PROYECTO APP LIGABETFEM23

Pajarito Santamaria Nicole Dayana
nicole.pajarito@pi.edu.co
Programación de Software y aplicativos móviles, Politécnico Internacional
Autopista Sur No. 67 - 71

Resumen - Este proyecto consiste en una aplicación de interfaz gráfica desarrollada en Java que utiliza Programación Orientada a Objetos (POO) para organizar y administrar los datos de los equipos participantes en la Liga Femenina BetPlay 2023. Se hacen uso de clases y objetos para representar y controlar diversos componentes de la aplicación, como ventanas, etiquetas, campos de texto y botones. Asimismo, se implementa herencia para crear instancias específicas de equipos y se incorpora un sistema de autenticación para acceder a la aplicación. La interfaz gráfica está diseñada para ofrecer una experiencia de usuario intuitiva y atractiva, facilitando la interacción con la información de los equipos.

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

Abstract – This project consists of a graphical interface application developed in Java that uses Object Oriented Programming (OOP) to organize and manage the data of the teams participating in the BetPlay Women's League 2023. Classes and objects are used to represent and control various application components, such as windows, labels, text fields, and buttons. In addition, inheritance is implemented to create specific instances of computers and an authentication system is incorporated to access the application. The graphical interface is designed to offer an intuitive and attractive user experience, facilitating interaction with equipment information.

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

I. INTRODUCCIÓN

A través de esta aplicación en Java, los usuarios tendrán la oportunidad de explorar la Liga BetPlay Femenino 2023, descubrir detalles de equipos y partidos, y al mismo tiempo, experimentar de manera práctica conceptos clave de la POO. Desde herencia hasta polimorfismo, esta iniciativa ofrece una introducción única a la POO mientras disfrutan del entusiasmo del fútbol femenino.

Así mismo contará con un entorno gráfico que tendrá las siguientes opciones:

. **Acceso al aplicativo:** Al ingresar a la aplicación, se muestra una ventana donde los usuarios deben ingresar un nombre de usuario y una contraseña. El nombre de usuario correcto es "Proyecto" y la contraseña correcta es "LBF". Si los datos ingresados son correctos, el usuario podrá ingresar a la interfaz principal de la aplicación. En caso de datos incorrectos, se muestra un mensaje de error.

. **Interfaz principal:** Una vez autenticado, se muestra la interfaz principal que contiene una ventana con título "Liga BetPlay - Femenino 2023". En esta interfaz, hay un menú desplegable llamado "Equipos" que contiene opciones para los 5 equipos finalistas.

. **Selección de equipo**: Al seleccionar cada uno de los equipos en el menú desplegable "Equipos", se muestra una ventana que presenta las estadísticas del equipo seleccionado. Las estadísticas incluyen el número de partidos ganados perdidos, empatados, goles a favor, goles en contra, puntos acumulados, goleadora destacada y goles anotados.

. **Botón de Datos**: En la ventana de autenticación tiene un botón donde apareceran los datos de acceso para facilitar el ingreso.

II. CÓDIGO DEL APLICATIVO BETPLAY FEMENINO 2023

A continuación, el código del aplicativo comentado para mejor entendimiento:

1. **Librerías importadas.**

Figura 1.

```
import javax.swing.*;  // Crear interfaces graficas
import java.awt.*;     // Importa clases para manejar componentes gráficos
import java.awt.event.ActionEvent;   // Importa clases para manejar eventos
import java.awt.event.ActionListener;  // Importa clases para manejar eventos de acción
```

2. La clase principal Main, donde se crea una instancia “VentanaAutenticación” para mostrar el inicio de sesión en una interfaz gráfica de usuario (GUI).
Figura 2

```
public class LIGABETPLAY2023 {// clases principal
    public static void main(String[] args) {
        // Crea una nueva instancia de VentanaAutenticacion
        VentanaAutenticacion ventanaAutenticacion = new VentanaAutenticacion();
    }

}
```

3. La clase VentanaAutenticacion, aquí se implementa toda la interfaz grafica del código.
Figura 3

```
public class VentanaAutenticacion {

    private JFrame frame; // Se declara la variable 'frame' de tipo JFrame. Para representar interfaz grafica

    // Constructor de la clase
    public VentanaAutenticacion() {
        // Crear una instacia JFrame para la ventana de autenticación
        frame = new JFrame(title: "Liga BetPlay - Futbol Femenino Colombiano 2023");
        //configura la accion de cierre de ventana en la X
        frame.setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE);
        //establece el tamaño de la ventana de autenticacion
        frame.setSize(width: 500, height: 300);

        // Crear un JPanel que cubra toda la ventana y configurar el color de fondo
        JPanel panelFondo = new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {//se llama este metodo para pintar el panel
                super.paintComponent(g);// para mantener la funcionalidad adicional que se haga
                g.setColor(new Color(r: 169, g: 12, b: 45)); // configuracion de color
                // Dibuja un rectángulo del tamaño del panel que será el fondo que vamos a pintar
                g.fillRect(x: 0, y: 0, width: getWidth(), height: getHeight());
            }
        };
```

Contiene un constructor :
Figura 4

```
// Constructor de la clase
public VentanaAutenticacion() {
    // Crear una instacia JFrame para la ventana de autenticación
    frame = new JFrame(title: "Liga BetPlay - Futbol Femenino Colombiano 2023");
    //configura la accion de cierre de ventana en la X
    frame.setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE);
    //establece el tamaño de la ventana de autenticacion
    frame.setSize(width: 500, height: 300);
```

Se crea una instancia Jframe para la ventana de autenticación :
Figura 5

```
// Crear una instacia JFrame para la ventana de autenticación
frame = new JFrame(title: "Liga BetPlay - Futbol Femenino Colombiano 2023");
//configura la accion de cierre de ventana en la X
frame.setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE);
//establece el tamaño de la ventana de autenticacion
frame.setSize(width: 500, height: 300);
```

Se crea un JPanel para cubrir toda la ventana y modificar el color del fondo y que tenga un desplazamiento flexible
Figura 6

```
// Crear un JPanel que cubra toda la ventana y configurar el color de fondo
JPanel panelFondo = new JPanel() {
    @Override
    protected void paintComponent(Graphics g) {//se llama este metodo para pintar el panel
        super.paintComponent(g);// para mantener la funcionalidad adicional que se haga
        g.setColor(new Color(r: 169, g: 12, b: 45)); // configuracion de color
        // Dibuja un rectángulo del tamaño del panel que será el fondo que vamos a pintar
        g.fillRect(x: 0, y: 0, width: getWidth(), height: getHeight());
    }
};
```

Se crea un JLabel para el título y su configuración
Figura 7

```
    // Etiqueta del título
JLabel labelTitulo = new JLabel(text: "BIENVENIDO A LA APP LIGA FEMENINA BETPLAY 2023");
labelTitulo.setFont(new Font(name: "Arial", style: Font.BOLD, size: 15)); // Establece fuente y tamaño
labelTitulo.setForeground(fg: Color.WHITE); // Color del texto
//Establece la posición de la etiqueta en la cuadrícula del GridBagLayout. En este caso, se coloca
gbc.gridx = 0;  // Posición en la columna 0
gbc.gridy = 0;  // Posición en la fila 0
gbc.gridwidth = 3; // Ocupar dos columnas
gbc.anchor = GridBagConstraints.PAGE_START; // Alinear al principio del panel

panelFondo.add(comp: labelTitulo, constraints: gbc);  // Agrega la etiqueta al panelFondo

gbc.insets = new Insets(top: 25, left: 10, bottom: 0, right: 5); // Establece márgenes para los siguientes

// Etiqueta y campo de texto para el nombre de usuario
JLabel labelUsuario = new JLabel(text: "Usuario:");  // Crea una etiqueta con el texto "Usuario:"
labelUsuario.setForeground(fg: Color.WHITE); // Establece el color del texto como blanco
```

Se crea un JButton para configurar el botón de “ingresar” en la ventana de autenticación
Figura 8

```
// Botón para ingresar
JButton botonIngresar = new JButton(text: "Ingresar");  // Crea un botón con el texto "Ingresar"

botonIngresar.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String usuario = campoUsuario.getText();  // Obtiene el texto del campo de usuario
        String contraseña = new String(value: campoContraseña.getPassword());  // Obtiene la contraseña

        if (autenticar(usuario, contraseña)) {  // Verifica la autenticación
            mostrarVentanaMenu();  // Muestra la ventana del menú
            frame.dispose();  // Cierra la ventana de autenticación
        } else {
            // Muestra un mensaje de error si la autenticación falla
            JOptionPane.showMessageDialog(parentComponent: frame, message: "Usuario o contraseña incorrectos"
        }
    }
});
```

Se crea un método para mostrar la ventana del menú principal
Figura 9

```
// Método para mostrar la ventana del menú principal
private void mostrarVentanaMenu() {
    // Crear un JFrame para el menú principal
    JFrame menuFrame = new JFrame(title: "Menú Principal");
    menuFrame.setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE);
    menuFrame.setSize(width: 800, height: 600);// tañamo

    // Crear un JPanel para el menú con GridBagLayout
    JPanel panelMenu = new JPanel(new GridBagLayout());
    GridBagConstraints gbc = new GridBagConstraints();
    gbc.insets = new Insets(top:20, left:20, bottom: 20, right: 20); // Espaciado en todos los l
    panelMenu.setBackground(new Color(r: 5, g: 31, b: 94));// se define el color del panel

    // Añadir título centrado en la parte superior
    JLabel labelTitulo = new JLabel(text:"LIGA FEMENINA BETPLAY 2023");  // Crea una etique
    labelTitulo.setFont(new Font(name:"Arial", style: Font.BOLD, size:20));  // fuente de text
    labelTitulo.setHorizontalAlignment(alignment: JLabel.CENTER);  // Centrar el texto horiz
    labelTitulo.setForeground(fg: Color.WHITE); // Color de texto
```

Se crea un método para mostrar la información detallada de un equipo en una ventana emergente
Figura 10

```
    // Método para mostrar información detallada de un equipo en un cuadro de diálogo
private void mostrarInformacion(Equipo equipo) {
    // Verificar si el equipo no es nulo
    if (equipo != null) {
        // Obtener colores para la ventana de informacion Equipos
        Color colorFondo = obtenerColorEquipo(equipo); // Color de fondo del equipo
        Color colorTexto = obtenerColorTextoEquipo(equipo); // Color del texto

        // Establecer el color de texto en el cuadro de diálogo
        UIManager.put(key:"OptionPane.messageForeground", value: colorTexto);

        // Crear un cuadro de diálogo con la información del equipo
        JOptionPane optionPane = new JOptionPane(
            // Construir el mensaje con detalles del equipo
            "Información de " + equipo.getNombre() + "\n" +
            "Partidos jugados: " + equipo.getPartidosJugados() + "\n" +
            "Partidos Ganados: " + equipo.getPartidosGanados() + "\n" +
            "Partidos Empatados: " + equipo.getPartidosEmpatados() + "\n" +
            "Partidos Perdidos: " + equipo.getPartidosPerdidos() + "\n" +
            "Goles a favor: " + equipo.getGolesAFavor() + "\n" +
            "Goles en contra: " + equipo.getGolesEnContra() + "\n" +
            "Diferencia de goles: " + equipo.getDiferenciaDeGoles() + "\n" +
            "Puntos: " + equipo.getPuntos() + "\n" +
```

Se crea un método para asignar un color específico a un equipo de fútbol según su nombre
Figura 11

```
private Color obtenerColorEquipo(Equipo equipo) {
    Color colorFondo = Color.BLACK; //color fondo

    if (equipo != null) {
        switch (equipo.getNombre()) {
            case "América de Cali":
                colorFondo = new Color(r: 255, g: 0, b: 0);
                break;
            case "Independiente Santa Fe":
                colorFondo = new Color(r: 255, g: 99, b: 71);
                break;
            case "Atlético Nacional":
                colorFondo = new Color(r: 34, g: 139, b: 34);
                break;
            case "Deportivo Pereira":
                colorFondo = new Color(r: 184, g: 134, b: 11);
                break;
            case "Deportivo Cali":
                colorFondo = new Color(r: 34, g: 139, b: 34);
                break;
            default:
                colorFondo = Color.BLACK;
                break;
        }
    }

    // Imprimir el color en la consola
```

4. La clase Equipo es la clase abstracta, de esta clase se heredan atributos a las subclases
Figura 12

```
public abstract class Equipo {
    private String nombre;
    private int partidosJugados;
    private int partidosGanados;
    private int partidosEmpatados;
    private int partidosPerdidos;
    private int golesAFavor;
    private int golesEnContra;
    private int diferenciaDeGoles;
    private int puntos;
    private String goleadora;
    private int cantidadGolesGoleadora;
```

Contiene un constructor para inicializar los atributos del equipo
Figura 13

```
// Constructor para inicializar los atributos del equipo
public Equipo(String nombre, int partidosJugados, int partidosGanados, int partidosEmpatados,
              int partidosPerdidos, int golesAFavor, int golesEnContra, int diferenciaDeGoles,
              int puntos, String goleadora, int cantidadGolesGoleadora) {
    this.nombre = nombre;
    this.partidosJugados = partidosJugados;
    this.partidosGanados = partidosGanados;
    this.partidosEmpatados = partidosEmpatados;
    this.partidosPerdidos = partidosPerdidos;
    this.golesAFavor = golesAFavor;
    this.golesEnContra = golesEnContra;
    this.diferenciaDeGoles = diferenciaDeGoles;
    this.puntos = puntos;
    this.goleadora = goleadora;
    this.cantidadGolesGoleadora = cantidadGolesGoleadora;
}
```

Contiene métodos get y set para obtener y modificar información
Figura 14

```
// Métodos getter para acceder a la información de los atributos. Encapsulamiento y m

public String getNombre() {
    return nombre;
}

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

public int getPartidosJugados() {
    return partidosJugados;
}

public void setPartidosJugados(int partidosJugados) {
    this.partidosJugados = partidosJugados;
}

public int getPartidosGanados() {
    return partidosGanados;
}

public void setPartidosGanados(int partidosGanados) {
    this.partidosGanados = partidosGanados;
}

public int getPartidosEmpatados() {
    return partidosEmpatados;
}
```

5. Subclases AmércíaDeCali, IndependienteSantaFe, AtléticoNacional, DeportivoCali y DeportivoPereira; heredaron de la super clase Euipos atributos para modificar la información de cada equipo

Figura 15

```
public class IndependienteSantaFe extends Equipo {// hereda información de clase abstracta Equipo HERENCIA
    public IndependienteSantaFe() {//información de los detalles del constructor
        super(nombre: "Independiente Santa Fe", partidosJugados:16, partidosGanados:10, partidosEmpatados: 5, partidosPerdidos
              goleadora: "Liana Salazar, Maria Reyes", cantidadGolesGoleadora: 10);// se llama al contructor de la s
    }
}
```

```
public class AmericaDeCali extends Equipo {// hereda información de clase abstracta Equipo HERENCIA
    public AmericaDeCali() {//información de los detalles del constructor
        super(nombre: "América de Cali", partidosJugados:16, partidosGanados:13, partidosEmpatados: 1, partidosPerdidos
    }// se llama al contructor de la super clase
}
```

```
 */
// Subclase de Equipo para representar el equipo "Atlético Nacional"
class AtleticoNacional extends Equipo {
    public AtleticoNacional() {
        super (nombre: "Atlético Nacional", partidosJugados:16, partidosGanados:8, partidosEr
    }
}
```

```
 */
// Subclase de Equipo para representar el equipo "Deportivo Cali"
class DeportivoCali extends Equipo {
    public DeportivoCali() {
        super(nombre: "Deportivo Cali", partidosJugados:16, partidosGanados:7, partidosEmpatados: 6,
    }
}
```

```
 */
public class DeportivoPereira extends Equipo {// hereda información de clase abstracta Equipo HERENCIA
    public DeportivoPereira() {//información de los detalles del constructor
        super(nombre: "Deportivo Pereira", partidosJugados:16, partidosGanados:8, partidosEmpatados: 5, partidosPerdidos: 3,
              goleadora: "Ana Milé González", cantidadGolesGoleadora: 8);// se llama al contructor de la super cl
    }
}
```

En el aplciativo **LIGABETPLAYFEMINFO** tenemos el siguiente código comentado para su mejor entendimiento:

1. Clase principal VentanaAutenticacion donde configuramos toda la parte de interfaz gráfica para el inicio de sesión

Figura 16

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

// Definición de la clase VentanaAutenticacion
class VentanaAutenticacion extends JFrame implements ActionListener {
    // Declaración de los componentes de la ventana
    private JTextField usuarioField;  // Campo de texto para el usuario
    private JPasswordField contraseñaField;  // Campo de texto para la contraseña
    private JButton botonAutenticar;  // Botón de autenticación
    private JButton botonDatos;
    private boolean autenticado = false;  // Variable que indica si el usuario ha sido autenticado
```

2. Tenemos un constructor de la ventana de autenticación

Figura 17

```
// Constructor de la ventana de autenticación
public VentanaAutenticacion() {
    // Configuración de la ventana
    setTitle(title: "INFOBETPLAYFEM2023");
    setSize(width: 600, height: 300);  // Tamaño aumentado
    setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE);
    setLocationRelativeTo(c: null);
    getContentPane().setBackground(new Color(r: 246, g: 186, b: 202 )); // Cambia el color de fondo

    // Inicialización de los componentes
    usuarioField = new JTextField(columns:10);
    contraseñaField = new JPasswordField(columns:10);
    botonAutenticar = new JButton(text:"Ingresar");
    botonDatos = new JButton(text:"Datos");  // Inicializa el botón "Datos"
```

3. Configuración de los botones

Figura 18

```
// Configuración del layout
setLayout(new GridBagLayout()); // Usamos un GridBagLayout para mejor control de posición

// Configuración de GridBagConstraints
GridBagConstraints gbc = new GridBagConstraints();
gbc.insets = new Insets(top:5, left:5, bottom: 2, right: 5); // Margen entre componentes

// Agrega un JLabel con el título
JLabel titulo = new JLabel(text: "A continuación verás datos generales de la LIGA FEMENINA BETPLAY 2023");
gbc.gridx = 0;
gbc.gridy = 0; // Puedes ajustar este valor para controlar la posición vertical del título
gbc.gridwidth = 1;
gbc.anchor = GridBagConstraints.CENTER;
add(comp:titulo, constraints:gbc);

// Agregar componente: Usuario
gbc.gridx = 0;
gbc.gridy = 1;
gbc.anchor = GridBagConstraints.WEST; // Alinea a la izquierda
add(new JLabel(text: "Usuario: "), constraints:gbc);

gbc.gridx = 0;
gbc.gridy = 2;
gbc.anchor = GridBagConstraints.WEST; // Alinea a la izquierda
add(comp:usuarioField, constraints:gbc);

// Agregar componente: Contraseña
gbc.gridx = 0;
gbc.gridy = 3;
gbc.anchor = GridBagConstraints.WEST; // Alinea a la izquierda
add(new JLabel(text: "Contraseña: "), constraints:gbc);
```

4. Métodos para validar autenticación

Figura 19

```
ublic void actionPerformed(ActionEvent e) {
    if (e.getSource() == botonAutenticar) {
        String usuario = usuarioField.getText();
        String contraseña = new String(value: contraseñaField.getPassword());

        // Verifica las credenciales
        if (validarCredenciales(usuario, contraseña)) {
            autenticado = true;
            dispose(); // Cierra la ventana
        } else {
            JOptionPane.showMessageDialog(parentComponent:this, message:"Credenciales incorrectas. Intente de nuevo.");
        }
    } else if (e.getSource() == botonDatos) {
        JOptionPane.showMessageDialog(parentComponent:this, message:"Usuario: LIGA y Contraseña: 2023", title: "CREDENCIALES DE ACCESO",
    }



    // Método para validar las credenciales (usuario y contraseña)
    private boolean validarCredenciales(String usuario, String contraseña) {
        return usuario.equals(anObject:"LIGA") && contraseña.equals(anObject:"2023");
    }
```

5. La clase main donde tenemos toda la información del aplicativo

Figura 20

```
public class ProyectoBetPlayFem2023 {
    public static void main(String[] args) {
        VentanaAutenticacion ventanaAutenticacion = new VentanaAutenticacion();
        ventanaAutenticacion.setVisible(b: true);

        while (!ventanaAutenticacion.estaAutenticado()) {
            try {
                Thread.sleep(millis: 1000); // Esperar 1000 milisegundos antes de volver a verificar
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        System.out.println(x: "BIENVENIDO\n");

        int opcion;
        Scanner scr = new Scanner(source: System.in);

        do {
            System.out.println(x: "\nMenu:");
            System.out.println(x: "1. Datos de la Liga BetPlayFem2023");
            System.out.println(x: "2. Equipos que clasifican para la final");
            System.out.println(x: "3. Datos sobre el club");
            System.out.println(x: "4. Total de puntos anotados en la LigaBetPlayFemenina2023");
            System.out.println(x: "5. Posicion jugadora");
            System.out.println(x: "6. Estadios de Colombia");
            System.out.println(x: "7. Creditos");
            System.out.print(s: "Seleccione una opcion: ");
            opcion = scr.nextInt();
```

6. Tenemos una clase llamada TotalPuntos con el método pila

Figura 21

```
public class TotalPuntos {
    public static void main(String[] args) {
        // Crear una pila llamada "TotalPuntos" para simular el total de puntos anotados por los equipos
        Stack<Integer> TotalPuntos = new Stack<>();
        TotalPuntos.push(item: 40); // puntos del equipo America De cali
        TotalPuntos.push(item: 35); // puntos del equipo SantaFe
        TotalPuntos.push(item: 30); // puntos del equipo Atletico Nacional
        TotalPuntos.push(item: 29); // puntos del equipo Pereira
        TotalPuntos.push(item: 27); // puntos del equipo Deportivo Cali
        TotalPuntos.push(item: 27); // puntos del equipo Independiente Medellin
        TotalPuntos.push(item: 25); // puntos del equipo Cor
        TotalPuntos.push(item: 23); // puntos del equipo Equidad
        TotalPuntos.push(item: 23); // puntos del equipo Millonarios
        TotalPuntos.push(item: 22); // puntos del equipo Llaneros Femenina
        TotalPuntos.push(item: 22); // puntos del equipo Junior
        TotalPuntos.push(item: 21); // puntos del equipo Boyacá Chicó Femenina
        TotalPuntos.push(item: 20); // puntos del equipo Huila
        TotalPuntos.push(item: 20); // puntos del equipo RST
        TotalPuntos.push(item: 13); // puntos del equipo Pasto
        TotalPuntos.push(item: 12); // puntos del equipo Bucaramanga
        TotalPuntos.push(item: 6); // puntos del equipo Tolima

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

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

7. Clase llamada Posiciones donde aplicamos una lista de posición ordinal

Figura 22

```
import java.util.ArrayList;
import java.util.List;

public class Posiciones {
    public static void main(String[] args) {
        // Crear una "Lista Posición Ordinal" para los finalistas de la LigaBetPlayfem2023
        List<String> finalistas = new ArrayList<>();
        finalistas.add(e: "America");
        finalistas.add(e: "SantaFe");
        finalistas.add(e: "Atletico Nacional");
        finalistas.add(e: "Pereira");
        finalistas.add(e: "Deportaivo Cali");

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


        // Mostrar cada nombre con su posición en el equipo
        for (int i = 0; i < finalistas.size(); i++) {
            int posicion = i + 1; // Sumar 1 para que las posiciones comiencen desde 1
            String nombre = finalistas.get(index: i);
            System.out.println("Finalista " + posicion + ": " + nombre);
        }
    }
}
```

8. Clase llamada PosicionJugadora donde aplicamos un método de iteracción
Figura 23

```java
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

class JugadoraDeFutbol {
    String nombre;
    String posicion;

    public JugadoraDeFutbol(String nombre, String posicion) {
        this.nombre = nombre;
        this.posicion = posicion;
    }
}
// método de iteracción
public class PosicionJugadora {
    public static void main(String[] args) {
        // Crear una lista con las jugadoras de fútbol
        List<JugadoraDeFutbol> jugadoras = new ArrayList<>();
        jugadoras.add(new JugadoraDeFutbol(nombre: "Maria", posicion: "Delantera"));
        jugadoras.add(new JugadoraDeFutbol(nombre: "Laura", posicion: "Defensa"));
        jugadoras.add(new JugadoraDeFutbol(nombre: "Ana", posicion: "Centrocampista"));

        // Método para buscar una jugadora de fútbol por su posición
        String posicionJugadoraBuscada = "Defensa";
        Optional<JugadoraDeFutbol> jugadoraBuscada = buscarJugadoraPorPosicion(listaJugadoras: jugadoras, posicion: posicionJugadoraBuscada);

        // Mostrar información de la jugadora de fútbol buscada
        if (jugadoraBuscada.isPresent()) {
            JugadoraDeFutbol jugadoraEncontrada = jugadoraBuscada.get();
            System.out.println(jugadoraEncontrada.nombre + " juega como " + jugadoraEncontrada.posicion);
        } else {
            System.out.println("No se encontró ninguna jugadora en la posición " + posicionJugadoraBuscada);
```

9. Clase llamada EstadiosColombia donde aplicamos una lista genérica
Figura 24

```java
import java.util.ArrayList;
import java.util.List;

public class EstadiosColombia {
    public static void main(String[] args) {
        // Crear una "Lista Genérica" para almacenar goleadoras de los equipos
        List<String> EstadiosColombia = new ArrayList<>();

        EstadiosColombia.add(e: "Estadio Monumental de Palmaseca");
        EstadiosColombia.add(e: "Metropolitano Roberto Melendez ");
        EstadiosColombia.add(e: "Atanasio Girardot");
        EstadiosColombia.add(e: "Nemesio Camacho El Campín");
        EstadiosColombia.add(e: "Olimpico Pascual Guerrero");

        // Mostrar los nombres de todas la goleadoras

        for (String Estadios : EstadiosColombia) {
            System.out.println(x: Estadios);
        }
    }
}
```

10. En esta clase Equipos utilizamos una lista tipada
Figura 25

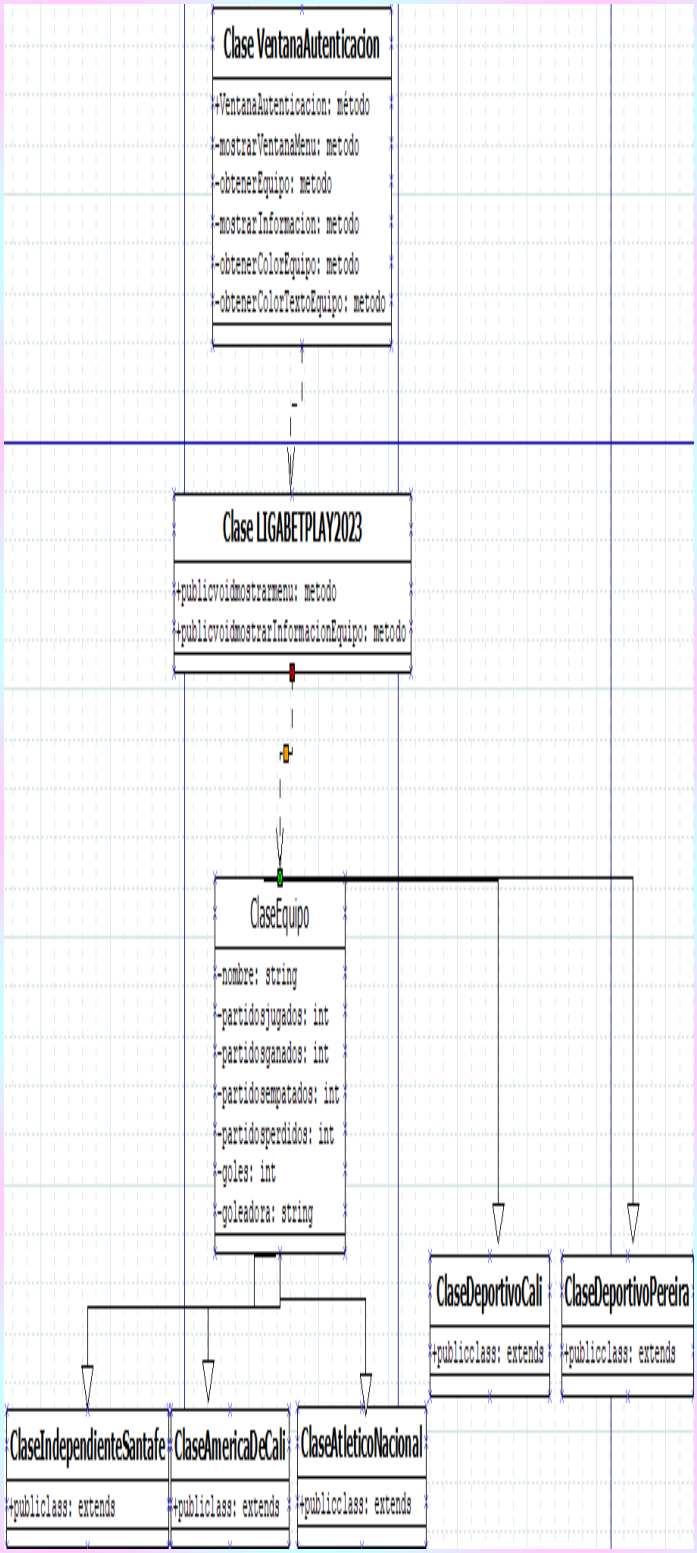
```java
import java.util.ArrayList;
import java.util.List;

public class Equipos {
    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(e: "America De cali");
        listaTipada.add(e: "Deportivo Cali");
        listaTipada.add(e: "Independiente SantaFe");
        listaTipada.add(e: "Millonarios");
        listaTipada.add(e: "Independiente Medellin");
        listaTipada.add(e: "Atletico Nacional");
        listaTipada.add(e: "Deportivo Pereira");
        listaTipada.add(e: "Junior FC");
        listaTipada.add(e: "Deportes Tolima");
        listaTipada.add(e: "La Equidad");
        listaTipada.add(e: "Real Santander");
        listaTipada.add(e: "Boyaca Chico");

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

        // Imprimir los elementos de la lista
        System.out.println(x: "Equipos de la Liga BetPlay Femenina 2023 :\n");
        for (String elemento : listaTipada) {
            System.out.println(x: elemento);
        }

    }
}
```

11. Clase llamada Club donde utilizamos una lista generica diversa
Figura 26

```java
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

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

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

        // Agregar información de participantes
        Map<String, Object> participante1 = new HashMap<>();
        participante1.put(key:"equipo", value: "America De Cali");
        participante1.put(key:"presidente", value: "Marcela Gomez");
        participante1.put(key:"director", value: "Carlos Hernandez");
        participante1.put(key:"goles", value: 11);
        participantesLiga.add(e: participante1);

        Map<String, Object> participante2 = new HashMap<>();
        participante2.put(key:"equipo", value: "Atletico Nacional");
        participante2.put(key:"presidente", value: "Mauricio Navarro");
        participante2.put(key:"director", value: "Jorge Barreneche");
        participante2.put(key:"goles", value: 8);
        participantesLiga.add(e: participante2);

        Map<String, Object> participante3 = new HashMap<>();
        participante3.put(key:"equipo", value: "SantaFe");
        participante3.put(key:"presidente", value: "Eduardo Mendez");
        participante3.put(key:"director", value: "Omar Ramirez");
        participante3.put(key:"goles", value: 20);
```

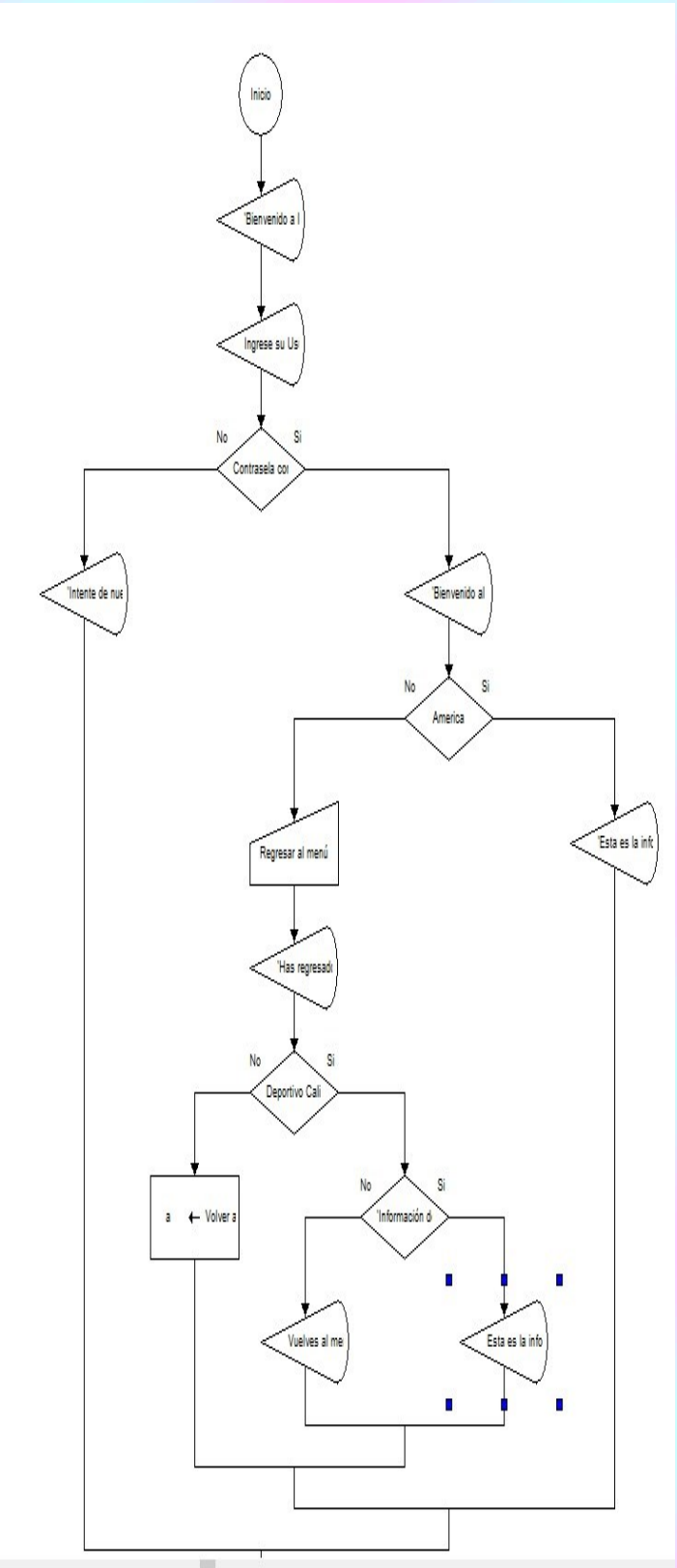
III. DIAGRAMA UML

Figura 27



IV. DIAGRAMA DE FLUJO

Figura 28



V. DICCIONARIO DE DATOS

Tabla 1

DICCIONARIO DE DATOS		
NOMBRE	**TIPO DE DATO**	**DESCRIPCIÓN**
Scanner	scanner	imprimir datos ingresados por teclado
opcion	int	Entero para las opciones del menú
do	ciclo	ciclo hacer para que el menú se repita
mostrarinforequipo	método	mostrar detalles en cada subclase, se llamada desde la superclase
nombre	string	nombre de equipos
partidosjugados	int	partidos jugados por los equipos
partidosganados	int	partidos ganados por los equipos
partidosempatados	int	partidos empatados por los equipos
golesafavor	int	goles a favor
diferenciasdegoles	int	diferencia de goles
puntos	int	puntos anotados por cada equipo
goleadora	string	goleadora destacada de cada equipo
equipo	clase	clase abstracta que herencia
publicequipo	constructor	inicializar atributos de equipo
getter	método	obtener información
setter	método	modificar información
Ventana Autenticacion	Clase	representa la ventana de autenticación en la interfaz gráfica.
JFrame	Clase	representa una ventana en una interfaz gráfica.
JPanel	Clase	Proporciona un contenedor para organizar componentes en una interfaz gráfica.
GridBagConstraints	Clase	Especifica restricciones para la colocación de componentes en un GridBagLayout.
Color	Clase	Representa un color en el espacio de color RGB.
JLabel	Clase	Muestra texto o imagen en una interfaz gráfica
JTextField	Clase	proporciona un campo de texto para que los usuarios ingresen texto.
JPasswordField	Clase	proporciona un campo de texto para contraseñas.
JButton	Clase	un botón en una interfaz gráfica.
Font	Clase	Representa una fuente de texto.
JOptionPane	Clase	Proporciona cuadros de diálogo para interactuar con el usuario.
private void mostrarVentanaMenu()	metodo	Muestra la ventana del menú principal.
private void mostrarInformacion (Equipo equipo	metodo	Muestra información detallada de un equipo en un cuadro de diálogo.
private boolean autenticar(String usuario, String contraseña)	metodo	Verifica la autenticación del usuario y la contraseña.

private Equipo obtenerEquipo(String nombreEquipo)	metodo	Obtiene una instancia de un equipo según su nombre.

VI. ANÁLISIS PROGRAMACIÓN ORIENTADA A OBJETOS

ABSTRACCIÓN: La abstracción simplifica y representa un objeto en forma de clases; en este caso se crea la clase abstracta "Equipo" donde se define el método abstracto "mostrarDetallesEquipo", esto quiere decir que las subclases implementan este método, pero lo pueden hacer de diferentes maneras según sus características.

HERENCIA: La herencia permite definir una estructura común en la clase base y luego extenderla a las subclases.
Figura 29

```
public void mostrarInformaciónEquipo(Equipo equipo) {//método para mostrar los detalles en cada subclase llamandolo de la superclase Equipo

    System.out.println("Información "+ equipo.getNombre());//imprimir el nombre del equipo
    System.out.println("Partidos jugados: " + equipo.getPartidosJugados());// imprimir los atributos del equipo obteniendo los metodos de la
    System.out.println("Partidos Ganados: " + equipo.getPartidosGanados());
    System.out.println("Partidos Empatados: " + equipo.getPartidosEmpatados());
    System.out.println("Partidos Perdidos: " + equipo.getPartidosPerdidos());
    System.out.println("Goles a favor: " + equipo.getGolesAFavor());
    System.out.println("Goles en contra: " + equipo.getGolesEnContra());
    System.out.println("Diferencia de goles: " + equipo.getDiferenciaDeGoles());
    System.out.println("Puntos: " + equipo.getPuntos());
    System.out.println("Goleadora destacada : " + equipo.getGoleadora());
    System.out.println("Cantidad de goles marcados por la goleadora: " + equipo.getCantidadGolesGoleadora());

    System.out.println();// se crea un espacio para que separe la información de la opcion elegida por el usuario y del menú dentro del cicl

}
```

POLIMORFISMO: El polimorfismo es la capacidad de tratar las diferentes clases de objetos para ser tratados como objetos de una clase base común a través de su interfaz común. En este caso lo podemos ver en el método "mostrarDetallesEquipo" de la clase "Equipo" al heredar este método en la subclase especifica.
Figura 30

```
LIGABETPLAY2023.java | Equipo.java | IndependienteSantaFe.java | AmericaDeCali.java | AtleticoNacional.java | DeportivoCali.java

                System.out.println("¡ Gracias por usar el programa !");
                break;

            default:
                System.out.println(x: "Opción inválida. Por favor, ingrese una opción valida");//si el usuario ingresa u
        }
    } while (opcion != 6);//El bucle se repite mientras la opción ingresada por el usuario no sea 6

}


public void mostrarInformaciónEquipo(Equipo equipo) {//método para mostrar los detalles en cada subclase llamandolo de

    System.out.println("Información "+ equipo.getNombre());//imprimir el nombre del equipo
    System.out.println("Partidos jugados: " + equipo.getPartidosJugados());// imprimir los atributos del equipo obtenie
    System.out.println("Partidos Ganados: " + equipo.getPartidosGanados());
    System.out.println("Partidos Empatados: " + equipo.getPartidosEmpatados());
    System.out.println("Partidos Perdidos: " + equipo.getPartidosPerdidos());
    System.out.println("Goles a favor: " + equipo.getGolesAFavor());
    System.out.println("Goles en contra: " + equipo.getGolesEnContra());
    System.out.println("Diferencia de goles: " + equipo.getDiferenciaDeGoles());
    System.out.println("Puntos: " + equipo.getPuntos());
    System.out.println("Goleadora destacada : " + equipo.getGoleadora());
    System.out.println("Cantidad de goles marcados por la goleadora: " + equipo.getCantidadGolesGoleadora());

    System.out.println();// se crea un espacio para que separe la información de la opcion elegida por el usuario y del
```

ENCAPSULAMIENTO: El encapsulamiento se refleja en este proyecto en la protección de datos y control de acceso a los datos internos de las clases a través de atributos privados y métodos de acceso.
Atributos privados:
Figura 31

```
public abstract class Equipo {
    private String nombre;
    private int partidosJugados;
    private int partidosGanados;
    private int partidosEmpatados;
    private int partidosPerdidos;
    private int golesAFavor;
    private int golesEnContra;
    private int diferenciaDeGoles;
    private int puntos;
    private String goleadora;
    private int cantidadGolesGoleadora;
```

Métodos de acceso Getters y Setters:
Figura 32

```
public String getNombre() {
    return nombre;
}

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

public int getPartidosJugados() {
    return partidosJugados;
}

public void setPartidosJugados(int partidosJugados) {
    this.partidosJugados = partidosJugados;
}

public int getPartidosGanados() {
    return partidosGanados;
}

public void setPartidosGanados(int partidosGanados) {
    this.partidosGanados = partidosGanados;
}

public int getPartidosEmpatados() {
    return partidosEmpatados;
}

public void setPartidosEmpatados(int partidosEmpatados) {
    this.partidosEmpatados = partidosEmpatados;
}
```

Uso del constructor :
Figura 32

```
// Constructor para inicializar los atributos del equipo
public Equipo(String nombre, int partidosJugados, int partidosGanados, int partidosEmpatados,
              int partidosPerdidos, int golesAFavor, int golesEnContra, int diferenciaDeGoles,
              int puntos, String goleadora, int cantidadGolesGoleadora) {
    this.nombre = nombre;
    this.partidosJugados = partidosJugados;
    this.partidosGanados = partidosGanados;
    this.partidosEmpatados = partidosEmpatados;
    this.partidosPerdidos = partidosPerdidos;
    this.golesAFavor = golesAFavor;
    this.golesEnContra = golesEnContra;
    this.diferenciaDeGoles = diferenciaDeGoles;
    this.puntos = puntos;
    this.goleadora = goleadora;
    this.cantidadGolesGoleadora = cantidadGolesGoleadora;
}
```

VISIBILIDAD RESTRINGIDA : Al definir atributos y métodos como privados, protegemos la implementación interna de las clases y eviamos que se realicen cambios no deseados.

LISTAS ABSTRACTAS: Las listas abstractas en programación proporcionan una especificación de las operaciones que se pueden realizar en una estructura de datos, sin entrar en detalles de su implementación. Esto facilita su uso sin la necesidad de comprender los aspectos internos, lo que promueve la flexibilidad y la reutilización del código. Por ejemplo, se pueden realizar operaciones como añadir, eliminar o buscar elementos sin necesidad de conocer la forma en que se almacenan internamente.

Figura 33

```
public interface ListaAbstracta<T> {
    void agregar(T elemento);
    void insertar(int indice, T elemento);
    void eliminar(int indice);
    T obtener(int indice);
    void actualizar(int indice, T elemento);
    int longitud();
    boolean estaVacia();
}

public class ListaConcreta<T> implements ListaAbstracta<T> {
    // Implementa los métodos de la interfaz aquí
    // ...
}
```

DIAGRAMA DE LISTAS SENCILLAS: Un diagrama de listas sencillas es una representación visual que muestra cómo los elementos de una lista están conectados. Cada nodo contiene datos y una referencia al siguiente elemento. El primer nodo es la "cabeza" y el último apunta a un valor especial que indica el final de la lista. Estos diagramas son útiles para comprender y manipular listas en programación.

CLASE INTERNA DE ENLACE: Una clase interna es una clase definida dentro de otra clase. Tiene acceso completo a todos los miembros de la clase contenedora, incluso los privados. Pueden ser regulares, estáticas, anónimas o locales. Se utilizan para mejorar la organización y modularidad del código. No pueden existir de manera independiente y solo tienen sentido dentro de la clase contenedora.

Figura 34

```
public class ListaEnlazada {
    private class Nodo {
        int dato;
        Nodo siguiente;
        Nodo(int dato) { this.dato = dato; }
    }

    private Nodo primero;

    public void agregarAlFinal(int dato) {
        Nodo nuevoNodo = new Nodo(dato);
        if (primero == null) primero = nuevoNodo;
        else {
            Nodo actual = primero;
            while (actual.siguiente != null) actual = actual.siguiente;
            actual.siguiente = nuevoNodo;
        }
    }
}
```

LISTAS GENÉRICAS: Las listas genericas facilitan el almacenamiento seguro de elementos de distintos tipos de datos. Esto posibilita la creación de listas con una variedad de elementos, y el compilador garantiza la corrección de las operaciones según el tipo especificado. Esto aporta versatilidad y confianza al manipular datos, permitiendo la creación de código reutilizable con diversos tipos sin requerir conversiones manuales. Lenguajes como Java y C# son ejemplos que ofrecen esta característica.

Figura 35

```
import java.util.HashMap;
import java.util.List;
import java.util.Map;

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

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

        // Agregar información de participantes
        Map<String, Object> participante1 = new HashMap<>();
        participante1.put(key:"equipo", value: "America De Cali");
        participante1.put(key:"presidente", value: "Marcela Gomez");
        participante1.put(key:"director", value: "Carlos Hernandez");
        participante1.put(key:"goles", value: 11);
        participantesLiga.add(e: participante1);

        Map<String, Object> participante2 = new HashMap<>();
        participante2.put(key:"equipo", value: "Atletico Nacional");
        participante2.put(key:"presidente", value: "Mauricio Navarro");
        participante2.put(key:"director", value: "Jorge Barreneche");
        participante2.put(key:"goles", value: 8);
        participantesLiga.add(e: participante2);

        Map<String, Object> participante3 = new HashMap<>();
        participante3.put(key:"equipo", value: "SantaFe");
        participante3.put(key:"presidente", value: "Eduardo Mendez");
        participante3.put(key:"director", value: "Omar Ramirez");
        participante3.put(key:"goles", value: 20);
        participantesLiga.add(e: participante3);
```

```
// Mostrar información de Equipo  AmericaDeCali
Map<String, Object> infoAmerica = participantesLiga.get(index: 0);
System.out.println(x: "Informacion del primer equipo finalista:");
System.out.println("Equipo: " + infoAmerica.get(key:"equipo"));
System.out.println("Presidente: " + infoAmerica.get(key:"presidente"));
System.out.println("Director Tecnico: " + infoAmerica.get(key:"director"));
System.out.println("Total Goles: " + infoAmerica.get(key:"goles"));

System.out.println();
// Mostrar información de Equipo  Atlético Nacional
Map<String, Object> infoNacional = participantesLiga.get(index: 1);
System.out.println(x: "Informacion del segundo equipo finalista:");
System.out.println("Equipo: " + infoNacional.get(key:"equipo"));
System.out.println("Presidente: " + infoNacional.get(key:"presidente"));
System.out.println("Director Tecnico: " + infoNacional.get(key:"director"));
System.out.println("Total Goles: " + infoNacional.get(key:"goles"));

System.out.println();
// Mostrar información de Equipo  SantaFe
Map<String, Object> infoSantaFe = participantesLiga.get(index: 2);
System.out.println(x: "Informacion del tercer equipo finalista:");
System.out.println("Equipo: " + infoSantaFe.get(key:"equipo"));
System.out.println("Presidente: " + infoSantaFe.get(key:"presidente"));
System.out.println("Director Tecnico: " + infoSantaFe.get(key:"director"));
System.out.println("Total Goles: " + infoSantaFe.get(key:"goles"));
```

LISTAS TIPADAS: Las listas tipadas, también conocidas como listas genéricas, son estructuras de datos que permiten almacenar y manipular elementos de un tipo específico de manera segura, sin necesidad de conversiones explícitas. Al ser creadas, se especifica el tipo de datos que contendrán, lo que proporciona seguridad y claridad en el código. Ofrecen ventajas como seguridad de tipo, claridad en el código, reutilización de código y evita conversiones explícitas. Son una característica esencial en muchos lenguajes de programación modernos como Java, C#, Python (a través de bibliotecas como typing).

Figura 36

```
import java.util.ArrayList;
import java.util.List;

public class Equipos {
    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(e: "America De cali");
        listaTipada.add(e: "Deportivo Cali");
        listaTipada.add(e: "Independiente SantaFe");
        listaTipada.add(e: "Millonarios");
        listaTipada.add(e: "Independiente Medellin");
        listaTipada.add(e: "Atletico Nacional");
        listaTipada.add(e: "Deportivo Pereira");
        listaTipada.add(e: "Junior FC");
        listaTipada.add(e: "Deportes Tolima");
        listaTipada.add(e: "La Equidad");
        listaTipada.add(e: "Real Santander");
        listaTipada.add(e: "Boyaca Chico");

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

        // Imprimir los elementos de la lista
        System.out.println(x: "Equipos de la Liga BetPlay Femenina 2023 :\n");
        for (String elemento : listaTipada) {
            System.out.println(x: elemento);
        }

    }
}
```

LISTAS POSICIÓN ORDINAL : Una lista posición ordinal es una estructura de datos que asigna una posición única y específica a cada elemento en relación con los demás. Esta posición se basa en un orden natural, como números enteros o índices, permitiendo acceder a los elementos de manera secuencial. Es útil para mantener un orden específico entre los elementos, como en listas de reproducción de música o clasificaciones de equipos en un torneo.

Figura 37

```
import java.util.ArrayList;
import java.util.List;

public class Posiciones {
    public static void main(String[] args) {
        // Crear una "Lista Posición Ordinal" para los finalistas de la LigaBetPlayfem2023
        List<String> finalistas = new ArrayList<>();
        finalistas.add(e: "America");
        finalistas.add(e: "SantaFe");
        finalistas.add(e: "Atletico Nacional");
        finalistas.add(e: "Pereira");
        finalistas.add(e: "Deportaivo Cali");

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


        // Mostrar cada nombre con su posición en el equipo
        for (int i = 0; i < finalistas.size(); i++) {
            int posicion = i + 1; // Sumar 1 para que las posiciones comiencen desde 1
            String nombre = finalistas.get(index: i);
            System.out.println("Finalista " + posicion + ": " + nombre);
        }
    }
}
```

MÉTODOS DE ITERACCION: Los métodos de iteración en programación son técnicas para recorrer y procesar elementos de una estructura de datos, como listas o conjuntos. Permiten acceder a los elementos de forma secuencial y realizar operaciones específicas en cada uno. Los métodos incluyen bucles (como for y while), foreach, recursión, y funciones de alto orden. También existen herramientas específicas en algunos lenguajes, como la Stream API en Java o los métodos de map y filter en Python, que facilitan la manipulación de datos de manera más eficiente y expresiva. La elección del método dependerá del lenguaje y el tipo de datos que se esté utilizando.

Figura 38

```
import java.util.List;


public class IteradorLista {

    public static void iterar(List<String> lista) {
        for (String elemento : lista) {
            // Haz algo con cada elemento, por ejemplo imprimirlo
            System.out.println(x: elemento);
        }
    }

    public static void main(String[] args) {
        List<String> miLista = List.of(e1: "Elemento 1", e2: "Elemento 2", e3: "Elemento 3");
        iterar(lista: miLista);
    }
}
```

PILA: Una pila es una estructura de datos que sigue el principio de "último en entrar, primero en salir" (LIFO). Esto significa que el último elemento añadido es el primero en ser retirado. Las operaciones básicas en una pila son "push" para añadir un elemento en la cima, "pop" para retirar el elemento de la cima, y "push" para obtener el elemento superior sin retirarlo. Las pilas se utilizan para rastrear el orden de procesamiento en situaciones donde es importante. Pueden ser implementadas de manera estática o dinámica según los requisitos y el lenguaje de programación utilizado.
Figura 39

```java
import java.util.Stack;

public class TotalPuntos {
    public static void main(String[] args) {
        // Crear una pila llamada "TotalPuntos" para simular el total de puntos anotados por los equipos
        Stack<Integer> TotalPuntos = new Stack<>();
        TotalPuntos.push(item: 40); // puntos del equipo America De cali
        TotalPuntos.push(item: 35); // puntos del equipo SantaFe
        TotalPuntos.push(item: 30); // puntos del equipo Atletico Nacional
        TotalPuntos.push(item: 29); // puntos del equipo Pereira
        TotalPuntos.push(item: 27); // puntos del equipo Deportivo Cali
        TotalPuntos.push(item: 27); // puntos del equipo Independiente Medellin
        TotalPuntos.push(item: 25); // puntos del equipo Cor
        TotalPuntos.push(item: 23); // puntos del equipo Equidad
        TotalPuntos.push(item: 23); // puntos del equipo Millonarios
        TotalPuntos.push(item: 22); // puntos del equipo Llaneros Femenina
        TotalPuntos.push(item: 22); // puntos del equipo Junior
        TotalPuntos.push(item: 21); // puntos del equipo Boyacá Chicó Femenina
        TotalPuntos.push(item: 20); // puntos del equipo Huila
        TotalPuntos.push(item: 20); // puntos del equipo RST
        TotalPuntos.push(item: 13); // puntos del equipo Pasto
        TotalPuntos.push(item: 12); // puntos del equipo Bucaramanga
        TotalPuntos.push(item: 6); // puntos del equipo Tolima

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

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

JTABLE: Es un elemento gráfico de Java Swing que se emplea para representar datos en una estructura tabular con filas y columnas. Facilita la adaptación de su apariencia, la selección de celdas, así como la capacidad de ordenar y filtrar datos. Además, permite reaccionar a eventos específicos. Se basa en un modelo de datos y es comúnmente empleado en aplicaciones de escritorio para presentar información de manera ordenada y de fácil manipulación.
Figura 40

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

public class EjemploJTable {

    public static void main(String[] args) {
        // Crear datos de ejemplo (una matriz bidimensional)
        Object[][] datos = {
                {"Juan", "Pérez", 25},
                {"María", "Gómez", 30},
                {"Pedro", "López", 28},
                {"Laura", "Díaz", 35}
        };

        // Definir nombres de columnas
        String[] columnas = {"Nombre", "Apellido", "Edad"};

        // Crear el JTable
        JTable tabla = new JTable(rowData:datos, columnNames:columnas);

        // Crear un JFrame
        JFrame frame = new JFrame(title: "Ejemplo de JTable");
        frame.setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new BorderLayout());

        // Agregar la JTable a un JScrollPane (para permitir desplazamiento si hay muchas filas/columnas)
        JScrollPane scrollPane = new JScrollPane(view: tabla);
        frame.getContentPane().add(comp: scrollPane, constraints: BorderLayout.CENTER);

        // Ajustar el tamaño y hacer visible el JFrame
        frame.setSize(width: 400, height: 300);
        frame.setVisible(b: true);
    }
```

JLIST: En Java Swing es un componente de la interfaz gráfica que se emplea para presentar una lista vertical de elementos. Ofrece la capacidad de seleccionar uno o múltiples elementos, emplea un modelo de datos para suministrar dichos elementos y puede ser personalizado mediante renderizadores. Asimismo, permite gestionar eventos asociados con la lista y facilita la navegación cuando la lista es extensa. Es una herramienta comúnmente empleada en aplicaciones de escritorio para elegir opciones de una lista.
Figura 41

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

public class EjemploJList {

    public static void main(String[] args) {
        // Crear el Modelo de Datos
        DefaultListModel<String> modeloLista = new DefaultListModel<>();
        modeloLista.addElement(element:"Elemento 1");
        modeloLista.addElement(element:"Elemento 2");
        modeloLista.addElement(element:"Elemento 3");

        // Crear la Instancia de JList
        JList<String> lista = new JList<>(dataModel: modeloLista);

        // Crear un JFrame
        JFrame frame = new JFrame(title: "Ejemplo de JList");
        frame.setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new BorderLayout());

        // Agregar la JList a un JScrollPane (para permitir desplazamiento si hay muchos elementos)
        JScrollPane scrollPane = new JScrollPane(view: lista);
        frame.getContentPane().add(comp: scrollPane, constraints: BorderLayout.CENTER);

        // Ajustar el tamaño y hacer visible el JFrame
        frame.setSize(width: 300, height: 200);
        frame.setVisible(b: true);
    }
}
```

JPROGRESSBAR: Es un componente de la interfaz gráfica en Java Swing que se utiliza para mostrar el progreso de una operación en curso. Es especialmente útil cuando se ejecutan tareas que pueden tomar un tiempo significativo y se desea informar al usuario sobre el estado de la operación.
Figura 42

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

public class EjemploJProgressBar {
    public static void main(String[] args) {
        JFrame frame = new JFrame(title: "Ejemplo de JProgressBar");
        frame.setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE);
        frame.setSize(width: 300, height: 200);

        JProgressBar progressBar = new JProgressBar();
        progressBar.setStringPainted(b: true); // Muestra el texto de progreso

        JButton startButton = new JButton(text: "Iniciar");
        startButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Simulación de una tarea que toma tiempo
                Thread thread = new Thread(new Runnable() {
                    public void run() {
                        for (int i = 0; i <= 100; i++) {
                            try {
                                Thread.sleep(millis: 50); // Simula una operación que toma tiempo
                            } catch (InterruptedException ex) {
                                ex.printStackTrace();
                            }
                            progressBar.setValue(n: i); // Actualiza el valor de la barra de progreso
                        }
                    }
                });
                thread.start();
            }
        });

        JPanel panel = new JPanel();
```

JTREE: Es un componente de la biblioteca Swing de Java que permite mostrar datos de manera jerárquica en forma de árbol. Se utiliza para representar información organizada en una estructura de árbol, donde cada nodo puede tener nodos hijos.

Figura 43

```
import javax.swing.*;
import javax.swing.tree.DefaultMutableTreeNode;

public class EjemploJTree extends JFrame {
    public EjemploJTree() {
        // Crear un nodo raíz
        DefaultMutableTreeNode raiz = new DefaultMutableTreeNode(userObject: "Raíz");

        // Crear nodos hijos
        DefaultMutableTreeNode hijo1 = new DefaultMutableTreeNode(userObject: "Hijo 1");
        DefaultMutableTreeNode hijo2 = new DefaultMutableTreeNode(userObject: "Hijo 2");

        // Agregar hijos al nodo raíz
        raiz.add(newChild: hijo1);
        raiz.add(newChild: hijo2);

        // Crear el JTree con la raíz como punto de partida
        JTree arbol = new JTree(root: raiz);

        // Agregar el JTree al JFrame
        add(new JScrollPane(view: arbol));

        // Configurar la ventana
        setTitle(title: "Ejemplo JTree");
        setSize(width: 200, height: 200);
        setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(c: null);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            EjemploJTree ejemplo = new EjemploJTree();
```

SETTITLE: Es un método de la clase JFrame en Java. Este método se utiliza para establecer el título de una ventana o marco (frame) en una interfaz gráfica de usuario.

Figura 44

```
SetTittle SetTittle = new JLabel(text: "BIENVENIDO A LA APP LIGA FEMENINA BETPLAY 2023")
SetTittle.setFont(new Font(name: "Arial", style: Font.BOLD, size: 15)); // Establece fuente
SetTittle.setForeground(Color.WHITE); // Color del texto
```

JCOMBOBOX : Es un componente de interfaz gráfica en Java que proporciona una lista desplegable de opciones para que los usuarios seleccionen.

Figura 45

```
JComboBox<String> comboBoxMenu = new JComboBox<String>(items: opcionesMenu);
comboBoxMenu.setAlignmentX(alignmentX: Component.CENTER_ALIGNMENT); // Centrar en X
comboBoxMenu.setPreferredSize(new Dimension(width: 200, height: 40)); // Tamaño menu
```

JBUTTON : Es un componente de interfaz gráfica en Java que representa un botón. Los usuarios pueden hacer clic en él para desencadenar acciones en una aplicación.

Figura 46

```
// Crea un botón con el texto "Salir"
JButton botonSalir = new JButton(text: "Salir");

botonSalir.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        System.exit(status: 0);  // Termina la ejecución del programa con código de salida 0
    }
});
```

JFRAME : Es una clase en la biblioteca Swing de Java que representa una ventana con una interfaz gráfica de usuario. Es una de las clases más fundamentales para la creación de aplicaciones de escritorio en Java.

Figura 47

```
    // Método para mostrar la ventana del menú principal
rivate void mostrarVentanaMenu() {
    // Crear un JFrame para el menú principal
    JFrame menuFrame = new JFrame(title: "Menú Principal");
    menuFrame.setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE);
    menuFrame.setSize(width: 800, height: 600);// tañamo
```

JLABEL: Es un componente de interfaz gráfica en Java que proporciona un área para mostrar texto o una imagen. Es utilizado para proporcionar información o etiquetar otros componentes en una GUI.

Figura 48

```
// Añadir título centrado en la parte superior
JLabel labelTitulo = new JLabel(text: "LIGA FEMENINA BETPLAY 2023");  // Crea una etiqueta c
labelTitulo.setFont(new Font(name: "Arial", style: Font.BOLD, size: 20));  // fuente de texto
labelTitulo.setHorizontalAlignment(alignment: JLabel.CENTER);  // Centrar el texto horizontal
labelTitulo.setForeground(fg: Color.WHITE); // Color de texto
```

VII. CÓDIGO EN PSEINT

Figura 49

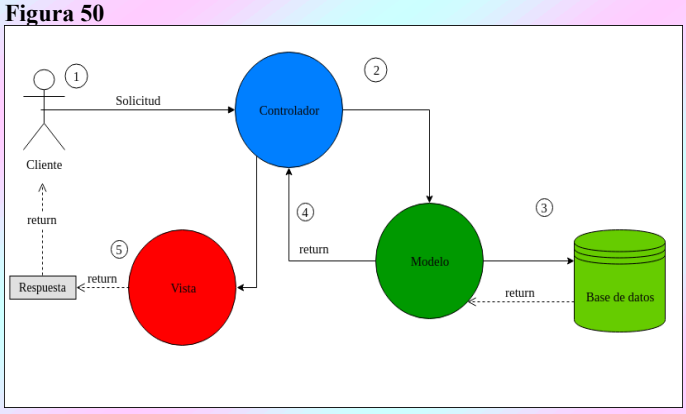
```
Algoritmo AutenticacionBasica

    // Definir variables
    Cadena usuarioCorrecto = "Proyecto"
    Cadena contraseñaCorrecta = "LBF"
    Cadena usuarioIngresado, contraseñaIngresada

    // Solicitar usuario y contraseña
    Escribir "Ingrese su nombre de usuario:"
    Leer usuarioIngresado
    Escribir "Ingrese su contraseña:"
    Leer contraseñaIngresada

    // Verificar autenticación
    Si usuarioIngresado = usuarioCorrecto Y contraseñaIngresada = contraseñaCorrecta Entonces
        Escribir "Autenticación exitosa"
    Sino
        Escribir "Usuario o contraseña incorrectos"
    FinSi

FinAlgoritmo
```

```
// Mostrar ventana de autenticación
Escribir "BIENVENIDO A LA APP LIGA FEMENINA BETPLAY 2023"
Escribir "Usuario:"
Leer usuario
Escribir "Contraseña:"
Leer contraseña

// Autenticar usuario
autenticado = Autenticar(usuario, contraseña)

Si autenticado Entonces
    Escribir "Autenticación exitosa"
    Escribir "Menú Principal"
    Escribir "1. Equipos"
    Escribir "2. América de Cali"
    Escribir "3. Independiente Santa Fe"
    Escribir "4. Atlético Nacional"
    Escribir "5. Deportivo Pereira"
    Escribir "6. Deportivo Cali"
    Leer opcion

    // Obtener información del equipo seleccionado
    Segun opcion Hacer
        1: Escribir "Opción no válida"
        2: MostrarInformacionEquipo(AmericaDeCali)
        3: MostrarInformacionEquipo(IndependienteSantaFe)
        4: MostrarInformacionEquipo(AtleticoNacional)
        5: MostrarInformacionEquipo(DeportivoPereira)
        6: MostrarInformacionEquipo(DeportivoCali)
        Sino: Escribir "Opción no válida"
    FinSegun

Sino
```

VIII. ARQUITECTURA MVC

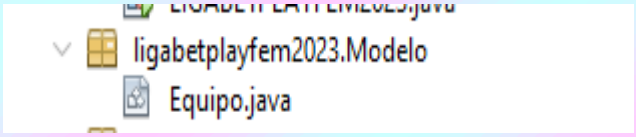
El proyecto seguirá la arquitectura Modelo Vista Controlador (MVC), la cual divide el código en distintas funciones, manteniendo capas separadas que se ocupan de tareas específicas. Esto proporciona una organización más efectiva, facilita el mantenimiento y promueve la reutilización del código en la aplicación.

Figura 50



MODELO: El modelo en una aplicación de software se encarga de la estructura lógica de los datos, sin incluir detalles sobre la interfaz de usuario. Su función principal es actuar como un intermediario entre la vista, el controlador y la base de datos.

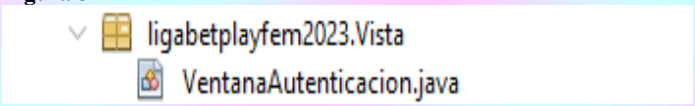
Figura 51



VISTA: La vista se encarga de mostrar al usuario la información que proviene del modelo. Esto se materializa a través de interfaces visuales como pantallas que contienen los datos del modelo. Estos datos pueden presentarse en campos, ventanas de edición, tablas, entre otros formatos.

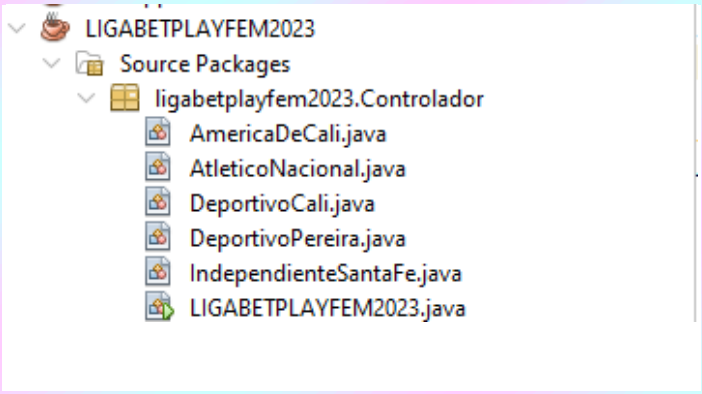
Además, es posible que los datos se muestren en modo de solo lectura o que se puedan editar.

Figura 52



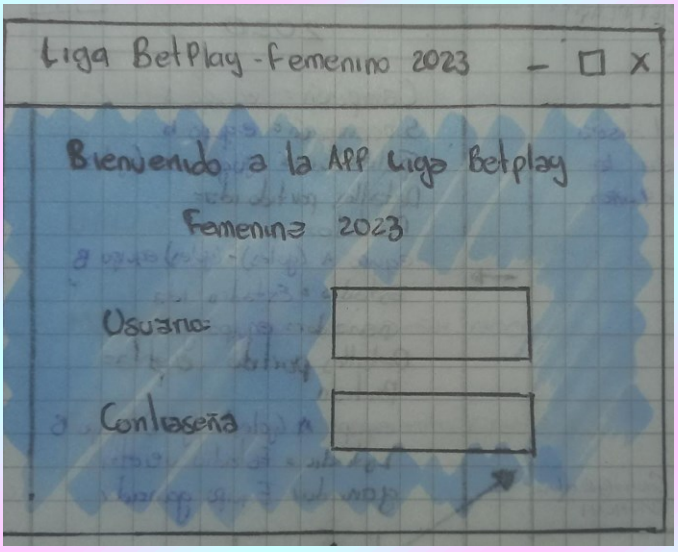
El CONTROLADOR: El Controlador responde a las peticiones del cliente (usuario) hechas a través del navegador. Se comunica con el Modelo para obtener los datos necesarios y luego elige la Vista apropiada para presentar esta información al usuario.

Figura 53



IX. MOCKUPS

Figura 54



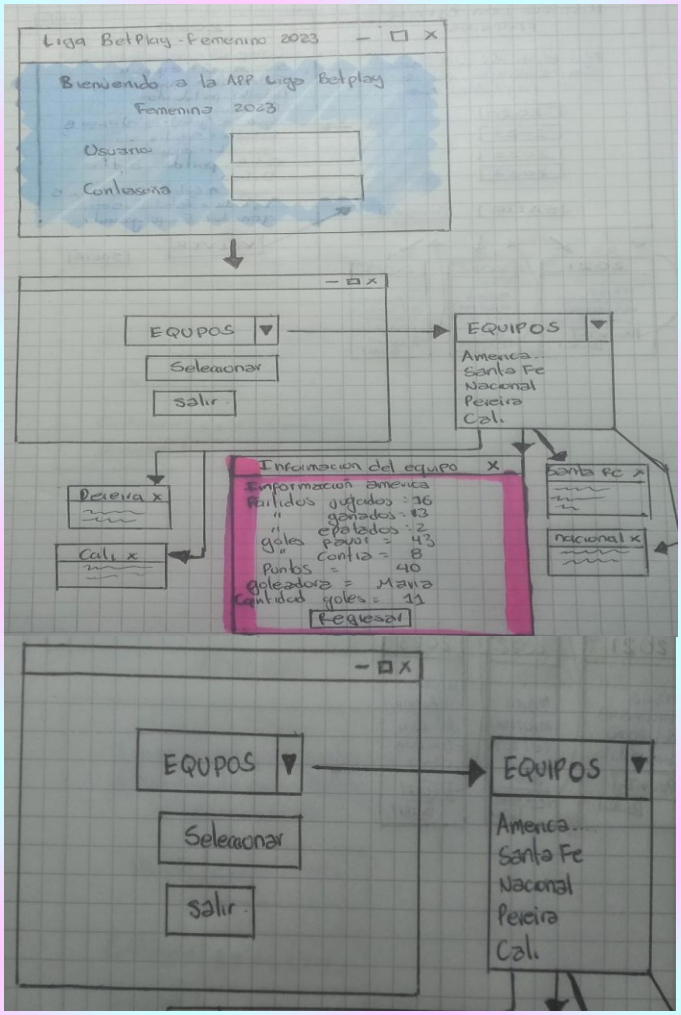
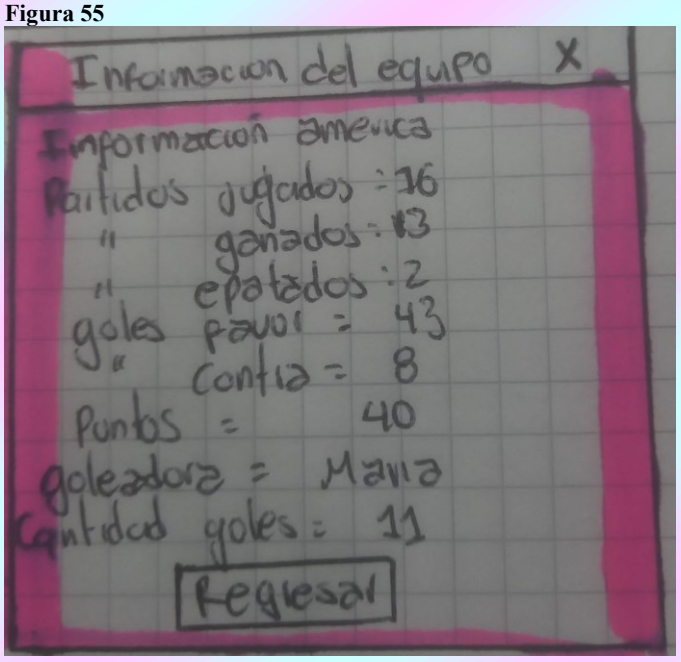


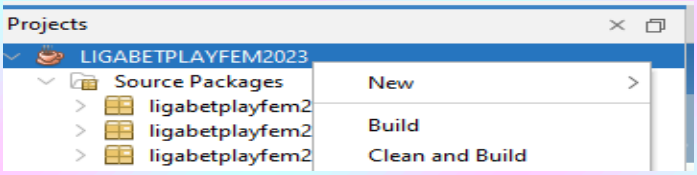
Figura 55



X. EJECUCIÓN DEL CÓDIGO CON .JAR

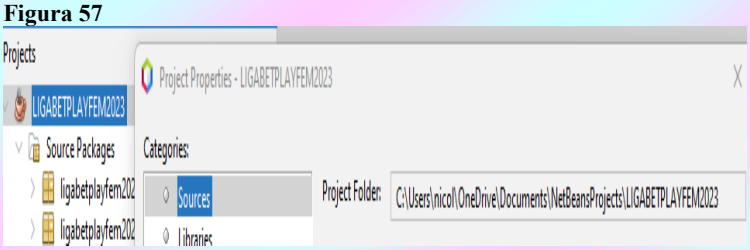
Para realizar este proceso debemos abrir NetBeans, seleccionar el proyecto a compilar y dar clic derecho y hacer clic en clean and build, esto genera un archivo .jar

Figura 56



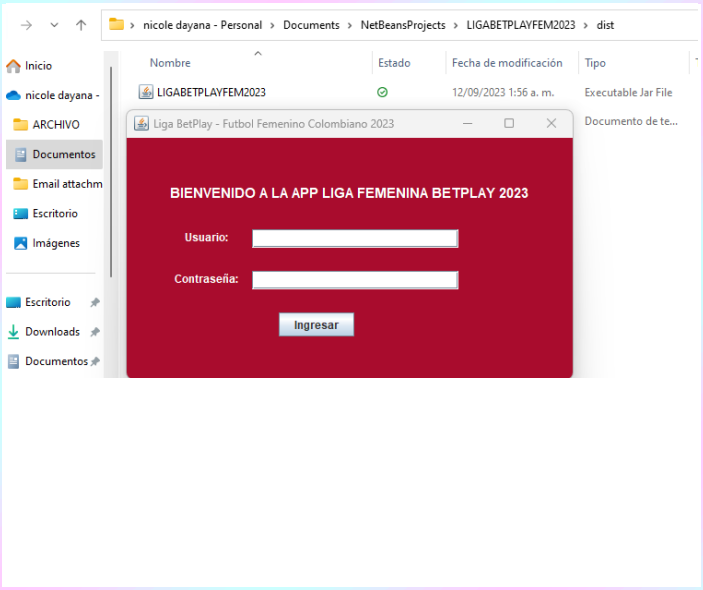
A continuación hacemos clic derecho, luego en “properties” ,luego en sourses, aquí se puede copiar la ruta del archivo en Project Folder

Figura 57



A continuación abrimos una ventana con tecla Windows + E y pegamos este link en la barra de inicio, de esta manera se ejecuta el archivo

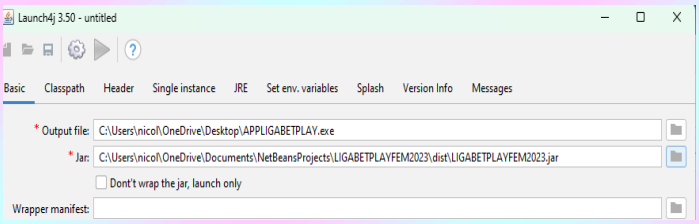
Figura 58



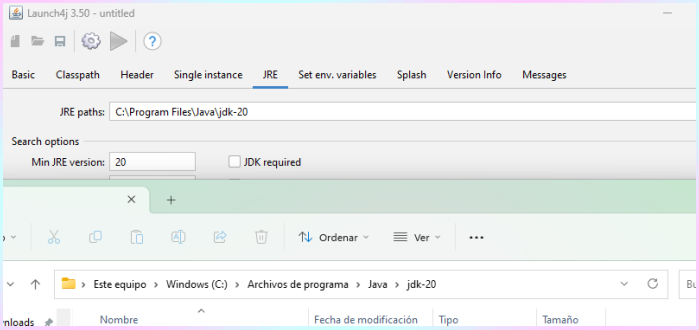
XI. PASAR CÓDIGO DE .JAR A .EXE

Para este proceso debemos descargar el siguiente archivo
Find out more about Launch4j Executable Wrapper | SourceForge.net

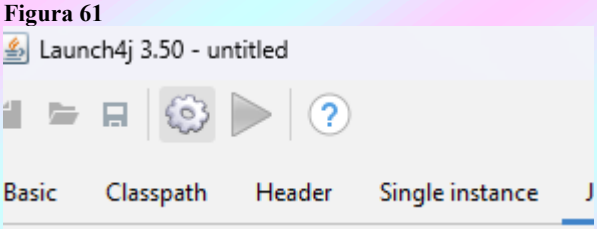
Una vez descargado seleccionamos donde se va a guardar el archivo (debe poner .exe al final del nombre) y ubicamos el archive .Jar
Figura 59



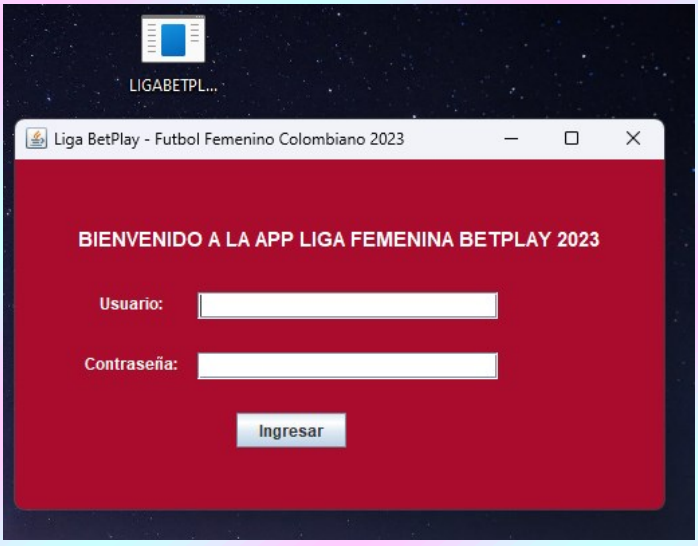
A continuación en la ventana "JRE " ubicamos el archivo JDK de neatbeans y pegamos el link en "JRE Paths" como se muestra a continuación:
Figura 60



Luego damos click en el simbolo de configuración :
Figura 61

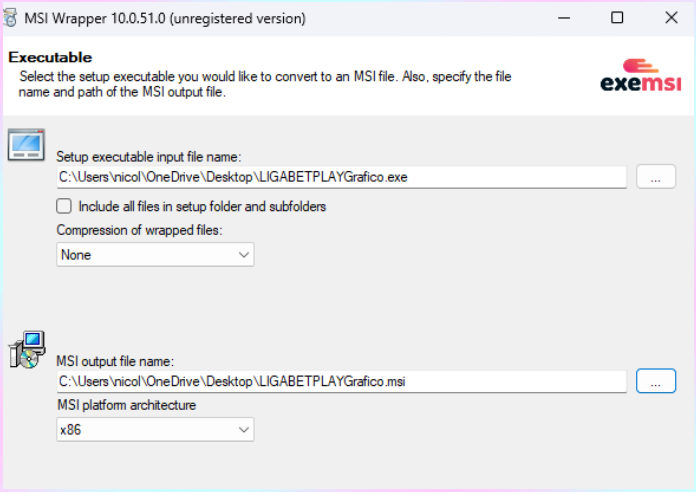


Y por último en el simbolo de "Play" para ejecutar el programa tal como se muestra a continuación:
Figura 62

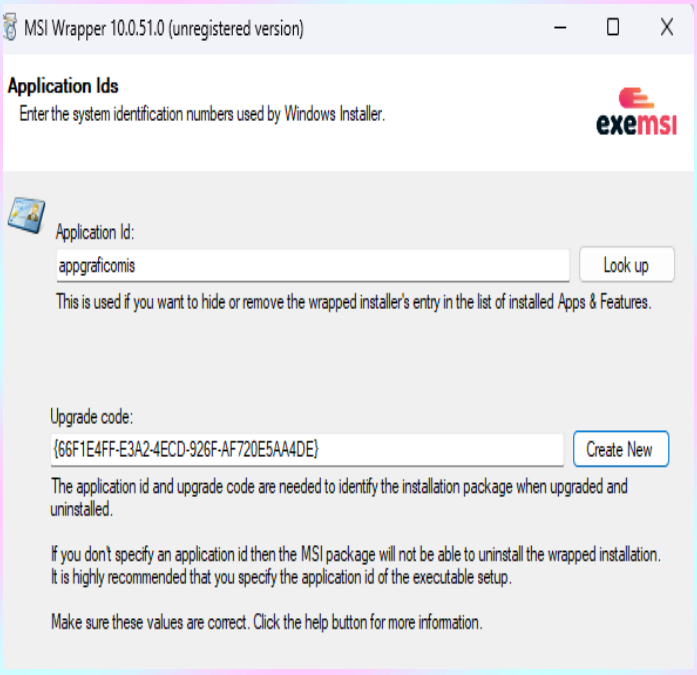


XII. PASAR CÓDIGO DE .EXE A MSI

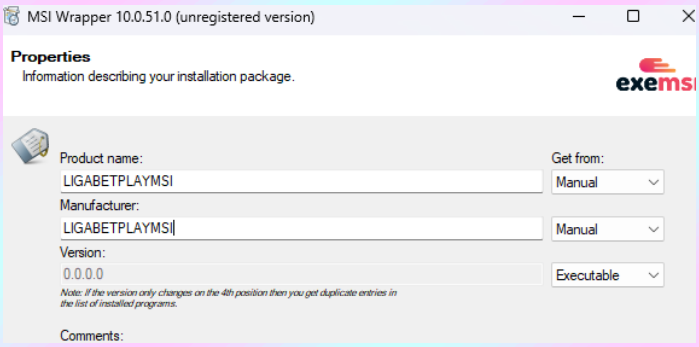
Para realizar este proceso instalamos el siguiente programa MSI_Wrapper_10_0_51_0 (mediafire.com), al ejecutarlo debemos ubicar la ruta del archivo .exe y guardar este archive que estamos creando en el escritorio
Figura 63



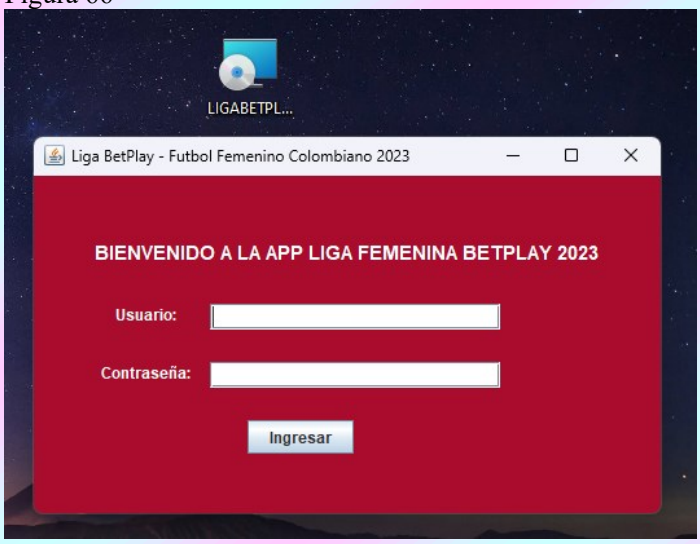
Luego, introducimos el nombre de nuestro archivo MSI y le damos click en crear new"CODE":
Figura 64



A continuación modificamos los siguientes campos con el nombre de nuestro archivo y con la opción “manual”:
Figura 65



Le damos next y se crea nuestro archvio listo para ser ejecutado:
Figura 66



XIII. CONCLUSIONES

1. Información de Equipos de Fútbol Femenino: Este proyecto se dedica a administrar datos concernientes a los equipos de fútbol femenino que participaron en la Liga BetPlay 2023. Esto implica que su propósito es ofrecer un análisis minucioso del rendimiento de dichos equipos en la liga.

2. Interfaz Gráfica Atractiva y Personalizada: La implementación de un sistema de autenticación subraya el interés por garantizar la seguridad y el acceso regulado a la aplicación. Esto resulta fundamental para salvaguardar la información y las funciones del programa.

3. Autenticación de Usuarios: La inclusión de un sistema de autenticación evidencia la preocupación por la seguridad y el acceso controlado a la aplicación. Esto es esencial para proteger la información y funcionalidades del aplicativo.

4. Aplicación de Programación Orientada a Objetos (POO): La implementación de clases y objetos demuestra una aplicación consistente del enfoque de Programación Orientada a Objetos (POO). Esto simplifica la estructuración y manipulación de la información, además de favorecer la reutilización de código de manera eficiente.

5. Facilidad de Interacción: La inclusión de elementos como botones y campos de texto indica un enfoque en la usabilidad y la interacción intuitiva. Esto es importante para que los usuarios puedan navegar y utilizar el aplicativo de manera sencilla.

6. Durante el desarrollo de este aplicativo aprendí a introducir el modelo vista controlador, me percarte de nuevas funcionalidades para la implementación de la interfaz gráfica mediante el uso del código. Recordé el uso del aplicativo Pseint, no se me facilitaron los diagramas pero los trate de implementar de la mejor manera.

XIV. REFERENCIAS

[1] Colombia.com. (s. f.). Tabla de posiciones - Liga Profesional Femenina de Fútbol de Colombia. https://www.colombia.com/futbol/femenino/tabla-de-posiciones

[2] Que es el modelo Vista Controlador (MVC) y como funciona. (s. f.). Educación para todos. https://articulosvirtuales.com/articles/educacion/que-es-el-modelo-vista-controlador-mvc-y-como-funciona

[3] La Dimayor – Dimayor. (s. f.). https://dimayor.com.co/la-dimayor/

[4] Harol. (s. f.). Java_Estructura_Datos/Presentacion_Contenidos.pdf at main · Harol003/Java_Estructura_Datos. GitHub. https://github.com/Harol003/Java_Estructura_Datos/blob/main/Presentacion_Contenidos.pdf

[4] Smad, R. (2021, 8 julio). ¡Prográmese! Este sábado comienza la Liga Betplay Femenina - Santa Marta al día. Santa Marta Al Día. https://santamartaaldia.co/programese-este-sabado-comienza-la-liga-betplay-femenina/

[5] Paul, J. (s. f.). Componentes básicos de Java Swing. https://javajhon.blogspot.com/2020/10/swing-b.html

[6] Liga Femenina BetPlay DIMAYOR – Dimayor. (2023, 7 septiembre). https://dimayor.com.co/liga-femenina/

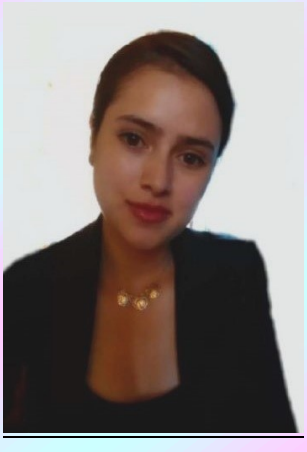
[7] MSI_Wrapper_10_0_51_0. (s. f.). MediaFire. https://www.mediafire.com/file/sjhfgeow8pykuqw/MSI_Wrapper_10_0_51_0.msi/file

[8] Find out more about Launch4J Executable Wrapper | SourceForge.net. (2022, 30 noviembre). SourceForge. https://sourceforge.net/projects/launch4j/postdownload

XV. LISTA DE IMÁGENES

XVI. LISTA DE TABLAS

XVII. CREDITOS



Mi nombre es Nicole Dayana Pajarito Santamaria, tengo 26 años, me gusta leer, correr, viajar, escuchar música, séptimo arte. Actualmente estoy estudiando una tecnología en Desarrollo de Software en el Politecnico Internacional. Este proyecto hace parte de Programación I, tercer ciclo.