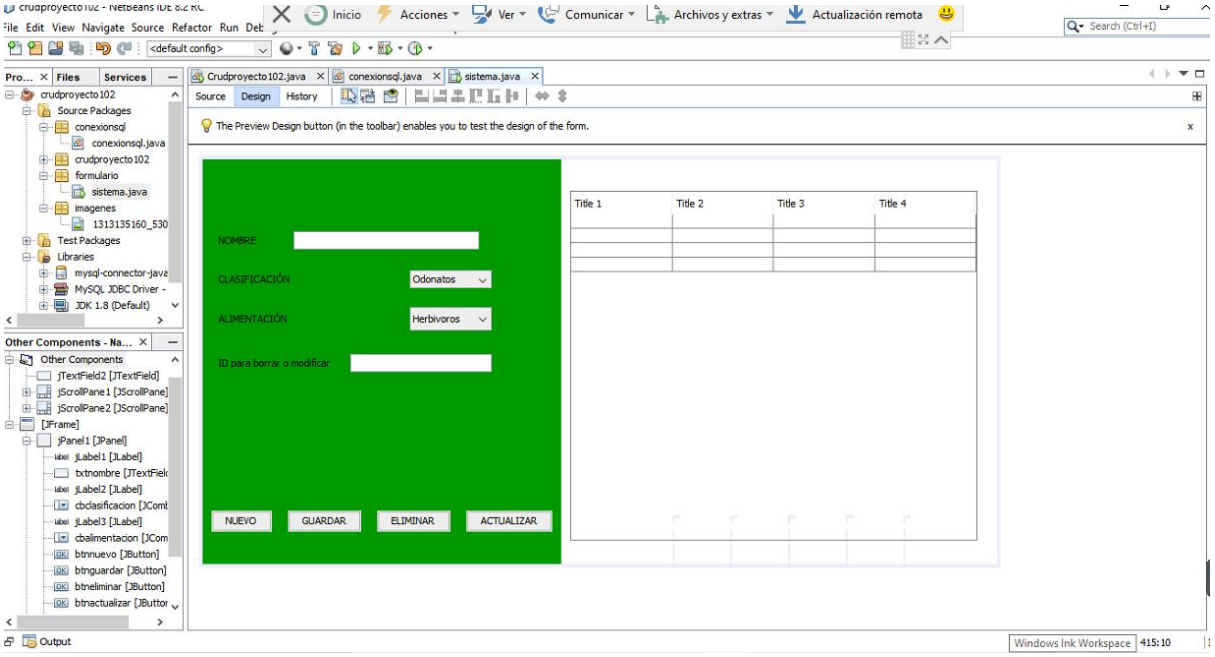
Documentación: Reporte del proyecto individual

Adriana Ivette Valadez Squivias

El IDE utilizado es NetBeans y se desarrolla la interfaz gráfica



El método main es el siguiente:

```
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)

    /* Create and display the form */

    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                new sistema().setVisible(true);

            } catch (SQLException ex) {
                Logger.getLogger(sistema.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    });
}
```

El cual nos muestra la interfaz gráfica de nuestra aplicación.

El método *mostrarDatos()* es importante ya que se llama cada vez que se ejecuta alguna acción sobre la base de datos y su función es actualizar la interfaz gráfica.

```
public void mostrarDatos() throws SQLException{

  String[] titulos={"ID","nombre","clasificación", "alimentación"};
  String[] registros=new String[4];


  DefaultTableModel modelo=new DefaultTableModel(null,titulos);

  String SQL="select * from insectos";

    try {
        Statement at=con.createStatement();
        ResultSet rs=at.executeQuery(SQL);

        while (rs.next()){

        registros[0]=rs.getString("idinsectos");
        registros[1]=rs.getString("nombre");
        registros[2]=rs.getString("clasificación");
        registros[3]=rs.getString("alimentación");

        modelo.addRow(registros);

        }

        tablainsectos.setModel(modelo);

      }catch (Exception e) {
```

Botón de nuevo

se utiliza la función de *limpiarCajas* para que al agregar un nuevo elemento en la base de datos lo muestre con elementos designados

```
public void limpiarCajas(){

    txtnombre.setText("Insecto");
    cbclasificacion.setSelectedItem(1);
    cbalimentacion.setSelectedItem(1);
    idborrar.setText("0");
```

Guardar datos
La función ligada al botón de agregar dato es:

```
private void btnguardarActionPerformed(java.awt.event.ActionEvent evt) {
    insertarDatos();
    limpiarCajas();
    try {
        mostrarDatos();
    } catch (SQLException ex) {
        Logger.getLogger(sistema.class.getName()).log(Level.SEVERE, null, ex);
    }


}
```

La cual llama el método *insertarDatos()* posteriormente limpia las entradas del usuario y actualiza la interfaz con el método *mostrarDatos()* explicado anteriormente.

El método *insertarDatos()* contiene el query de SQL capaz de insertar los datos en la tabla correspondiente:

```
public void insertarDatos(){
    try {
        int seleccionado=cbclasificacion.getSelectedIndex();
        int seleccionadol=cbalimentacion.getSelectedIndex();
        String SQL="INSERT INTO `insectos`(`nombre`, `clasificación`, `alimentación`) VALUES (?,?,?)";

        PreparedStatement pst=con.prepareStatement(SQL);

        pst.setString(1,txtnombre.getText());


        pst.setString(2,cbclasificacion.getItemAt(seleccionado));


        pst.setString(3,cbalimentacion.getItemAt(seleccionadol));

        pst.execute();

        JOptionPane.showMessageDialog(null,"Registro Exitoso");



    } catch (Exception e) {
        JOptionPane.showMessageDialog(null,"Error de Registro" +e.getMessage());
    }
}
```

Eliminar Datos:

La siguiente función se llama al presionar el botón de eliminar, la cual sigue una lógica similar al del botón de agregar dato.

```
private void btneliminarActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        eliminarRow();
        mostrarDatos();
    } catch (SQLException ex) {
        Logger.getLogger(sistema.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

La función de *eliminarRow()* contiene el query capaz de eliminar un dato a partir de su ID el cual se recupera de la interfaz gráfica.

```
public void eliminarRow() throws SQLException{

    String SQL="DELETE FROM `insectos` WHERE `idinsectos`=?";
    PreparedStatement pst=con.prepareStatement(SQL);
    pst.setString(1,idborrar.getText());
    pst.execute();

}
```

Modificar datos

La siguiente función se llama al hacer click en el botón de modificar:

```
private void btnactualizarActionPerformed(java.awt.event.ActionEvent evt) {
    updateRow();
    try {
        mostrarDatos();
    } catch (SQLException ex) {
        Logger.getLogger(sistema.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

La cual llama la función *updateRow()* la cual tiene una lógica similar a la de guardar una entrada nueva con la única diferencia del query al ser un UPDATE en lugar de un INSERT y agregar el filtro por ID.

```
public void updateRow(){
     try {
        int seleccionado=cbclasificacion.getSelectedIndex();
        int seleccionadol=cbalimentacion.getSelectedIndex();
        String SQL="UPDATE `insectos` SET `nombre`=?,`clasificación`=?,`alimentación`=

        PreparedStatement pst=con.prepareStatement(SQL);

        pst.setString(1,txtnombre.getText());


        pst.setString(2,cbclasificacion.getItemAt(seleccionado));


        pst.setString(3,cbalimentacion.getItemAt(seleccionadol));

        pst.setString(4,idborrar.getText());

        pst.execute();

        JOptionPane.showMessageDialog(null,"Modificación Exitoso");



    } catch (Exception e) {
        JOptionPane.showMessageDialog(null,"Error de Registro" +e.getMessage());
    }
```