**DESARROLLO DE INTERFACES**

**Creación y Validación de formularios**

**public FormularioPrincipal() {**

initComponents();

this.setTitle("Clínica veterinaria 'El Ángel'");

this.setLocationRelativeTo(null);

this.jButtonAnadir.setBackground(new Color(34, 139, 34));

this.jButtonAnadir.setForeground(Color.WHITE);

this.jButtonAnadir.setContentAreaFilled(true);

this.jButtonSalir.setBackground(new Color(220, 20, 60));

this.jButtonSalir.setForeground(Color.WHITE);

this.jButtonSalir.setContentAreaFilled(true);

**}**

**private void confirmarSalida() {**

String[] opciones = {"Sí", "No"};

int resultado = JOptionPane.showOptionDialog(

null,

"¿Estás segur@ que deseas salir de la aplicación?",

"Información de la aplicación",

JOptionPane.DEFAULT\_OPTION,

JOptionPane.INFORMATION\_MESSAGE,

null,

opciones,

opciones[0] // Botón por defecto

);

switch (resultado) {

case 0: // Índice del botón "Sí"

this.dispose();

break;

case 1: // Índice del botón "No"

System.out.println("Operación cancelada");

break;

default: // Caso en que se cierra el diálogo

break;

}

**}**

**private void jButtonAnadirActionPerformed(java.awt.event.ActionEvent evt) {**

// TODO add your handling code here:

DialogoNuevoCliente nuevoCliente = new DialogoNuevoCliente(this, rootPaneCheckingEnabled);

nuevoCliente.setVisible(true);

**}**

**public DialogoNuevoCliente(java.awt.Frame parent, boolean modal) {**

super(parent, modal);

initComponents();

this.setTitle("Añadir nuevo cliente");

this.setLocationRelativeTo(null);

// Crear el modelo del Spinner (SpinnerDateModel)

SpinnerDateModel dateModel = new SpinnerDateModel();

dateModel.setValue(new Date()); // Establece la fecha actual como valor inicial

// Asignar el modelo al JSpinner (jSpinner1)

jSpinner1.setModel(dateModel);

// Configurar el editor del Spinner con el formato deseado

JSpinner.DateEditor dateEditor = new JSpinner.DateEditor(jSpinner1, "dd/MM/yyyy");

jSpinner1.setEditor(dateEditor);

**}**

**private void guardar() {**

StringBuilder errores = new StringBuilder(); // Para acumular los mensajes de error

// Verificar que el campo "NombreCliente" no esté vacío y sea alfabético

if (jTextFieldNombreCliente.getText().trim().isEmpty()

|| !esAlfabetico(jTextFieldNombreCliente.getText())) {

errores.append("El nombre del cliente debe ser un texto alfabético.\n");

}

// Verificar que el campo "PrimerApe" no esté vacío y sea alfabético

if (jTextFieldPrimerApe.getText().trim().isEmpty()

|| !esAlfabetico(jTextFieldPrimerApe.getText())) {

errores.append("El primer apellido debe ser un texto alfabético.\n");

}

// Verificar que el campo "SegundoApe" sea alfabético (opcional, puede estar vacío)

if (!jTextFieldSegundoApe.getText().trim().isEmpty()

&& !esAlfabetico(jTextFieldSegundoApe.getText())) {

errores.append("El segundo apellido debe ser un texto alfabético.\n");

}

// Verificar que el campo "NumIdentificacion" contenga solo dígitos

if (jTextFieldNumIdentificacion.getText().trim().isEmpty()

|| !esNumerico(jTextFieldNumIdentificacion.getText())) {

errores.append("El número de identificación debe contener solo dígitos.\n");

}

// Verificar que el campo "Telefono" contenga solo dígitos

if (jTextFieldTelefono.getText().trim().isEmpty() || !esNumerico(jTextFieldTelefono.getText())) {

errores.append("El número de teléfono debe contener solo dígitos.\n");

}

// Verificar que el campo "Email" tenga formato válido

if (jTextFieldEmail.getText().trim().isEmpty() || !esEmailValido(jTextFieldEmail.getText())) {

errores.append("El email no tiene un formato válido.\n");

}

// Verificar que el campo "Direccion" no esté vacío

if (jTextFieldDireccion.getText().trim().isEmpty()) {

errores.append("La dirección no puede estar vacía.\n");

}

// Verificar que el sexo esté seleccionado

if (buttonGroupSexo.getSelection() == null) {

errores.append("Debe seleccionar el sexo (Macho o Hembra).\n");

}

// Verificar si hubo errores

if (errores.length() > 0) {

// Mostrar los errores acumulados

JOptionPane.showMessageDialog(

this,

errores.toString(),

"Errores de validación",

JOptionPane.ERROR\_MESSAGE);

} else {

// Continuar con la lógica de guardado

JOptionPane.showMessageDialog(

this,

"Todos los datos son válidos. Guardando...",

"Éxito",

JOptionPane.INFORMATION\_MESSAGE);

}

**}**

**private boolean esAlfabetico(String texto) {**

return texto.matches("^[a-zA-ZáéíóúÁÉÍÓÚñÑ ]+$");

**}**

**private boolean esNumerico(String texto) {**

return texto.matches("^\\d+$");

**}**

**private boolean esEmailValido(String email) {**

return email.matches("^[\\w.-]+@[\\w.-]+\\.[a-zA-Z]{2,}$");

**}**

**private boolean esFechaValida(String fecha) {**

// Define el formato esperado de la fecha

SimpleDateFormat formatoFecha = new SimpleDateFormat("dd/MM/yyyy");

formatoFecha.setLenient(false); // Asegura que la fecha sea estrictamente válida

try {

// Intenta analizar la fecha

formatoFecha.parse(fecha);

return true; // La fecha es válida

} catch (ParseException e) {

return false; // La fecha no es válida

}

**}**

**private void limpiar() {**

limpiarRecursivo(this.getContentPane()); // Llama al método recursivo en el contenedor principal

this.buttonGroupSexo.clearSelection(); // Limpia la selección del grupo de botones

this.jComboBoxTipo.setSelectedIndex(0); // Selecciona la primera opcion del combobox por defecto

**}**

**private void limpiarRecursivo(Container container) {**

for (Component component : container.getComponents()) {

if (component instanceof JTextField) {

JTextField textField = (JTextField) component;

textField.setText(""); // Limpiar el campo de texto

} else if (component instanceof Container) {

// Si el componente es un contenedor (como un JPanel), llama al método recursivo

limpiarRecursivo((Container) component);

}

}

**}**

**private void confirmarSalidadSinGuardar() {**

if (todosCamposTextoVacios(this) && this.buttonGroupSexo.getSelection() == null) {

this.dispose();

System.out.println(todosCamposTextoVacios(this));

} else {

String[] opciones = {"Sí", "No"};

int resultado = JOptionPane.showOptionDialog(

null,

"Hay información sin guardar, ¿Estás segur@ que deseas cerrar?",

"Información de la aplicación",

JOptionPane.DEFAULT\_OPTION,

JOptionPane.INFORMATION\_MESSAGE,

null,

opciones,

opciones[0] // Botón por defecto

);

switch (resultado) {

case 0: // Índice del botón "Sí"

this.dispose();

break;

case 1: // Índice del botón "No"

System.out.println("Operación cancelada");

break;

default: // Caso en que se cierra el diálogo

break;

}

}

**}**

**private boolean todosCamposTextoVacios(Container container) {**

for (Component component : container.getComponents()) {

if (component instanceof JTextField) {

JTextField textField = (JTextField) component;

if (!textField.getText().trim().isEmpty()) {

return false; // Si un campo no está vacío, devuelve false

}

} else if (component instanceof Container) {

// Llamada recursiva para contenedores anidados

if (!todosCamposTextoVacios((Container) component)) {

return false;

}

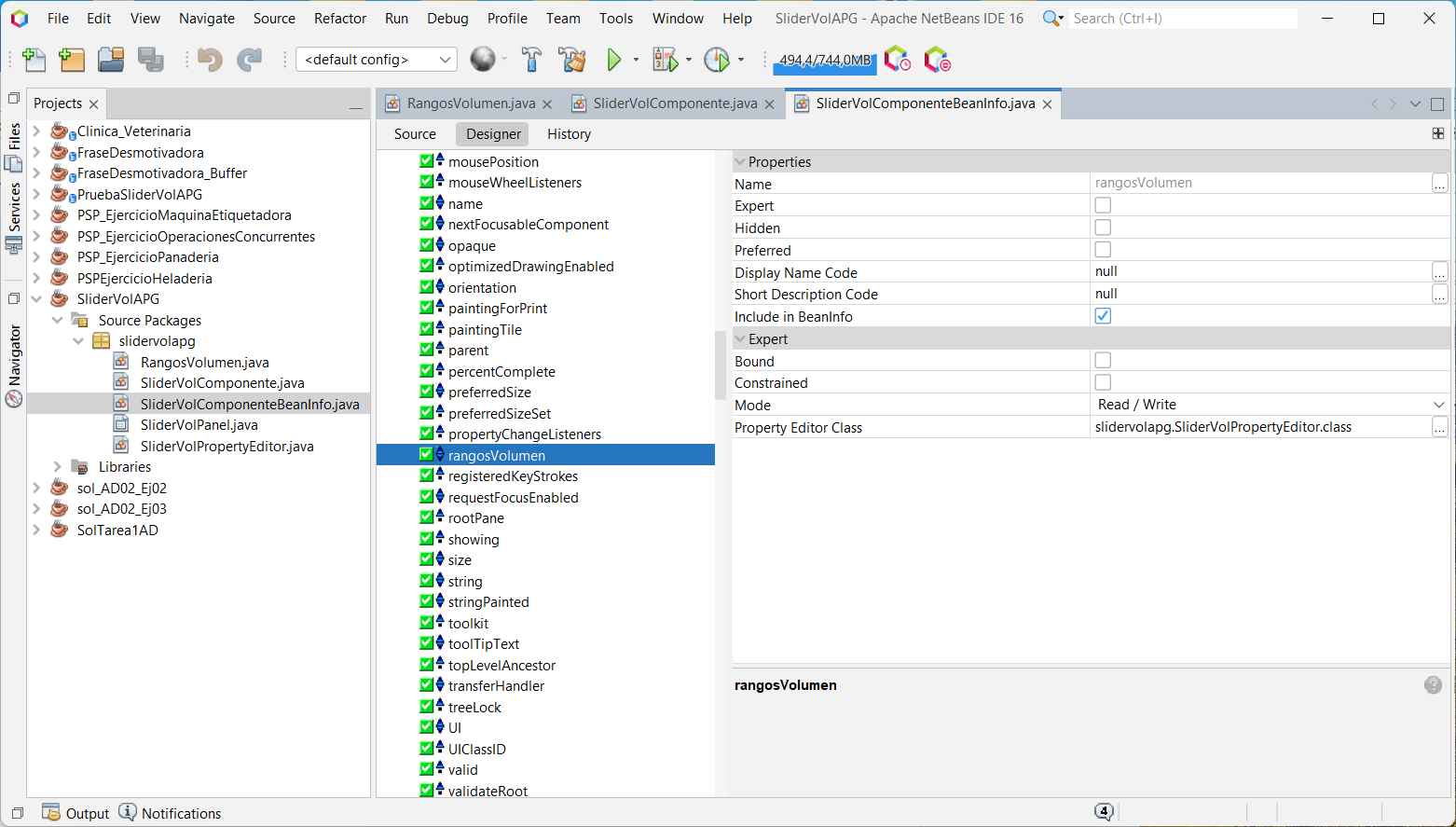
}

}

return true; // Si todos los campos están vacíos, devuelve true

**}**

**CREACIÓN DEL COMPONENTE SliderVolAPG**

****

**Código del archivo panel:**

**package slidervolapg;**

/\*\*

\*

\* @author andres

\*/

**public class SliderVolPanel extends javax.swing.JPanel {**

/\*\*

\* Creates new form NotaPanel

\*/

**public SliderVolPanel() {**

initComponents();

**}**

**public RangosVolumen getSelectedValue() {**

return new RangosVolumen(jSliderRango1.getValue(), jSliderRango2.getValue(), jSliderRango3.getValue());

**}**

**public void mostrarValores() {**

this.jLabelRango1.setText(this.jSliderRango1.getValue() + "%");

this.jLabelRango2.setText(this.jSliderRango2.getValue() + "%");

this.jLabelRango3.setText(this.jSliderRango3.getValue() + "%");

**}**

**// Generated Code**

**private void jSliderRango1StateChanged(javax.swing.event.ChangeEvent evt) {**

if (this.jSliderRango1.getValue() >= this.jSliderRango2.getValue()) {

this.jSliderRango2.setValue(this.jSliderRango1.getValue());

}

this.mostrarValores();

**}**

**private void jSliderRango2StateChanged(javax.swing.event.ChangeEvent evt) {**

if (this.jSliderRango2.getValue() >= this.jSliderRango3.getValue()) {

this.jSliderRango3.setValue(this.jSliderRango2.getValue());

}

this.mostrarValores();

**}**

**private void jSliderRango3StateChanged(javax.swing.event.ChangeEvent evt) {**

this.jSliderRango3.setValue(this.jSliderRango3.getValue());

this.mostrarValores();

**}**

// Variables declaration - do not modify

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabelRango1;

private javax.swing.JLabel jLabelRango2;

private javax.swing.JLabel jLabelRango3;

private javax.swing.JSlider jSliderRango1;

private javax.swing.JSlider jSliderRango2;

private javax.swing.JSlider jSliderRango3;

// End of variables declaration

**}**