

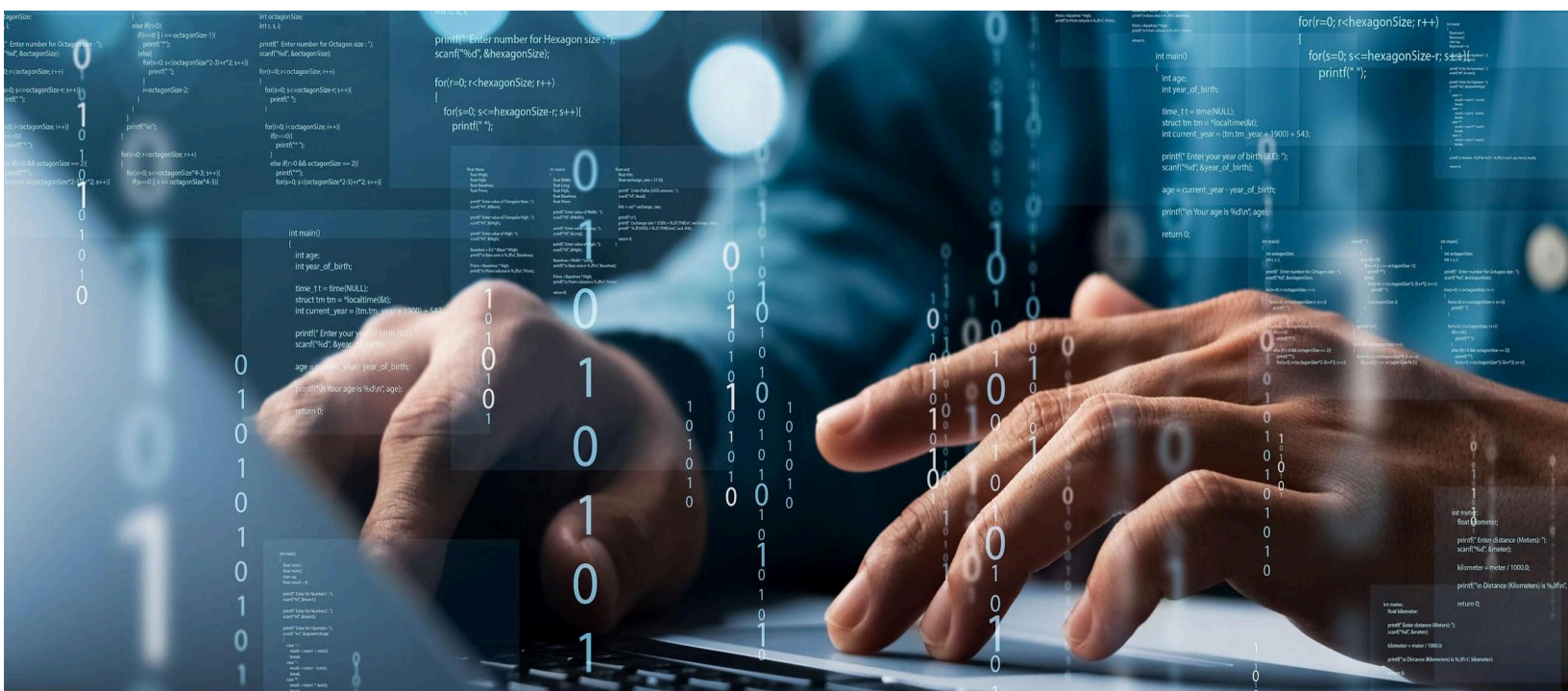
LENGUAJES DE PROGRAMACIÓN III



Práctica N° 10: JAVA SWING AVANZADO

Elaborado por:

DIAZ ACOSTA KAHORI FERNANDA



GRUPO N° 05

LENGUAJES DE PROGRAMACIÓN

Presentado por:

2023242762 DIAZ ACOSTA KAHORI FERNANDA 100%

ÍNDICE

1. ACTIVIDADES	5
EXPERIENCIA DE PRÁCTICA 1:	5
2. EJERCICIOS	12
2.1 EJERCICIO 1	12
1. Crea una aplicación que gestione la información de un producto en una tienda. La aplicación debe permitir ingresar el nombre del producto, precio, cantidad en stock y categoría. Al hacer clic en el botón "Actualizar Producto", los datos deben actualizarse en el modelo de datos y mostrarse en una etiqueta en la interfaz. Archivo del código: GestionProducto.java	12
2. Crea una aplicación para registrar y visualizar la temperatura diaria de una semana mediante un gráfico de líneas. La aplicación debe permitir al usuario ingresar la temperatura para cada día de la semana y mostrar una línea que conecte los puntos correspondientes a cada día en el gráfico. Archivo del código: GraficoTemperaturas.java	14
3. Crea una aplicación de Swing que funcione como un reproductor de efectos de sonido. La aplicación debe permitir reproducir diferentes sonidos al hacer clic en botones específicos. Cada botón representa un efecto de sonido distinto (por ejemplo, "Aplausos", "Campana" y "Explosión"). Diseñar una interfaz con varios botones, cada uno representando un efecto de Sonido. Archivo del código: ReproductorEfectosSonido.java	16
4. Crea una aplicación de Java Swing que permita al usuario reproducir, pausar y reanudar una pista de música. La aplicación debe tener botones de "Reproducir", "Pausar" y "Reanudar" que controlen la reproducción del audio.	18
2.2 EJERCICIO 2	20
3. CUESTIONARIO	24
4. BIBLIOGRAFÍA	29

1. ACTIVIDADES

ENLACE GITHUB:

https://github.com/KahoriDiazUCSM/Laboratorio_10/tree/main/ACTIVIDAD1

EXPERIENCIA DE PRÁCTICA 1:

```
/* *****  
 * NOMBRE : GraficosAvanzadosPanel.java  
 * ***** */  
package actividad1;  
  
import javax.swing.*;  
import java.awt.*;  
import java.awt.geom.AffineTransform;  
  
public class GraficosAvanzadosPanel extends JPanel {  
    @Override  
    protected void paintComponent(Graphics g) {  
        super.paintComponent(g);  
        Graphics2D g2d = (Graphics2D) g;  
  
        g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,  
RenderingHints.VALUE_ANTIALIAS_ON);  
  
        // Rectángulo girado  
        AffineTransform original = g2d.getTransform();  
        g2d.rotate(Math.toRadians(45), 100, 100);  
        g2d.setColor(Color.MAGENTA);  
        g2d.fillRect(50, 50, 100, 50);  
        g2d.setTransform(original);  
    }  
}
```

```
/* *****  
 * NOMBRE : GraficosSimplesPanel.java  
 * ***** */  
package actividad1;  
  
import javax.swing.*;  
import java.awt.*;  
  
class GraficosSimplesPanel extends JPanel {  
    @Override  
    protected void paintComponent(Graphics g) {  
        super.paintComponent(g);  
  
        g.setColor(Color.BLUE);  
    }  
}
```

```
        g.drawRect(50, 50, 100, 50); // Rectángulo
        g.setColor(Color.RED);
        g.fillOval(200, 50, 50, 50); // Círculo lleno
        g.setColor(Color.BLACK);
        g.drawString("Gráficos Simples", 50, 150); // Texto
    }
}
```

```
/******
 * NOMBRE : Producto.java
 *****/
package actividad1;

class Producto {
    private String nombre;
    private double precio;
    private int cantidadStock;

    public Producto(String nombre, double precio, int cantidadStock) {
        this.nombre = nombre;
        this.precio = precio;
        this.cantidadStock = cantidadStock;
    }

    public String getNombre() { return nombre; }
    public void setNombre(String nombre) { this.nombre = nombre; }

    public double getPrecio() { return precio; }
    public void setPrecio(double precio) { this.precio = precio; }

    public int getCantidadStock() { return cantidadStock; }
    public void setCantidadStock(int cantidadStock) { this.cantidadStock =
cantidadStock; }

    @Override
    public String toString() {
        return "Nombre: " + nombre + ", Precio: " + precio + ", Stock: " +
cantidadStock;
    }
}
```

```

/*****
* NOMBRE : VentanaBinding.java
*****/
package actividad1;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;

class VentanaBinding extends JFrame {
    public VentanaBinding() {
        setTitle("Binding de Datos Manual");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 300);
        setLocationRelativeTo(null);

        // Modelo
        Producto producto = new Producto("Producto A", 100.0, 10);

        // Componentes
        JTextField nombreField = new JTextField(producto.getNombre(), 15);
        JTextField precioField = new JTextField(String.valueOf(producto.getPrecio()),
15);
        JTextField stockField = new
JTextField(String.valueOf(producto.getCantidadStock()), 15);
        JButton actualizarButton = new JButton("Actualizar Producto");

        // Panel y disposición
        setLayout(new GridLayout(5, 2));
        add(new JLabel("Nombre:"));
        add(nombreField);
        add(new JLabel("Precio:"));
        add(precioField);
        add(new JLabel("Stock:"));
        add(stockField);
        add(new JLabel());
        add(actualizarButton);

        // Acción al presionar el botón
        actualizarButton.addActionListener(e -> {
            try {
                producto.setNombre(nombreField.getText());
                producto.setPrecio(Double.parseDouble(precioField.getText()));
                producto.setCantidadStock(Integer.parseInt(stockField.getText()));
                JOptionPane.showMessageDialog(this, "Producto actualizado:\n" +
producto);
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(this, "Datos inválidos. Verifique los
campos.");
            }
        });

        setVisible(true);
    }
}

```

```
}
```

```

/*****
* NOMBRE : VentanaGraficosAvanzados .java
*****/
package actividad1;

import javax.swing.*;

public class VentanaGraficosAvanzados extends JFrame {
    public VentanaGraficosAvanzados() {
        setTitle("Gráficos Avanzados");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(500, 500);
        setLocationRelativeTo(null);

        add(new GraficosAvanzadosPanel());
        setVisible(true);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(VentanaGraficosAvanzados::new);
    }
}

```

```

/*****
* NOMBRE : VentanaGraficosSimples.java
*****/
package actividad1;
import javax.swing.*;

public class VentanaGraficosSimples extends JFrame {
    public VentanaGraficosSimples() {
        setTitle("Gráficos Simples");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 300);
        setLocationRelativeTo(null);

        add(new GraficosSimplesPanel()); // No pasa el parámetro aquí
        setVisible(true);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(VentanaGraficosSimples::new); // Llama al
        constructor sin parámetros
    }
}

```



```

/*****
* NOMBRE : VentanaMultimedia.java
*****/
package actividad1;

import javax.swing.*;
import java.awt.*;
import javax.sound.sampled.*;
import java.io.File;

public class VentanaMultimedia extends JFrame {
    public VentanaMultimedia() {
        setTitle("Reproductor Multimedia");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 200);
        setLocationRelativeTo(null);

        JButton playButton = new JButton("Reproducir Sonido");
        playButton.addActionListener(e -> reproducirSonido("success.wav"));

        add(playButton, BorderLayout.CENTER);
        setVisible(true);
    }

    private void reproducirSonido(String filePath) {
        try {
            File audioFile = new File(filePath);
            AudioInputStream audioStream = AudioSystem.getAudioInputStream(audioFile);
            Clip clip = AudioSystem.getClip();
            clip.open(audioStream);
            clip.start();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(VentanaMultimedia::new);
    }
}

```

```

/*****
* NOMBRE : MainApp.java
*****/
package actividad1;

import javax.swing.*;
import java.awt.*;

public class MainApp {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            JFrame menuFrame = new JFrame("Aplicación Educativa");
            menuFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            menuFrame.setSize(300, 200);
            menuFrame.setLocationRelativeTo(null);

            JPanel panel = new JPanel(new GridLayout(4, 1, 5, 5));

            JButton bindingButton = new JButton("Gestión de Datos");
            bindingButton.addActionListener(e -> new VentanaBinding());

            JButton graficosSimplesButton = new JButton("Gráficos Simples");
            graficosSimplesButton.addActionListener(e -> new VentanaGraficosSimples());

            JButton graficosAvanzadosButton = new JButton("Gráficos Avanzados");
            graficosAvanzadosButton.addActionListener(e -> new
VentanaGraficosAvanzados());

            JButton multimediaButton = new JButton("Multimedia");
            multimediaButton.addActionListener(e -> new VentanaMultimedia());

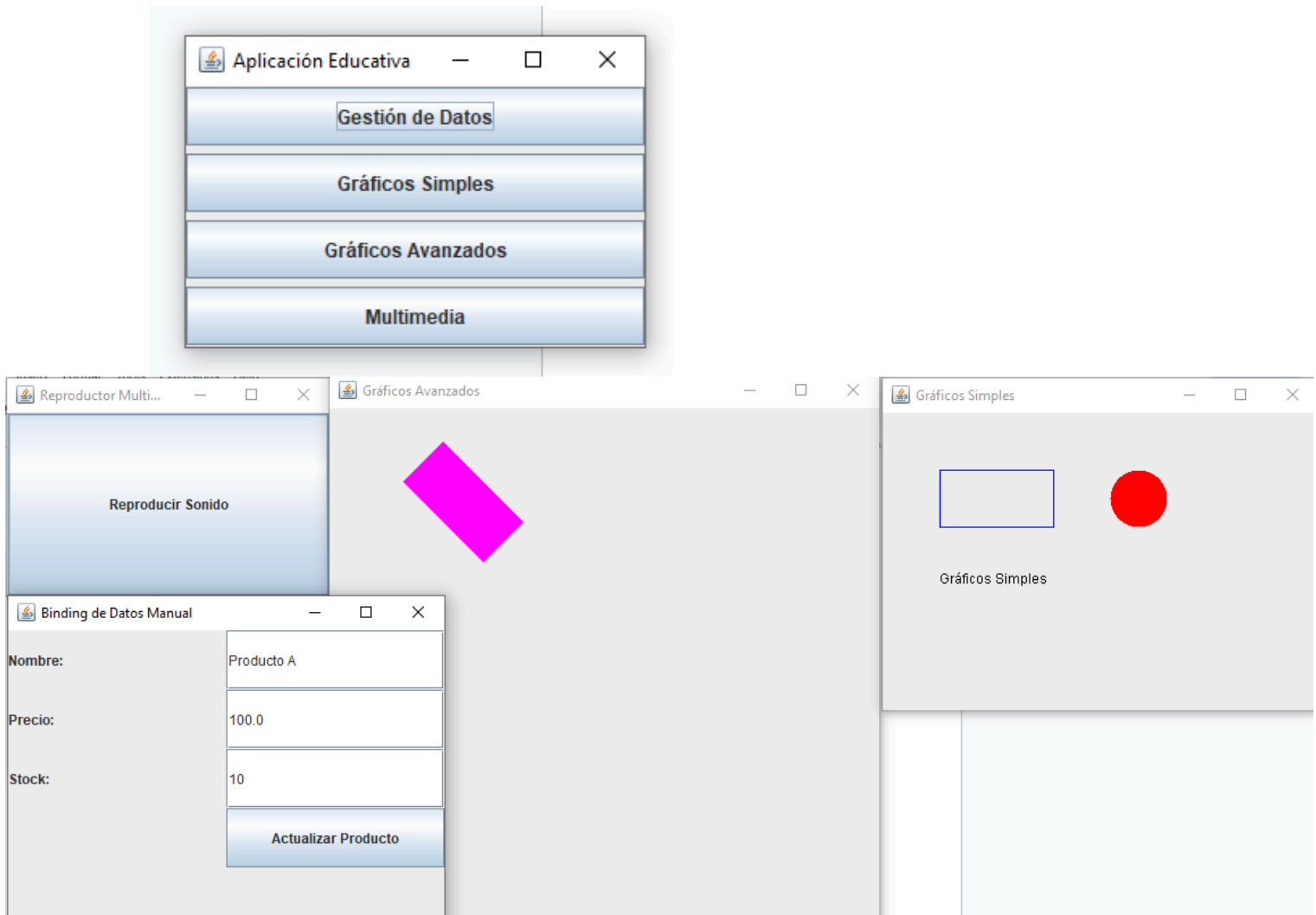
            panel.add(bindingButton);
            panel.add(graficosSimplesButton);
            panel.add(graficosAvanzadosButton);
            panel.add(multimediaButton);

            menuFrame.add(panel);
            menuFrame.setVisible(true);

        });
    }
}

```

RESULTADO:



2. EJERCICIOS

ENLACE GITHUB:

https://github.com/KahoriDiazUCSM/Laboratorio_10/tree/main/EJERCICIO1

2.1 EJERCICIO 1

1. Crea una aplicación que gestione la información de un producto en una tienda. La aplicación debe permitir ingresar el nombre del producto, precio, cantidad en stock y categoría. Al hacer clic en el botón "Actualizar Producto", los datos deben actualizarse en el

modelo de datos y mostrarse en una etiqueta en la interfaz. Archivo del código: GestionProducto.java

```

/*****
* NOMBRE : GestionProducto.java
*****/
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

class Producto {
    private String nombre;
    private double precio;
    private int cantidadStock;
    private String categoria;

    public Producto(String nombre, double precio, int cantidadStock, String categoria)
    {
        this.nombre = nombre;
        this.precio = precio;
        this.cantidadStock = cantidadStock;
        this.categoria = categoria;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public void setPrecio(double precio) {
        this.precio = precio;
    }

    public void setCantidadStock(int cantidadStock) {
        this.cantidadStock = cantidadStock;
    }

    public void setCategoria(String categoria) {
        this.categoria = categoria;
    }

    @Override
    public String toString() {
        return String.format("Producto: %s | Precio: %.2f | Stock: %d | Categoría: %s",
            nombre, precio, cantidadStock, categoria);
    }
}

public class GestionProducto {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Gestión de Producto");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

```

```

        frame.setSize(450, 300);

        Producto producto = new Producto("", 0.0, 0, "");

        JPanel panel = new JPanel(new GridLayout(6, 2, 10, 10));
        JTextField nombreField = new JTextField();
        JTextField precioField = new JTextField();
        JTextField stockField = new JTextField();
        JTextField categoriaField = new JTextField();
        JLabel resultadoLabel = new JLabel("Información del producto:",
SwingConstants.CENTER);
        JButton actualizarButton = new JButton("Actualizar Producto");

        panel.add(new JLabel("Nombre:"));
        panel.add(nombreField);
        panel.add(new JLabel("Precio:"));
        panel.add(precioField);
        panel.add(new JLabel("Cantidad en Stock:"));
        panel.add(stockField);
        panel.add(new JLabel("Categoría:"));
        panel.add(categoriaField);
        panel.add(actualizarButton);
        panel.add(new JLabel());
        panel.add(resultadoLabel);

        actualizarButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    String nombre = nombreField.getText();
                    double precio = Double.parseDouble(precioField.getText());
                    int cantidadStock = Integer.parseInt(stockField.getText());
                    String categoria = categoriaField.getText();

                    producto.setNombre(nombre);
                    producto.setPrecio(precio);
                    producto.setCantidadStock(cantidadStock);
                    producto.setCategoria(categoria);

                    resultadoLabel.setText(producto.toString());
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(frame, "Por favor, ingrese valores
válidos.", "Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        });

        frame.add(panel);
        frame.setVisible(true);
    }
}

```

2. Crea una aplicación para registrar y visualizar la temperatura diaria de una semana mediante un gráfico de líneas. La aplicación debe permitir al usuario ingresar la temperatura para cada día de la semana y mostrar una línea que conecte los puntos correspondientes a cada día en el gráfico. Archivo del código: GraficoTemperaturas.java

```

/*****
* NOMBRE : GraficoTemperaturas .java
*****/
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class GraficoTemperaturas {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Gráfico de Temperaturas");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 400);

        JPanel inputPanel = new JPanel(new GridLayout(8, 2, 10, 10));
        JLabel[] dayLabels = {
            new JLabel("Lunes:"), new JLabel("Martes:"), new JLabel("Miércoles:"),
            new JLabel("Jueves:"), new JLabel("Viernes:"), new JLabel("Sábado:"),
            new JLabel("Domingo:")
        };
        JTextField[] tempFields = new JTextField[7];
        for (int i = 0; i < tempFields.length; i++) {
            tempFields[i] = new JTextField();
            inputPanel.add(dayLabels[i]);
            inputPanel.add(tempFields[i]);
        }

        JButton drawButton = new JButton("Mostrar Gráfico");
        inputPanel.add(drawButton);
        inputPanel.add(new JLabel()); // Espaciador

        GraficoPanel graficoPanel = new GraficoPanel();
        frame.setLayout(new BorderLayout());
        frame.add(inputPanel, BorderLayout.NORTH);
        frame.add(graficoPanel, BorderLayout.CENTER);

        drawButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    int[] temperaturas = new int[7];
                    for (int i = 0; i < tempFields.length; i++) {

```

```

        temperaturas[i] = Integer.parseInt(tempFields[i].getText());
    }
    graficoPanel.setTemperaturas(temperaturas);
    graficoPanel.repaint();
} catch (NumberFormatException ex) {
    JOptionPane.showMessageDialog(frame, "Ingrese valores numéricos
válidos.", "Error", JOptionPane.ERROR_MESSAGE);
}
}
});

frame.setVisible(true);
}
}

class GraficoPanel extends JPanel {
    private int[] temperaturas = new int[7];
    private final String[] dias = { "Lun", "Mar", "Mié", "Jue", "Vie", "Sáb", "Dom" };

    public void setTemperaturas(int[] temperaturas) {
        this.temperaturas = temperaturas;
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g;
        int width = getWidth();
        int height = getHeight();
        int margin = 50;

        g2d.drawLine(margin, height - margin, width - margin, height - margin); // Eje
X        g2d.drawLine(margin, margin, margin, height - margin); // Eje Y

        int stepX = (width - 2 * margin) / (dias.length - 1);
        for (int i = 0; i < dias.length; i++) {
            int x = margin + i * stepX;
            g2d.drawString(dias[i], x - 10, height - margin + 20);
        }

        int maxTemp = Integer.MIN_VALUE;
        int minTemp = Integer.MAX_VALUE;
        for (int temp : temperaturas) {
            maxTemp = Math.max(maxTemp, temp);
            minTemp = Math.min(minTemp, temp);
        }
        if (maxTemp == minTemp) {
            maxTemp++;
        }

        int[] puntosX = new int[dias.length];
        int[] puntosY = new int[dias.length];
        for (int i = 0; i < temperaturas.length; i++) {

```

```

        puntosX[i] = margin + i * stepX;
        puntosY[i] = height - margin - (temperaturas[i] - minTemp) * (height - 2 *
margin) / (maxTemp - minTemp);
    }

    g2d.setColor(Color.BLUE);
    for (int i = 0; i < puntosX.length - 1; i++) {
        g2d.drawLine(puntosX[i], puntosY[i], puntosX[i + 1], puntosY[i + 1]);
    }

    g2d.setColor(Color.RED);
    for (int i = 0; i < puntosX.length; i++) {
        g2d.fillOval(puntosX[i] - 5, puntosY[i] - 5, 10, 10);
    }
}
}

```

3. Crea una aplicación de Swing que funcione como un reproductor de efectos de sonido. La aplicación debe permitir reproducir diferentes sonidos al hacer clic en botones específicos. Cada botón representa un efecto de sonido distinto (por ejemplo, "Aplausos", "Campana" y "Explosión"). Diseñar una interfaz con varios botones, cada uno representando un efecto de Sonido. Archivo del código: ReproductorEfectosSonido.java

```

/*****
* NOMBRE : ReproductorEfectosSonido .java
*****/
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.sound.sampled.*;
import java.io.File;
import java.io.IOException;

public class ReproductorEfectosSonido {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Reproductor de Efectos de Sonido");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 200);

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(1, 3, 10, 10));

        JButton aplausosButton = new JButton("Aplausos");
    }
}

```



```

JButton campanaButton = new JButton("Campana");
JButton explosionButton = new JButton("Explosión");

panel.add(aplausosButton);
panel.add(campanaButton);
panel.add(explosionButton);

aplausosButton.addActionListener(e -> reproducirSonido("aplausos.wav"));
campanaButton.addActionListener(e -> reproducirSonido("campana.wav"));
explosionButton.addActionListener(e -> reproducirSonido("explosion.wav"));

frame.setLayout(new BorderLayout());
frame.add(panel, BorderLayout.CENTER);
frame.setVisible(true);
}

private static void reproducirSonido(String nombreArchivo) {
    try {
        File archivoSonido = new File(nombreArchivo);
        if (!archivoSonido.exists()) {
            JOptionPane.showMessageDialog(null, "Archivo de sonido no encontrado: " +
+ nombreArchivo, "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        AudioInputStream audioStream =
AudioSystem.getAudioInputStream(archivoSonido);
        Clip clip = AudioSystem.getClip();
        clip.open(audioStream);
        clip.start();
    } catch (UnsupportedAudioFileException | IOException | LineUnavailableException
e) {
        JOptionPane.showMessageDialog(null, "Error al reproducir el sonido: " +
e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
}
}

```

4. Crea una aplicación de Java Swing que permita al usuario reproducir, pausar y reanudar una pista de música. La aplicación debe tener botones de "Reproducir", "Pausar" y "Reanudar" que controlen la reproducción del audio.

```

/*****
* NOMBRE : ReproductorMusica .java
*****/
import javax.sound.sampled.*;
import javax.swing.*;
import java.awt.*;
import java.io.File;
import java.io.IOException;

public class ReproductorMusica extends JFrame {
    private Clip clip;
    private Long posicionPausa = 0L;
    private boolean enReproduccion = false;

    public ReproductorMusica() {
        setTitle("Reproductor de Música");
        setSize(400, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        JPanel panelBotones = new JPanel();
        JButton btnReproducir = new JButton("Reproducir");
        JButton btnPausar = new JButton("Pausar");
        JButton btnReanudar = new JButton("Reanudar");

        panelBotones.add(btnReproducir);
        panelBotones.add(btnPausar);
        panelBotones.add(btnReanudar);

        add(panelBotones, BorderLayout.CENTER);

        btnReproducir.addActionListener(e -> reproducir());
        btnPausar.addActionListener(e -> pausar());
        btnReanudar.addActionListener(e -> reanudar());
    }

    private void reproducir() {
        try {
            if (clip != null && clip.isRunning()) {
                clip.stop();
            }
            File archivoMusica = new File("musica.wav");
            if (!archivoMusica.exists()) {
                JOptionPane.showMessageDialog(this, "Archivo de música no encontrado.",
                "Error", JOptionPane.ERROR_MESSAGE);
                return;
            }
            AudioInputStream audioStream =
AudioSystem.getAudioInputStream(archivoMusica);
            clip = AudioSystem.getClip();
            clip.open(audioStream);
            clip.setFramePosition(0);
            clip.start();
            enReproduccion = true;

```

```

        posicionPausa = 0L;
    } catch (UnsupportedAudioFileException | IOException | LineUnavailableException
e) {
        mostrarError("Error al reproducir la música: " + e.getMessage());
    }
}

private void pausar() {
    if (clip != null && clip.isRunning()) {
        posicionPausa = clip.getMicrosecondPosition();
        clip.stop();
        enReproduccion = false;
    }
}

private void reanudar() {
    if (clip != null && !enReproduccion) {
        clip.setMicrosecondPosition(posicionPausa);
        clip.start();
        enReproduccion = true;
    }
}

private void mostrarError(String mensaje) {
    JOptionPane.showMessageDialog(this, mensaje, "Error",
JOptionPane.ERROR_MESSAGE);
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        ReproductorMusica reproductor = new ReproductorMusica();
        reproductor.setVisible(true);
    });
}
}

```

2.2 EJERCICIO 2

Aplicación

Diagrama de Tablas

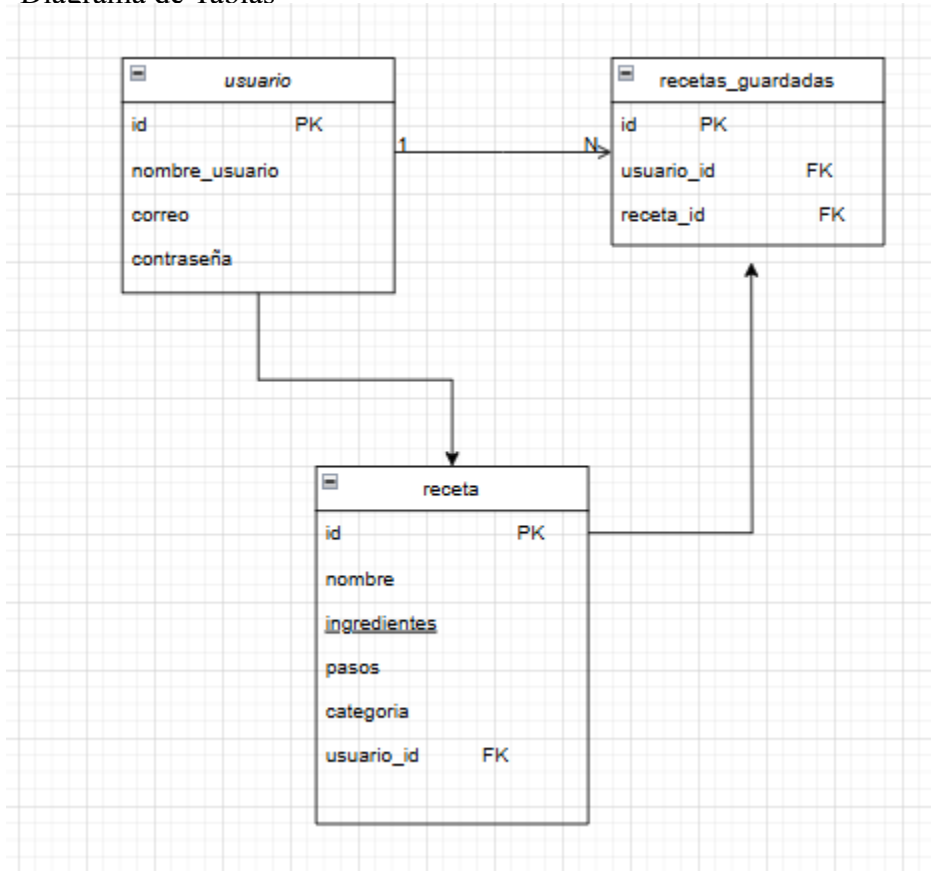


Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	R
<input type="checkbox"/> recetas	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	10	InnoDB	utf8_general_ci	32.0 KB	
<input type="checkbox"/> recetas_guardadas	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8_general_ci	48.0 KB	
<input type="checkbox"/> usuario	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	8	InnoDB	utf8_general_ci	48.0 KB	
3 tablas	Número de filas	20	InnoDB	utf8_general_ci	128.0 KB	

☐ Seleccionar todo

https://github.com/KahoriDiazUCSM/Laboratorio_10/tree/main/EJERCICIO2

Estructura del Proyecto

El proyecto está organizado en diferentes paquetes y clases para mantener un diseño modular y escalable. Aquí está la estructura general:

Inicio de la Aplicación:

La aplicación comienza en la clase Main, que crea una instancia del Controlador.

El Controlador inicializa la LoginVista para que el usuario inicie sesión o se registre.

Inicio de Sesión o Registro:



The screenshot shows a Java Swing window titled "Bienvenido a PicaApp". It features the PicaApp logo (a red stylized 'P' with a fork) and three input fields: "Nombre de usuario:", "Contraseña:", and "Confirmar contraseña:". Below the fields are three buttons: "Iniciar Sesión" (highlighted in orange), "Registrarse", and "Recuperar Contraseña".



The screenshot shows a Java Swing window titled "Registro en PicaApp". It features a cartoon illustration of a girl cooking. Below the illustration are four input fields: "Nombre de usuario:", "Correo electrónico:", "Contraseña:", and "Confirmar contraseña:". At the bottom are two buttons: "Registrar" (highlighted in blue) and "Volver".

Si el usuario ya tiene una cuenta, puede iniciar sesión.

Si el usuario no tiene una cuenta, puede registrarse.

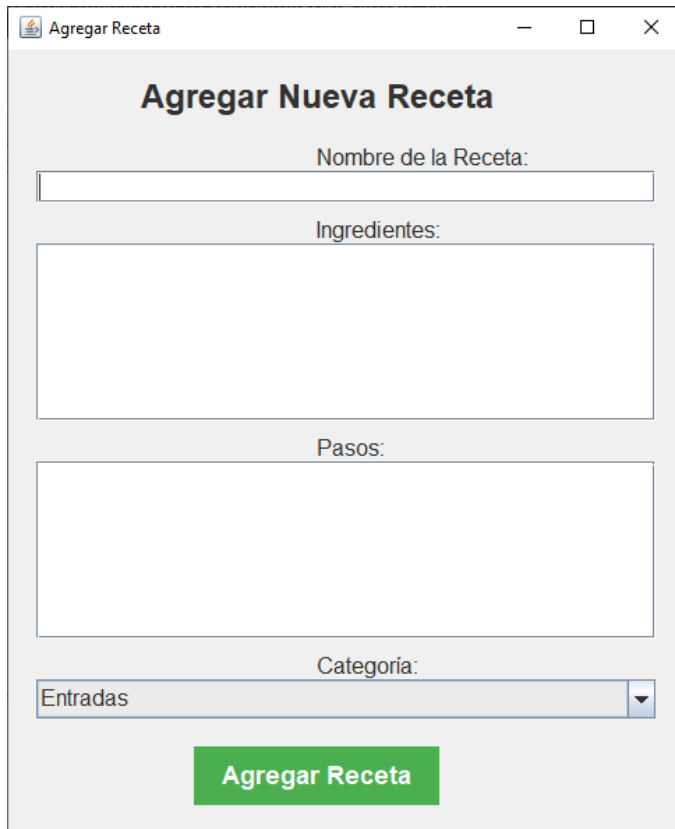
El Controlador valida las credenciales o registra al usuario en la base de datos.

Interfaz Principal:

Una vez que el usuario inicia sesión, se muestra la PrincipalVista.

El usuario puede buscar recetas por ingredientes, ver categorías y acceder a su cuenta.





A screenshot of a Java Swing window titled "Agregar Receta". The window has a light gray background and a title bar with standard Windows controls (minimize, maximize, close). The main content area is titled "Agregar Nueva Receta" in bold black text. Below the title, there are four input fields: a single-line text field for "Nombre de la Receta:", a multi-line text area for "Ingredientes:", another multi-line text area for "Pasos:", and a dropdown menu for "Categoría:" with "Entradas" selected. At the bottom center, there is a green button with the text "Agregar Receta" in white.

Agregar Nueva Receta

Nombre de la Receta:

Ingredientes:

Pasos:

Categoría:

Entradas

Agregar Receta

3. CUESTIONARIO

1. **¿Cómo se puede sincronizar un modelo de datos y una interfaz gráfica en Swing de forma manual?**

Para sincronizar un modelo de datos y una interfaz gráfica en Swing de forma manual, se puede seguir el siguiente procedimiento: Tuatara. (s.f.).

- Crear un contenedor superior
- Obtener el contenedor intermedio
- Seleccionar un gestor de esquemas para el contenedor intermedio
- Crear los componentes necesarios
- Agregar los componentes al contenedor intermedio

2. **¿Qué métodos de los componentes de Swing se utilizan comúnmente para obtener y establecer valores al realizar binding de datos manual?**

Componentes de entrada de texto (JTextField, JTextArea, etc.)

Obtener valores:

getText() → Retorna el texto introducido por el usuario.

Establecer valores:

setText(String text) → Establece el texto mostrado en el componente.

Botones y casillas de verificación (JCheckBox, JRadioButton, etc.)

Obtener valores:

isSelected() → Retorna un booleano indicando si el botón o casilla está seleccionada.

Establecer valores:

setSelected(boolean selected) → Selecciona o deselecciona el componente.

Listas desplegables y listas (JComboBox, JList)

Obtener valores:

getSelectedItem() → Retorna el elemento seleccionado (puede necesitar casting si se usa un modelo específico).

getSelectedIndex() → Retorna el índice del elemento seleccionado.

Establecer valores:

setSelectedItem(Object anObject) → Establece el elemento seleccionado.

setSelectedIndex(int index) → Establece el índice del elemento seleccionado.

Barras deslizantes (JSlider)

Obtener valores:

getValue() → Retorna el valor actual del deslizador.

Establecer valores:

setValue(int value) → Establece el valor del deslizador.

3. ¿Cómo se maneja la validación de datos en un formulario que utiliza binding de datos manual?

Pasos para manejar la validación de datos:

- Interceptar los datos ingresados por el usuario: Capturar los datos de los componentes antes de transferirlos al modelo.
- Validar los datos
- Notificar errores
- Actualizar el modelo solo si los datos son válidos

4. ¿Cómo podrías implementar un binding de datos manual para una lista de objetos y mostrar los datos en un componente JTable?

Crear una clase modelo que represente los datos de cada fila.

Crear un modelo de tabla que contenga los datos y se encargue de mostrar la lista de objetos en el JTable.

Actualizar la tabla manualmente cuando cambien los datos.

5. ¿Cómo se pueden crear gráficos interactivos que respondan a eventos de ratón y teclado en un JPanel en Swing?

Primero implementar eventos de ratón (MouseListener, MouseMotionListener).

Pasos para crear gráficos interactivos con eventos:

- Crear un JPanel que gestione eventos de ratón y teclado.
- Implementar los eventos de ratón y teclado.
- Redibujar el gráfico según la interacción del usuario.
- Actualizar la vista del gráfico tras los eventos (usando repaint()).

6. ¿Cómo podrías implementar un gráfico de datos (barras o líneas) utilizando Graphics en Swing para visualizar valores numéricos y permitir que se actualicen dinámicamente?

Para implementar un gráfico de barras o líneas utilizando Graphics en Swing y permitir actualizaciones dinámicas, sigue estos pasos:

1. Crear un panel personalizado para dibujar el gráfico:

```
import javax.swing.*;
import java.awt.*;
import java.util.List;

public class GraficoPanel extends JPanel {
    private List<Integer> datos; // Datos numéricos a graficar

    public GraficoPanel(List<Integer> datos) {
        this.datos = datos;
    }

    // Método para actualizar los datos
    public void actualizarDatos(List<Integer> nuevosDatos) {
        this.datos = nuevosDatos;
        repaint(); // Redibuja el gráfico
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        if (datos != null && !datos.isEmpty()) {
            int width = getWidth();
            int height = getHeight();
            int maxValor = datos.stream().max(Integer::compareTo).orElse(0);

            // Dibujar barras
            int barraAncho = width / datos.size();
            for (int i = 0; i < datos.size(); i++) {
                int barraAltura = (int) ((double) datos.get(i) / maxValor *
height);
                g.fillRect(i * barraAncho, height - barraAltura, barraAncho
- 1, barraAltura);
            }
        }
    }
}
```

2. Actualizar los datos y redibujar el gráfico:

```
List<Integer> datos = Arrays.asList(5, 10, 15, 20, 25);
GraficoPanel panel = new GraficoPanel(datos);

// Para actualizar los datos y redibujar:
panel.actualizarDatos(Arrays.asList(10, 15, 20, 30, 40));
```

7. ¿Cómo funciona el método `paintComponent(Graphics g)` y cuándo se llama automáticamente?

El método `paintComponent(Graphics g)` se usa para realizar el dibujo personalizado de un componente. Recibe un objeto `Graphics` y proporciona métodos para realizar el dibujo, como `drawLine()`, `fillRect()`, etc.

¿Cuándo se llama automáticamente?

Cuando el componente necesita ser redibujado.

Cuando se invoca `repaint()`

Cuando el tamaño del componente cambia

Cuando el componente es cubierto y luego descubierto

8. ¿Qué es antialiasing y cómo se aplica en gráficos creados con Graphics2D en Java Swing?

Antialiasing es una técnica utilizada para suavizar los bordes de las líneas y formas en gráficos, el uso de esta técnica reduce los bordes que suelen aparecer en líneas diagonales o curvas.

En `Graphics2D` de Java Swing, se puede activar el antialiasing utilizando el método `setRenderingHint()` para mejorar la calidad visual de los gráficos.

9. ¿Qué clases de Java estándar permiten trabajar con audio en una aplicación Swing y cuáles son sus principales métodos?

En Java, las clases estándar para trabajar con audio en una aplicación Swing son:

1. Clip (`javax.sound.sampled.Clip`)

Métodos principales:

`open(AudioInputStream stream)` - Carga un archivo de audio.

`start()` - Inicia la reproducción.

`stop()` - Detiene la reproducción.

`loop(int count)` - Reproduce en bucle.

2. AudioSystem (`javax.sound.sampled.AudioSystem`)

Métodos principales:

`getAudioInputStream(File file)` - Obtiene un flujo de audio desde un archivo.

`getClip()` - Obtiene un Clip vacío para cargar audio.

3. `AudioInputStream` (`javax.sound.sampled.AudioInputStream`)

Métodos principales:

`read(byte[] b)` - Lee datos de audio.

`close()` - Cierra el flujo de audio.

10. ¿Cuál es la diferencia entre los formatos de audio soportados nativamente en Java (WAV, AU) y otros formatos populares como MP3? ¿Cómo se pueden reproducir estos formatos en Java Swing?

1. WAV y AU:

Soporte nativo en Java: Java tiene soporte directo para los formatos WAV (Waveform Audio File) y AU a través de las clases `AudioSystem` y `Clip`.

Ventajas: Estos formatos no tienen casi nada de compresión o es mínima, lo que los hace adecuados para aplicaciones donde la calidad es más importante que el tamaño del archivo.

Limitación: No soportan compresión eficiente como MP3, lo que puede hacer que los archivos sean grandes.

2. MP3:

No soporte nativo: Java no soporta de forma nativa la reproducción de archivos MP3, OGG, o FLAC. Para reproducir estos formatos, se requieren bibliotecas externas, ya que Java Sound API no los maneja de forma directa.

Soluciones externas: Se pueden usar bibliotecas como `JLayer` (para MP3) o `JavaZoom` para poder trabajar con estos formatos.

Reproducción de MP3 en Java Swing (con `JLayer`):

Para reproducir MP3 en Java Swing, una opción común es usar la librería `JLayer`, que es una implementación de MP3 para Java.

11. ¿Cómo se usa la clase `Clip` en `javax.sound.sampled` para cargar y reproducir archivos de audio en Java Swing?

La clase Clip de javax.sound.sampled se utiliza para cargar y reproducir archivos de audio en Java. A continuación se describe cómo se utiliza esta clase en una aplicación Swing.

Pasos básicos para usar Clip en Java Swing:

Cargar el archivo de audio usando un `AudioInputStream`.

Obtener un objeto Clip desde `AudioSystem.getClip()`.

Abrir el clip con `clip.open(audioStream)`.

Reproducir el audio con `clip.start()`.

Detener la reproducción con `clip.stop()`.

12. ¿Cómo se pueden pausar, detener y reanudar archivos de audio en una aplicación de Java Swing utilizando la clase Clip?

`clip.getMicrosecondPosition()` se utiliza para obtener la posición actual de reproducción en microsegundos.

Pausar: Se guarda la posición actual con `clip.getMicrosecondPosition()` y luego se detiene el clip con `clip.stop()`.

Reanudar: Al presionar "Resume", se **restablece** la **posición** de reproducción usando `clip.setMicrosecondPosition(clipTimePosition)` y luego se reanuda con `clip.start()`.

Detener: Al presionar "Stop", el clip se **detiene** con `clip.stop()` y se reinicia al principio con `clip setFramePosition(0)`.

4. BIBLIOGRAFÍA

ITLP. (s.f.). Tutorial de Java - Capítulo 5: Audio. Recuperado de <http://www.itlp.edu.mx/web/java/Tutorial%20de%20Java/Cap5/audio.html>.

YouTube. (s.f.). Tutorial de Java - Login Recuperado de
https://www.youtube.com/watch?v=ZUGkQHtLS4U&list=PLyt2v1LVXYS0_-_nE4ZMfJvhjDlx9kRqL.