Ejercicio Resuelto - Java Swing

Docente: Samuel Sepúlveda Cuevas

Ayudante: Jesús Tapia Martin

26/06/2025

1 Requerimientos

Este reporte corresponde a la solución/apunte del ejercicio de la semana, se extenderá la funcionalidad del sistema mediante la incorporación de una **interfaz gráfica de usuario (GUI)** utilizando la biblioteca **Swing** de Java. Esta mejora reemplazará el enfoque por consola utilizado hasta ahora, permitiendo una interacción más intuitiva a través de ventanas, botones y campos de entrada visuales.

> Para realizar esta transición desde la consola a una interfaz gráfica, **no se debe modificar el Modelo ni el Controlador**, ya que la responsabilidad de la Vista es únicamente mostrar componentes al usuario e invocar métodos ya definidos en el sistema. Por lo tanto, las ventanas no deben contener lógica de negocio, manteniendo así una correcta separación de responsabilidades.

Aunque existen herramientas visuales que permiten construir interfaces mediante el arrastre de componentes (como los archivos *.form* en entornos gráficos), en este ejercicio se construirá la GUI **manualmente mediante código Java**. Esto permitirá comprender de manera profunda cómo se estructura y organiza una interfaz desde cero, fortaleciendo el dominio de los conceptos fundamentales de Swing y del patrón arquitectónico MVC.

Nota

La solución presente en este ejercicio resuelto, es solo una aproximación al uso de Java con la biblioteca Swing, por lo que esta solución puede ser implementada de mejor forma, por lo que se deja abierta la invitación a estudiar desde la documentación y a realizar una mejor versión de esta solución.

2 Clase LoginView

2.1 Paso 1

```
public class LoginView extends JFrame {
    // Aquí se definen los componentes y la lógica de la ventana
}
```

2.1.1. JFrame

- JFrame es una clase de Java Swing que representa una ventana completa del sistema operativo, con borde, botón cerrar, minimizar, etc.
- Cuando se extiende de JFrame se indica que la clase será una ventana, a la cual se podrá agregar elementos visuales como botones, campos de texto, etiquetas, etc.

2.2 Paso 2

```
private final JTextField usuarioField = new JTextField(15);
private final JPasswordField claveField = new JPasswordField(15);
private final JButton loginButton = new JButton("Iniciar Sesión");
private final DatosLogin datosLogin = new DatosLogin();
private final Login loginController = new Login();
```

- **JTextField:** Campo de texto normal (para escribir texto).
- **JPasswordField:** Campo de texto oculto (para contraseñas).
- **JButton:** Un botón con texto que se puede presionar.

> El «15» en JTextField(15) significa que el campo mostrará aproximadamente 15 letras.

2.3 Paso 3

```
public LoginView() {
    setTitle("Login");
```

```
    setSize(300, 200);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setLocationRelativeTo(null);
```

El constructor de una clase que extiende de JFrame, es el lugar donde se configura el diseño, comportamiento, contenido y ubicación de la ventana cuando se crea una nueva instancia.

- **setTitle("Login"):** Define el título que aparecerá en la barra superior de la ventana.
- **setSize(300, 200):** Establece el tamaño de la ventana en 300 píxeles de ancho y 200 de alto.
- **setDefaultCloseOperation(EXIT_ON_CLOSE):** Le dice a Java que cierre el programa cuando se presione la X de la ventana.
- **setLocationRelativeTo(null):** Centra la ventana en la pantalla al abrirse.

2.3.1. JPanel

JPanel es un contenedor, es decir, una zona dentro de un JFrame donde se pueden colocar componentes gráficos como botones, etiquetas o campos de texto. Su función es organizar visualmente los elementos de la interfaz, siendo un análogo a un en HTML, por lo que es común utilizar varios JPanel dentro de un mismo JFrame para estructurar la ventana por secciones.

Para organizar los componentes en forma de grilla (filas y columnas), se utiliza un **GridLayout**, el cual recibe cuatro parámetros: número de filas, número de columnas, espacio horizontal entre componentes y espacio vertical.

```
JPanel panel = new JPanel();
panel.setLayout(new GridLayout(3, 2, 5, 5));
```

En este caso, se usa un GridLayout(3, 2, 5, 5) que distribuye los componentes en 3 filas y 2 columnas, con márgenes de 5 píxeles.

2.3.2. addActionListener

```
loginButton.addActionListener(e -> autenticarUsuario());
```

«addActionListener» es un método de los botones (JButton) que permite escuchar cuando el usuario hace clic en el botón.

Hay listeners para distintos eventos:

- **ActionListener:** Clics en botones, opciones de menú.
- **MouseListener:** Clicks, movimiento del mouse, etc.
- **KeyListener:** Teclas presionadas.
- **WindowListener:** Eventos de la ventana (abrir, cerrar).
- **DocumentListener:** Cambios en campos de texto (JTextField).

- En el contexto del ejercicio, cada vez que se presiona «loginButton», se llama automáticamente al método autenticarUsuario().

2.4 Paso 4

```
private void autenticarUsuario() {
    String usuario = usuarioField.getText();
    String clave = new String(claveField.getPassword());

    Usuario user = loginController.autenticar(usuario, clave, datosLogin);
```

- Se crea el método «autenticarUsuario» el cual obtiene el texto que el usuario escribió en los campos.
- Por medio del método «autenticar» del package Controlador, se pasa el nombre, la clave y la lista de usuarios, para validar el usuario, devolviendo un objeto Usuario en caso de serlo.

2.4.1. JOptionPane.showMessageDialog

```
if (user != null) {
    new MainView(user).setVisible(true);
    this.dispose();
} else {
    JOptionPane.showMessageDialog(
        this,
        "Usuario no registrado",
        "Error",
```

```
            JOptionPane.ERROR_MESSAGE
        );
    }
```

- **new MainView(user):** crea una nueva ventana (de tipo JFrame) para mostrar las tareas del usuario que acaba de iniciar sesión.
- **.setVisible(true):** Esto muestra la ventana, es necesario ya que sin este cualquier instancia de la clase no se verá.
- **this.dispose():** Cierra la ventana actual para abrir una nueva.
- **JOptionPane:** Clase que permite mostrar cuadros de diálogo (ventanas pequeñas).
- **.showMessageDialog:** Muestra un mensaje informativo o de error.
- **this:** Se refiere al JFrame actual, en este contexto, la ventana LoginView.
- **JOptionPane.ERROR_MESSAGE:** Icono rojo con cruz para indicar un error.

2.5 Código Completo

```
package Vista;

import Controlador.Login;
import Modelo.DatosLogin;
import Modelo.Usuario;

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

public class LoginView extends JFrame {
    private final JTextField usuarioField = new JTextField(15);
    private final JPasswordField claveField = new JPasswordField(15);
    private final JButton loginButton = new JButton("Iniciar Sesión");
    private final DatosLogin datosLogin = new DatosLogin();
    private final Login loginController = new Login();
```

```
public LoginView() {
    setTitle("Login");
    setSize(300, 200);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setLocationRelativeTo(null);

    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(3, 2, 5, 5));

    panel.add(new JLabel("Usuario:"));
    panel.add(usuarioField);

    panel.add(new JLabel("Contraseña:"));
    panel.add(claveField);

    panel.add(new JLabel());
    panel.add(loginButton);

    add(panel);

    loginButton.addActionListener(e -> autenticarUsuario());
}

private void autenticarUsuario() {
    String usuario = usuarioField.getText();
    String clave = new String(claveField.getPassword());

    Usuario user = loginController
                        .autenticar(usuario, clave, datosLogin);
    if (user != null) {
        new MainView(user).setVisible(true);
```

```
            this.dispose();
        } else {
            JOptionPane.showMessageDialog(
                this,
                "Usuario no registrado",
                "Error",
                JOptionPane.ERROR_MESSAGE
                );
        }
    }
}
```

3 Clase MainView

3.1 Paso 1

```
public class MainView extends JFrame {
// Se define la ventana principal que muestra las tareas del usuario
}
```

- Esta clase extiende de **JFrame**, lo que significa que representa una ventana.
- Hereda todos los comportamientos de una ventana: borde, botones para cerrar, minimizar, etc.

3.2 Paso 2

```
private final Usuario usuario;
private final DatosSesion datosSesion;
private DefaultListModel<String> tareasModel;
private JList<String> tareasList;
private JTextField filtroField;
```

- **Usuario**: almacena el usuario autenticado.
- **DatosSesion**: permite registrar las tareas agregadas durante la sesión.
- **JList**: lista donde se mostrarán las tareas.
- **JTextField**: permite escribir texto para buscar tareas.

3.3 Paso 3

```
public MainView(Usuario usuario) {
    this.usuario = usuario;
    this.datosSesion = new DatosSesion(usuario);
```

```
    setTitle("Tareas de " + usuario.getNombre());
    setSize(500, 400);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(EXIT_ON_CLOSE);

    inicializarComponentes();
    cargarTareas();

    setVisible(true);
}
```

- Se define el título de la ventana usando el nombre del usuario.
- Se configura el tamaño y posición de la ventana.
- Se llama al método **inicializarComponentes** para preparar la interfaz.
- Finalmente, se hace visible con `setVisible(true)`.

3.4 Paso 4

```
private void inicializarComponentes() {
    // Panel principal con distribución BorderLayout.
    // Norte, Sur, Centro, Este y Oeste.
    JPanel panel = new JPanel();
    panel.setLayout(new BorderLayout());

    tareasModel = new DefaultListModel<>();
    tareasList = new JList<>(tareasModel);
    JScrollPane scrollPane = new JScrollPane(tareasList);

    filtroField = new JTextField();
    // Listener que detecta cuando se escribe en el campo de búsqueda.
    // Cada cambio actualiza automáticamente la lista filtrada.
    filtroField.getDocument().addDocumentListener(new DocumentListener() {
```

```
        public void insertUpdate(DocumentEvent e) { filtrar(); }
        public void removeUpdate(DocumentEvent e) { filtrar(); }
        public void changedUpdate(DocumentEvent e) { filtrar(); }
    });

    // Panel superior con campo de búsqueda
    JPanel topPanel = new JPanel(new BorderLayout());
    topPanel.add(new JLabel("Buscar: "), BorderLayout.WEST);
    topPanel.add(filtroField, BorderLayout.CENTER);

    JButton agregarBtn = new JButton("Agregar Tarea");
    agregarBtn.addActionListener(e -> mostrarDialogoAgregar());

    // Agregar los componentes al panel principal en sus respectivas zonas.
    panel.add(topPanel, BorderLayout.NORTH);
    panel.add(scrollPane, BorderLayout.CENTER);
    panel.add(agregarBtn, BorderLayout.SOUTH);

    add(panel);
}
```

- Se usa **BorderLayout** para distribuir los componentes.
- Se crean e inicializan los objetos.
- En la parte superior se coloca el campo de búsqueda.
- En el centro se muestra la lista de tareas.
- En la parte inferior se encuentra el botón para agregar nuevas tareas.

3.4.1. BorderLayout

BorderLayout es un administrador de diseño (layout manager) en Java Swing que permite organizar los componentes gráficos dentro de un contenedor (como un JPanel o JFrame) en cinco zonas: NORT, SOUTH, CENTER, EAST y WEST

3.5 Paso 5

```
private void filtrar() {
    // Obtener texto del usuario a minúsculas
    String texto = filtroField.getText().toLowerCase();
    tareasModel.clear();
    for (Tarea tarea : usuario.getTareas()) {
        // Si la descripción de la tarea contiene el texto buscado
        if (tarea.getDescripcion().toLowerCase().contains(texto)) {
            tareasModel.addElement(
                "[" +
                tarea.getPrioridad() +
                "] "
                + tarea.getDescripcion()
            );
        }
    }
}
```

3.6 Paso 6

```
private void mostrarDialogoAgregar() {
    JTextField descField = new JTextField();
    String[] opciones = {"BAJA", "MEDIA", "ALTA"};
    JComboBox<String> prioridadBox = new JComboBox<>(opciones);

    JPanel panel = new JPanel(new GridLayout(2, 2));
    panel.add(new JLabel("Descripci\'on:"));
    panel.add(descField);
    panel.add(new JLabel("Prioridad:"));
    panel.add(prioridadBox);
```

```
    // Ventana de diálogo para ingresar datos
    int resultado = JOptionPane
                        .showConfirmDialog(
                            this,
                            panel,
                            "Agregar Tarea",
                            JOptionPane.OK_CANCEL_OPTION,
                            JOptionPane.PLAIN_MESSAGE
                        );

    // Si el usuario presiona OK
    if (resultado == JOptionPane.OK_OPTION) {
        String desc = descField.getText();
        String prio = (String) prioridadBox.getSelectedItem();
        if (!desc.isBlank() && prio != null) {
            datosSesion.agregarTarea(
                new Tarea(
                        desc,
                        Prioridad.valueOf(prio),
                        false
                    )
                );
            cargarTareas();
        }
    }
}
```

3.7 Paso 7

```
private void cargarTareas() {
    tareasModel.clear();
    for (Tarea tarea : usuario.getTareas()) {
```

```
        tareasModel.addElement(
            "[" +
            tarea.getPrioridad() +
            "] " +
            tarea.getDescripcion()
        );
    }
}
```

3.8 Código Final

```
package Vista;

import Modelo.DatosSesion;
import Modelo.Tarea;
import Modelo.Usuario;

import javax.swing.*;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;
import java.awt.*;
import java.util.List;

public class MainView extends JFrame {
    private final Usuario usuario;
    private final DatosSesion datosSesion;

    private DefaultListModel<String> tareasModel;
    private JList<String> tareasList;
    private JTextField filtroField;

    public MainView(Usuario usuario) {
        this.usuario = usuario;
```

```
        this.datosSesion = new DatosSesion(usuario);

        setTitle("Tareas de " + usuario.getNombre());
        setSize(500, 400);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        inicializarComponentes();
        cargarTareas();
        setVisible(true);
    }

    private void inicializarComponentes() {
        JPanel panel = new JPanel();
        panel.setLayout(new BorderLayout());

        tareasModel = new DefaultListModel<>();
        tareasList = new JList<>(tareasModel);
        JScrollPane scrollPane = new JScrollPane(tareasList);

        filtroField = new JTextField();
        filtroField.getDocument().addDocumentListener(new DocumentListener() {
            public void insertUpdate(DocumentEvent e) { filtrar(); }
            public void removeUpdate(DocumentEvent e) { filtrar(); }
            public void changedUpdate(DocumentEvent e) { filtrar(); }
        });

        JPanel topPanel = new JPanel(new BorderLayout());
        topPanel.add(new JLabel("Buscar: "), BorderLayout.WEST);
        topPanel.add(filtroField, BorderLayout.CENTER);

        JButton agregarBtn = new JButton("Agregar Tarea");
```

```
        agregarBtn.addActionListener(e -> mostrarDialogoAgregar());

        panel.add(topPanel, BorderLayout.NORTH);
        panel.add(scrollPane, BorderLayout.CENTER);
        panel.add(agregarBtn, BorderLayout.SOUTH);

        add(panel);
    }

    private void cargarTareas() {
        tareasModel.clear();
        for (Tarea tarea : usuario.getTareas()) {
            tareasModel.addElement(
                "[" +
                tarea.getPrioridad() +
                "] " +
                tarea.getDescripcion()
            );
        }
    }

    private void filtrar() {
        String texto = filtroField.getText().toLowerCase();
        tareasModel.clear();
        for (Tarea tarea : usuario.getTareas()) {
            if (tarea.getDescripcion().toLowerCase().contains(texto)) {
                tareasModel.addElement(
                    "[" +
                    tarea.getPrioridad() +
                    "] " +
                    tarea.getDescripcion()
                );
```

```
            }
        }
    }

    private void mostrarDialogoAgregar() {
        JTextField descField = new JTextField();
        String[] opciones = {"BAJA", "MEDIA", "ALTA"};
        JComboBox<String> prioridadBox = new JComboBox<>(opciones);

        JPanel panel = new JPanel(new GridLayout(2, 2));
        panel.add(new JLabel("Descripción:"));
        panel.add(descField);
        panel.add(new JLabel("Prioridad:"));
        panel.add(prioridadBox);

        int resultado = JOptionPane
                            .showConfirmDialog(
                                this,
                                panel,
                                "Agregar Tarea",
                                JOptionPane.OK_CANCEL_OPTION,
                                JOptionPane.PLAIN_MESSAGE
                            );

        if (resultado == JOptionPane.OK_OPTION) {
            String desc = descField.getText();
            String prio = (String) prioridadBox.getSelectedItem();
            if (!desc.isBlank() && prio != null) {
                datosSesion
                    .agregarTarea(new Tarea(
                        desc,
                        Enum.
```

```
                        valueOf(Modelo.Prioridad.class, prio),
                        false)
                    );
                cargarTareas();
            }
        }
    }
}
```