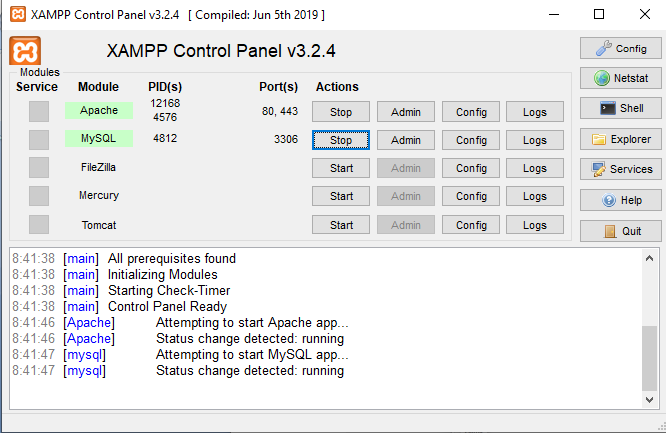
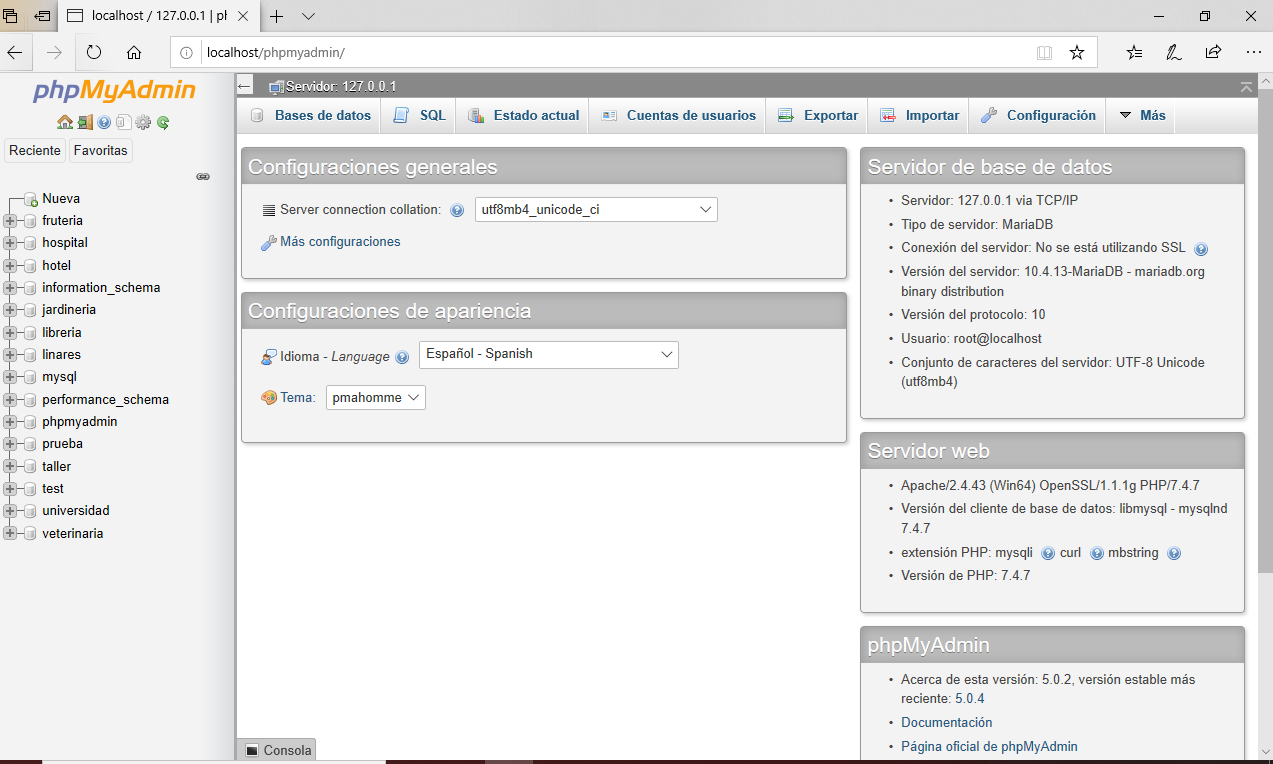
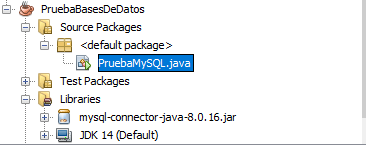
**Access to Relational Databases from Java**





In the Java program, we must have a connector:



Database access program from Java:

/\*\*

\*

\* @author adelapresa

\* Test program for connection to a MySQL database, working almost everything

\* from Java source code

\*Assumes that the database server is up, available,

\* on the default port. (For example, in the XAMPP that we have in the previous document)

\* The username and password for connecting to the database must be changed to the

\* suitable for our needs.

\* By adding the code that deletes the database, it is not necessary to have defined

\* previously the structure of it, since all the control of the program is now on the side of the

\* host language. However, it is necessary that it has been defined so that it does not give us problems the first time.

\* time -since there is nothing to delete.

\*

\* We create a test database

\* has a creditors table with three fields, like this:

\* numacree (creditor number)

\* creditname (creditor name)

\* address

\*

\* I fill the table with data and check if I can delete one of them

\*

\* Afterwards, another table is created in the database, called clients, with a structure

\* quite similar:

\* customernum

\* clientname

\* address

\*/

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

/\*\*

\* Connection test class with a MySQL database

\*/

public class TestMySQL {

/\*\*

\* Create an instance of the MySQL class and do all the code

\* connection, query and display of results.

\*/

public TestMySQL() throws SQLException

{

// Put everything in a try because of possible MySQL errors

try

{

// MySQL Driver is registered

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

} catch (Exception ex) {

// handle the error

System.out.println(ex.getMessage());

}

// A connection to the database is obtained. One has to

// change the user "root" and password ""(It goes without saying that in a real //production environment, the password could not be an empty field) for the

// appropriate to the database we are using.

Connection connection = DriverManager.getConnection(

"jdbc:mysql://localhost/test","root", "");

//If we wanted to access a database in MySQL Workbench, it would be necessary to modify the previous line as follows:

//"jdbc:mysql://localhost/test"+"?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC","root", "");

// A Statement is created to perform the query and it is declared that it can be updated

Statement s = connection.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE,ResultSet.CONCUR\_UPDATABLE);

//We are going to delete the database so that it does it from scratch every time it starts and does not give us errors

//Which will be the one we will then insert (For this to work, the database must have been previously created

//but we don't need to do anything else)

s.executeUpdate("DROP DATABASE test");

s.executeUpdate("CREATE DATABASE test");

s.executeUpdate("USE test");

s.executeUpdate("CREATE TABLE creditors(numacree INT, creename VARCHAR(25), address VARCHAR(25), PRIMARY KEY(numacree))");

//The following line would delete all the elements of the creditors table if we did not already have it deleted when deleting the database

//s.executeUpdate("DELETE FROM creditors");

//Enter data

s.executeUpdate("INSERT INTO creditors " + "VALUES (1,'Recaredo', 'General Dávila 27')");

s.executeUpdate("INSERT INTO creditors " + "VALUES (2,'Chindasvinto', 'General Dávila 17')");

s.executeUpdate("INSERT INTO creditors " + "VALUES (3,'Leovigildo', 'General Dávila 7')");

// The query is performed. The results are saved in the

// ResultSet rs

ResultSet rs = s.executeQuery("SELECT \* FROM creditors");

// The ResultSet is traversed, displaying the results on the screen.

while (rs.next())

{

System.out.println(rs.getInt("numacree") + " " + rs.getString(2)+

" " + rs.getString(3));

//Let's now delete the second of the inserted records

//using the ResultSet method

if (rs.getString(2).equals("Chindasvinto")){

rs.deleteRow();//This command deletes the record whose second value is Chindasvinto

}

}

System.out.println("Now we are going to view the list of creditors to verify that it has actually been deleted");

ResultSet rs2=s.executeQuery("select numacree, nombacree, address from creditors");

while (rs2.next())

{

System.out.println(rs2.getInt("numacree") + " " + rs2.getString (2)+

" " + rs2.getString(3));}

//Create a new table

s.executeUpdate("CREATE TABLE clients(clientnum INT, clientname VARCHAR(25),address VARCHAR(25))");

// The connection to the database is closed.

connection.close();}

//Let's create a new customers table

/\*\*

\* Main method, instantiates a TestMySQL class

\*

\* @param args the command line arguments

\*/

public static void main(String[] args) throws SQLException

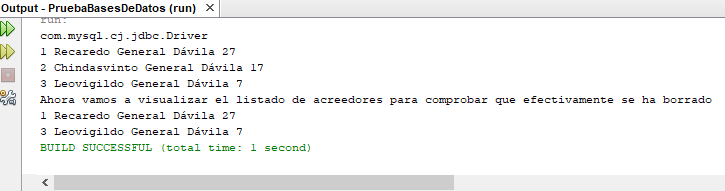
{

new TestMySQL();

}

}

Executing the above program produces the following output:



We can check the modifications in the database itself -in phpmyadmin-

