

**OBJETIVO:** Al finalizar de este taller, los estudiantes serán capaces de comprender y utilizar la clase JFrame para desarrollar una aplicación gráfica en relación a la liga BetPlay.

## INFORMACIÓN APLICACIÓN – CAMPEONATOFUTBOLAPP2

La aplicación "CampeonatoFutbolApp2" es una herramienta que permite a los usuarios acceder a estadísticas de equipos de fútbol de la Liga BetPlay - Futbol Femenino Colombiano 2023. La aplicación tiene una interfaz gráfica de usuario (GUI) que muestra diferentes opciones y funcionalidades relacionadas con los equipos y sus estadísticas. Los principales componentes y funcionalidades de la aplicación son los siguientes:

1. Autenticación de usuario:

Al ejecutar la aplicación, se muestra una ventana de autenticación donde los usuarios deben ingresar un nombre de usuario y una contraseña. El nombre de usuario correcto es "poli" y la contraseña correcta es "colombia". Si los datos ingresados son correctos, la ventana de autenticación se cierra y se muestra la interfaz principal de la aplicación. En caso de datos incorrectos, se muestra un mensaje de error.

2. Interfaz principal:

Una vez autenticado, se muestra la interfaz principal que contiene una ventana con título "Liga BetPlay - Futbol Femenino Colombiano 2023". En esta interfaz, hay un menú desplegable llamado "Equipos" que contiene opciones para cada uno de los equipos en la liga. Además, hay una opción llamada "Estadísticas Generales" que muestra estadísticas de todos los equipos.

3. Selección de equipo:

Al hacer clic en uno de los equipos en el menú desplegable "Equipos", se muestra una nueva interfaz que presenta las estadísticas del equipo seleccionado. Las estadísticas incluyen el número de partidos ganados (PG), perdidos (PP), empatados (PE), goles a favor (GF), goles en contra (GC) y puntos acumulados. Estas estadísticas se muestran en una tabla con el nombre del equipo en la primera columna y los valores de las estadísticas en las columnas siguientes. Además, el color de fondo de la tabla cambia según el equipo seleccionado.

4. Estadísticas generales:

Al seleccionar la opción "Estadísticas Generales" en el menú "Equipos", se muestra una tabla con las estadísticas generales de todos los equipos en la liga. Similar a la vista de equipo individual, esta tabla muestra el nombre del equipo y las estadísticas de partidos ganados, perdidos, empatados, goles a favor, goles en contra y puntos acumulados. El color de fondo de la tabla es oscuro para facilitar la lectura.

5. Botón de regreso:

En ambas vistas de estadísticas de equipo individual y estadísticas generales, hay un botón llamado "Regresar al Menú" que permite a los usuarios regresar a la interfaz principal para seleccionar otro equipo o ver las estadísticas generales.

La aplicación proporciona una manera fácil de acceder y visualizar las estadísticas de equipos de fútbol en la liga, permitiendo a los usuarios obtener información sobre el rendimiento de cada equipo en términos de partidos ganados, puntos y otros aspectos clave. La interfaz gráfica intuitiva y la capacidad de navegar entre diferentes equipos y estadísticas generales hacen que esta aplicación sea útil para aquellos interesados en el seguimiento de la liga de fútbol femenino en Colombia.

## CODIGO APP BETPLAY – CampeonatoFutbolApp2

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.HashMap;
import java.util.Map;
```

```
public class CampeonatoFutbolApp2 {
    private static final String[] equipos = {"Millonarios", "Santa Fe", "Nacional", "Tolima", "Pasto"};
    private static final Map<String, String[]> estadisticas = new HashMap<>();
    private static final String USUARIO_CORRECTO = "poli";
    private static final String CONTRASENA_CORRECTA = "colombia";

    public static void main(String[] args) {
        // Inicializar simulación de estadísticas
        estadisticas.put("Millonarios", new String[]{"3", "1", "1", "8", "5"});
        estadisticas.put("Santa Fe", new String[]{"2", "2", "1", "6", "7"});
        estadisticas.put("Nacional", new String[]{"4", "0", "0", "12", "3"});
        estadisticas.put("Tolima", new String[]{"1", "3", "1", "5", "9"});
        estadisticas.put("Pasto", new String[]{"0", "4", "1", "3", "10"});

        // Interfaz de autenticación de usuario
        SwingUtilities.invokeLater(() -> {
            JFrame authFrame = new JFrame("Autenticación");
            authFrame.setSize(300, 150);
            authFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

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

            JLabel userLabel = new JLabel("Usuario:");
            JTextField userField = new JTextField();

            JLabel passLabel = new JLabel("Contraseña:");
            JPasswordField passField = new JPasswordField();

            JButton loginButton = new JButton("Iniciar Sesión");
            loginButton.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    String usuario = userField.getText();
                    String contrasena = new String(passField.getPassword());

                    if (usuario.equals(USUARIO_CORRECTO) && contrasena.equals(CONTRASENA_CORRECTA)) {
                        authFrame.dispose(); // Cerrar ventana de autenticación
                        mostrarInterfazPrincipal();
                    } else {
                        JOptionPane.showMessageDialog(authFrame, "Usuario o contraseña incorrectos", "Error de Autenticación",
                            JOptionPane.ERROR_MESSAGE);
                    }
                }
            });

            authPanel.add(userLabel);
            authPanel.add(userField);
            authPanel.add(passLabel);
            authPanel.add(passField);
            authPanel.add(new JLabel()); // Espacio en blanco
            authPanel.add(loginButton);

            authFrame.add(authPanel);
            authFrame.setVisible(true);
        });

        private static void mostrarInterfazPrincipal() {
            // Lanzar la interfaz gráfica en el hilo de despacho de eventos
        }
    }
}
```



```
SwingUtilities.invokeLater(() -> {
    JFrame frame = new JFrame("Liga BetPlay - Futbol Femenino Colombiano 2023");
    frame.setSize(600, 400);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JPanel panel = new JPanel();
    panel.setLayout(new BorderLayout());

    // Crear menú con los equipos y botón de estadísticas generales
    JMenuBar menuBar = new JMenuBar();
    JMenu menuEquipos = new JMenu("Equipos");

    // Agregar opciones de equipo al menú
    for (String equipo : equipos) {
        JMenuItem menuItem = new JMenuItem(equipo);
        menuItem.addActionListener(new EquipoMenuItemListener(equipo, panel));
        menuEquipos.add(menuItem);
    }

    // Agregar opción de estadísticas generales al menú
    JMenuItem menuItemEstadisticasGenerales = new JMenuItem("Estadísticas Generales");
    menuItemEstadisticasGenerales.addActionListener(new EstadisticasGeneralesMenuItemListener(panel));
    menuEquipos.add(menuItemEstadisticasGenerales);

    // Agregar menú al menú bar
    menuBar.add(menuEquipos);

    // Configurar la interfaz
    frame.setJMenuBar(menuBar);
    frame.add(panel, BorderLayout.CENTER);
    frame.setVisible(true);
});
}
```

```
// Clase interna que maneja la acción de selección de un equipo
static class EquipoMenuItemListener implements ActionListener {
    private final String equipo;
    private final JPanel panel;

    public EquipoMenuItemListener(String equipo, JPanel panel) {
        this.equipo = equipo;
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        // Limpiar el panel y configurar el diseño
        panel.removeAll();
        panel.setBackground(new Color(0, 0, 128));
        panel.setLayout(new BorderLayout());

        // Crear botón para regresar al menú
        JButton botonRegresar = new JButton("Regresar al Menú");
        botonRegresar.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Limpiar el panel y redibujar
                panel.removeAll();
                panel.revalidate();
                panel.repaint();
            }
        });
    }
}
```

```
});
panel.add(botonRegresar, BorderLayout.SOUTH);

// Crear tabla de posiciones
String[] columnNames = {"Equipo", "PG", "PP", "PE", "GF", "GC", "Puntos"};
DefaultTableModel tableModel = new DefaultTableModel(columnNames, 0);

JTable table = new JTable(tableModel);
table.setForeground(Color.WHITE); // Letras en color blanco

// Obtener estadísticas del equipo o "N/A" si no hay información
String[] stats = estadisticas.getOrDefault(equipo, new String[]{"N/A", "N/A", "N/A", "N/A", "N/A"});
int partidosGanados = Integer.parseInt(stats[0]);
int puntos = partidosGanados * 3; // Cada partido ganado da 3 puntos
tableModel.addRow(new Object[]{equipo, stats[0], stats[1], stats[2], stats[3], stats[4], puntos});

// Configurar color de fondo de acuerdo al equipo
if (equipo.equals("Santa Fe")) {
    table.setBackground(Color.RED);
} else if (equipo.equals("Nacional")) {
    table.setBackground(Color.GREEN);
} else if (equipo.equals("Millonarios")) {
    table.setBackground(Color.BLUE);
} else if (equipo.equals("Pasto")) {
    table.setBackground(new Color(139, 69, 19)); // Color café
} else if (equipo.equals("Tolima")) {
    table.setBackground(Color.BLACK); // Color negro
} else {
    table.setBackground(Color.WHITE);
}

// Agregar tabla a un JScrollPane y al panel
JScrollPane scrollPane = new JScrollPane(table);
panel.add(scrollPane, BorderLayout.CENTER);

// Redibujar el panel
panel.revalidate();
panel.repaint();
}
}

// Clase interna que maneja la acción de estadísticas generales
static class EstadisticasGeneralesMenuItemListener implements ActionListener {
    private final JPanel panel;

    public EstadisticasGeneralesMenuItemListener(JPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        // Limpiar el panel y configurar el diseño
        panel.removeAll();
        panel.setBackground(new Color(0, 0, 128));
        panel.setLayout(new BorderLayout());

        // Crear botón para regresar al menú
        JButton botonRegresar = new JButton("Regresar al Menú");
        botonRegresar.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {

```

```
// Limpiar el panel y redibujar
panel.removeAll();
panel.revalidate();
panel.repaint();
}
});
panel.add(botonRegresar, BorderLayout.SOUTH);

// Crear tabla de estadísticas generales
String[] columnNames = {"Equipo", "PG", "PP", "PE", "GF", "GC", "Puntos"};
DefaultTableModel tableModel = new DefaultTableModel(columnNames, 0);

JTable table = new JTable(tableModel);
table.setForeground(Color.WHITE); // Letras en color blanco

// Agregar estadísticas de todos los equipos
for (String equipo : equipos) {
    String[] stats = estadisticas.getDefault(equipo, new String[]{"N/A", "N/A", "N/A", "N/A", "N/A"});
    int partidosGanados = Integer.parseInt(stats[0]);
    int puntos = partidosGanados * 3; // Cada partido ganado da 3 puntos
    tableModel.addRow(new Object[]{equipo, stats[0], stats[1], stats[2], stats[3], stats[4], puntos});
}

// Configurar colores de fondo
table.setBackground(new Color(30, 30, 30)); // Color oscuro
panel.setBackground(new Color(0, 0, 128)); // Color azul

// Agregar tabla a un JScrollPane y al panel
JScrollPane scrollPane = new JScrollPane(table);
panel.add(scrollPane, BorderLayout.CENTER);

// Redibujar el panel
panel.revalidate();
panel.repaint();
}
}
}
```

#### EXPLICACION DEL CODIGO

1. Antes de iniciar se debe importar las librerías:

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.HashMap;
import java.util.Map;
```

Estas importaciones son necesarias para utilizar las clases y funciones proporcionadas por las bibliotecas de Swing y otros paquetes de Java.

2. Clase **CampeonatoFutbolApp2**

Esta clase define toda la aplicación y contiene el método main, que es el punto de entrada de la aplicación.

3. Declaración de Variables

```
private static final String [] equipos = {"Millonarios", "Santa Fe", "Nacional", "Tolima", "Pasto"};
```



```
private static final Map<String, String[]> estadisticas = new HashMap<>();
private static final String USUARIO_CORRECTO = "poli";
private static final String CONTRASENA_CORRECTA = "colombia";
```

Aquí se declaran constantes como los nombres de los equipos, las estadísticas y las credenciales de usuario.

#### 4. Método **main**

Este método es el punto de entrada de la aplicación. Inicializa las estadísticas de los equipos y luego crea una interfaz de autenticación para el usuario.

#### 5. Interfaz de Autenticación

- Se crea un **JFrame** llamado **authFrame** para la ventana de autenticación.
- Se configura el diseño de la ventana con un **GridLayout**.
- Se crean componentes de interfaz de usuario como etiquetas, campos de texto y botones para que el usuario ingrese su nombre de usuario y contraseña.
- Se añade un **ActionListener** al botón "Iniciar Sesión" que verifica las credenciales ingresadas y muestra la interfaz principal si son correctas, o muestra un mensaje de error si no lo son.

#### 6. Método **mostrarInterfazPrincipal**:

Este método crea la interfaz principal después de que el usuario se autentique correctamente.

- Se crea un **JFrame** llamado **frame** para la ventana principal.
- Se configura el diseño de la ventana con un **BorderLayout**.
- Se crea un menú desplegable con opciones de equipos y un botón para estadísticas generales.
- Se añaden **ActionListeners** a cada opción del menú que mostrarán las estadísticas del equipo seleccionado o las estadísticas generales en la interfaz principal.

#### 7. Clase interna **EquipoMenuItemListener**:

- Esta clase interna implementa la interfaz **ActionListener** y maneja la acción de seleccionar un equipo en el menú.
- Al seleccionar un equipo, se muestra una tabla con sus estadísticas, incluyendo partidos ganados, perdidos, empatados, goles a favor, goles en contra y puntos.

#### 8. Clase interna **EstadisticasGeneralesMenuItemListener**:

- Esta clase interna implementa la interfaz **ActionListener** y maneja la acción de seleccionar la opción de estadísticas generales en el menú.
- Muestra una tabla con las estadísticas generales de todos los equipos, incluyendo los mismos datos que en la clase **EquipoMenuItemListener**.

Por lo tanto, este programa crea una aplicación de interfaz gráfica en Java usando la biblioteca Swing. Permite a los usuarios autenticarse, ver estadísticas de equipos individuales y estadísticas generales de la liga de fútbol femenino. Utiliza **JFrame**, **JPanel**, **JMenuBar**, **JMenuItem**, **JTable**, **JScrollPane**, y otros componentes de Swing para crear la interfaz y manejar las interacciones del usuario.

### DIAGRAMA DE CLASES

Un diagrama de clases es una representación visual que describe la estructura estática y las relaciones entre las clases en un sistema de software. Sirve como una herramienta fundamental en el diseño orientado a objetos, ayudando a los desarrolladores y diseñadores a visualizar y comprender cómo las clases se relacionan entre sí, qué atributos y métodos poseen, y cómo se comunican dentro del sistema. Este tipo de diagrama permite modelar la arquitectura y las interacciones esenciales de un programa, lo que facilita la comunicación entre equipos de desarrollo y la toma de decisiones informadas durante todo el ciclo de vida del software. A continuación, se presenta lo correspondiente para el software BetPlay.

```

+-----+
| CampeonatoFutbolApp2 |
+-----+
| - equipos: String[] |
| - estadísticas: Map<String, String[]> |
| - USUARIO_CORRECTO: String |
| - CONTRASEÑA_CORRECTA: String |
+-----+
| + main(args: String[]) |
| + mostrarInterfazPrincipal() |
+-----+

+-----+ +-----+
| EquipoMenuItemListener|<->| ActionListener |
+-----+ +-----+
| - equipo: String | | |
| - panel: JPanel | | |
+-----+ +-----+
| + actionPerformed(e: ActionEvent) |
+-----+

+-----+ +-----+
| EstadísticasGeneralesML|<->| ActionListener |
+-----+ +-----+
| - panel: JPanel | | |
+-----+ +-----+
| + actionPerformed(e: ActionEvent) |
+-----+

```

#### DICCIONARIO DE DATOS

Un diccionario de datos es una herramienta esencial en el diseño y desarrollo de sistemas de información, ya que proporciona una descripción detallada y organizada de los elementos de datos utilizados en un sistema, incluyendo su significado, formato, relaciones y reglas asociadas. Sirve como una referencia centralizada que ayuda a los diseñadores, desarrolladores y usuarios a entender y comunicar de manera clara y consistente cómo se capturan, almacenan, procesan y presentan los datos dentro de una aplicación o base de datos. Al facilitar la comprensión y alineación de los términos y conceptos relacionados con los datos, un diccionario de datos contribuye a la coherencia, calidad y eficiencia en el desarrollo y mantenimiento de sistemas de información.

NOMBRE	TIPO	DESCRIPCIÓN
equipos	String[ ]	Arreglo de nombres de los equipos en el campeonato.
estadísticas	Map<String, String [ ]>	Mapa que almacena las estadísticas de los equipos.
USUARIO_CORRECTO	String	Usuario correcto para la autenticación.
CONTRASEÑA_CORRECTA	String	Contraseña correcta para la autenticación.
authFrame	JFrame	Ventana de autenticación.
authPanel	JPanel	Panel en la ventana de autenticación.
userLabel	JLabel	Etiqueta "Usuario" en la ventana de autenticación.
userField	TextField	Campo de texto para ingresar el usuario.
passLabel	JLabel	Etiqueta "Contraseña" en la ventana de autenticación.
passField	JPasswordField	Campo de contraseña en la ventana de autenticación.
loginButton	JButton	Botón "Iniciar Sesión" en la ventana de autenticación.
frame	JFrame	Ventana principal de la interfaz.

panel	JPanel	Panel principal en la ventana principal.
menuBar	JMenuBar	Barra de menú en la ventana principal.
menuEquipos	JMenu	Menú desplegable "Equipos" en la barra de menú.
menuItem	JMenuItem	Elementos de menú para cada equipo en el menú de equipos.
menuItemEstadisticasGenerales	JMenuItem	Elemento de menú para las estadísticas generales en el menú.
tableModel	DefaultTableModel	Modelo de tabla para mostrar estadísticas.
table	JTable	Tabla para mostrar estadísticas.
botonRegresar	JButton	Botón "Regresar al Menú" en las vistas de estadísticas.
EquipoMenuItemListener	ActionListener	Clase interna para manejar la acción de selección de un equipo.
EstadisticasGeneralesMenuItemListener	ActionListener	Clase interna para manejar la acción de estadísticas generales.

#### ANÁLISIS - PROGRAMACIÓN ORIENTADA A OBJETOS

### POLIMORFISMO

El polimorfismo se refleja en la implementación de las clases internas `EquipoMenuItemListener` y `EstadisticasGeneralesMenuItemListener`. Ambas clases implementan la interfaz `ActionListener` y sobrescriben el método `actionPerformed(ActionEvent e)`. Aunque comparten la misma interfaz, cada clase proporciona una implementación diferente de cómo responder ante un evento, lo que demuestra el concepto de polimorfismo, donde diferentes clases pueden responder de manera distinta a la misma interfaz o método.

### ABSTRACCIÓN

El código demuestra abstracción al encapsular la funcionalidad en clases y métodos con nombres significativos. Por ejemplo, la clase `CampeonatoFutbolApp2` actúa como una abstracción que representa la aplicación de gestión de estadísticas de fútbol. Cada clase interna (como `EquipoMenuItemListener` y `EstadisticasGeneralesMenuItemListener`) también abstrae la lógica específica para manejar acciones de eventos. Esto permite a los desarrolladores interactuar con estas abstracciones en lugar de preocuparse por los detalles internos.

### ENCAPSULAMIENTO


El encapsulamiento se aprecia en el diseño de las clases internas. Cada una de ellas encapsula la funcionalidad relacionada con la interfaz gráfica y la lógica de negocio correspondiente, manteniendo su implementación y propiedades ocultas al exterior. Por ejemplo, `EquipoMenuItemListener` maneja la visualización de estadísticas de equipos y `EstadisticasGeneralesMenuItemListener` maneja la visualización de estadísticas generales, ambos encapsulan la funcionalidad en métodos y estructuras de datos privados.

### HERENCIA

Aunque no se observa una herencia directa en el código proporcionado, es importante mencionar que la interfaz `ActionListener` que implementan las clases internas (`EquipoMenuItemListener` y `EstadisticasGeneralesMenuItemListener`) es una forma de herencia en Java. Estas clases heredan el contrato de la interfaz y deben proporcionar su propia implementación para el método `actionPerformed(ActionEvent e)`, permitiendo la reutilización de la funcionalidad genérica de escucha de eventos en diferentes contextos.



## INTERFAZ



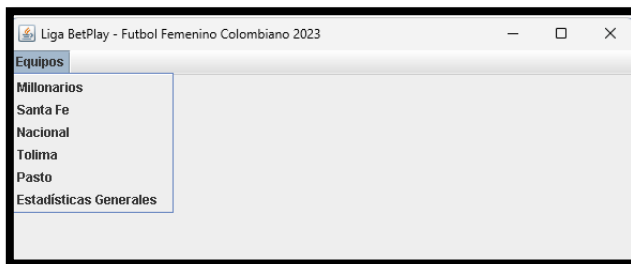
Autenticación

Usuario:

Contraseña:

**Iniciar Sesión**

→ Login



Liga BetPlay - Futbol Femenino Colombiano 2023

**Equipos**

- Millonarios
- Santa Fe
- Nacional
- Tolima
- Pasto
- Estadísticas Generales

→ Menú Interactivo



Liga BetPlay - Futbol Femenino Colombiano 2023

**Equipos**

Equipo	PG	PP	PE	GF	GC	Puntos
Santa Fe	2	2	1	6	7	6

**Regresar al Menú**

→ Informacion de estadísticas del Equipo.



Liga BetPlay - Futbol Femenino Colombiano 2023

**Equipos**

Equipo	PG	PP	PE	GF	GC	Puntos
Millonarios	3	1	1	8	5	9
Santa Fe	2	2	1	6	7	6
Nacional	4	0	0	12	3	12
Tolima	1	3	1	5	9	3
Pasto	0	4	1	3	10	0

**Regresar al Menú**

→ Informacion de Estadísticas Globales