CÓDIGO: Mostrar Datos

```
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
public class MostrarDatos {
  public static void mostrarDatos(DefaultTableModel model) {
     String sql = "SELECT fruta, precio FROM frutas WHERE precio IS NOT NULL";
     try (Connection conn = ConexionBD.getConnection();
        PreparedStatement pstmt = conn.prepareStatement(sql);
        ResultSet rs = pstmt.executeQuery()) {
       model.setRowCount(0); // Limpiar la tabla antes de agregar nuevos datos
       while (rs.next()) {
         Object[] rowData = {
              rs.getString("fruta"),
              rs.getDouble("precio")
         };
          model.addRow(rowData);
       }
    } catch (SQLException e) {
       JOptionPane.showMessageDialog(null, "Error al recuperar datos de la base de
datos: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
}
```

CÓDIGO: Calcular la Relación de Frutas

```
private void calcularRelacion() {
  // Obtener el precio del plátano del JTextField
  double precioPlatano = Double.parseDouble(jTextField1.getText());
  // Variables para la conexión a la base de datos
  String url = "jdbc:postgresql://oregon-postgres.render.com:5432/sisinfo2db";
  String user = "kevin_123";
  String password = "8baO6G0RmY0HjY2PqSrScxaR3p4al3We";
  // Consulta SQL para obtener los precios de las otras frutas
  String sql = "SELECT fruta, precio FROM frutas WHERE fruta <> 'Plátano' AND precio
>=?";
  try (Connection conn = DriverManager.getConnection(url, user, password);
     PreparedStatement pstmt = conn.prepareStatement(sql)) {
    // Establecer el parámetro de precio del plátano en la consulta SQL
     pstmt.setDouble(1, precioPlatano);
     try (ResultSet rs = pstmt.executeQuery()) {
       // Limpiar la tabla antes de agregar nuevos datos
       DefaultTableModel model = (DefaultTableModel) jTable2.getModel();
       model.setRowCount(0);
       // Iterar sobre los resultados y calcular la relación con el precio del plátano
       while (rs.next()) {
          String fruta = rs.getString("fruta");
         double precioFruta = rs.getDouble("precio");
         double relacion = precioFruta / precioPlatano;
```

```
int relacion_entero = Math.round((float) relacion);
         model.addRow(new Object[]{fruta, "vale " + relacion_entero + " veces más"});
       }
    }
  } catch (SQLException e) {
    JOptionPane.showMessageDialog(this, "Error al recuperar datos de la base de datos:
" + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
}
CÓDIGO: Cargar Datos de la tabla escoger la fruta para las
interacciones
private void cargarFrutasEnComboBox() {
  try {
     Connection con = DriverManager.getConnection("jdbc:postgresql://oregon-
postgres.render.com:5432/sisinfo2db", "kevin_123",
"8baO6G0RmY0HjY2PqSrScxaR3p4al3We");
    String query = "SELECT fruta FROM frutas where 1=1";
    PreparedStatement pst = con.prepareStatement(query);
     ResultSet rs = pst.executeQuery();
    while (rs.next()) {
       String nombreFruta = rs.getString("fruta");
      // Imprime el nombre de la fruta
       ¡ComboBox1.addItem(nombreFruta);
    }
    con.close();
  } catch (SQLException ex) {
    ex.printStackTrace();
  }
```

}

CÓDIGO: Actualizar para ordenar los elementos.

```
import javax.swing.table.DefaultTableModel;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
public class Actualizar {
  public static void actualizarTabla(DefaultTableModel model) {
     model.setRowCount(0); // Limpiar la tabla
     try {
       Connection con = ConexionBD.getConnection();
       String query = "SELECT fruta, precio FROM frutas ORDER BY precio";
       PreparedStatement pst = con.prepareStatement(query);
       ResultSet rs = pst.executeQuery();
       while (rs.next()) {
          Object[] row = {
               rs.getString("fruta"),
               rs.getDouble("precio")
          };
          model.addRow(row);
       }
       ConexionBD.close(con, pst, rs);
     } catch (SQLException ex) {
       ex.printStackTrace();
     }
}
```

CÓDIGO: Eliminar los datos

```
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
public class EliminarDatos {
  public static void eliminarDatos(DefaultTableModel model, int selectedRow) {
     if (selectedRow != -1) {
       try (Connection con = ConexionBD.getConnection()) {
          String query = "DELETE FROM frutas WHERE fruta = ? AND precio = ?";
          PreparedStatement pst = con.prepareStatement(query);
         pst.setString(1, (String) model.getValueAt(selectedRow, 0)); // Suponiendo que
la primera columna es el nombre de la fruta
         pst.setDouble(2, (Double) model.getValueAt(selectedRow, 1)); // Suponiendo
que la segunda columna es el precio de la fruta
         pst.executeUpdate();
          model.removeRow(selectedRow);
          JOptionPane.showMessageDialog(null, "Fruta eliminada correctamente.");
       } catch (SQLException ex) {
          JOptionPane.showMessageDialog(null, "Error al eliminar la fruta: " +
ex.getMessage());
       }
    } else {
       JOptionPane.showMessageDialog(null, "Por favor, selecciona una fila para
eliminar.");
    }
  }
}
```

CÓDIGO: Modificar

```
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
public class Modificar {
  public static void modificarFruta(DefaultTableModel model, int filaSeleccionada, String
nuevoNombreFruta, double nuevoPrecioFruta) {
     String nombreFrutaSeleccionada = (String) model.getValueAt(filaSeleccionada, 0);
     // Verificar si el nombre de la fruta seleccionada es igual al nuevo nombre de la fruta
     if (!nombreFrutaSeleccionada.equals(nuevoNombreFruta)) {
       try {
          Connection con = ConexionBD.getConnection();
          // Consulta para verificar si el nuevo nombre de fruta ya existe
          String queryVerificar = "SELECT COUNT(*) FROM frutas WHERE fruta = ?";
          PreparedStatement pstVerificar = con.prepareStatement(queryVerificar);
          pstVerificar.setString(1, nuevoNombreFruta);
          ResultSet rs = pstVerificar.executeQuery();
          // Verificar si hay algún resultado en la consulta
          rs.next();
          int count = rs.getInt(1);
          if (count > 0) {
            JOptionPane.showMessageDialog(null, "El nombre de la fruta ya existe en la
base de datos.");
```

```
return;
         }
       } catch (SQLException ex) {
          JOptionPane.showMessageDialog(null, "Error al verificar la fruta en la base de
datos: " + ex.getMessage());
         return;
       }
    }
    // Si el nombre de la fruta seleccionada es igual al nuevo nombre de la fruta
    // o si el nuevo nombre de la fruta no existe en la base de datos, procedemos con la
modificación
     try {
       Connection con = ConexionBD.getConnection();
       // Consulta para actualizar la fruta
       String queryUpdate = "UPDATE frutas SET fruta = ?, precio = ? WHERE fruta = ?";
       PreparedStatement pst = con.prepareStatement(gueryUpdate);
       pst.setString(1, nuevoNombreFruta);
       pst.setDouble(2, nuevoPrecioFruta);
       pst.setString(3, nombreFrutaSeleccionada); // Utilizamos el nombre de la fruta
seleccionada como criterio de búsqueda
       pst.executeUpdate();
       // Actualizar la tabla mostrando la fruta modificada
       model.setValueAt(nuevoNombreFruta, filaSeleccionada, 0); // Suponiendo que el
nombre de la fruta está en la primera columna
       model.setValueAt(nuevoPrecioFruta, filaSeleccionada, 1); // Suponiendo que el
precio de la fruta está en la segunda columna
       JOptionPane.showMessageDialog(null, "Fruta modificada correctamente.");
     } catch (SQLException ex) {
```

```
JOptionPane.showMessageDialog(null, "Error al modificar la fruta: " + ex.getMessage());
}
}
```

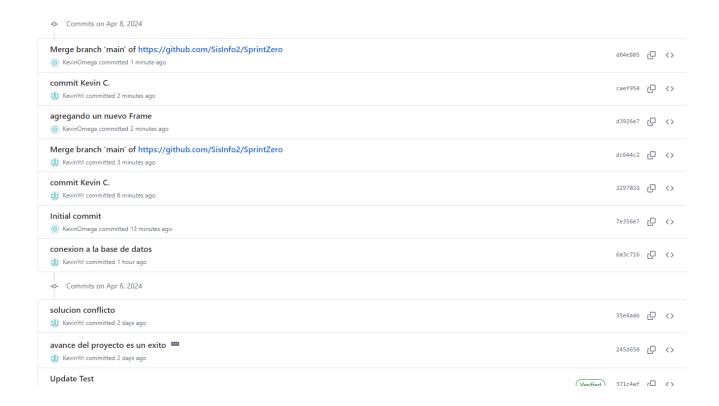
CÓDIGO: Para los bonotes de NetBeans

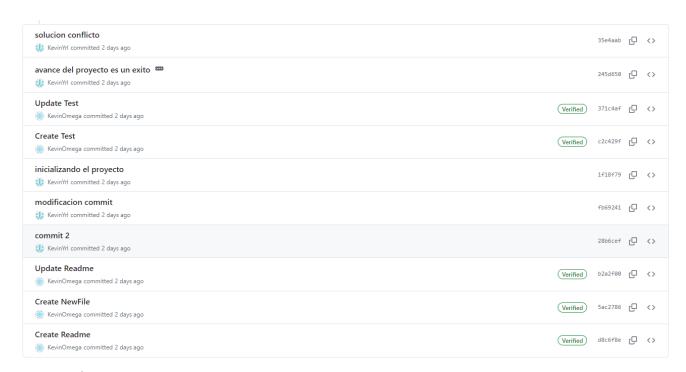
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
   MostrarDatos.mostrarDatos((DefaultTableModel) jTable1.getModel());
  }
  private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    calcularRelacion();
  }
  private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
  int selectedRow = jTable1.getSelectedRow();
  EliminarDatos.eliminarDatos((DefaultTableModel) jTable1.getModel(), selectedRow);
  }
  private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    String nombreFruta = jTextField2.getText();
  double precioFruta = Double.parseDouble(jTextField3.getText());
  Agregar.agregarFruta((DefaultTableModel) jTable1.getModel(), nombreFruta,
precioFruta);
  }
```

```
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
     int filaSeleccionada = jTable1.getSelectedRow();
  if (filaSeleccionada == -1) {
    JOptionPane.showMessageDialog(null, "Por favor, selecciona una fila para
modificar.");
    return:
  }
  String nuevoNombreFruta = jTextField2.getText();
  double nuevoPrecioFruta;
  try {
    nuevoPrecioFruta = Double.parseDouble(jTextField3.getText());
  } catch (NumberFormatException ex) {
    JOptionPane.showMessageDialog(null, "Por favor, ingresa un valor numérico válido
para el precio.");
    return;
  }
  DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
  Modificar.modificarFruta(model, filaSeleccionada, nuevoNombreFruta,
nuevoPrecioFruta);
  }
  private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
    cargarFrutasEnComboBox();
  }
  private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
    Actualizar.actualizarTabla((DefaultTableModel) jTable1.getModel());
```

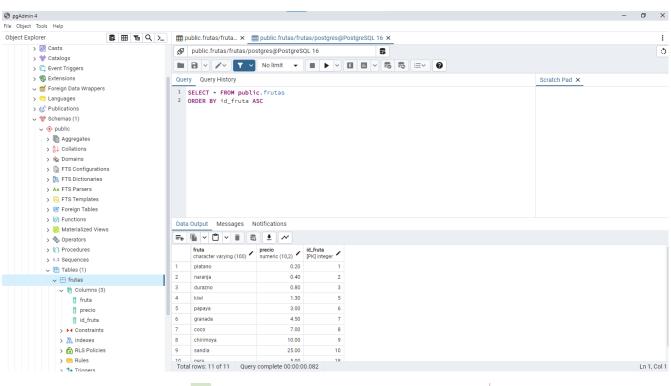
```
private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
   guardarCambios();
}
```

COMMMIT REALIZADOS:





CONEXIÓN A LA BASE DE DATOS:



private static final String URL = "jdbc:postgresql://oregon-postgres.render.com:5432/sisinfo2db";
private static final String USER = "kevin_123";
private static final String PASSWORD = "8baO6GORmYOHjY2PqSrScxaR3p4aI3We";

ESTADO DEL GITHUB(MAIN) CON LOS COMMITS REALIZADOS

