



Esta no es una licencia de Cultura Libre.



De Los Rios Rodriguez, Cristian Camilo  
Correo: [cristian.de.los.rios@pi.edu.co](mailto:cristian.de.los.rios@pi.edu.co)

Programación de Software y aplicativos móviles, Politécnico Internacional  
Autopista Sur No. 67 - 71



1

**Resumen** - “HISTOBET-FEM” es una aplicación para los fanáticos del fútbol profesional colombiano femenino y los que quieran conocer más sobre el mismo. La persona contará con un usuario y contraseña para el acceso con la que podrá acceder y seleccionar diferentes años entre el 2020 y el 2023, los cuales mostrarán en pantalla detalles de las finales de esos años seleccionados, así como el nombre de la goleadora y cantidad de goles anotados por la misma en ese año. Para esto usaremos Netbeans y modelo vista controlador (MVC).

Palabras clave: Modelo Vista controlador (MVC), Pilares de la POO, Entorno gráfico.

**Abstract** – " HISTOBET-FEM" is an application for fans of Colombian professional women's soccer and those who want to know more about it. The person will have a username and password for access with which they can access and select different years between 2020 and 2023, which will show details of the finals of those selected years on the screen, as well as the name of the scorer and amount of goals scored by it in that year. For this we will use Netbeans and a Model View Controller (MVC).

Keywords: Model View Controller (MVC), OOP Pillars, Graphical environment.

## I. INTRODUCCIÓN

Este proyecto se realizará mediante el aplicativo Apache Netbeans IDE 17, mediante arquitectura de modelo vista controlador (MVC), enfocando sus procesos en “Histórico BetPlay femenino”, que permitirá a sus usuarios tener la visualización del histórico de la Liga BetPlay - Fútbol Femenino Colombiano durante los últimos 4 años, en cuanto a finales jugadas y estadísticas de las mismas.

“HISTOBET-FEM” contará con un entorno gráfico que permitirá seleccionar diferentes opciones y funciones asociadas al año que el usuario seleccione como ejemplo:

### 1. Acceso del usuario:

Al iniciar la aplicación, el usuario visualizará una ventana en la que deberá acceder con su respectivo usuario y contraseña. El nombre de usuario será “ciclo3” y la contraseña de acceso será “proyecto”. Si los datos son correctos, podrá acceder al menú principal o en caso de ser errados, el sistema mostrará una alerta para que ingrese con los datos correctos.

### 2. Menú principal:

Luego de iniciar sesión, el usuario podrá visualizar el menú principal del aplicativo llamado "Histórico BetPlay Femenino". En el menú encontrará un listado de por lo menos 4 años descendiendo del 2023 a años anteriores.

### 3. Elección de año:

Al seleccionar uno de los años listados en pantalla, el usuario visualizará ahora en pantalla las estadísticas de las finales jugadas en el respectivo año, sea 2023, 2022, 2021 o 2020. Las estadísticas que aparecerán en pantalla de cada año son: Equipos finalistas, resultado de partido de ida, resultado de partido de vuelta, estadio en que se jugó cada partido, ganador de cada partido, nombre de la goleadora de ese año y cantidad de goles anotados por la misma.

### 4. Botones:

El aplicativo contendrá 2 botones, uno para finalizar el proceso o salir de la aplicación y otro dentro de cada año para retornar al menú principal para elegir otro año.

Se podrá acceder fácilmente al histórico de finales de cada año seleccionado y retornar para elegir cualquier otro año

sin problemas, de esta forma podrán recopilar ágilmente los datos del año que deseen, con un entorno gráfico amigable para cualquier usuario.

## II. CÓDIGO DE HISTOBET-FEM.

Código comentado para su fácil entendimiento, además de un comentario al inicio de cada clase para mayor comprensión.

### 1. Se importan las librerías a utilizar.

```
import java.util.Scanner;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

### 2. Se crea el Main que básicamente funciona para llamar la “ventanainiciodesesion”. Que es el método para el acceso del usuario y contraseña.

```
public class BetplayLigaFemenina_Grafico {

    public static void main(String[] args) {
        // Crear una ventana de inicio de sesión
        SwingUtilities.invokeLater() -> {
            VentanaInicioSesion ventanaInicioSesion = new
            VentanaInicioSesion();
            ventanaInicioSesion.setVisible(true);
        };
    }
}
```

### 3. Se crea la superclase “Anio”, que llamará los métodos de los demás años y especificará los datos de las finales de cada año.

```
package betplayligafemenina_grafico;

public abstract class Anio {
    // Atributos privados para almacenar los detalles del
    año
    private int anio; // El año del torneo
    private Equipo equipoCampeon; // El equipo campeón
    del torneo en ese año
    private Equipo equipoSubcampeon; // El equipo
    subcampeón del torneo en ese año
    private Final finalPartidoIda; // Los detalles del partido
    de ida de la final en ese año
    private Final finalPartidoVuelta; // Los detalles del
    partido de vuelta de la final en ese año
```

```
private Jugadora goleadora; // La jugadora goleadora
del torneo en ese año
```

```
// Constructor para crear un objeto de tipo Anio con los
detalles proporcionados
```

```
public Anio(int anio, Equipo equipoCampeon, Equipo
equipoSubcampeon, Final finalPartidoIda, Final
finalPartidoVuelta, Jugadora goleadora) {
```

```
    this.anio = anio; // Inicializar el año con el valor
proporcionado
```

```
    this.equipoCampeon = equipoCampeon; //
Inicializar el equipo campeón con el valor proporcionado
```

```
    this.equipoSubcampeon = equipoSubcampeon; //
Inicializar el equipo subcampeón con el valor
proporcionado
```

```
    this.finalPartidoIda = finalPartidoIda; // Inicializar
los detalles del partido de ida con el valor proporcionado
```

```
    this.finalPartidoVuelta = finalPartidoVuelta; //
Inicializar los detalles del partido de vuelta con el valor
proporcionado
```

```
    this.goleadora = goleadora; // Inicializar la jugadora
goleadora con el valor proporcionado
}
```

```
// Método abstracto que debe ser implementado en las
subclases para obtener información detallada sobre el año
public abstract String obtenerInfoAnio();
```

```
// Métodos para acceder a los detalles del año
```

```
public int getAnio() {
    return anio;
}
```

```
public Equipo getEquipoCampeon() {
    return equipoCampeon;
}
```

```
public Equipo getEquipoSubcampeon() {
    return equipoSubcampeon;
}
```

```
public Final getFinalPartidoIda() {
    return finalPartidoIda;
}
```

```
public Final getFinalPartidoVuelta() {
    return finalPartidoVuelta;
}
```

```
public Jugadora getGoleadora() {
    return goleadora;
}
```

// Sobrescribir el método toString para obtener información del año al convertir el objeto a cadena de texto

```
@Override
public String toString() {
    return obtenerInfoAnio();
}
```

**4. Se crea la clase “Anio2020” y otras clases con “Anio2021, 2022 y 2023” en la cual se ingresan los datos de las finales de cada año. Se crean métodos “obtenerInfoAnio” para que pueda ser llamado desde otra clase. Clases Anio, extends Anio.**

```
public class Anio2020 extends Anio {

    // Constructor de la subclase Anio2020
    public Anio2020() {
        // Llamada al constructor de la clase base (Anio) para
        // crear un objeto Anio2020 con detalles específicos del año
        // 2020
        super(
            2020, // Año
            new Equipo("Santa Fe"), // Equipo campeón
            new Equipo("América de Cali"), // Equipo
            subcampeón
            new Final(
                new Equipo("América de Cali"), // Equipo local
                new Equipo("Santa Fe"), // Equipo visitante
                1, 2, // Resultado ida
                "Estadio Pascual Guerrero, Cali", // Lugar ida
                "América de Cali" // Equipo ganador ida
            ),
            new Final(
                new Equipo("Santa Fe"), // Equipo local
                new Equipo("América de Cali"), // Equipo
            visitante
            2, 0, // Resultado vuelta
            "Estadio El Campín, Bogotá", // Lugar vuelta
            "Santa Fe" // Equipo ganador vuelta
        ),
            new Jugadora("Ysaura Viso", 13) // Goleadora del
            año 2020
        );
    }

    // Implementación del método abstracto
    obtenerInfoAnio() para mostrar información detallada del
    año 2020
    @Override
    public String obtenerInfoAnio() {
        // Crear una cadena con la información detallada del
        año 2020
```

```
        return "Año 2020:\n" +
            "Campeón: " +
            + getEquipoCampeon().getNombre() + "\n" + // Obtener
            información detallada del campeón
            "Subcampeón: " +
            + getEquipoSubcampeon().getNombre() + "\n" + // Obtener
            información detallada del subcampeón
            "Goleadora: " + getGoleadora().getNombre() + "
            (" + getGoleadora().getGoles() + " goles)\n" + // Obtener
            información detallada del nombre y goles de la goleadora
            "Detalles del partido ida:\n" +
            getFinalPartidoIda().obtenerInfo() + "\n" + //
            Obtener información detallada del partido de ida
            "Detalles del partido vuelta:\n" +
            getFinalPartidoVuelta().obtenerInfo(); // Obtener
            información detallada del partido de vuelta
    }
}
```

**5. Se crea la clase Equipo que se encargará de crear el método para obtener el nombre de los equipos.**

```
public class Equipo {
    private String nombre;

    // Constructor que recibe el nombre del equipo
    public Equipo(String nombre) {
        this.nombre = nombre;
    }

    // Método para obtener el nombre del equipo
    public String getNombre() {
        return nombre;
    }
}
```

**6. Se crea la clase “Final”, que representa los detalles de los partidos de la final en el año que sea solicitado.**

```
public class Final {
    private Equipo equipoLocal;
    private Equipo equipoVisitante;
    private int golesEquipoLocal;
    private int golesEquipoVisitante;
    private String estadio;
    private String ganador;

    // Constructor que recibe los detalles del partido final
```

```

public Final(Equipo equipoLocal, Equipo
equipoVisitante, int golesEquipoLocal, int
golesEquipoVisitante, String estadio, String ganador) {
    this.equipoLocal = equipoLocal;
    this.equipoVisitante = equipoVisitante;
    this.golesEquipoLocal = golesEquipoLocal;
    this.golesEquipoVisitante = golesEquipoVisitante;
    this.estadio = estadio;
    this.ganador = ganador;
}

// Métodos para obtener los detalles del partido final
public Equipo getEquipoLocal() {
    return equipoLocal;
}

public Equipo getEquipoVisitante() {
    return equipoVisitante;
}

public int getGolesEquipoLocal() {
    return golesEquipoLocal;
}

public int getGolesEquipoVisitante() {
    return golesEquipoVisitante;
}

public String getEstadio() {
    return estadio;
}

public String getGanador() {
    return ganador;
}

// Método para obtener una cadena de información
detallada sobre el partido final
public String obtenerInfo() {
    return "Equipo Local: " + equipoLocal.getNombre()
+ "\n" +
    "Equipo Visitante: " +
equipoVisitante.getNombre() + "\n" +
    "Goles Equipo Local: " + golesEquipoLocal +
"\n" +
    "Goles Equipo Visitante: " +
golesEquipoVisitante + "\n" +
    "Estadio: " + estadio + "\n" +
    "Ganador: " + ganador + "\n";
}
}

```

**7. Se crea la clase “Jugadora” que contendrá el método para llamar por el nombre de la jugadora y la cantidad de goles anotados, según el año seleccionado.**

```

public class Jugadora {
    private String nombre;
    private int goles;

    // Constructor que recibe el nombre de la jugadora y la
    cantidad de goles
    public Jugadora(String nombre, int goles) {
        this.nombre = nombre;
        this.goles = goles;
    }

    // Método para obtener el nombre de la jugadora
    public String getNombre() {
        return nombre;
    }

    // Método para obtener la cantidad de goles que ha
    marcado la jugadora
    public int getGoles() {
        return goles;
    }
}

```

**8. Se crea clase “VentanaInformacionAnio”, la cual se encarga de generar un entorno gráfico en una ventana emergente que muestra la información del año solicitado. Adicional, se configura su presentación como tamaño de la ventana, título de la ventana, botón para “salir” y botón para “regresar”.**

```

package betplayligafemenina_grafico.Modelo;

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

public class VentanaInformacionAnio extends JFrame {

    public VentanaInformacionAnio(String tituloVentana,
String informacionAnio) {
        super(tituloVentana);

        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setSize(600, 400);
        setLocationRelativeTo(null);

        // Crear un panel izquierdo en blanco

```



```

JPanel panelIzquierdo = new JPanel();

// Establecer el tamaño del panel izquierdo
panelIzquierdo.setPreferredSize(new
Dimension(150, getHeight()));

// Crear un JPanel para la información con fondo
blanco
JPanel panelInformacion = new JPanel(new
BorderLayout());

// Crear un JTextArea para mostrar la información
del año con fondo transparente
JTextArea textAreaInformacion = new
JTextArea(informacionAnio);
textAreaInformacion.setEditable(false);
textAreaInformacion.setFont(new Font("Arial",
Font.PLAIN, 16));

// Crear paneles en blanco arriba y debajo del texto
JPanel panelArriba = new JPanel();
JPanel panelAbajo = new JPanel();

// Configurar el diseño de la ventana principal con
un BorderLayout
Container container = getContentPane();
container.setLayout(new BorderLayout());

// Agregar el panel izquierdo en el lado izquierdo de
la ventana
container.add(panelIzquierdo,
BorderLayout.WEST);

// Agregar el panel de arriba, el JPanel de
información en el centro y el panel de abajo
container.add(panelArriba,
BorderLayout.NORTH);
container.add(panelInformacion,
BorderLayout.CENTER);
container.add(panelAbajo, BorderLayout.SOUTH);

// Agregar el JTextArea al panel de información
panelInformacion.add(textAreaInformacion,
BorderLayout.CENTER);

// Crear un panel para los botones "Volver" y
"Salir"
JPanel panelBotones = new JPanel();

// Crear un botón "Volver" para cerrar esta ventana
y volver a la ventana anterior
JButton botonAtras = new JButton("Volver");
botonAtras.addActionListener(new
ActionListener() {

```

```

@Override
public void actionPerformed(ActionEvent e) {
    dispose();
}
});

// Crear un botón "Salir" para cerrar toda la
aplicación
JButton botonSalir = new JButton("Salir");
botonSalir.addActionListener(new ActionListener()
{
    @Override
    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }
});

// Agregar los botones "Volver" y "Salir" al panel
de botones
panelBotones.add(botonAtras);
panelBotones.add(botonSalir);

// Agregar el panel de botones en la parte inferior
container.add(panelBotones,
BorderLayout.SOUTH);
}
}

```

**9. Se crea la clase “VentanaInicioSesion” que es la que contendrá los JLabel, JTextField, JPasswordField y JButton, para el menú o pantalla de inicio de sesión del usuario, en el cual el usuario será: ciclo3 y la contraseña: proyecto.**

**Adicional, se crean métodos para la apariencia de la ventana y espacio entre botones.**

```

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

public class VentanaInicioSesion extends JFrame {

    // Constructor de la clase
    public VentanaInicioSesion() {
        // Configuración de la ventana principal
        setTitle("***HISTOBET-FEM***"); // Título de la
        ventana
        setSize(400, 300); // Tamaño de la ventana

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // Cerrar la aplicación al cerrar la ventana
        setLocationRelativeTo(null); // Centrar la ventana en
        la pantalla
    }
}

```

```

        setResizable(false); // Evitar que la ventana sea
redimensionable

// Creación de un panel para organizar los
componentes
JPanel panel = new JPanel();
panel.setLayout(new GridBagLayout()); // Utilizar
un diseño de cuadrícula para organizar los componentes
panel.setBackground(new Color(28, 190, 243)); //
Establecer el color de fondo del panel

// Configuración de restricciones para el diseño de
cuadrícula
GridBagConstraints gbc = new
GridBagConstraints();
gbc.insets = new Insets(10, 10, 10, 10); // Espaciado
interno entre los componentes

// Etiqueta que muestra un mensaje informativo
JLabel labelMensaje = new JLabel(" Inicia sesión
para acceder al histórico BetPlay Femenino");

labelMensaje.setHorizontalAlignment(JLabel.CENTER)
; // Alinear el texto al centro
gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridwidth = 2;
panel.add(labelMensaje, gbc);

// Etiqueta y campo de texto para ingresar el nombre
de usuario
JLabel labelUsuario = new JLabel("Usuario:");
JTextField campoUsuario = new JTextField();
campoUsuario.setPreferredSize(new
Dimension(250, 30)); // Establecer el tamaño del campo
de texto

// Etiqueta y campo de contraseña
JLabel labelContraseña = new
JLabel("Contraseña:");
JPasswordField campoContraseña = new
JPasswordField();
campoContraseña.setPreferredSize(new
Dimension(250, 30)); // Establecer el tamaño del campo
de contraseña

// Botón para iniciar sesión
JButton botonIniciarSesion = new JButton("Iniciar
Sesión");

// Botón para mostrar datos de acceso
JButton botonDatosAcceso = new JButton("Datos
de Acceso");

```

```

        botonDatosAcceso.setBackground(new Color(28,
147, 250)); // Cambiar el color del botón a verde

// Botón para salir de la aplicación
JButton botonSalir = new JButton("Salir");

// Configuración de las posiciones de los
componentes en la cuadrícula
gbc.gridx = 0;
gbc.gridy = 1;
gbc.gridwidth = 1;
panel.add(labelUsuario, gbc);

gbc.gridy = 2;
panel.add(labelContraseña, gbc);

gbc.gridx = 1;
gbc.gridy = 1;
panel.add(campoUsuario, gbc);

gbc.gridy = 2;
panel.add(campoContraseña, gbc);

gbc.gridx = 0;
gbc.gridy = 3;
gbc.gridwidth = 2;
panel.add(botonIniciarSesion, gbc);

gbc.gridy = 4;
gbc.insets = new Insets(5, 10, 5, 10); // Espacio entre
el botón "Datos de Acceso" y "Salir"
panel.add(botonDatosAcceso, gbc);

gbc.anchor = GridBagConstraints.SOUTHEAST; //
Alinear el botón "Salir" en la esquina inferior derecha
gbc.gridy = 5;
panel.add(botonSalir, gbc);

// Acción del botón "Iniciar Sesión"
botonIniciarSesion.addActionListener(new
ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Obtener el usuario y la contraseña ingresados
        String usuario = campoUsuario.getText();
        char[] contraseña =
        campoContraseña.getPassword();
        String contraseñaStr = new String(contraseña);

        // Verificar si el usuario y la contraseña son
correctos
        if (usuario.equals("ciclo3") &&
        contraseñaStr.equals("proyecto")) {

```

```

        dispose(); // Cerrar la ventana de inicio de
sesión
        VentanaMenu ventanaMenu = new
VentanaMenu(); // Crear una ventana de menú
        ventanaMenu.setVisible(true); // Mostrar la
ventana de menú
    } else {
        JOptionPane.showMessageDialog(null,
"Usuario o contraseña incorrectos", "Error de inicio de
sesión", JOptionPane.ERROR_MESSAGE);
    }
}
});

// Acción del botón "Datos de Acceso"
botonDatosAcceso.addActionListener(new
ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Mostrar un mensaje con los datos de acceso
        JOptionPane.showMessageDialog(null, "El
usuario es: ciclo3\nLa contraseña es: proyecto", "Datos de
Acceso", JOptionPane.INFORMATION_MESSAGE);
    }
});

// Acción del botón "Salir"
botonSalir.addActionListener(new ActionListener()
{
    @Override
    public void actionPerformed(ActionEvent e) {
        System.exit(0); // Cerrar la aplicación
    }
});

add(panel); // Agregar el panel a la ventana principal
}

// Método principal para ejecutar la aplicación
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new VentanaInicioSesion(); // Crear y mostrar
la ventana de inicio de sesión
        }
    });
}
}

```

**10. Se crea la clase: “VentanaMenu”, que será el entorno gráfico donde se encontrará el menú con los años “botones” a seleccionar. Estos botones de cada año estarán asociados al método**

**“mostrarVentanaInformacionAnio” para que, al hacer clic en ellos, se abra la ventana emergente con la información correspondiente a cada año. Se usa el JFrame para la ventana emergente de cada año. Se adiciona el botón de créditos, se ajustan tamaños y se centra el espacio.**

```

package betplayligafemenina_grafico.Vista;

import betplayligafemenina_grafico.controlador.Anio;
import
betplayligafemenina_grafico.controlador.Anio2020;
import
betplayligafemenina_grafico.controlador.Anio2021;
import
betplayligafemenina_grafico.controlador.Anio2022;
import
betplayligafemenina_grafico.controlador.Anio2023;
import
betplayligafemenina_grafico.Modelo.VentanaInformacio
nAnio;

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

public class VentanaMenu extends JFrame {

    public VentanaMenu() {
        // Configuración de la ventana principal
        setTitle("Histórico HISTOBET-FEM"); // Título de
la ventana

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // Cerrar la aplicación al cerrar la ventana
        setSize(600, 500); // Tamaño de la ventana
        setLocationRelativeTo(null); // Centrar la ventana en
la pantalla
        setResizable(false); // Evitar que la ventana sea
redimensionable

        // Establecer el color de fondo de la ventana
        getContentPane().setBackground(new Color(28,
190, 243));

        // Contenedor para el título
        JPanel panelTitulo = new JPanel();
        panelTitulo.setBackground(new Color(28, 190,
243));

        // Contenedor para el mensaje
        JPanel panelMensaje = new JPanel();
    }
}

```

```

panelMensaje.setBackground(new Color(28, 190,
243));

// Contenedor para los botones
JPanel panelBotones = new JPanel(new
GridBagLayout());
panelBotones.setBackground(new Color(28, 190,
243));

// Crear botones
JButton[] botones = new JButton[4];
for (int i = 0; i < 4; i++) {
    botones[i] = crearBoton("Año " + (2020 + i));
    // Establecer el tamaño preferido de los botones
    botones[i].setPreferredSize(new Dimension(200,
40));
}

// Botón de créditos
JButton botonCreditos = crearBoton("Créditos");
botonCreditos.setBackground(new Color(198, 148,
3)); // Cambiamos el color de fondo del botón Créditos
// Establecer el tamaño del botón de créditos
botonCreditos.setPreferredSize(new
Dimension(200, 40));

// Botón de salir
JButton botonSalir = crearBoton("Salir");
botonSalir.setBackground(new Color(192, 192,
192)); // Color gris plata
botonSalir.setForeground(Color.RED); // Cambiar
el color del texto a rojo
// Establecer el tamaño del botón de salir
botonSalir.setPreferredSize(new Dimension(200,
40));

// Agregar acciones a los botones de año
for (int i = 0; i < 4; i++) {
    final int anio = 2020 + i;
    botones[i].addActionListener(new
ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            mostrarInformacionAnio(anio);
        }
    });
}

// Agregar acción al botón de créditos
botonCreditos.addActionListener(new
ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        mostrarCreditos();
    }
}

```

```

    }
});

// Agregar acción al botón de salir
botonSalir.addActionListener(new ActionListener()
{
    @Override
    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }
});

// Crear etiquetas de bienvenida y mensaje
JLabel titulo = new JLabel("Bienvenido a
HISTOBET-FEM");
titulo.setFont(new Font("Arial", Font.BOLD, 18));

titulo.setHorizontalAlignment(SwingConstants.CENTE
R);
titulo.setForeground(Color.BLUE);

JLabel mensaje = new JLabel("Elija el año del cual
desea saber los detalles de la final");
mensaje.setFont(new Font("Arial", Font.BOLD,
14));

mensaje.setHorizontalAlignment(SwingConstants.CENT
ER);
mensaje.setForeground(Color.BLACK);

mensaje.setBorder(BorderFactory.createEmptyBorder(1
0, 0, 20, 0)); // Agrega espacio alrededor del mensaje

// Configurar las restricciones para centrar los
botones
GridBagConstraints gbc = new
GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 0;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.insets = new Insets(10, 0, 10, 0); // Espacio entre
los botones

// Agregar los botones al panel de botones
for (int i = 0; i < 4; i++) {
    panelBotones.add(botones[i], gbc);
    gbc.gridy++; // Avanzar a la siguiente fila
}
panelBotones.add(botonCreditos, gbc);
gbc.gridy++;
panelBotones.add(botonSalir, gbc);

// Agregar etiquetas y paneles de botones a los
contenedores

```



```

panelTitulo.add(titulo);
panelMensaje.add(mensaje);

// Agregar los contenedores al contenedor principal
Container container = getContentPane();
container.setLayout(new BorderLayout());
container.add(panelTitulo, BorderLayout.NORTH);
container.add(panelMensaje,
BorderLayout.CENTER);
container.add(panelBotones,
BorderLayout.SOUTH);

setVisible(true);
}

// Método para crear botones con un estilo común
private JButton crearBoton(String texto) {
    JButton boton = new JButton(texto);
    boton.setFont(new Font("Arial", Font.ITALIC, 16)); //
    Aumentamos el tamaño de la fuente

    // Establecer el fondo del botón
    boton.setBackground(new Color(28, 147, 250));
    boton.setForeground(Color.WHITE); // Cambiamos
    el color del texto a blanco
    boton.setFocusPainted(false); // Quitamos el efecto
    de enfoque

    return boton;
}

// Método para mostrar información de un año
específico
private void mostrarInformacionAnio(int anio) {
    Anio anioSeleccionado = null;
    switch (anio) {
        case 2020:
            anioSeleccionado = new Anio2020();
            break;
        case 2021:
            anioSeleccionado = new Anio2021();
            break;
        case 2022:
            anioSeleccionado = new Anio2022();
            break;
        case 2023:
            anioSeleccionado = new Anio2023();
            break;
    }
    if (anioSeleccionado != null) {
        mostrarVentanaInformacionAnio(anioSeleccionado);
    }
}

```

```

// Método para mostrar la ventana de información de un
año
private void mostrarVentanaInformacionAnio(Anio
anio) {
    VentanaInformacionAnio ventanaInformacionAnio
= new VentanaInformacionAnio("Información del Año",
anio.obtenerInfoAnio());
    ventanaInformacionAnio.setVisible(true);
}

// Método para mostrar los créditos
private void mostrarCreditos() {
    JOptionPane.showMessageDialog(this,
"Créditos:\nCristian De Los
Rios\ncristian.de.los.rios@pi.edu.co");
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new VentanaMenu();
        }
    });
}
}

```

### III. PSEUDO CÓDIGO

#### Pseudocódigo Proyecto con listas:

Figura 1. Main

```

Inicio

Definir una lista de equipos (equiposLiga) de tipo Lista de Cadenas

Definir una lista de detalles de campeonatos (detallesCampeonatos) de tipo Lista de Mapas

Repetir
    Mostrar "BIENVENIDO A SU HISTORICO BETPLAYFEM2023"
    Mostrar Menú:
        Mostrar "1. Subcampeones de los últimos 4 años"
        Mostrar "2. Total de Equipos que participaron"
        Mostrar "3. Datos de cada Año"
        Mostrar "4. Salir"
        Leer opción desde el teclado

    Según opción Hacer
        Caso 1:
            Llamar al método MostrarSubcampeones()
        Caso 2:
            Llamar al método MostrarEquipos()
        Caso 3:
            Llamar al método MostrarDetallesCampeonatos()
        Caso 4:
            Mostrar "BYE!"
        Defecto:
            Mostrar "Opción no válida. Por favor, seleccione una opción válida."

Fin Según

```

Figura 2. Pseudocódigo

```

Mientras opción sea diferente de 4
|
Fin

Método MostrarSubcampeones()
  Para cada equipo en equiposLiga Hacer
    Mostrar "Subcampeón: " + equipo
  Fin Para
Fin Método

Método MostrarEquipos()
  Para cada equipo en equiposLiga Hacer
    Mostrar "Equipo: " + equipo
  Fin Para
Fin Método

Método MostrarDetallesCampeonatos()
  Para cada detalleCampeonato en detallesCampeonatos Hacer
    Mostrar "Información del Campeonato:"
    Mostrar "Campeón: " + detalleCampeonato["campeon"]
    Mostrar "Subcampeón: " + detalleCampeonato["Subcampeon"]
    Mostrar "Partido Ida: " + detalleCampeonato["Partido Ida"]
    Mostrar "Partido Vuelta: " + detalleCampeonato["Partido vuelta"]
    Mostrar "Total Goles temporada: " + detalleCampeonato["Goles"]
  Fin Para
Fin Método

Fin

```

```

CASO 3:
  anioSeleccionado <- ObtenerInformacionAnio(2022)
  MOSTRAR anioSeleccionado

CASO 4:
  anioSeleccionado <- ObtenerInformacionAnio(2023)
  MOSTRAR anioSeleccionado

CASO 5:
  MOSTRAR "Saliendo del programa."

DE OTRO MODO:
  .
  .

DE OTRO MODO:
  MOSTRAR "Opción no válida. Por favor, seleccione una opción válida."

FIN SEGUN

  HASTA QUE opcionMenu = 5 // Salir del programa si se selecciona la opción 5

FIN

```

#### IV. DIAGRAMA UML Y DE FLUJO

##### Pseudocódigo, proyecto entorno gráfico:

Figura 3. pseudocodigo

```

INICIO
  // Declarar variables
  ANIO anioSeleccionado
  MENSAJE mensajeInicio
  OPCION opcionMenu

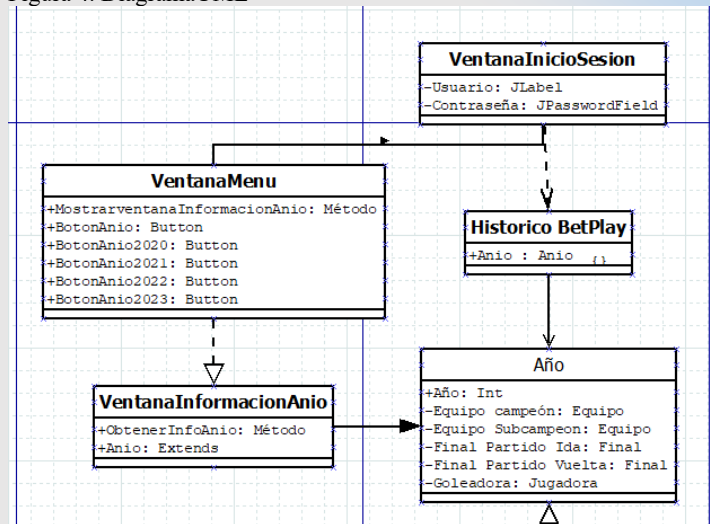
  // Mostrar mensaje de bienvenida
  mensajeInicio <- "Bienvenido al Historico BetPlay Femenino"
  MOSTRAR mensajeInicio

  // Bucle principal del programa
  REPETIR
    // Mostrar menú de opciones
    MOSTRAR "Menú Principal:"
    MOSTRAR "1. Ver información del año 2020"
    MOSTRAR "2. Ver información del año 2021"
    MOSTRAR "3. Ver información del año 2022"
    MOSTRAR "4. Ver información del año 2023"
    MOSTRAR "5. Salir"
    LEER opcionMenu

    // Realizar acciones según la opción seleccionada
    SEGUN opcionMenu
      CASO 1:
        anioSeleccionado <- ObtenerInformacionAnio(2020)
        MOSTRAR anioSeleccionado
      CASO 2:
        anioSeleccionado <- ObtenerInformacionAnio(2021)
        MOSTRAR anioSeleccionado

```

Figura 4. DiagramaUML



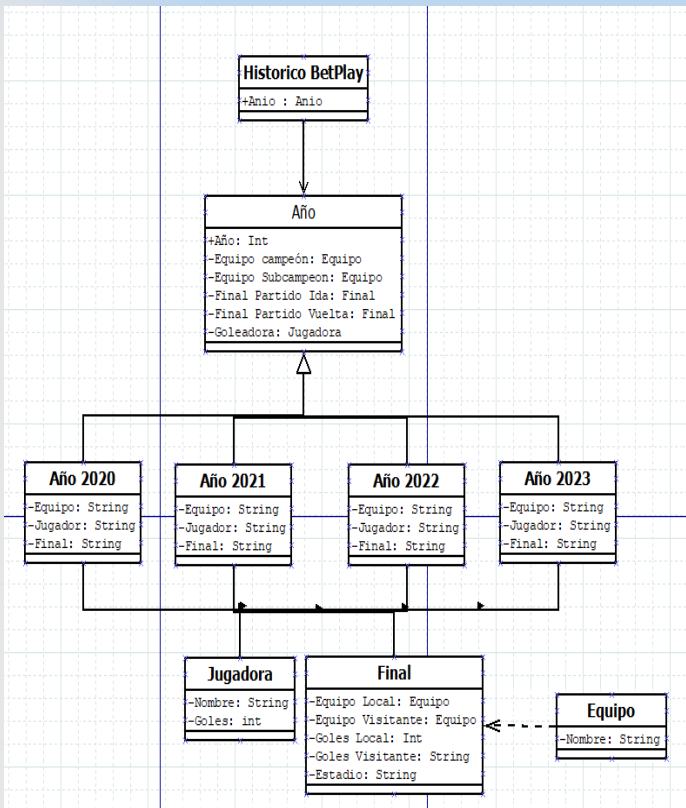
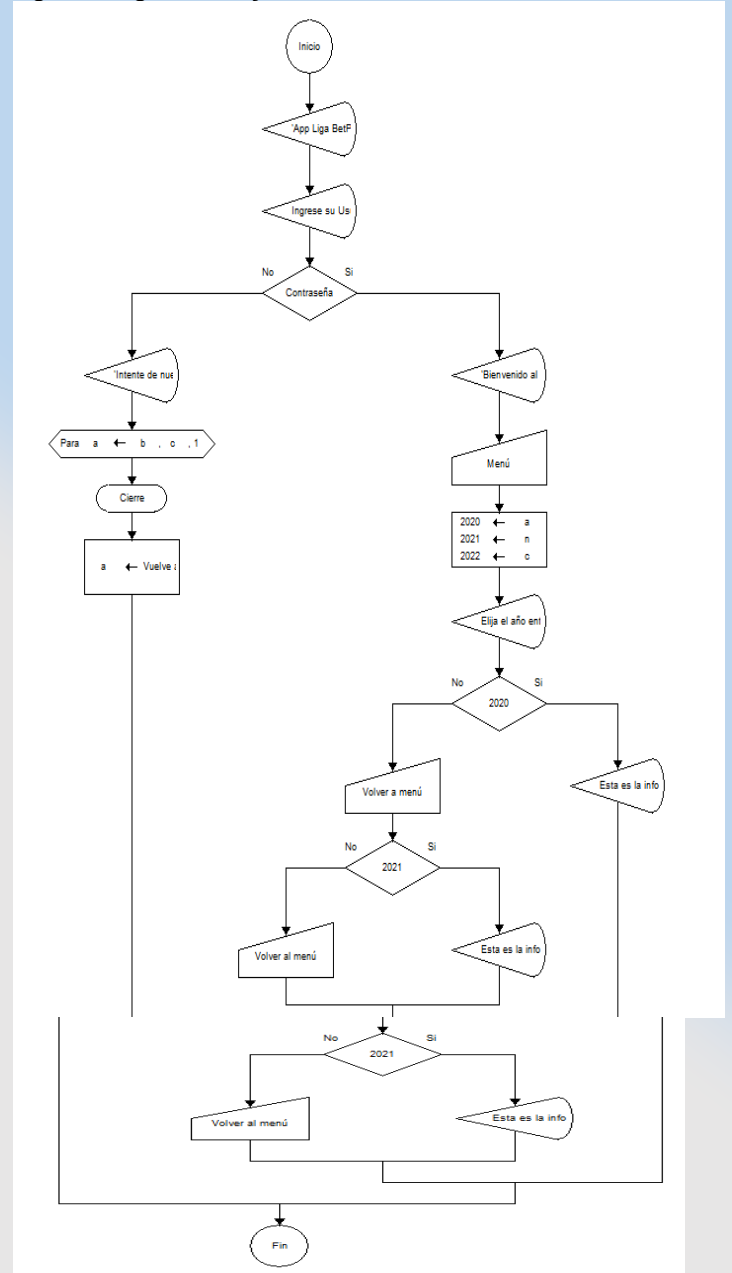


Figura 5 DiagramaDeFlujo



## V. DICCIONARIO DE DATOS

Tabla 1. Diccionario datos, creación propia.

Diccionario de datos		
Nombre	Tipo de dato	Descripción
Scanner	Scanner	Para leer el dato ingresado por el usuario
opcion	Int	Entero para las opciones del menú
do	do	Ciclo para el menú
opcion	switch	Crea el menú

opcion	while	Cierra el ciclo del menú
nombre	String	Nombre del equipo
nombre	String	Nombre de la goleadora
goles	Int	Cantidad de goles de la goleadora
golesLocal	Int	Cantidad de goles anotados por el equipo local en la final
golesVisitante	Int	Cantidad de goles anotados por el equipo visitante en la final
estadio	String	Estadio donde se jugó la final
ganador	String	Equipo que ganó la final
anio	Int	Represente el año en el que se jugó la final
Equipo	clase	Representa un equipo participante en la final del torneo.
Jugadora	clase	Representa una jugadora destacada en el torneo, con nombre y cantidad de goles.
Final	clase	Representa los detalles de una final, incluyendo los equipos participantes, los resultados de los partidos de ida y vuelta, el lugar y el equipo ganador.
obtenerInfoAnio	Método	Método abstracto en la clase "Anio" que devuelve una cadena con información detallada del año.
obtenerInfo	Método	Método en la clase "Final" que devuelve una cadena con información

		detallada sobre una final.
BetPlayLigaFemenina	Clase (principal)	Representa el punto de entrada de la aplicación y contiene el bucle principal para mostrar el menú de años y detalles.
mostrarInfoAnio	Método	Método en la clase "BetPlayLigaFemenina" que muestra información detallada sobre un año específico.
VentanaInformacionAnio	clase	Ventana del entorno gráfico en donde es llamado el detalle de cada año.
textoInformacion	JTextArea	Espacio para ingresar texto
scrollPane	JScrollPane	Panel para deslizar de ser necesario
botonSalir	JButton	Botón para salir del aplicativo
panelBotones	JPanel	Contenedor para seccionar el menú.
Container	container	Genera espacios no visibles para seccionar en cuadrículas o dividir el menú
javax.swing.*	Biblioteca	Biblioteca de entorno gráfico
labelUsuario	JLabel	Campo de texto predefinido para que se visualice "Usuario"
campoUsuario	JTextField	Para que el usuario ingrese su "usuario".
labelContraseña	JLabel	Campo predefinido para que se visualice la palabra "Contraseña".
campoContraseña	JPasswordField	Espacio para ingresar la contraseña



botonIniciarSesion	JButton	Botón de inicio de sesión
gbc	GridBagConstraints	Cuadriculas dentro del panel.
contrasenaStr	String	Recibe la contraseña ingresada
contrasena	char	Muestra asteriscos en el campo de contraseña
setTitle	setTitle	Define un título
panelBotones	JPanel	Crea un panel para ajustar allí los botones
titulo	JLabel	Predefine un título para la sección de menú
mensaje	JLabel	Muestra un mensaje de bienvenida en el menú.
botonAnio2020	JButton	Botón para el año 2020, se crea uno para cada año, hasta el 2023.
Dimension	buttonSize	Se generan dimensiones para los botones

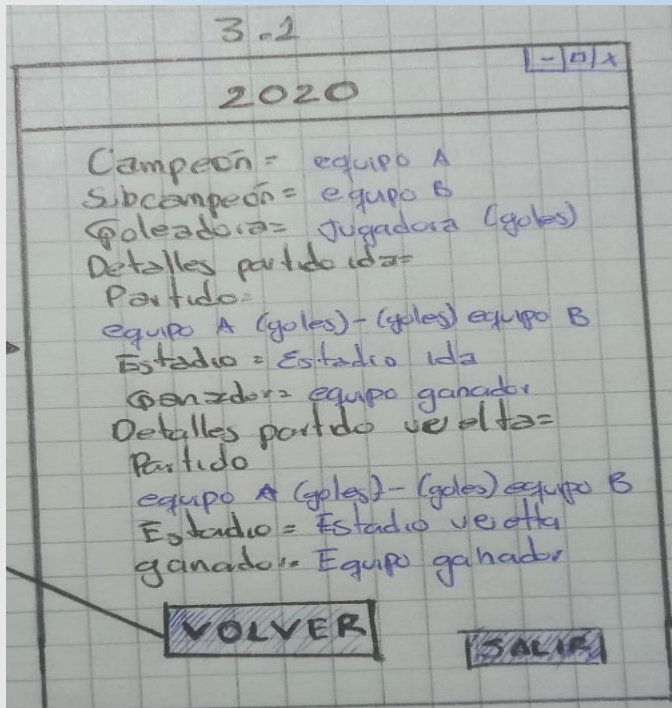
## VI. MOCKUPS.

Figura 6. Singin

Figura 7. Menú

Figura 8. Vista general

Figura 9. Descrip año.



## VII. CONCEPTOS.

**POLIMORFISMO:** Permite que diferentes clases se comporten de manera similar mediante la implementación de métodos en común. Con esto se puede reutilizar el código y genera flexibilidad en el mismo.

En este proyecto, un ejemplo claro de polimorfismo se encuentra en la implementación del método **obtenerInfoAnio()** en cada una de las subclases **Anio2020**, **Anio2021**, **Anio2022** y **Anio2023**. Aunque estos métodos están definidos en la clase abstracta **Anio**, cada subclase los implementa de manera específica para mostrar la información detallada de cada año en el torneo. Sin embargo, desde el punto de vista del método **mostrarInfoAnio** en el **BetPlayLigaFemenina**, se puede llamar al método **obtenerInfoAnio()** en cualquier objeto de tipo **Anio**, independientemente de su subclase, lo que demuestra el polimorfismo.

**ABSTRACCIÓN:** Consiste en simplificar y representar objetos del mundo real en modelos concretos y manejables en el código. En esencia, la abstracción implica centrarse en los aspectos esenciales de un objeto y ocultar los detalles innecesarios para lograr una representación más clara y eficiente.

La clase abstracta **Anio** es un ejemplo de abstracción, ya que encapsula los detalles generales de cada año en el

torneo, como el equipo campeón, el equipo subcampeón, los detalles de las finales y la jugadora goleadora. Aunque cada año tiene detalles únicos, la clase **Anio** proporciona una representación abstracta de estos detalles comunes.

**HERENCIA:** permite crear nuevas clases basadas en clases existentes, compartiendo sus atributos y métodos, extendiendo o modificando su comportamiento según requiera. Las clases derivadas heredan las características de la clase base, lo que promueve la reutilización de código y la organización por jerarquía de objetos.

En mi proyecto, la herencia se encuentra en la relación entre las clases base y las subclases que extienden sus características. Ejemplos de herencia en tu proyecto son:

La clase **Anio** es una clase base abstracta que proporciona una estructura general para representar un año en el torneo. Las subclases como **Anio2020**, **Anio2021**, **Anio2022** y **Anio2023** heredan de **Anio** y extienden su funcionalidad específica para cada año.

**ENCAPSULAMIENTO:** Permite definir los atributos y métodos de una clase de manera que solo puedan ser accedidos y modificados de acuerdo con las reglas establecidas por la clase, protegiendo así la integridad y la consistencia de los datos.

En mi proyecto, el encapsulamiento se encuentra en la definición de las clases y en cómo se controla el acceso a los atributos y métodos de estas clases. Aquí hay algunos ejemplos de encapsulamiento en tu proyecto:

En la clase **Equipo**, los atributos son privados, lo que significa que solo pueden ser accedidos y modificados desde dentro de la clase. El encapsulamiento asegura que estos atributos solo sean modificados de acuerdo con la lógica interna de la clase, evitando modificaciones no deseadas y garantizando la consistencia de los datos.

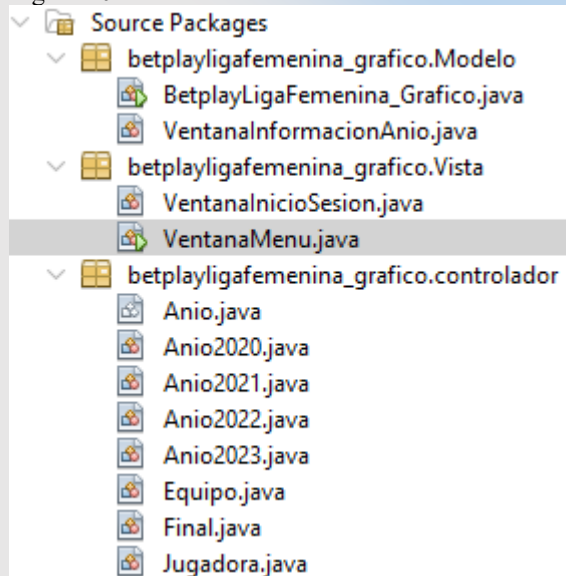
En la clase **Jugadora**, los atributos **nombre**, **goles** y **equipo** son privados, lo que significa que solo pueden ser accedidos y modificados mediante métodos públicos definidos en la misma clase. Esto controla el acceso a la información de las jugadoras y permite establecer reglas para la asignación de goles y la asociación con un equipo.

**MODELO VISTA CONTROLADOR:** MVC es un acrónimo que significa Modelo-Vista-Controlador. Es usado para separar y organizar el código en tres componentes principales, lo que facilita la gestión y mantenimiento de las aplicaciones.

En este proyecto es usado directamente para definir las diferentes clases en sus respectivos ámbitos como

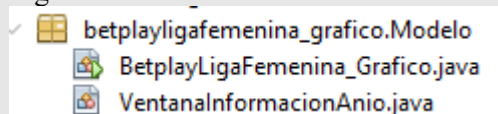
“modelo, lo que une a las diferentes clases, “vista” la sección que conlleva a lo relacionado con la visualización en pantalla y “controlador” con todo lo relacionado al código de funcionamiento:

Figura 10. mvc



**Modelo:** Representa los datos y la lógica de la aplicación. Contiene la estructura de datos, la lógica para acceder y manipular dichos datos.

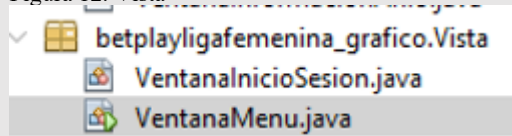
Figura 11. modelo



Contiene el Main y la clase VentanaInfoAnio que enlazan a todas las demás clases para que funcionen en conjunto.

**Vista:** Presenta los datos al usuario y a su vez la interfaz. Representa cómo se visualiza y se interactúa con los datos.

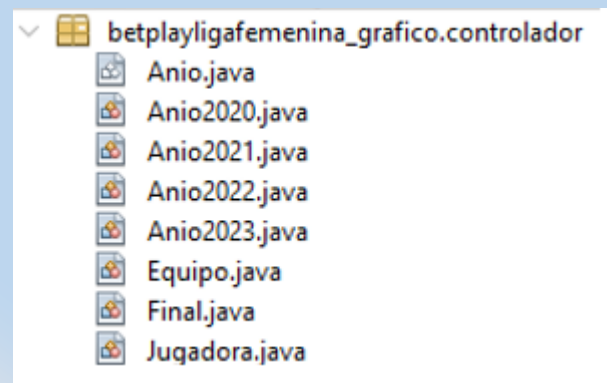
Figura 12. Vista



Contiene a las clases VentanaInicioSesion y VentanaMenú, que son básicamente las 2 clases que contienen todo lo referente al entorno grafico del proyecto.

**Controlador:** Es el intermediario entre el modelo y la vista. Procesa las solicitudes del usuario desde la vista, interactúa con el modelo para obtener o actualizar los datos, y actualiza la vista con los resultados.

Figura 13. controlador



Este controlador, contiene todas las demás clases que son las encargadas de contener el código de funcionamiento del proyecto.

Cada uno de estos componentes se centra en tareas específicas para facilitar la comprensión, el mantenimiento y las actualizaciones del código.

Ya que todo está separado, favorece la re utilización del código en diferentes partes de la aplicación. Adicional los componentes, al estar separados, son más fáciles de probar y esto mejora enormemente la calidad del Software.

Se puede trabajar desde diferentes equipos de trabajo a la vez en diferentes partes del código, sin afectar los avances de los otros colaboradores.

**LISTA ABSTRACTA:** Estas encapsulan operaciones de las listas para simplificar su implementación. Define el comportamiento de una estructura de datos sin detallar su implementación. Esto ayuda a la modularidad y la reutilización del código.

**CLASE INTERNA DE ENLACE:** Las clases internas son clases anidadas dentro de otras clases o métodos, que conectan a su vez con otra clase o subclase. Se caracteriza por su eficiencia en la inserción y eliminación de datos.

**LISTA GENERICA:** Una lista es genérica cuando se pueden insertar o extraer datos de cualquier parte de la lista. Pueden manejar datos variables evitando así la redundancia en la implementación de estructuras de datos.

En nuestro proyecto, la podemos encontrar en:



Figura 14. Lista Gen.

```
public static void main(String[] args) {
    // Crear una "Lista Genérica Diversa" para los Equipos finalistas
    List<Map<String, Object>> Anios = new ArrayList<>();

    System.out.println();//agrega una línea en blanco

    // Agregar información de participantes para varios años
    Map<String, Object> Anio2023 = new HashMap<>();
    Anio2023.put("campeon", "SantaFe");
    Anio2023.put("Subcampeon", "AmericaDeCali");
    Anio2023.put("Partido Ida", "SantaFe 2 - 0 AmericaDeCali");
    Anio2023.put("Partido vuelta", "America de Cali 0 - 0 Santa Fe");
    Anio2023.put("Goles", 11);
    Anios.add(Anio2023);

    Map<String, Object> Anio2022 = new HashMap<>();
    Anio2022.put("campeon", "AmericaDeCali");
    Anio2022.put("Subcampeon", "Deportivo Cali");
    Anio2022.put("Partido Ida", "Deportivo Cali 2 - 1 America de Cali");
    Anio2022.put("Partido vuelta", "America de Cali 3 - 1 Deportivo Cali");
    Anio2022.put("Goles", 15);
    Anios.add(Anio2022);
}
```

**LISTA TIPADA:** Estas restringen la lista a un tipo de dato específico como lo puede ser, cadena de texto, entero, Booleano o demás, lo que previene error en los mismos durante la compilación. Con lo anterior, se asegura que solo los elementos compatibles sean almacenados en las listas, generando confiabilidad y el fácil mantenimiento del código.

En nuestro código se encuentra en:

Figura 15. Lista Tip

```
public class TotalEquipos {
    public static void main(String[] args) {
        // Crear una lista tipada para almacenar cadenas de caracteres
        List<String> listaTipada = new ArrayList<>();

        // Agregar elementos a la lista
        listaTipada.add("America De Cali");
        listaTipada.add("Deportivo Cali");
        listaTipada.add("SantaFe");
    }
}
```

**LISTA DE POSICIÓN ORDINAL:** Esta permite llevar un seguimiento ordenado de los elementos de la lista, numerándolos desde el inicio hasta el final de la misma. Es ideal para realizar Rankings, histórico de transacciones o registro de actividades.

En nuestro proyecto la podemos encontrar en:

Figura 16. Lista PO.

```
public class SUBCAMPEONES {
    public static void main(String[] args) {
        // Crear una "Lista Posición Ordinal" para los subcampeones de los últimos 4 años
        List<String> Subcampeon = new ArrayList<>();
        Subcampeon.add("America De Cali");
        Subcampeon.add("Deportivo Cali");
        Subcampeon.add("SantaFe");
        Subcampeon.add("America De Cali");

        System.out.println(); // Agrega una línea en blanco
    }
}
```

**MÉTODO DE INTERACCIÓN:** Esta permite recorrer y manipular las colecciones de datos de manera eficiente y entendible. Pueden ayudar en la simplificación de tareas como filtrar datos, lo que reduce la necesidad de bucles. métodos, como "forEach" y "map", mejoran la mantenibilidad y comprensión del código, al tiempo que promueven la escritura de programas más eficientes y funcionales.

Figura 23. Met. Iteración,

```
// método de iteración
public class Posicionfinalistas {
    public static void main(String[] args) {
        // Crear una lista con los equipos finalistas de fútbol
        System.out.println(""); //Se deja espacio para separar del título
        System.out.println("****Aquí verá al equipo que ocupó el tercer puesto en el 2023****");
        System.out.println(""); //Se deja espacio para separar del título

        List<Equipofinalista> Equipo = new ArrayList<>();
        Equipo.add(new Equipofinalista("America De Cali", "Tercer Lugar"));
        Equipo.add(new Equipofinalista("Millonarios", "Segundo Lugar"));
        Equipo.add(new Equipofinalista("SantaFe", "Primer Lugar"));

        // Método para buscar un equipo de fútbol por su posición
        String posicionEquipoBuscado = "Tercer Lugar";
        Optional<Equipofinalista> EquipoBuscado = buscarEquipoPorPosicion(Equipo, posicionEquipoBuscado);

        // Mostrar información del equipo de fútbol buscado
        if (EquipoBuscado.isPresent()) {
            Equipofinalista EquipoEncontrado = EquipoBuscado.get();
            System.out.println(EquipoEncontrado.nombre + " quedó en " + EquipoEncontrado.posicion);
        } else {
            System.out.println("No se encontró ningún equipo en la posición " + posicionEquipoBuscado);
        }
    }

    // Método para buscar un equipo de fútbol por su posición
    private static Optional<Equipofinalista> buscarEquipoPorPosicion(List<Equipofinalista> listaEquipos, String posicion) {
        return listaEquipos.stream()
            .filter(Equipo -> Equipo.posicion.equals(posicion))
            .findFirst();
    }
}
```



**PILA:** Son implementadas para mantener el flujo de ejecución en la programación. Se mantiene mediante las operaciones “push” para agregar datos y “pop” para retirar los mismos y que el mismo sistema los clasifique ordenadamente lo que permite controlar y gestionar estados. En esta se mantiene el proceso que el último dato en entrar, es el primero en salir.

Figura 22. Pila

```
public class TotalGolesLiga {
    public static void main(String[] args) {
        // Crear una pila llamada "TotalGolesLiga" para simular el total de goles anotados por los equipos

        System.out.println(""); // Se deja espacio para separar del titulo
        System.out.println("Aquí verá el total de goles del último torneo");
        System.out.println(""); // Se deja espacio para separar del titulo

        Stack<Integer> TotalGoles = new Stack<>();
        TotalGoles.push(10); // goles del equipo America De cali
        TotalGoles.push(25); // goles del equipo SantaFe
        TotalGoles.push(10); // goles del equipo Atletico Nacional
        TotalGoles.push(29); // goles del equipo Pereira
        TotalGoles.push(17); // goles del equipo Deportivo Cali
        TotalGoles.push(7); // goles del equipo Independiente Medellin
        TotalGoles.push(3); // goles del equipo Equidad
        TotalGoles.push(10); // goles del equipo Millonarios
        TotalGoles.push(12); // goles del equipo Llaneros Femenina
        TotalGoles.push(12); // goles del equipo Junior
        TotalGoles.push(12); // goles del equipo Boyacá Chicó Femenina
        TotalGoles.push(10); // goles del equipo Huila
        TotalGoles.push(10); // goles del equipo RST
        TotalGoles.push(13); // goles del equipo Pasto
        TotalGoles.push(12); // goles del equipo Bucaramanga
        TotalGoles.push(6); // goles del equipo Tolima

        // Calcular los goles anotados en este año 2023
        int goles = 0;
        while (!TotalGoles.isEmpty()) {
            goles += TotalGoles.pop();
        }

        // Mostrar los Goles totales anotados en la LigaBetPlayFem 2023
        System.out.println("El total de goles anotados por los equipos: " + goles + " goles");
    }
}
```

**JTABLE:** Este es un componente visual de Java que nos permite dibujar una tabla, de forma que en cada fila/columna de la tabla podamos poner el dato que queramos; un nombre, un apellido, una edad, un número, etc.

**JLIST:** Es un componente de Java que permite que permite seleccionar uno o más elementos de una lista, o un grupo de elementos para elegir.

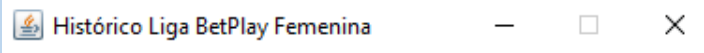
**JPROGRESSBAR:** Esta se usa para mostrar visualmente una barra de carga, indicando el tiempo que puede tardar en completarse una tarea.

**JTREE:** Es un componente visual de Java que permite ordenar algunos datos jerárquicamente o como “árbol”, permitiendo seleccionar un elemento el cual desplegará en carpetas internas nuevos elementos que este esté conteniendo.

**SETTITLE:** Este permite asignar un título a nuestro JFrame o pantalla visual en la que nos encontremos actualmente, en nuestro proyecto podemos encontrarlo en los encabezados de cada ventana, por ejemplo:

Figura 17. Settitle

```
setTitle("Histórico Liga BetPlay Femenina");
```



**JCOMBOBOX:** También se conoce como lista desplegable, es un componente de combinación entre botón y una lista desplegable.

**JBUTTON:** Es un componente de Java que funciona como “botón” y permite llevar a otra parte del código.

En este código se utiliza para realizar la creación de los botones “Acceder”, “Salir”, “Atrás” y el correspondiente a cada uno de los años en el “menú”, por ejemplo:

Figura 18. Jbutton

```
// Crea un botón "Salir" para cerrar toda la aplicación
JButton botonSalir = new JButton("Salir");
botonSalir.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        System.exit(0); // Sale de la aplicación
    }
});
```

**JFRAME:** Es una clase utilizada por la biblioteca Swing (gráfica) para generar nuevas ventanas emergentes sobre las cuales el usuario puede o no interactuar, además posee por defecto las opciones de ampliar, minimizar o cerrar.

**JLABEL:** Este permite ingresar un texto estático en un panel, para que sea mostrada en pantalla, esta información no puede variar, también se puede usar para mostrar en pantalla algunos resultados.

En nuestro proyecto se utiliza para crear textos en el entorno gráfico, en este caso predefinidos, como, por ejemplo, el texto de “Usuario”, “Contraseña”, el mensaje de bienvenida, o el que informa que se debe iniciar sesión para acceder al histórico, en la ventana de inicio de sesión y en las demás:

Figura 19. JLabel

```
labelMensaje = new JLabel(" Inicia sesión para acceder a historico BetPlay Femenino");
labelMensaje.setHorizontalAlignment(JLabel.CENTER);
```

```
labelUsuario = new JLabel("Usuario:");
campoUsuario = new JTextField("456");
```

**JTEXTFIELD:** Se utiliza para crear un cuadro de texto en el que el usuario puede ingresar información, la cual se puede almacenar, comparar o simplemente visualizar en pantalla.

En este proyecto se implementa en la sección de “Usuario”, en donde se solicita el ingreso por teclado del “Usuario” para compararlo con el almacenado, esto en la ventana de “inicio de sesión”.

Figura 20. Jtextfield

```
campoUsuario = new JTextField("456");
```

**JSCROLLPANE:** Se utiliza para crear una barra de desplazamiento en el entorno gráfico en caso de ser necesaria. En nuestro proyecto es implementada, pero por la poca cantidad de datos, no se logra visualizar en pantalla.

Figura 21. Jscroll

```
JScrollPane scrollPane = new JScrollPane(textoInformacion);
```

## VIII. TECNOLOGÍAS UTILIZADAS.

Este proyecto fue desarrollado en:

Product Version: **Apache NetBeans IDE 17**  
 Updates: Updates available  
 Java: 15.0.2; Java HotSpot(TM) 64-Bit Server VM  
 15.0.2+7-27  
 Runtime: Java(TM) SE Runtime Environment 15.0.2+7-27  
 System: Windows 10 version 10.0 running on amd64;  
 Cp1252; es\_CO (nb)

Para el diagrama se ha utilizado:

**dia.exe 0.97.2** // Un programa para dibujar diagramas estructurados. // (C) 1998-2009 The Free Software Foundation and the authors

## IX. CONCLUSIONES

Por medio de este proyecto he ampliado mis conocimientos en cuanto a Modelo Vista Controlador y entorno gráfico mediante Java Netbeans. Por lo que ya soy capaz de implementar entornos gráficos a proyectos y códigos generados, además de poder agregar listas y tablas.

Se implementó procesos de ciclos anteriores con diversas clases, con herencia polimorfismo y demás, pero ahora unificando todo con un menú y su respectivo entorno gráfico.

Ahora soy capaz de generar un inicio de sesión con usuario y contraseña, así como alertas de contraseña errónea.

Desde la realización del proyecto, puedo crear contenedores para la distribución visual de elementos dentro de una ventana, además de generar color y modificación en los mismos.

He logrado mejorar en gran medida mis habilidades generales de programación y conocer más sobre Java y Neatbens así como de sus respectivos componentes.

## X. REFERENCIAS.

Colaboradores de los proyectos Wikimedia. "Liga Profesional Femenina de Fútbol 2020 (Colombia) - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Liga\\_Profesional\\_Femenina\\_de\\_Fútbol\\_2020\\_\(Colombia\)](https://es.wikipedia.org/wiki/Liga_Profesional_Femenina_de_Fútbol_2020_(Colombia)) (accedido el 15 de agosto de 2023).

Colaboradores de los proyectos Wikimedia. "Liga Profesional Femenina de Fútbol 2021 (Colombia) - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Liga\\_Profesional\\_Femenina\\_de\\_Fútbol\\_2021\\_\(Colombia\)](https://es.wikipedia.org/wiki/Liga_Profesional_Femenina_de_Fútbol_2021_(Colombia)) (accedido el 15 de agosto de 2023).

Colaboradores de los proyectos Wikimedia. "Liga Profesional Femenina de Fútbol 2022 (Colombia) - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Liga\\_Profesional\\_Femenina\\_de\\_Fútbol\\_2022\\_\(Colombia\)](https://es.wikipedia.org/wiki/Liga_Profesional_Femenina_de_Fútbol_2022_(Colombia)) (accedido el 15 de agosto de 2023).

Colaboradores de los proyectos Wikimedia. "Liga Profesional Femenina de Fútbol 2023 (Colombia) - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Liga\\_Profesional\\_Femenina\\_de\\_Fútbol\\_2023\\_\(Colombia\)](https://es.wikipedia.org/wiki/Liga_Profesional_Femenina_de_Fútbol_2023_(Colombia))

a\_de\_Fútbol\_2023\_(Colombia) (accedido el 15 de agosto de 2023).

Harol. (s. f.). GitHub - Harol003/Java\_Estructura\_Datos: Java\_Estructura\_Datos. GitHub.  
[https://github.com/Harol003/Java\\_Estructura\\_Datos](https://github.com/Harol003/Java_Estructura_Datos)

ABCdatos.com. (s. f.). Ejemplo de uso de un JTable con un TableModel en Java.  
<https://www.abcdatos.com/tutoriales/tutorial/z3128.html>

ABCdatos.com. (s. f.). Ejemplo de uso de un JTable con un TableModel en Java.  
<https://www.abcdatos.com/tutoriales/tutorial/z3128.html>

colaboradores de Wikipedia. (2022). JFrame. Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/Jframe>

Jlabel en java. (2018, 16 abril). PROGRAMACION.  
<https://victomanolo.wordpress.com/jlabel-en-java/>

## XI. LISTA DE IMÁGENES

Figura 1. Main - Creación propia.  
 Figura 2. Pseudocódigo - Creación propia.  
 Figura 3. Pseudocódigo - Creación propia.  
 Figura 4. DiagramaUML- Creación propia.  
 Figura 5 DiagramaDeFlujo - Creación propia.  
 Figura 6. Singin - Creación propia.  
 Figura 7. Menú - Creación propia.  
 Figura 8. Vista general - Creación propia.  
 Figura 9. Descripción año - Creación propia.  
 Figura 10. mvc - Creación propia.  
 Figura 11. modelo - Creación propia.  
 Figura 12. vista - Creación propia.  
 Figura 13. Controlador - Creación propia.  
 Figura 14. Lista Gen. - Creación propia.  
 Figura 15. Lista Tip. - Creación propia.  
 Figura 16. Lista PO. - Creación propia.  
 Figura 17. Settitle - Creación propia.  
 Figura 18. Jbutton - Creación propia.  
 Figura 19. JLabel - Creación propia.  
 Figura 20. Jtextfield - Creación propia.  
 Figura 21. Jscroll - Creación propia.  
 Figura 22. Pila - Creación propia.  
 Figura 23. Met. Iteración - Creación propia.

## XII. LISTA DE TABLAS.

Tabla1. Diccionario de datos. Creación propia.

## XIII. CRÉDITOS.



Cristian Camilo De Los Rios Rodríguez.  
[Cristian.de.los.rios@pi.edi.co](mailto:Cristian.de.los.rios@pi.edi.co)

Me gustan los video juegos, las películas, escuchar música y viajar por carretera.

Actualmente en proceso como tecnólogo de desarrollo, espero ser un excelente programador a futuro y poner en práctica todo lo aprendido en el transcurso de estos ciclos.