-05
12	Fernando Calderón	21	7667-7890	Hombre	2	2001-05-07
15	Emerson Torres	22	7123-9800	Hombre	3	1999-08-03

Podemos observar que obtenemos todos los registros y las columnas de la tabla persona, lo más relevante que podemos notar es que en el campo `id_ocupacion` obtenemos el valor de la relación y no el significado real.

2. **SELECT * FROM ocupaciones**

Realizaremos entonces los mismos pasos del punto anterior, pero ahora buscamos obtener la data de la tabla ocupaciones.

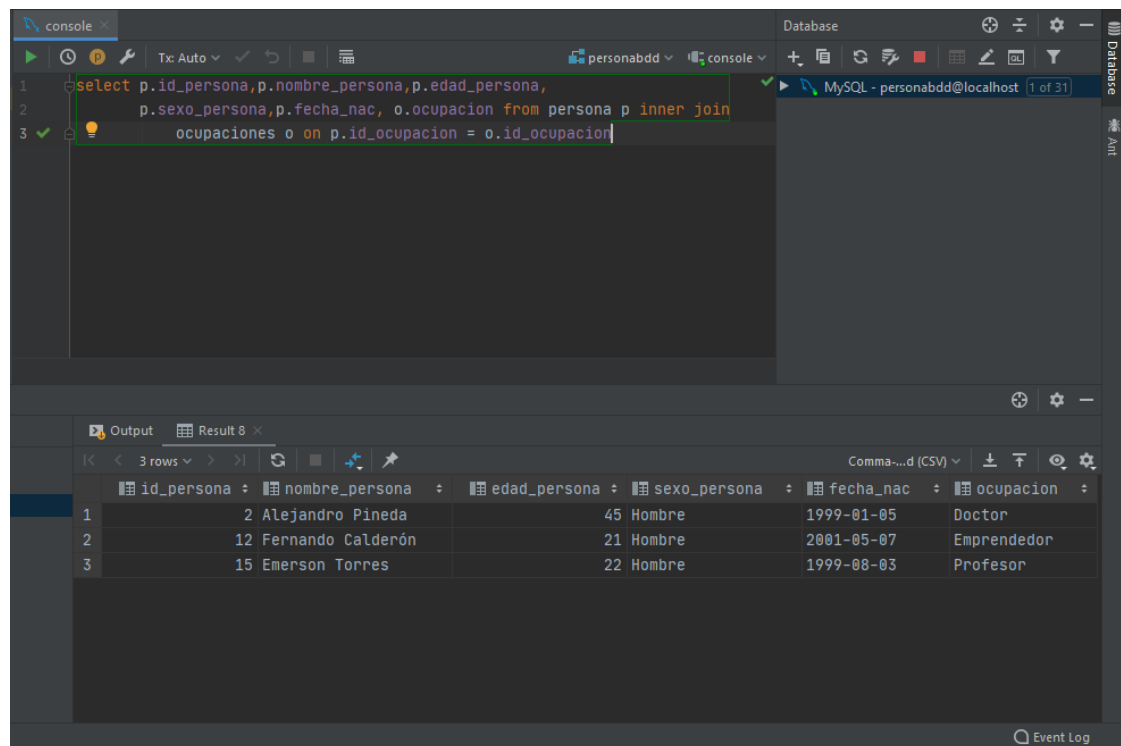
The screenshot shows a database console interface. The top panel displays the SQL query `select * from ocupaciones`. The bottom panel shows the output of the query as a table with 2 columns: `id_ocupacion` and `ocupacion`. The results are as follows:

id_ocupacion	ocupacion
1	Doctor
2	Emprendedor
3	Profesor

Aquí notaremos entonces que las profesiones asignadas a los campos `id_ocupacion` de la tabla `persona` son: **Doctor, Emprendedor y Profesor**, correspondientemente.

3. **SELECT** `p.id_persona, p.nombre_persona, p.edad_persona, p.sexo_persona, p.fecha_nac, o.ocupacion` **FROM** `persona p` **INNER JOIN** `ocupaciones o` **ON** `p.id_ocupacion = o.id_ocupacion`

Ahora podemos notar que aplicamos un filtro en cuanto a las columnas que mandamos a llamar para ver en la data, podemos notar que para funciones prácticas, esta vez mandamos a llamar todas las columnas, excepto **`id_ocupacion`**. La razón de este filtro es mostrar al cliente solamente información relevante, puesto que 1,2,3 correspondientemente no tienen un significado relevante para el cliente, sin embargo con esta consulta nos es mucho más fácil identificar las profesiones de los sujetos en cuestión.



The screenshot shows a MySQL database interface with a console window displaying a SQL query and its results. The query is:

```
select p.id_persona, p.nombre_persona, p.edad_persona,
       p.sexo_persona, p.fecha_nac, o.ocupacion from persona p inner join
       ocupaciones o on p.id_ocupacion = o.id_ocupacion
```

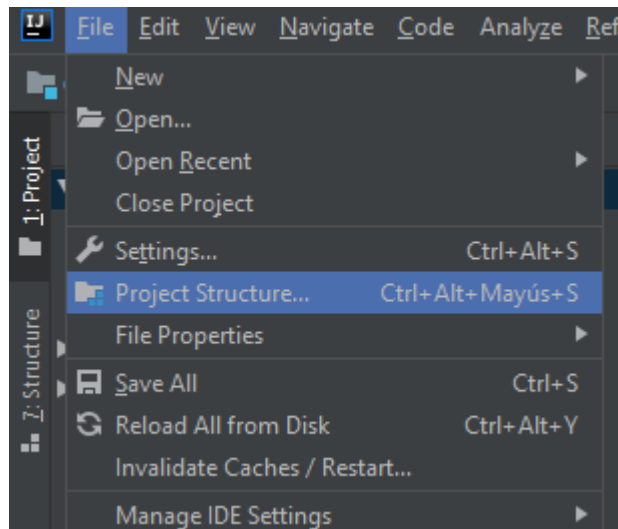
The results are displayed in a table with 6 columns: `id_persona`, `nombre_persona`, `edad_persona`, `sexo_persona`, `fecha_nac`, and `ocupacion`. The table contains 3 rows of data:

	<code>id_persona</code>	<code>nombre_persona</code>	<code>edad_persona</code>	<code>sexo_persona</code>	<code>fecha_nac</code>	<code>ocupacion</code>
1	2	Alejandro Pineda	45	Hombre	1999-01-05	Doctor
2	12	Fernando Calderón	21	Hombre	2001-05-07	Emprendedor
3	15	Emerson Torres	22	Hombre	1999-08-03	Profesor

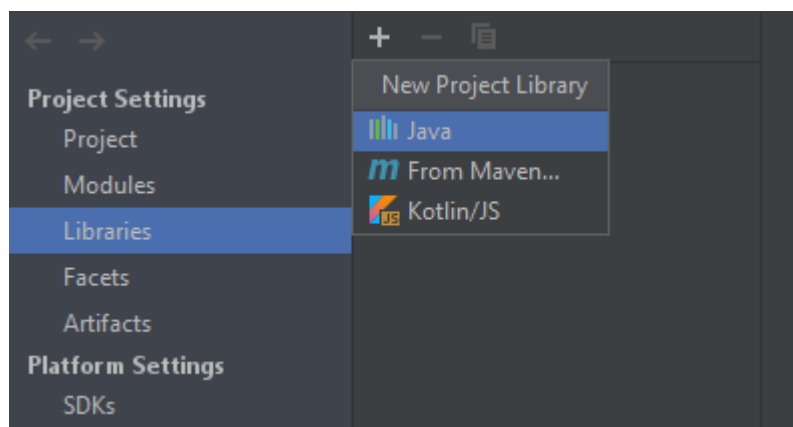
Acceder a los datos de una BD desde una aplicación Java

1. Primero vamos a integrar nuestro **mysql-connector-java-8.0.27.jar** a nuestro proyecto.

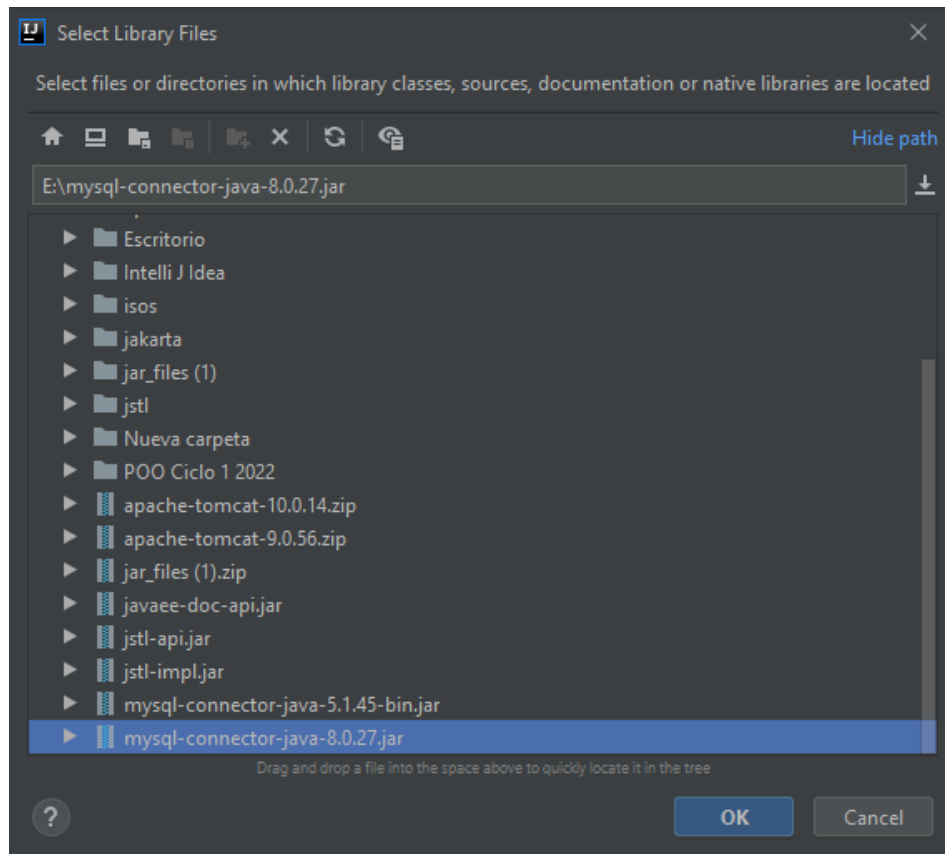
1- Ir a **File > Project Structure...**



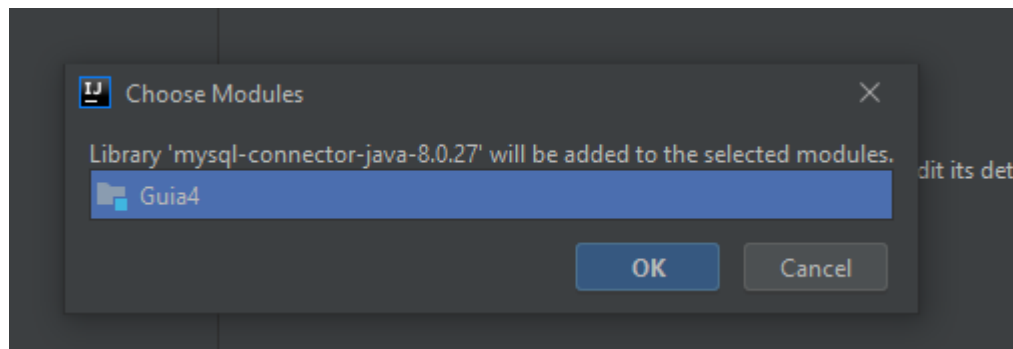
- 2- En **Project Settings**, buscar **libraries** y dar click en el botón (+) como se muestra en la figura. Finalmente seleccionara **Java**, como dependencia para el proyecto.



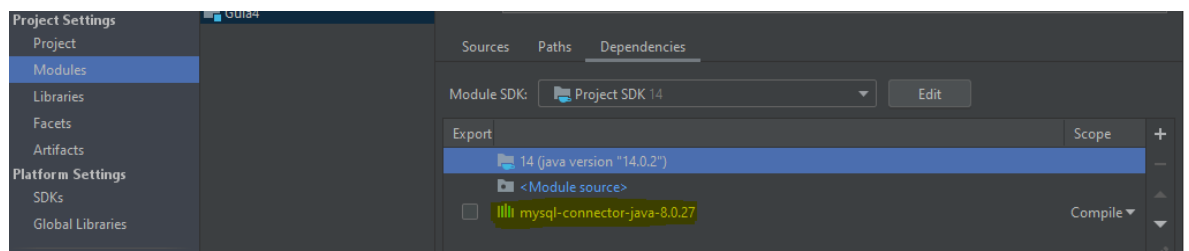
- 3- Ahora buscara el directorio en el que ha descomprimido su archivo **mysql-connector-java-8.0.27.jar** para integrarlo a su proyecto como librería local, posteriormente dar click en **ok**.



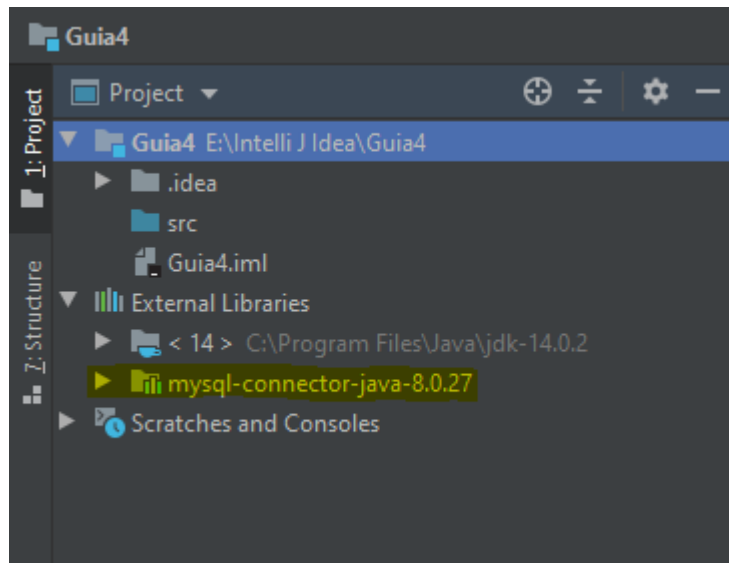
- 4- Es imprescindible que asigne esta librería a los módulos del proyecto.



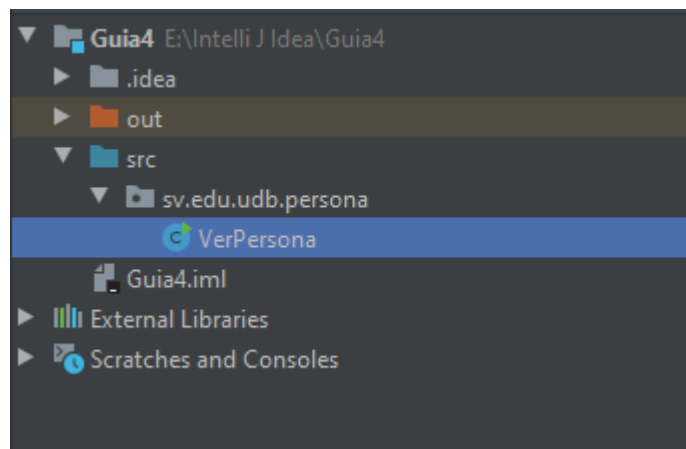
- 5- Bastara con que revise en la sección de módulos siempre en **Project Settings** y verifique que su librería ha sido incluida correctamente.



- 6- Finalmente notara que todas las librerías que se incluyan de ahora en adelante, se verán reflejadas en el directorio principal del proyecto.



2. En el proyecto Guia4 debe crear una clase y colocar el nombre **VerPersona**, dentro del paquete “**sv.edu.udb.persona**”.



3. Digitar el siguiente código:

```
import java.sql.*;

public class VerPersona {

    public VerPersona()
    {
        // Se utiliza un try por los posibles errores de MySQL
        try
        {
            //obtenemos el driver de para mysql

```



```

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

// Se obtiene una conexión con la base de datos.
Connection conexion = DriverManager.getConnection (
"jdbc:mysql://localhost/personabdd","root", "");
//      IMPORTANTE: EL CAMPO PASSWORD POR DEFECTO DEBE IR EN BLANCO,
//      DEBE ASIGNAR EL PASSWORD CORRECTO

// Permite ejecutar sentencias SQL sin parámetros
Statement s = conexion.createStatement();

// Contiene la tabla resultado de la pregunta SQL que se haya realizado
ResultSet rs = s.executeQuery ("select * from persona");

//Se recorre el ResultSet, mostrando por pantalla los resultados.

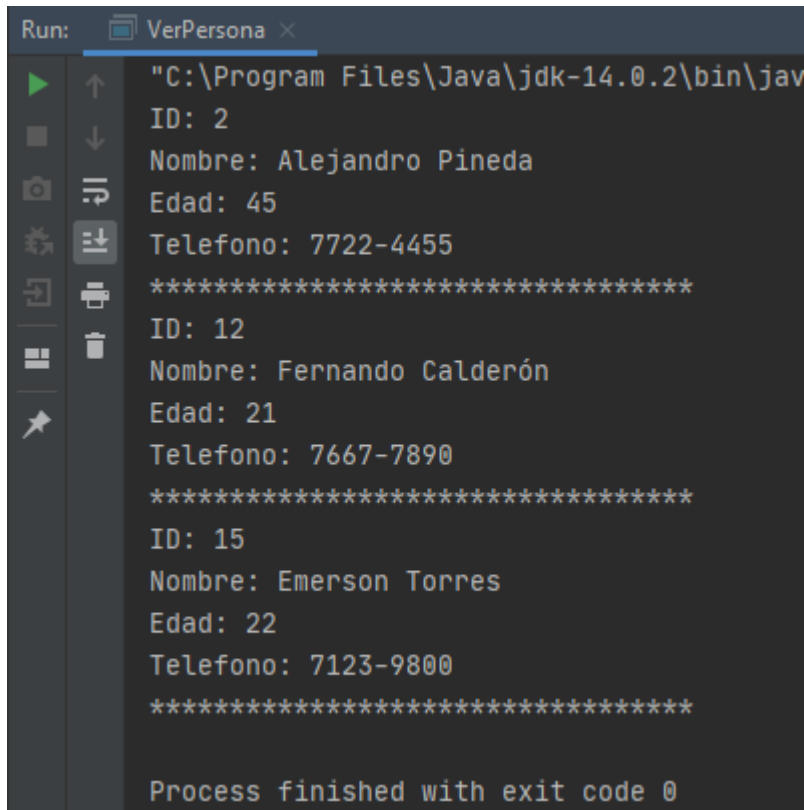
while (rs.next())
{
//Podemos mostrar los datos de otra forma ver mas abajo e la guia.
System.out.println ("ID: "+rs.getInt(1)+
"\nNombre: "+rs.getString(2)+
"\nEdad: "+rs.getString(3)+
"\nTelefono: "+rs.getString(4));
System.out.println("*****");
}

// Se cierra la conexión con la base de datos. NUNCA OLVIDE CERRARLA
conexion.close();
}
catch (ClassNotFoundException e1) {
//Error si no puedo leer el driver de MySQL
System.out.println("ERROR:No encuentro el driver de la BD:
"+e1.getMessage());
}
catch (SQLException e2) {
//Error SQL: login/passwdó sentencia sqlerronea
System.out.println("ERROR:Fallo en SQL: "+e2.getMessage());
}
}
}
/**
 * Método principal, instancia una clase PruebaMySQL
 *
 * @param args the command line arguments
 */
public static void main(String[] args)

```

```
{  
new VerPersona();  
}  
}
```

4. Ejecutar la aplicación y obtendremos los siguientes resultados.



```
Run: VerPersona x  
"C:\Program Files\Java\jdk-14.0.2\bin\jav  
ID: 2  
Nombre: Alejandro Pineda  
Edad: 45  
Telefono: 7722-4455  
*****  
ID: 12  
Nombre: Fernando Calderón  
Edad: 21  
Telefono: 7667-7890  
*****  
ID: 15  
Nombre: Emerson Torres  
Edad: 22  
Telefono: 7123-9800  
*****  
Process finished with exit code 0
```

5. Forma alternativa de mostrar los datos.

Para obtener los distintos atributos de la base se utilizarán, normalmente, los métodos `getString(atributo)`, cuyo parámetro es una cadena con el nombre del atributo que queremos recuperar.

```
//Se recorre el ResultSet, mostrando por pantalla los resultados.  
while (rs.next())  
{  
System.out.println ("ID: "+rs.getInt("id_persona")+  
"\nNombre: "+rs.getString("nombre_persona")+  
"\nEdad: "+rs.getString("edad_persona")+  
"\nTelefono: "+rs.getString("telefono_persona"));  
System.out.println("*****");  
}
```

Ejercicio 3: Ingresar datos desde una aplicación a una base de datos

1. En el proyecto Guia4 debe crear una clase y colocar el nombre **IngresoDatos**, dentro del paquete “**sv.edu.udb.persona**”.
2. Digitar el siguiente código.

```
package sv.edu.udb.persona;
import java.sql.*;
import javax.swing.JOptionPane;
import sv.edu.udb.util.* ;

public class IngresoDatos {
    private int id;
    private String ids;
    private String nombre;
    private int edad;
    private String edads;
    private String telefono;
    private String sexo;
    private int idocupacion;
    private String idocupacions;
    private String fechanac;

    private Connection conexion;
    private ResultSet rs;
    private Statement s;

    public IngresoDatos()
    {
        // Se utiliza un try por los posibles errores de MySQL
        try
        {
            //obtenemos el driver de para mysql
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Se obtiene una conexión con la base de datos.
            conexion = DriverManager.getConnection (
                "jdbc:mysql://localhost/personabdd","root", "");
            //
            // IMPORTANTE: EL CAMPO PASSWORD DEBE IR EN BLANCO POR DEFECTO,
            // DEBE ASIGNAR EL PASSWORD CORRECTO

            // Permite ejecutar sentencias SQL sin parámetros
            s = conexion.createStatement();
            //Metodo para ingresar valores
            ingreso();
            s.executeUpdate("Insert into persona
values("+id+",\""+nombre+"\","+edad+",\""+telefono+"\","+sexo+"\","+idocupaci
on+",\""+fechanac+"\")");

            JOptionPane.showMessageDialog(null,"Persona Ingresada
Correctamente");
        }
        catch (ClassNotFoundException el) {
            //Error si no puedo leer el driver de MySQL
            System.out.println("ERROR:No encuentro el driver de la BD:
```

```

"+e1.getMessage());
        System.exit(0);
    }
    catch (SQLException e2) {
//Error SQL: login/passwdó sentencia sql erronea
        System.out.println("ERROR:Fallo en SQL: "+e2.getMessage());
        System.exit(0);
    }
}

public void obtenerOcupacion() throws SQLException {
    rs = s.executeQuery("select count(*) from ocupaciones");
    int tamanio=0;
    while (rs.next()) {
        tamanio= rs.getInt(1);
    }
    String[] options2 = new String[tamanio];
    int contador = 0;
    rs = s.executeQuery("select * from ocupaciones");
    String texto="";
    while (rs.next()) {
        options2[contador] =rs.getString("ocupacion");
        contador++;
    }
    idocupacions=(String) JOptionPane.showInputDialog(null, "Seleccione una
ocupacion", "ocupacion persona", JOptionPane.QUESTION_MESSAGE, null, options2,
options2[0]);
    rs= s.executeQuery("select id_ocupacion from ocupaciones where
ocupacion ='"+idocupacions+"'" );
    while (rs.next()){
        idocupacion= rs.getInt(1);
    }
}

public void ingreso() throws SQLException {
    ids = JOptionPane.showInputDialog("Ingreso el ID");
    id = Integer.parseInt(ids);
    nombre = JOptionPane.showInputDialog("Ingreso el Nombre");
    edads = JOptionPane.showInputDialog("Ingreso la edad");
    edad = Integer.parseInt(edads);
    telefono = JOptionPane.showInputDialog("Ingreso su Telefono");
    do {
        if ( MatchTelephone.compareTelephone(telefono) == true)
        {
            break;
        }else{
            JOptionPane.showMessageDialog(null, "Numero de Telefono
invalido");
            telefono=JOptionPane.showInputDialog("Ingreso su Telefono");
        }
    } while (true);

    String[] options = {"Femenino", "Masculino"};
    sexo = (String) JOptionPane.showInputDialog(null, "¿Ingreso el sexo?",
"Sexo persona", JOptionPane.QUESTION_MESSAGE, null, options, options[0]);
    obtenerOcupacion();
}

```

```

        fechanac= JOptionPane.showInputDialog("Ingrese la fecha de
nacimiento");
        do {
            if ( MatchDate.compareDate(fechanac) == true)
            {
                break;
            }else{
                JOptionPane.showMessageDialog(null, "Formato fecha invalido");
                fechanac=JOptionPane.showInputDialog("Ingrese la fecha de
nacimiento");
            }
        } while (true);

    }

    public void mostrardatos() throws SQLException{
        rs = s.executeQuery ("select * from persona");
        while (rs.next())
        {
            JOptionPane.showMessageDialog(null,"ID:
"+rs.getString("id_persona")+
            "\nNombre: "+rs.getString("nombre_persona")+
            "\nEdad: "+rs.getInt("edad_persona")+
            "\nSexo: "+rs.getString("telefono_persona")

        );
        }
    }

    public void cierreconexion() throws SQLException{
// Se cierra la conexión con la base de datos.
        if (conexion != null){
            conexion.close();
        }
    }

    public static void main(String[] args) throws SQLException
    {
        IngresoDatos ing=new IngresoDatos();
        ing.mostrardatos();
        ing.cierreconexion();
    }
}

```

3. Crear la clase llamada **MatchTelephone** dentro del paquete “sv.edu.udb.util” y agregar el siguiente código, esta clase utilizara expresiones regulares para validar el formato de un número de teléfono.

```

package sv.edu.udb.util;

import static java.lang.Boolean.FALSE;
import static java.lang.Boolean.TRUE;
import java.util.regex.Matcher;

```

```

import java.util.regex.Pattern;

/**
 *
 * @author Rafael Torres
 */
public class MatchTelephone {

    // public static void main(String[] args) {
    //     compareTelephone("7123-4444");
    // }

    public static boolean compareTelephone (String telefono) {

        String expresion="(2|7)\\d{3}-\\d{4}";

        Pattern pat = Pattern.compile(expresion);
        Matcher mat = pat.matcher(telefono);
        if (mat.matches()) {
            System.out.println("SI");
            return TRUE;
        }

        return FALSE;
    } //Cierre del main

}

```

4. Crear la clase llamada **MatchDate** dentro del paquete “**sv.edu.udb.util**” y agregar el siguiente código, esta clase utilizara expresiones regulares para validar el formato de la fecha.

```

package sv.edu.udb.util;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

import static java.lang.Boolean.FALSE;
import static java.lang.Boolean.TRUE;

public class MatchDate {
    // public static void main(String[] args) {
    //     compareTelephone("1999-03-08");
    // }

```

```

public static boolean compareDate (String date) {

    String expresion="\d{4}-\d{2}-\d{2}";

    Pattern pat = Pattern.compile(expresion);
    Matcher mat = pat.matcher(date);
    if (mat.matches()) {
        System.out.println("SI");
        return TRUE;
    }

    return FALSE;
} //Cierre del main
}

```

Ejercicio 4: Nulos y vacios.

1. Crear la base de datos **guia4** y correr el siguiente script.

```

create table Empleados
(Codigo int primary key,
Nombre varchar(25),
Apellidos varchar(25),
Telefono varchar(9)
);
INSERT into Empleados VALUES
(1,'Roberto Mario','Rodríguez','2589-8585'),
(2,'Maria Gabriela','Carranza','7895-7858'),
(3,'José Fernando','Martinez','2698-4576');

```

2. Para este ejemplo se realizara una análisis que nos permita comprender que no es lo mismo manejar datos “Vacios o Nulos”, en primer lugar crearemos la clase **Conexion** para que este en un paquete diferente “**sv.edu.udb.util**” y solo la invoquemos para futuros desarrollos.

```
package sv.edu.udb.util;
```

```

/**
 *
 * @author Rafael Torres
 */
import java.sql.*;

public class Conexion {
    private Connection conexion =null;
    private Statement s =null;
    private ResultSet rs=null;
    private String ingresoempleados="";

    //Contructor
    public Conexion() throws SQLException{
        try
        {
            //obtenemos el driver de para mysql
            Class.forName("com.mysql.jdbc.Driver");
            // Se obtiene una conexión con la base de datos.
            conexion = DriverManager.getConnection (
                "jdbc:mysql://localhost/guia4","root", "rafael");
            // Permite ejecutar sentencias SQL sin parámetros
            s = conexion.createStatement();

            System.out.println("Conexion Exitosa");

        }
        catch (ClassNotFoundException e1) {
            //Error si no puedo leer el driver de MySQL
            System.out.println("ERROR:No encuentro el driver de la BD:
"+e1.getMessage());
        }
    }

    //Metodo que permite obtener los valores del resulset
    public ResultSet getRs() {
        return rs;
    }

    //Metodo que permite fijar la tabla resultado de la pregunta
    //SQL realizada
    public void setRs(String sql) {
        try {

            this.rs = s.executeQuery(sql);
        }
    }
}

```



```

    } catch (SQLException e2) {
        //Error SQL: login/passwd ó sentencia sql errónea
        System.out.println("ERROR:Fallo en SQL: "+e2.getMessage());
    }
}

//Metodo que recibe un sql como parametro que sea un update,insert.delete
public void setQuery(String sql) throws SQLException {
    this.s.executeUpdate(sql);
}

//Metodo que cierra la conexion
public void cerrarConexion() throws SQLException{
    conexion.close();
}
}

```

3. Crear la clase llamada “**InsertNulos**” dentro del paquete “**sv.edu.udb.nulos**” y agregar el siguiente código.

```

package sv.edu.udb.nulos;
import java.sql.ResultSet;
import java.sql.SQLException;
import sv.edu.udb.util.Conexion;

/**
 *
 * @author Rafael Torres
 */
public class InsertNulos {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws SQLException {
        // TODO code application logic here

        Conexion con = new Conexion();
        String sql1 = "insert into empleados values(7,','Torres',null)";
        //
        con.setQuery(sql1);

        String sql = "select nombres from empleados ";
        ResultSet rs ;

        con.setRs(sql);
    }
}

```

```

rs= con.getRs();

String nombre;

while (rs.next()){
    nombre=rs.getString(1);

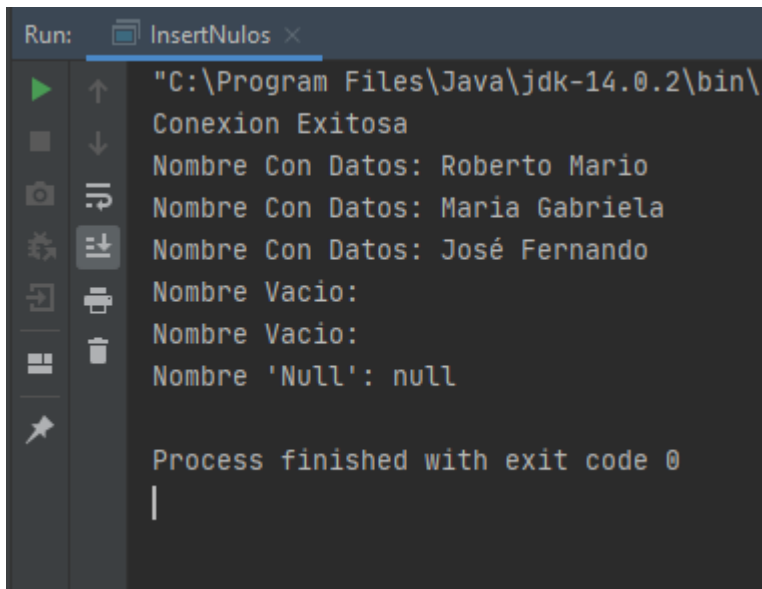
    if (nombre == null){
        System.out.println("Nombre 'Null': " + nombre);
    }else if(nombre.equals("")){
        System.out.println("Nombre Vacio: " + nombre);
    }else{
        System.out.println("Nombre Con Datos: " + nombre);
    }

}

con.cerrarConexion();
}
}

```

4. Ejecutar un par de sentencias update para la tabla **empleados**.
 - insert into empleados values(5, 'Torres', null);
 - insert into empleados values(6, null, Rodriguez, null);
5. Ejecutar la clase **InsertNulos** para ver un resultado como el de la siguiente imagen



```

Run: InsertNulos x
"C:\Program Files\Java\jdk-14.0.2\bin\
Conexion Exitosa
Nombre Con Datos: Roberto Mario
Nombre Con Datos: Maria Gabriela
Nombre Con Datos: José Fernando
Nombre Vacio:
Nombre Vacio:
Nombre 'Null': null

Process finished with exit code 0
|

```

IV. EJERCICIOS COMPLEMENTARIOS

➤ Crear las siguientes tablas en MySql:

Tabla alumno

Nombre Columna	Tipo de dato
Cod_alumno	int primary key
Nombre	varchar(80)
Apellido	varchar(80)
Edad	int
Direccion	varchar(100)

Tabla materia

Nombre Columna	Tipo de dato
Cod_materia	int primary key
Nombre	varchar(25)
Descripción	varchar(100)

Tabla alumno_materia

Nombre Columna	Tipo de dato
Cod_alumno	int (llave foránea)
Cod_materia	int (llave foránea)

Crear el CRUD a las 3 tablas anteriores:

- ❖ Alumno
- ❖ Materia
- ❖ Alumno_materia

Reportes

1. Crear una aplicación donde muestre la información de cada una de las tablas anteriores, para ello crear tres métodos diferentes que serán invocados según la necesidad.
2. Mostrar las materias que está cursando un alumno específico.

Tip: Usará la cláusula where en la sentencia SQL

Nota: Para cada ejercicio solicitado utilizar JOptionPane y validar los datos. Debe usar una clase pre-fabricada para validar.