USO DEL ENTORNO DE NETBEANS PARA EL ACCESO DE LA BASE DE DATOS Y EL PAQUETE JAVA.SQL

LOGRO DE SESIÓN: Al finalizar la sesión el estudiante es capaz de implementar base de datos desde Netbeands y desarrollar aplicaciones graficas que involucren controles de lista.

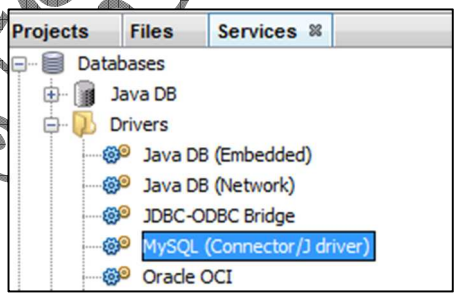
ACCESO A LA BASE DE DATOS DESDE NETBEANS

1

NetBeans nos permite hacer operaciones sobre la base de datos como crear y borrar tablas, agregar y eliminar columnas, agregar, modificar y eliminar registros de datos como realizar consultas.
Vamos a proceder hacer algunas operaciones:

1) CONFIGURACIÓN PARA ACCEDO A DATOS DESDE NETBEANDS.

A) En el Explorador de proyecto Accede a la ficha “Servicios”.

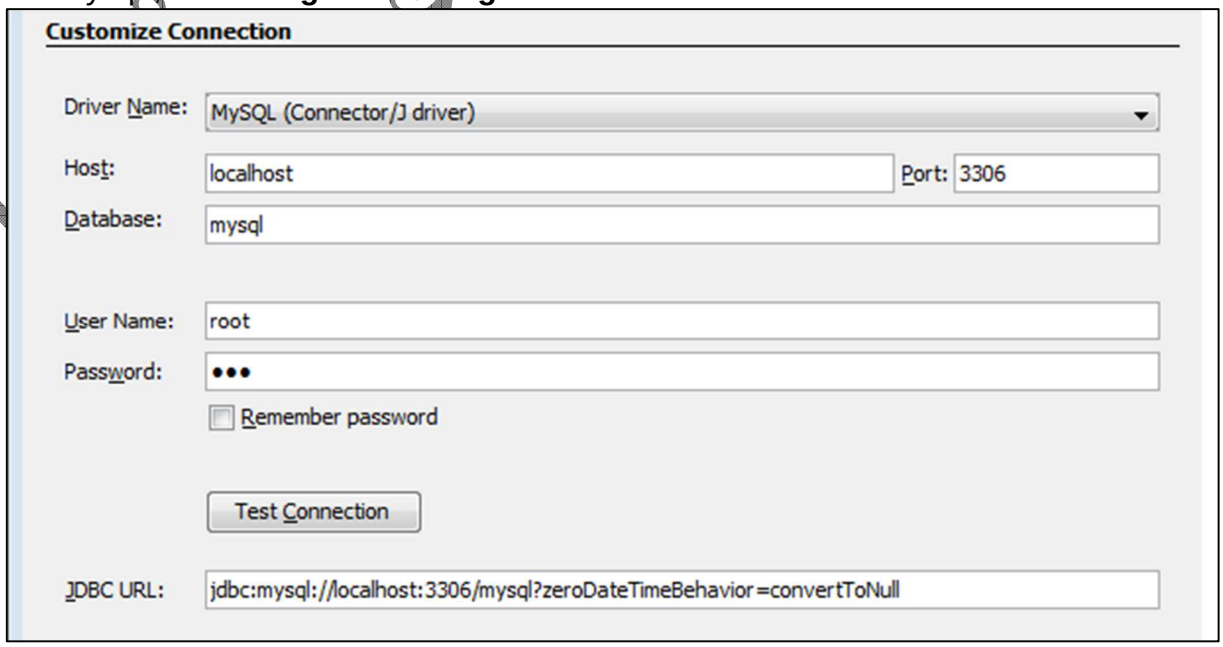


B) Expandir el icono **Databases,** Drivers.

C) Clic Secundario en el Driver: **“Mysql(Connector/JDriver)”**

D) Elegir la opcion “Usar Conexion”

E) En el asistente de configuración ajustar algunas opciones como contraseña de “Mysql”. **Vea la siguiente imagen:**

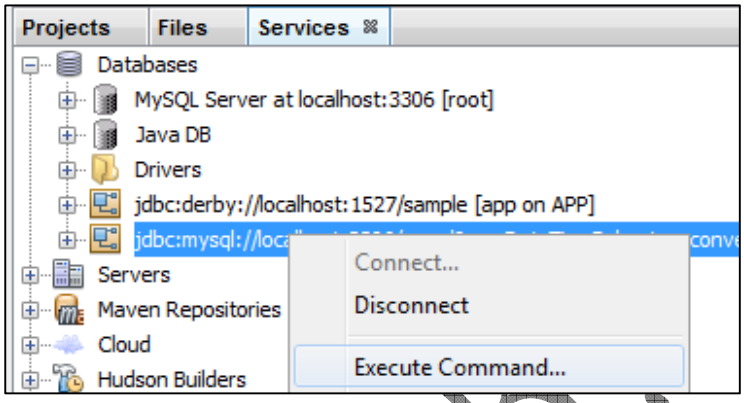


F) Finaliza la configuración.

2) ELABORAR UNA BASE DE DATOS EN MYSQL DESDE NETBEANDS

A) Clic Secundario en la Conexión Creada, Seleccionar la opción "**Execute Command**".

IMPORTANTE. Debe abrir una ventana que servirá para escribir "CODIGO SQL"



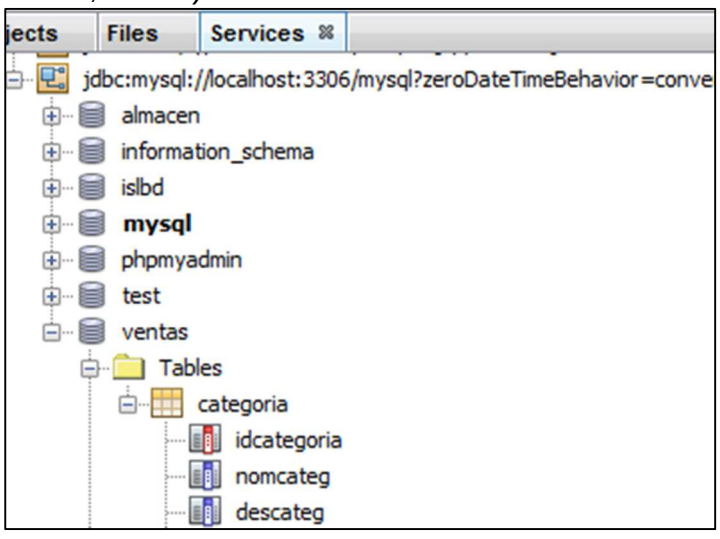
B) Crear la base de datos "ventas" y una tabla "Categoría". **Vea siguiente imagen:**

```
Source  History  Connection: jdbc:mysql://localhost:3306/mysql?zeroDa...
1  --Creando la Base de datos ventas
2  CREATE DATABASE ventas;
3  --Seleccionado la Base de datos
4  USE ventas;
5  --Creando la Tabla "Categoria"
6  CREATE TABLE categoria
7  (
8  idcategoria    int primary key,
9  nomcateg       varchar(25) not null,
10 descateg       varchar(50) null
11 );
12 --Insertar Registro
13 INSERT INTO categoria VALUES(1,'Carne','');
14 INSERT INTO categoria VALUES(2,'Lacteo','');
15 --Consulta de información
16 SELECT * FROM categoria;
```

IMPORTANTE
Para la ejecución del código Seleccionar el código, Clic Secundario "**Ejecutar**"

C) Revisar los objetos (Base de datos, tablas) elaborados.

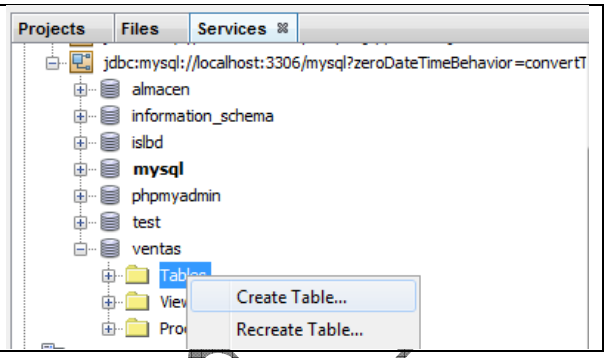
Observamos en la figura anterior las carpetas referidas a Tables (tablas), Views (vistas) y Procedures (procedimientos). Si expandimos el nodo referido a Tables veremos la tabla estadio.



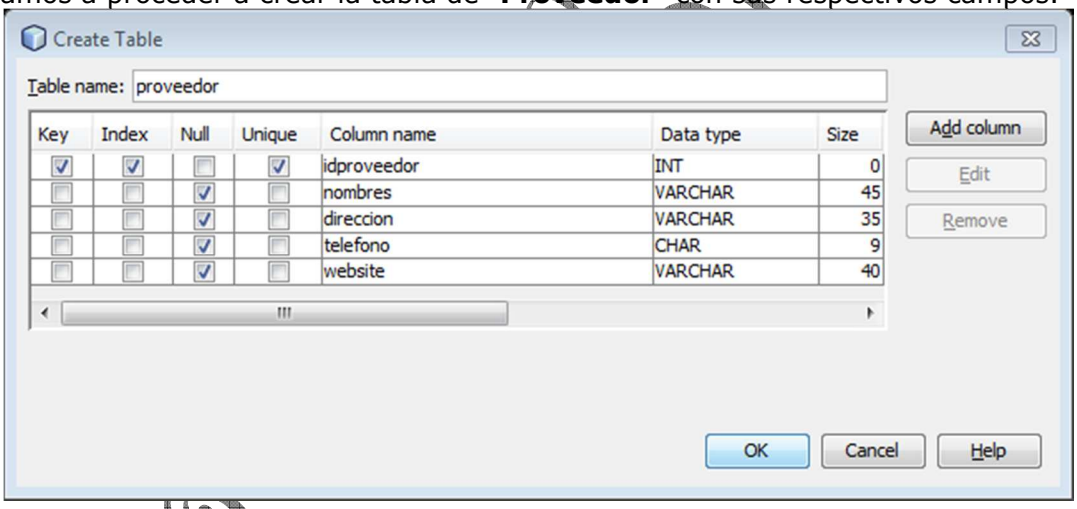
IMPORTANTE. El usuario también tiene la posibilidad de elaborar la base de datos de forma gráfica.

Ejemplo: Vamos a elaborar la tabla "Proveedor".

A) Si seleccionamos el nodo referido a Tables y damos clic boton derecho, se muestra un menu flotante y luego elegimos la opcion **Create Table**.



B) Vamos a proceder a crear la tabla de "**Proveedor**" con sus respectivos campos.



C) Si deseamos insertar un registro a una de las tablas a través del comando **insert,** seleccionamos el nodo referido a Tables. Dando clic botón derecho se muestra un menú flotante y luego seleccionamos la opcion **Execute Command...**

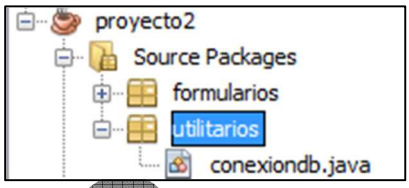
Actividad propuesto, Adicione 3 proveedores usando código SQL.

CONTROLES DE LISTA (JDBC-MYSQL)

Vamos a proceder a elaborar una aplicación gráfica para visualizar en objetos Jlist, los datos contenidos de la tabla "Proveedor".

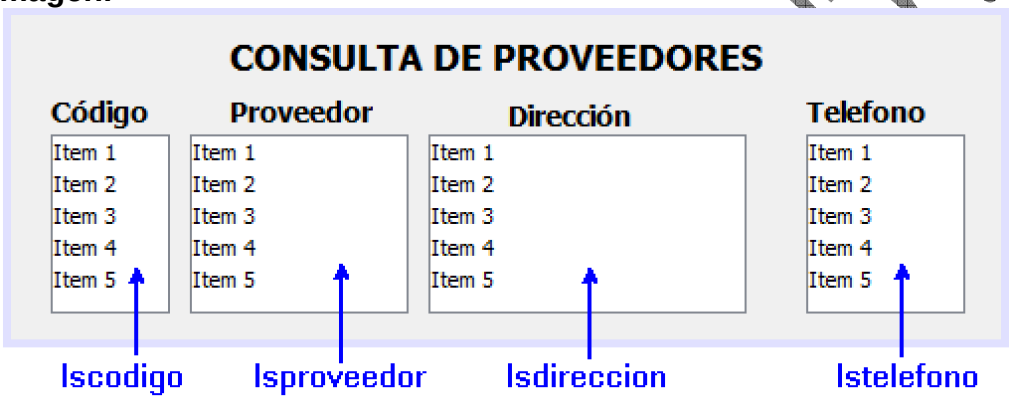
1) REALIZAR EL DISEÑO DEL FORMULARIO.

A) Crear 2 paquetes, "formularios" y "utilitarios". Vea la imagen:



B) En el paquete "utilitarios" copiar la clase "Conexionbd" implementada en la sesión anterior.

C) Diseñamos el formulario, donde debemos incluir el **jList. Vea la siguiente imagen:**



IMPORTANTE: Borrar los Items de los JList

2) PROGRAMAR EL FORMULARIO.

A) Importar el Espacio de Nombres.

```
package formularios;
//1. IMPORTAR EL ESPACIO DE NOMBRES
import java.sql.*;
```

B) Programar a nivel de clase

```
public class frm_Verproveedor extends javax.swing.JFrame {
    //CREANDO VARIABLES GENERALES
    Connection conn = conexiondb.conetarbd();
    static Statement st = null;
    static ResultSet rs = null;

    //CREANDO EL MODELO DE LA LISTA
    DefaultListModel modelo1 = new DefaultListModel();
    DefaultListModel modelo2 = new DefaultListModel();
    DefaultListModel modelo3 = new DefaultListModel();
    DefaultListModel modelo4 = new DefaultListModel();
}
```

5

C) Programar en el constructor del formulario.

```
public frm_Verproveedor() {
    initComponents();
    lsCodigo.setModel(modelo1);
    lsProveedor.setModel(modelo2);
    lsDireccion.setModel(modelo3);
    lsTelefono.setModel(modelo4);

    try {
          st = conn.createStatement();
          rs = st.executeQuery("select * from proveedor;");
        while(rs.next()){
          modelo1.addElement(rs.getString(1));
          modelo2.addElement(rs.getString("nombres"));
          modelo3.addElement(rs.getString(3));
          modelo4.addElement(rs.getString(4));
        }

    } catch (Exception e) {
       System.out.println(e);
    }
}
```

D) Probar la aplicación.

CONTROL JTABLE EN GESTION DE DATOS (JDBC-MYSQL)

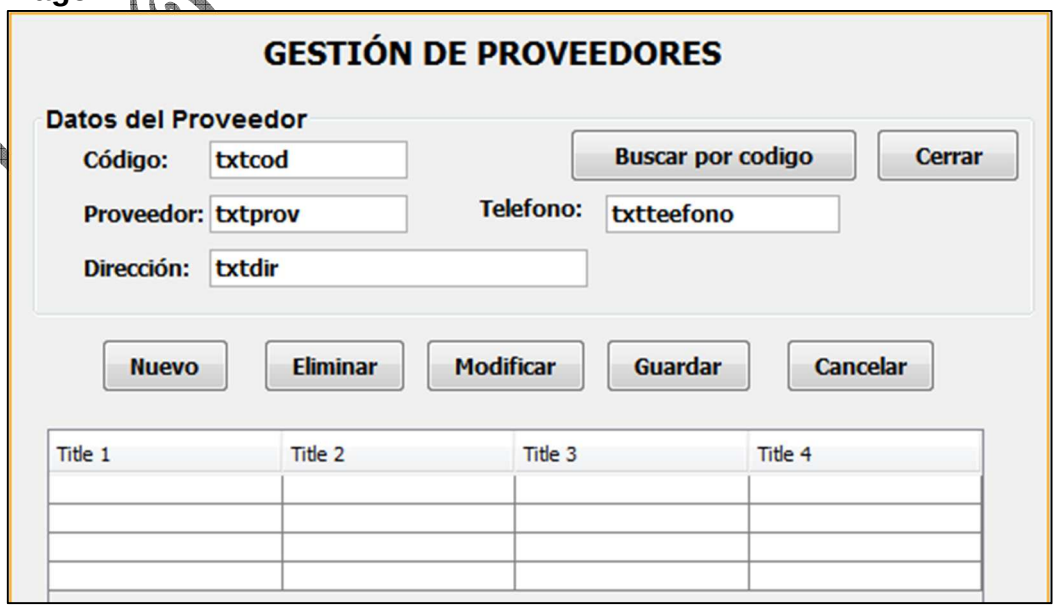
La presentación de datos tabulados es una de las tareas más comunes que se presentan al momento de crear interfaces graficas; desde la simple tabla que permite únicamente mostrar el resultado de una consulta, **hasta las que permiten editar directamente el contenido de cada celda, ordenar las columnas, personalizar su apariencia, etc.** Todas las tareas antes descritas, y muchas otras, son posibles de realizar utilizando la clase **JTable**; por supuesto, mientras más complejo sea el requerimiento a cubrir, se requerirá en igual medida utilizar más métodos o recursos de la clase. El objeto Jtable como los *modelos* de la tabla representados a través de la interfaz **TableModel**, pertenecen al paquete javax.swing.

El siguiente grafico intenta mostrar cómo cada componente JTable obtiene siempre sus Datos desde un *modelo de tabla*.

El TableModel se implementa a partir de la clase AbstractTableModel, aunque existe un modelo de tabla predeterminado denominado la clase DefaiultTableModel. Las propiedad más usada es ***model*** que permite definir el número de filas y columnas, siendo los metodos más usados: ***setModel(),*** que permite vincular un modelo al obejto Jtable y ***getRowCount()***, devuelve el número de filas en la tabla. Para la clase DefaultTableModel los metodos más utilizados son: ***AddRow()***, anade una fila al final del modelo, ***getRowCount()*** devuelve el número de filas de la tabla de datos, ***getValueAt()*** devuelve el dato ubicado en la posicion fila y columna y removeRow() elimina una fila del modelo segun posicion indicada.

1. REALIZAR EL DISEÑO DEL FORMULARIO.

A) Diseñamos el formulario, donde debemos incluir el **jTABLE. Vea la siguiente imagen:**



IMPORTANTE: Asignar un nombre a cada uno de los Objetos

2. PROGRAMAR EL FORMULARIO.

A. Importar el Espacio de Nombres

```
package formularios;
//1. IMPORTAR EL ESPACIO DE NOMBRES
import java.sql.*;
```

B. Programar a nivel de clase

```
public class frm_Verproveedor extends javax.swing.JFrame {
  //2. PROGRAMAR A NIVEL DE CLASE
    //A) Objetos Conexion, Statement, Resulset
    static  Connection cone = conexiondb.conetarbd();
    static Statement st =null;
    static PreparedStatement pst =null;
    static ResultSet rs=null;
    //B. Obeto ModeloTabla
    DefaultTableModel  dtm  = new DefaultTableModel();

  public void  TraerProveedor(){
     try {
        setSize(600,500);//Ancho y Alto del formulario
        st = cone.createStatement();//Conexion a bd
        rs = st.executeQuery("select * from proveedor;");
        String datos[] = new String[4]; //Declaracón de Array
        //Limpiar Tabla
       int fila = dtm.getRowCount(); //Retornando el # Filas
          if (fila>0) //Si tiene al menos un registro
                for (int i=0;i<fila;i++)
                   dtm.removeRow(0);
        //Leer Datos
        while(rs.next()){   //Guardando datos en Vector
          datos[0] = rs.getString(1);
          datos[1] = rs.getString(2);
          datos[2] = rs.getString(3);
          datos[3] = rs.getString(4);
          dtm.addRow(datos); //Adicionando al Modelo
        }
      } catch (Exception e) {
         System.out.println(e);
      }
   }
}
```

C. Programar en el constructor del formulario.

```
public frm_Verproveedor() {
    initComponents();
  //4. CONFIGURACION PLANTILLA DE TABLA
    String titulos[] ={"CÓDIGO","PROVEEDOR", "DIRECCIÓN", "TELEFONO"};
    dtm.setColumnIdentifiers(titulos);
    tablaproveedor.setModel(dtm);
    //MOSTRAR LOS ESTADIOS
    TraerProveedor();
}
```

D. Programar en el botón GUARDAR.

```
// 5. BOTON GUARDAR
    try {
       String sql = "insert into proveedor values(?,?,?,?,'')";

       pst = cone.prepareStatement(sql);
       //ADICIONANDO DATOS A PARAMETROS
       pst.setInt(1, Integer.parseInt(txtcod.getText()));
       pst.setString(2, txtprov.getText());
       pst.setString(3, txtdir.getText());
       pst.setString(4, txtteefono.getText());

       //EJECUCION DE SENTENCIA
       pst.executeUpdate();
       System.out.printf("Guardado Ok");
       //MOSTRANDO PROVEEDORES
       TraerProveedor();
    } catch (Exception e) {
       System.out.println(e);
    }
```

E. Programar en el botón ACTUALIZAR.

```
try {
   String sql;
   sql = "UPDATE proveedor SET  nombres=?, direccion=? , telefono=?, website='' where idproveedor=? ";
   pst = cone.prepareStatement(sql);
    //ADICIONANDO DATOS A PARAMETROS
    pst.setString(1, txtprov.getText());
    pst.setString(2, txtdir.getText());
    pst.setString(3, txtteefono.getText());
    pst.setInt(4, Integer.parseInt(txtcod.getText()));
    //EJECUCION DE SENTENCIA
    pst.executeUpdate();
    System.out.printf("Actualizado Ok");
    //MOSTRANDO PROVEEDORES
    TraerProveedor();

 } catch (Exception e) {
    System.out.println(e);
 }
```

F. Programar en el botón ELIMINAR

```
// ELMINAR REGISTRO
 try {
      st = cone.createStatement();
      //VARIABLES
      int id=Integer.parseInt(txtcod.getText());
      String sql = "delete from  proveedor where idproveedor= "+id;
      st.executeUpdate(sql);
      TraerProveedor();
   } catch (Exception e) {
      System.out.println(e);
   }
```

G. Programar en el evento “MouseCliked” del jTable.

```
//MOSTRANDO DATOS DE TABLA EN CAJAS DE TEXTO
txtcod.setText(""+dtm.getValueAt(tablaproveedor.getSelectedRow(), 0) );
txtprov.setText(""+dtm.getValueAt(tablaproveedor.getSelectedRow(), 1) );
txtdir.setText(""+dtm.getValueAt(tablaproveedor.getSelectedRow(), 2) );
txtteefono.setText(""+dtm.getValueAt(tablaproveedor.getSelectedRow(), 3) );
```

H. Programar en el botón CONSULTAR POR CODIGO

```
try {
      st = cone.createStatement();
      //VARIABLES
      int id= Integer.parseInt( JOptionPane.showInputDialog("Ingrese Codigo"));
      String sql = "Select *  from  proveedor where idproveedor="+id;
      rs = st.executeQuery(sql);

        String datos[] = new String[4];
        //Limpiar Tabla
        int fila = dtm.getRowCount();
          if (fila>0)
                for (int i=0;i<fila;i++)
                  dtm.removeRow(0);
        //Leer Datos
        while(rs.next()){
          datos[0] = rs.getString(1);
          datos[1] = rs.getString(2);
          datos[2] = rs.getString(3);
          datos[3] = rs.getString(4);
          dtm.addRow(datos);
        }
} catch (Exception e) {
   System.out.println(e);
}
```

I. Probar la Aplicación.