

Manual Tecnico

PROYECTO VETERINARIA



Presentado por

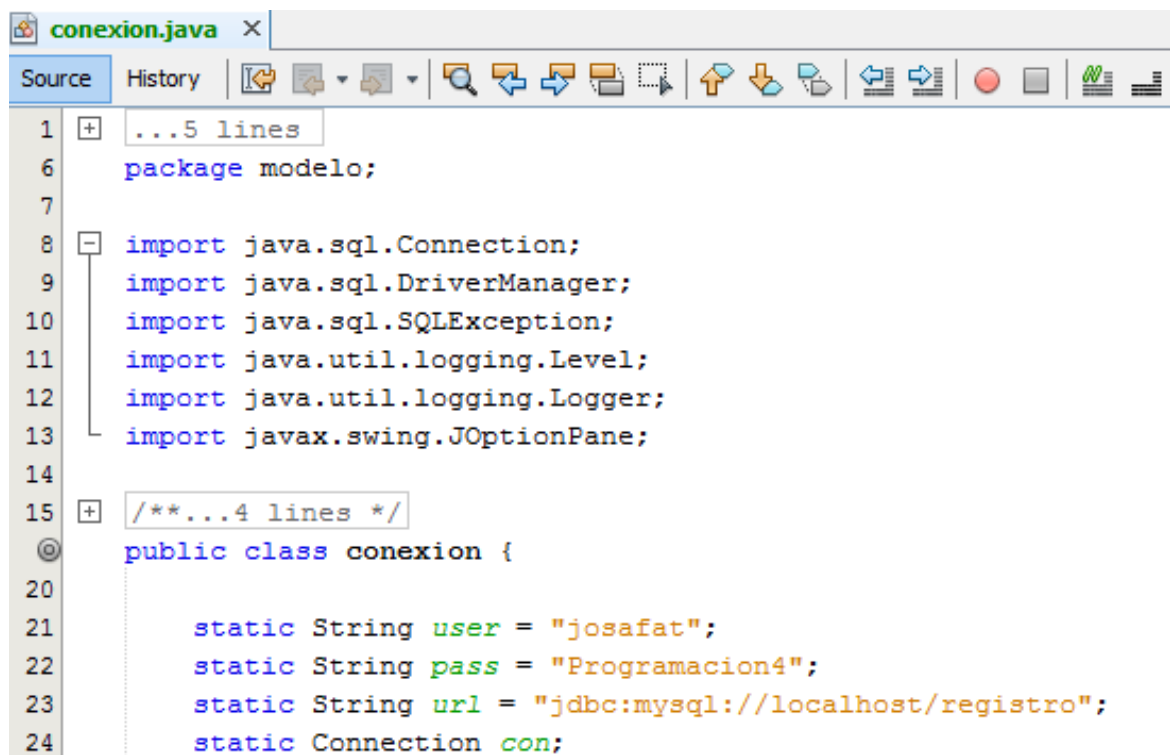
CESIA COTO,
NORMAN BARRIENTOS,
WALTER FLORES,
JOCELYN CLAVEL

Conexión

Como primer paso para empezar el desarrollo del proyecto se tiene la conexión, clase que nos permite tener relación con nuestra base de datos antes creada.

Debemos de estar seguro de tener bien el acceso con las credenciales convenientes para poder echar a andar la conexión sin ningún tipo de fallo.

importante llamar las librerías necesarias para la clase y los metodos que se ocupan.



```
conexion.java x
Source History
1 ...5 lines
6 package modelo;
7
8 import java.sql.Connection;
9 import java.sql.DriverManager;
10 import java.sql.SQLException;
11 import java.util.logging.Level;
12 import java.util.logging.Logger;
13 import javax.swing.JOptionPane;
14
15 /**...4 lines */
16 public class conexion {
17
18     static String user = "josafat";
19     static String pass = "Programacion4";
20     static String url = "jdbc:mysql://localhost/registro";
21     static Connection con;
```

Conexión

Como primer paso para empezar el desarrollo del proyecto se tiene la conexión, clase que nos permite tener relación con nuestra base de datos antes creada.

Debemos de estar seguro de tener bien el acceso con las credenciales convenientes para poder echar a andar la conexión sin ningún tipo de fallo.

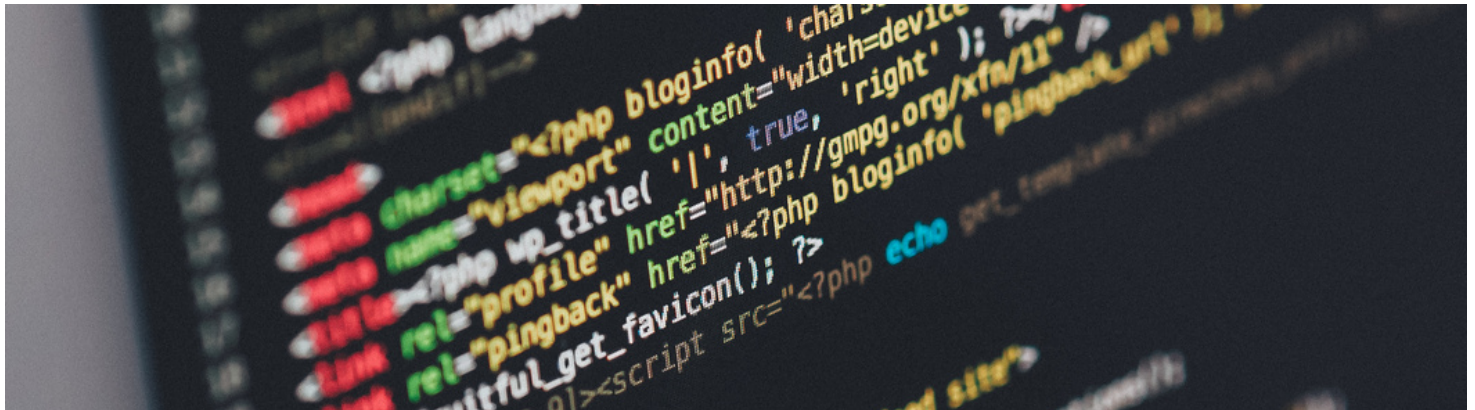
importante llamar las librerías necesarias para la clase y los métodos que se ocupan.

Se relaiza la conexión con el driver de MYSQL el cual identifica nuestras credenciales, url, usuario y contraseña de contar con una.

```
public static Connection Conectar() { //REALIZAMOS LA CONEXION LLAMANDO NUESTRAS CREDENCIALES
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection(url, user, pass);
    } catch (ClassNotFoundException cnfex) {
        JOptionPane.showMessageDialog(null, "ConexionMySQL:" + cnfex.getMessage());
    } catch (SQLException sqllex) {
        JOptionPane.showMessageDialog(null, "ConexionMySQL:" + sqllex.getMessage());
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(null, "ConexionMySQL:" + ex.getMessage());
    }
    return con;
}

public void cerrarConexion() { //CERRAMOS LA CONEXION
    try {
        con.close();
    } catch (SQLException ex) {
        Logger.getLogger(conexion.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Registrar usuarios



Para generar el registro desde el frame se necesita programar una acción en un botón que se encargue de guardar dicho registro en nuestra base de datos y se hizo de la siguiente manera.

```
SQLUsuarios modsq1 = new SQLUsuarios();
usuarios mod = new usuarios();

String pass = new String(txtPassword.getPassword());
String passCon = new String(txtConfirmarPassowrd.getPassword());

if (txtUsuario.getText().equals("") || pass.equals("") || passCon.equals("") ||
    txtNombre.getText().equals("") || txtCorreo.getText().equals(""))
{
    JOptionPane.showMessageDialog(null, "hay campos vacios, debe llenar todo");
}
else {
    if (pass.equals(passCon)) {
        if (modsq1.existeUsuario(txtUsuario.getText()) == 0) {

            String nuevoPass = hash.sha1(pass);

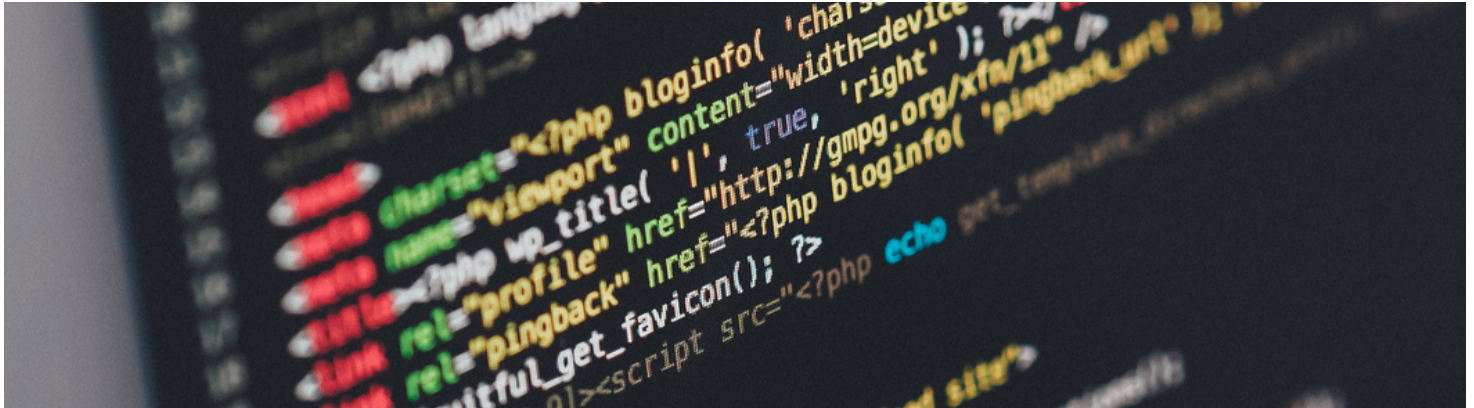
            mod.setUsuario(txtUsuario.getText());
            mod.setContraseña(nuevoPass);
            mod.setNombre(txtNombre.getText());
            mod.setCorreo(txtCorreo.getText());
            mod.setIdRoles(1);

            if (modsq1.registrar(mod)) {
                JOptionPane.showMessageDialog(null, "registro se guardo");
            }
        }
    }
}
```

Creamos nuestro objeto usuario el cual hace referencia a nuestra tabla en el SQL es decir nuestra base de datos.

Nuestros metodos getUsuario y getPassword son los que necesitamos para registrarnos, el programa identifica estas credenciales y se cerciora de que todo este segun lo pedido para poder guardar nuestro nuevo registro dentro de la base de datos.

Login usuarios



Una vez se almacenan los datos en la tabla lo siguiente es entrar y probar ingresar al programa con los datos antes registrados, para eso vamos al Login.

```
String pass = new String(txtPassword.getPassword());

if(!txtUsuario.getText().equals("") && !pass.equals("")) //encaso que
{
    String nuevoPass = hash.sha1(pass);

    mod.setUsuario(txtUsuario.getText());
    mod.setContraseña(nuevoPass);

    if(modsql.login(mod)) {
        Inicio.frmLog = null;
        this.dispose(); //cerramos ventana de login pero que continúe
```

Acá simplemente se comprueba que existan los datos almacenados, si el programa reconoce entonces deja pasar al usuario, si no el programa imprime un mensaje de error que puede ser: "Usuario o contraseña incorrecta" o "Usuario y contraseña no existen".

Además se cuenta con un método hash el cual su función es encriptar nuestra contraseña para que de manera segura nadie pueda intentar robar nuestros datos de ingreso.

```
public class hash {
    /* Retorna un hash a partir de un tipo y un texto */
    public static String getHash(String txt, String hashType) {
        try {
            java.security.MessageDigest md = java.security.MessageDigest.getInstance(hashType);
            byte[] array = md.digest(txt.getBytes());
            StringBuffer sb = new StringBuffer();
            for (int i = 0; i < array.length; ++i) {
                sb.append(Integer.toHexString((array[i] & 0xFF) | 0x100).substring(1, 3));
            }
            return sb.toString();
        } catch (java.security.NoSuchAlgorithmException e) {
            System.out.println(e.getMessage());
        }
        return null;
    }
}
```

Registros

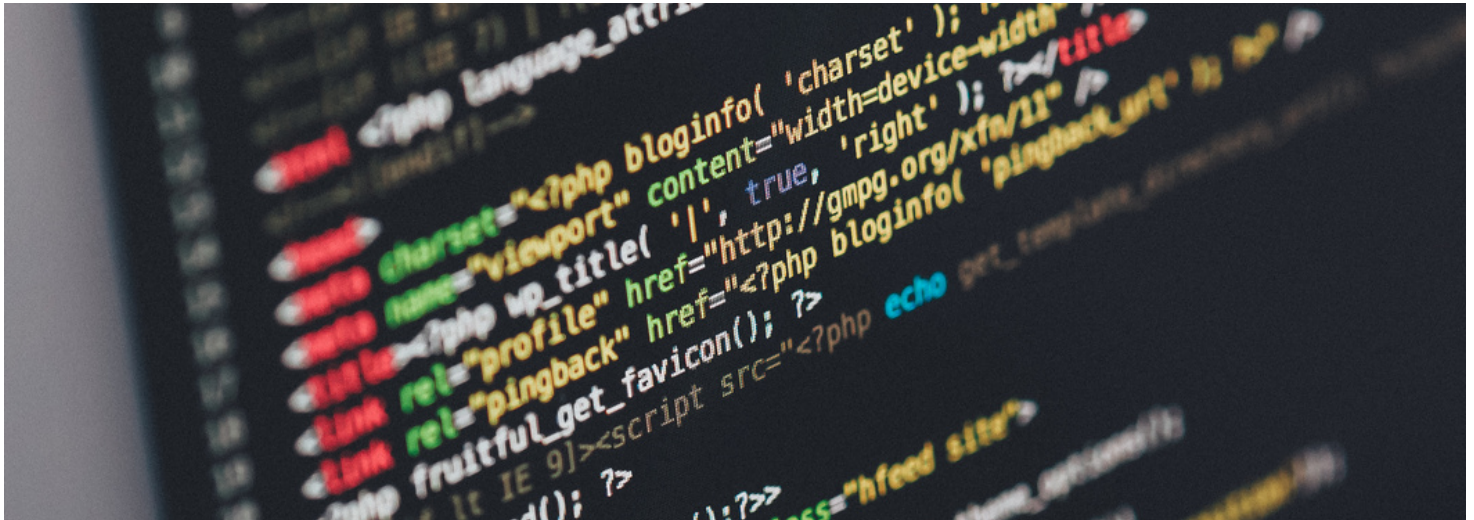
Al registrar el cliente se utilizó los procedimientos almacenados de SQL y los métodos. Ingresar, Modificar, Eliminar como se presenta a continuación.

Esto quiere decir que los datos son cargados desde la base de datos y al ingresarlos estos ya se almacenan como registros.

```
private void btnIngresarActionPerformed(java.awt.event.ActionEvent evt) {  
    String resultado = null;  
    resultado = user.IngresarDatos(  
        Integer.parseInt(txtCodigo.getText()),  
        txtNombre.getText(),  
        txtApellido.getText(),  
        txtEmail.getText(),  
        txtTelefono.getText(),  
        txtDireccion.getText());  
    CargarDatosLocal();  
    JOptionPane.showMessageDialog(null, resultado);  
}
```

```
private void btnModificarActionPerformed(java.awt.event.ActionEvent evt) {  
    String resultado = null;  
    resultado = user.ModificarDatos(  
        Integer.parseInt(txtCodigo.getText()),  
        txtNombre.getText(),  
        txtApellido.getText(),  
        txtEmail.getText(),  
        txtTelefono.getText());  
    CargarDatosLocal();  
    JOptionPane.showMessageDialog(null, resultado);  
}
```

```
private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {  
    String resultado = null;  
    resultado = user.EliminarDatos(Integer.parseInt(txtCodigo.getText()));  
    CargarDatosLocal();  
    JOptionPane.showMessageDialog(null, resultado);  
}
```



Los métodos cargar, ingresar, modificar y eliminar están dentro de la clase "Usuarios" en donde también se encuentran los métodos set y get

```
public DefaultTableModel CargarDatos() { ...25 lines }

public String IngresarDatos(int codigo, String nombres

public String ModificarDatos(int codigo, String nombre

public String EliminarDatos(int codigo) { ...18 lines }

public String getInsertar() { ...3 lines }

public void setInsertar(String Insertar) { ...3 lines }

public String getModificar() { ...3 lines }

public void setModificar(String Modificar) { ...3 lines }

public String getEliminar() { ...3 lines }
```