**Clase #8**

SQL es un lenguaje declarativo que nos permite acceder, consultar y modificar los datos en bases de datos relacionales. Las principales BDD del mercado son: DB2 (IBM), Oracle (Oracle), SQL Server (Microsoft), MySql (junto con el paquete de Java fue adquirida por Oracle).

MySQL es un sistema de gestión de bases de datos relacionales de código abierto (RDBMS, por sus siglas en inglés) con un modelo cliente-servidor. [RDBMS](https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_bases_de_datos_relacionales) es un software o servicio utilizado para crear y administrar bases de datos basadas en un modelo relacional.

Para poder instalar MySQL que es la que vamos a ver aquí debemos entrar a las siguientes url dependiendo la versión del sistema operativo:

dev.mysql.com/downloads **=> MySQLInstaller for Windows.** => 32 bits <https://dev.mysql.com/downloads/mysql/> => 64 bits.

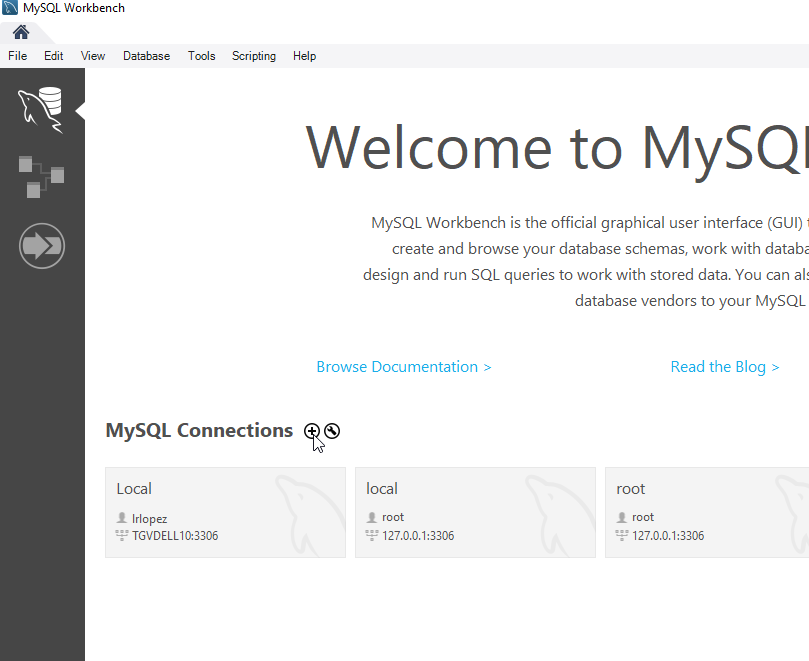
MYSQL + WORKBENCH



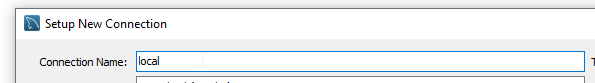
Y además de instalar mySql , también hay que instalar un **cliente** para poder interactuar con la BDD. En este caso utilizamos **mySQLWorkbench.**

La primera vez que se ingresa en MySQL Workbench hay que generar una conexión hacia el servidor de MySQL. Para eso se llevan a cabo los siguientes pasos:

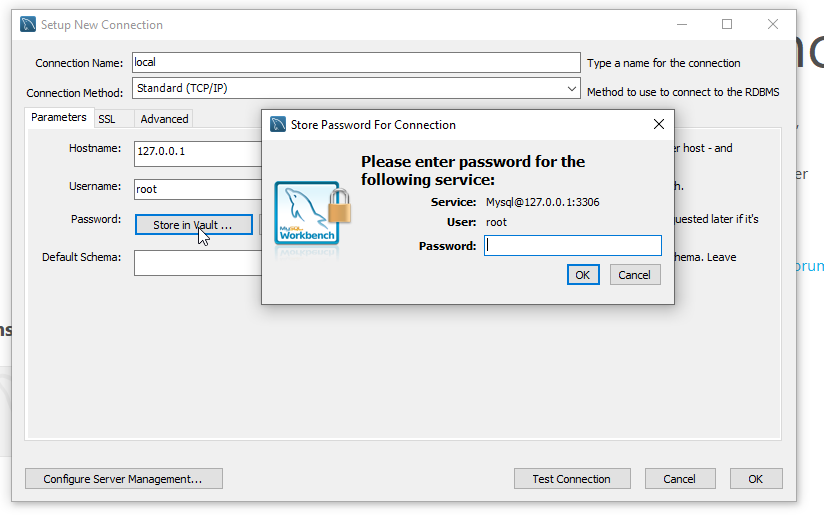
1. Seleccionar el icono +.



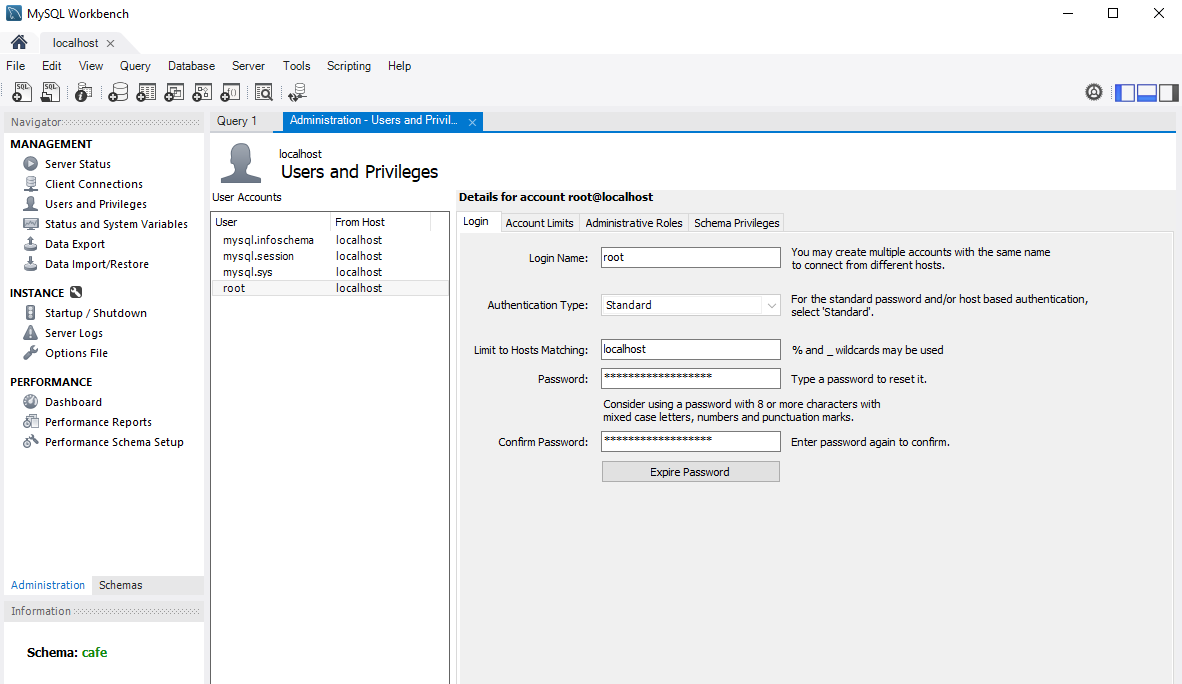
1. Ingresar un “connection name”, en este caso le ponemos local.



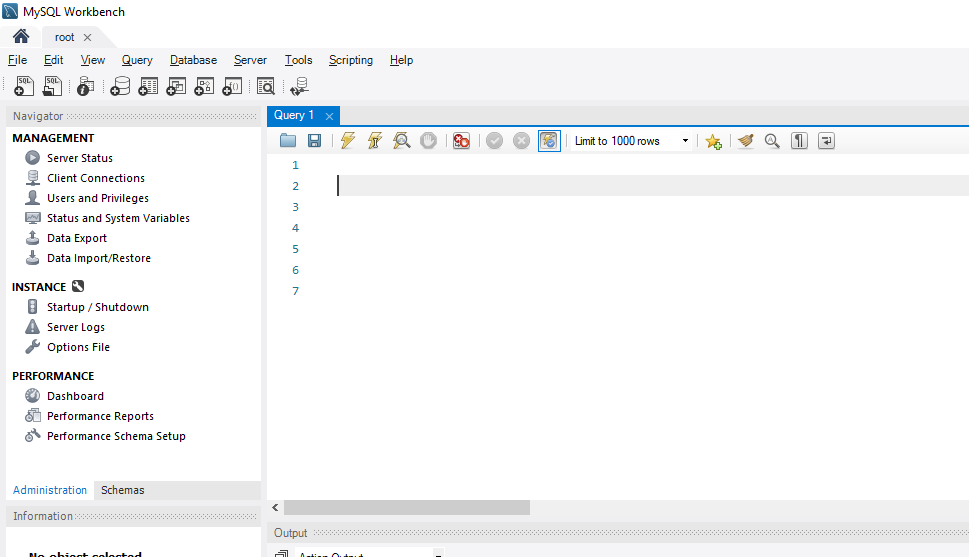
1. Ingresar una password:
2. Ir a “Store In vault”.
3. Ingresar la password que ingresaron en la instalación de MySQL.
4. Luego OK.



Para recuperar la password de Mysql si ya tienen habilitado el Workbench es:



1. Ingresar en la conexión:



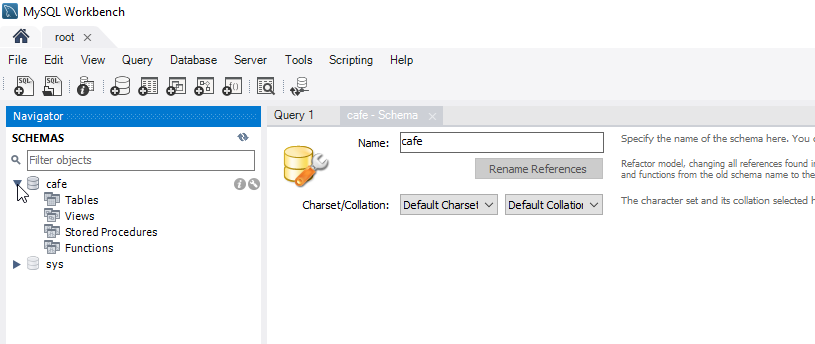
**Creamos Esquemas y tablas con MySQL**

**Creamos Esquema:**

El esquema es el concepto de una BDD.

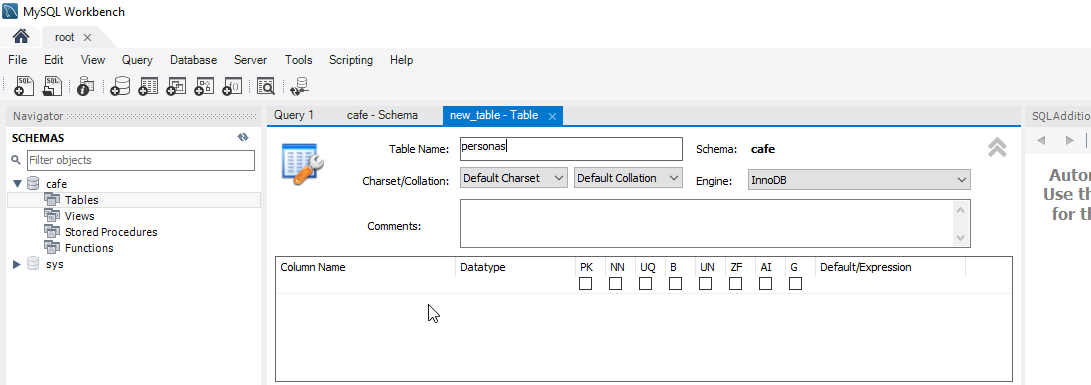
1. Botón derecho sobre “Schema”.
2. Le ponemos como nombre “café.

Nos queda:



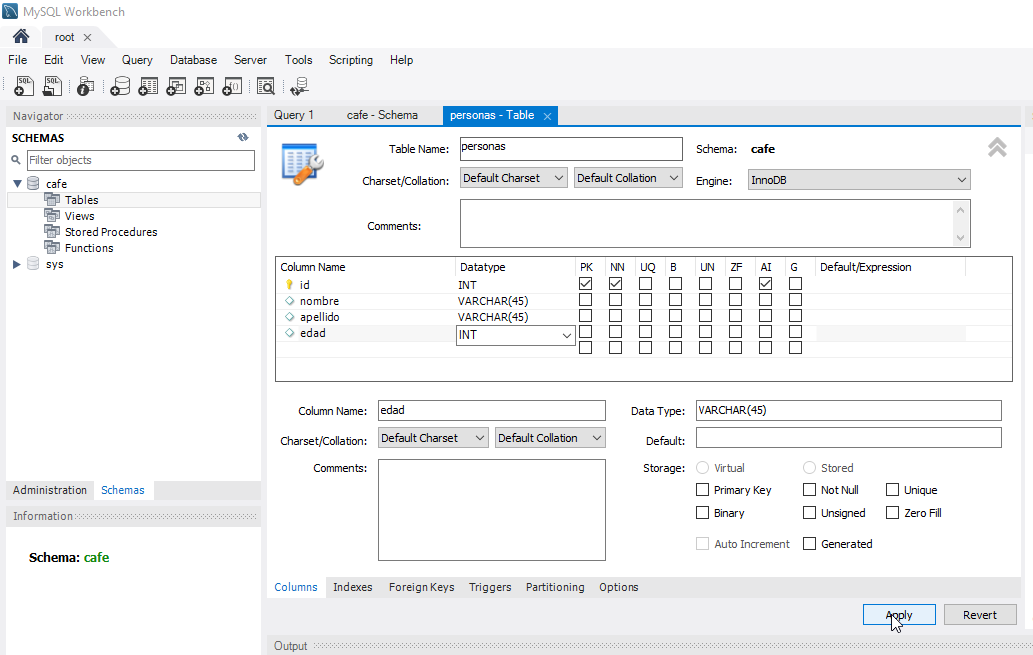
**Creamos Tabla:**

personas



Y dentro de la tabla personas le vamos a ingresar diversos tipos de datos.

Creamos 4 campos y luego “aplicar”.



**Comandos mas comunes dentro de SQL (Select, Insert, Delete, Update)**

**Insert:** Insertar datos.

**Select**: Seleccionar datos.

**Delete**: Borrar datos.

**Update**: Actualizar los datos.

**Vemos como funcionan los comandos básicos para impactar las tablas**

1. **Insert**

**Nota:** VARCHAR = STRING

insert into personas (nombre, apellido, edad) values ('Leo', 'Lopez', 40);

select \* from personas;

1. **Update**

update personas set edad=20 where id=1;

select \* from personas;

1. **Delete:**

delete from personas where id=2;

select \* from personas;

**mysql.jar (conjunto de clases compiladas) => es el connector sql que se setea en el proyecto**

Es necesario utilizarlo porque contiene la implementación del driver necesario para conectarse con la base de datos.

**CREAMOS CONEXIONES DESDE JAVA**

**JDBC** (api de java para trabajar con SQL)

Es el que permite la ejecución de operaciones sobre BDD relacionales desde Java.

Esta conformada por diversas clases e interfaces ubicadas en el paquete java.sql.

**Codigo Netbeans:**

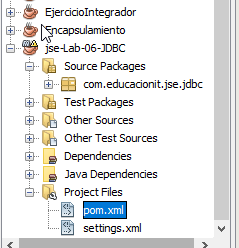


**Codigo Eclipse:**

****

Abrimos el paquete y nos ubicamos en la clase **AdministradorDeConexiones.java**.

Cuando se trabaja con JDBC hay que seleccionar una BDD.



Esta BDD además de tener el software instalado nos debe proveer de una librería para poder conectarnos a la misma.

**Para comenzar a COMPRENDER que es MAVEN se recomienda leer:**

1. Wikipedia => MAVEN.
2. Pagina oficial de MAVEN:

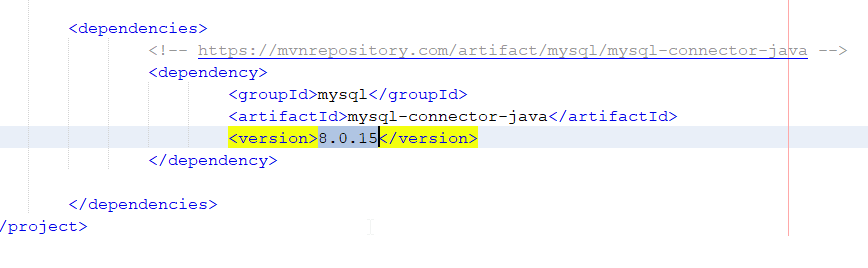
<http://maven.apache.org/what-is-maven.html>

1. El siguiente Word técnico sobre MAVEN.

DEV – QA – PRE-PROD (UAT) - PROD



En el POM.xml hay que configurar la versión de BDD con la que vamos a estar trabajando:



<version>8.0.20</version>

Luego Maven baja automáticamente las dependencias.

Vamos a mirar como esta compuesto el administrador de Conexiones. El cual es una clase que nos permite conectarnos con la BBDD.

Dentro de la clase **AdministradorDeConexiones**.java creo un método estatico que se llama obtenerConexion(). Este método debe recibir 4 datos.

1. **Clase driver**. Esta información nos la debe proveer el proveedor de la BDD. En este caso Oracle debe especificar cual es su clase driver.
2. **URL:** En nuestro caso se usa el **protocolo** jdbc:mysql. La **IP** es localhost. En algunas BDD se requiere también el puerto. En caso de MySql no es necesario por eso no se pone “:”. Y luego el **esquema** de la BBDD que me estoy conectando.
3. **Usuario.**
4. **Password.**
5. Comando **Class.forName** donde se le pasa el driver. Y esto disponibiliza el driver en la JVM.
6. Y la conexión se logra finalmente con el metodo **getConnection** del **DriverManager**. Que recibe la URL, el Usuario, y el password.

**PROBAMOS LA CONEXIÓN DESDE JAVA contra MySQL**

Previamente le indicamos al método ­**main**  que puede arrojar una excepción

…throws Exception

Importamos el paquete java.sql

Y luego agregamos el siguiente codigo.

Connection con = AdministradorDeConexiones.obtenerConexion();

if( ){

System.out.println("No estoy conectado");

} else{

System.out.println("Estoy conectado.");

}

Hacemos un par de modificaciones en la clase **AdministradorDeConexiones.java** para que nos funcione.

1. Colocar en la URL el serverTimezone. Es decir la zona horaria:

?serverTimezone=UTC //(al final de la URL)

1. Colocar la password de MySql en dbPassword. Y con eso debería andar.

Hay una forma que tiene JAVA de encarar las **clases** y las **conexiones** y la relación que tiene con las **BBDD**.

1 Tabla = 1 Clase

1 campo = 1 atributo

Se creo una entidad de tipo **Persona** que refleje exactamente todo lo que tiene mi tabla **persona** en la **BBDD**.

Para que un objeto de tipo **Persona** quede representado en mi BBDD es necesario crear un **DAO.**

**DAO - Data Access Object**

**Que es un DAO?**

Es un componente de software que suministra una [interfaz](https://es.wikipedia.org/wiki/Interfaz) común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una [Base de datos](https://es.wikipedia.org/wiki/Base_de_datos) o un [archivo](https://es.wikipedia.org/wiki/Archivo_(inform%C3%A1tica)).

**Relacion que se hace en los DAOs:**

“…por cada tabla, hay una clase y por campo de la tabla, hay un atributo en la clase…”

**Operaciones básicas**

Métodos: insertar(), modificar(), eliminar(), obtenerPorId(), etc

**Metodo createStatement()**

Se utiliza para crear un objeto que modela a una sentencia SQL

**Metodo executeQuery()**

Ejecuta una sentencia SQL y obtiene el resultado correspondiente de un objeto del tipo ResultSet. Es un conjunto de resultados como consecuencia de la sentencia SELECT.

**Codigo Clase06\_JDBC\_DAO**

**Dentro del main colocamos.**

//Indico si estoy conectado

Connection con = AdministradorDeConexiones.obtenerConexion();

if(con == null){

System.out.println("No estoy conectado");

} else{

System.out.println("Estoy conectado.");

}

//Defino la Conexion

Connection laConexion = AdministradorDeConexiones.obtenerConexion();

//Arma la consulta

Statement stmtConsulta = laConexion.createStatement();

//Ejecuta la consulta

ResultSet rs = stmtConsulta.executeQuery("SELECT \* FROM Personas");

//Muestra los datos

while(rs.next()){

System.out.println("\nID: "+rs.getInt("id")+"\nNombre: "+

rs.getString("nombre")+"\nApellido: "+rs.getNString("apellido")+

"\nEdad: "+rs.getInt("edad"));

}

//Cierra el Statement y la Connection

stmtConsulta.close();

laConexion.close();

**INSERTAR DATOS**

En el mismo codigo ponemos lo siguiente:

//INSERTAR DATOS

//Defino la Conexion

Connection laConexion = AdministradorDeConexiones.obtenerConexion();

//Arma la query

String query = "INSERT INTO personas(nombre, apellido, edad)VALUES('Juan', 'Santa Cruz', 33)";

Statement stmtInsercion = laConexion.createStatement();

//Ejecuta la consulta

stmtInsercion.execute(query);

//Cierra el Statement y la conexion

stmtInsercion.close();

laConexion.close();

**ACTUALIZAR DATOS**

//ACTUALIZAR DATOS

//Defino la conexion

Connection laConexion = AdministradorDeConexiones.obtenerConexion();

//Armar el UPDATE

String laActualizacion = "UPDATE personas SET id=2 WHERE nombre='Juan'";

Statement stmtActualizacion = laConexion.createStatement();

//Ejecuta el UPDATE

stmtActualizacion.execute(laActualizacion);

//Cierra el Statement y la conexion

stmtActualizacion.close();

laConexion.close();

**ELIMINAR DATOS**

//ELIMINAR DATOS

//Defino la Conexion

Connection laConexion = AdministradorDeConexiones.obtenerConexion();

//Armar la sentencia de eliminacion

String laEliminacion = "DELETE FROM personas WHERE Id=2";

Statement stmtEliminacion = laConexion.createStatement();

//Ejecutar la sentencia de Eliminacion

stmtEliminacion.execute(laEliminacion);

//Cierra el Statement y la Connection

stmtEliminacion.close();

laConexion.close();

**Ejemplo 2. Codigo “Clase06\_JDBC\_CONSULTA”**

Por ejemplo queremos consultar aquellos registros cuya edad sea menor a 21. En este caso corresponde al método getPersona(edad, signo, conexión).

Desde el workbench

select \* from personas where edad < 21;

Para eso adaptamos el método getPersona() pasandole 3 parametros: la edad, el signo matemático y la conexión. Luego nos devuelve un ArrayList el cual lo recibimos desde el main y mostramos su contenido.

**Codigo Netbeans:**



**Codigo Eclipse:**

****

**Error:**

En caso del siguiente error “Cannot run program "cmd" “ aplicar el siguiente workarround:

**https://www.youtube.com/watch?v=\_sVpzJJ9q4M**

**Ejemplo 3: CLASE\_8\_JDBC\_DAO\_VENTANA:**

**Codigo Netbeans:**



**Codigo Eclipse:**



1. Vemos como ingresar los datos con un **JOptionPane**. => INSERTAR

(Paquete mavenproject1).

1. Vemos como ingresar los datos con un **JFrameForm** => INSERTAR

(Paquete Auxiliar).

Ver la clase Formulario.java la cual contiene el siguiente codigo:

package Auxiliar;

import java.util.logging.Level;

import java.util.logging.Logger;

import com.mycompany.mavenproject1.Persona;

import com.mycompany.mavenproject1.PersonaDAO;

/\*\*

\*

\* @author lrlopez

\*/

public class Formulario extends javax.swing.JFrame {

String nombre1;

/\*\*

\* Creates new form Ventana

\*/

public Formulario() {

initComponents();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jLabel1 = new javax.swing.JLabel();

jButton1 = new javax.swing.JButton();

nombre = new javax.swing.JTextField();

jLabel2 = new javax.swing.JLabel();

jTextField1 = new javax.swing.JTextField();

jTextField2 = new javax.swing.JTextField();

jLabel3 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

jLabel1.setText("Nombre");

jLabel1.setToolTipText("");

jButton1.setText("OK");

jButton1.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jButton1MouseClicked(evt);

}

});

jLabel2.setText("Apellido:");

// jTextField1.setText("jTextField1");

// jTextField2.setText("jTextField2");

jLabel3.setText("Edad: ");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(49, 49, 49)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jButton1)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 53, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, 47, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel3))

.addGap(71, 71, 71)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(nombre, javax.swing.GroupLayout.DEFAULT\_SIZE, 104, Short.MAX\_VALUE)

.addComponent(jTextField1)

.addComponent(jTextField2))))

.addContainerGap(123, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(60, 60, 60)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 30, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(nombre, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, 30, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel3,javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

.addComponent(jButton1)

.addContainerGap(103, Short.MAX\_VALUE))

);

pack();

}// </editor-fold>

private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {

nombre1 = nombre.getText();

String apellido1 = jTextField1.getText();

String edad1 = jTextField2.getText();

int edad2 = Integer.valueOf(edad1);

Persona pepe1 = new Persona(nombre1, apellido1, edad2);

nombre.setText("");

jTextField1.setText("");

jTextField2.setText("");

try {

PersonaDAO.inserta(pepe1);

} catch (Exception ex) {

ex.printStackTrace();

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Formulario.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Formulario.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Formulario.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Formulario.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new Formulario().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JTextField jTextField1;

private javax.swing.JTextField jTextField2;

private javax.swing.JTextField nombre;

// End of variables declaration

}