



APP FEMDATA PRO

David Sebastian Enciso Lopez david.enciso@pi.edu.co

Desarrollo de software Politécnico Internacional estructura de datos y programación I
Autopista Sur N° 67-71



Resumen – Este proyecto tiene como objetivo crear una aplicación con la ayuda del IDE NetBeans, Java Development Kit (JDK) y las diversas opciones que se nos presentan en este IDE abordando las temáticas vistas en estructura de datos y programación 1 como lo son: herencia, polimorfismo, encapsulamiento, abstracción entre otros que nos darán como resultado una muy buena aplicación. Esta aplicación nos mostrara las estadísticas (tabla de posiciones) de la liga femenina 2023, equipos campeones en la historia y plantilla de los primeros 5 equipos de la liga 2023 para ser consultada por los fanáticos del futbol femenino. Se quiere que esta aplicación sea agradable al usuario final y que sea una guía de consulta en todos los ámbitos deportivos por medio de una interfaz gráfica agradable e intuitiva y que pueda ser usada por chicos y grandes y hacer de ella la favorita de todos en sus consultas.

Palabras claves – Java Development Kit, NetBeans, , Programación, estructura de datos, interfaz gráfica.

Abstract – This project aims to create an application with the help of NetBeans IDE, Java Development Kit (JDK) and the various options presented in this IDE addressing the issues seen in data structure and programming 1 such as: inheritance, polymorphism, encapsulation, abstraction and others that will result in a very good application. This application will show us the statistics (table of positions) of the women's league 2023, champion teams in history and template of the first 5 teams of the league 2023 to be consulted by the fans of women's soccer. We want this application to be pleasant to the end user and be a reference guide in all areas of sports through a nice and intuitive graphical interface that can be used by young and old and make it the favorite of all in their queries.

Se plantea escribir un código 100% funcional con interfaz gráfica que será intuitivo y de fácil acceso con las siguientes características:

1. Validar el nombre de usuario y contraseña que para este caso se usara por defecto usuario “David” y contraseña “Clave123”.

2. Si la validación es correcta nos mostrará un menú de opciones, la primera opción mostrará los equipos campeones en una lista con sus años ordenados del más reciente al más antiguo, la segunda permitirá consultar la plantilla de las 11 jugadoras titulares y el DT de los primeros 5 equipos de la liga 2023 y como última opción nos permitirá consultar las estadísticas de la liga actual.
3. La primera opción mostrará los equipos campeones de la liga femenina ordenados del más reciente al más antiguo.
4. La opción 2 desplegará un menú donde se va a poder elegir 1 de los 5 equipos habilitados que para el caso serán los 5 primeros de la liga 2023 fase todos contra todos. (América, Santa Fe, Nacional, Pereira y Cali)
5. La opción 3 mostrará las estadísticas de la liga femenina 2023 con partidos jugados PJ, ganados PG, empatados PE, perdidos PP, goles a favor GF, en contra GC, diferencia de goles DG y puntos totales obtenidos PTS.

VERSIONES DE DESARROLLO

```
Product Version: Apache NetBeans IDE 17
Java: 20.0.1; Java HotSpot(TM) 64-Bit Server VM 20.0.1+9-29
Runtime: Java(TM) SE Runtime Environment 20.0.1+9-29
System: Windows 10 version 10.0 running on amd64; UTF-8; es_ES (nb)
User directory: C:\Users\Enciso Suarez\AppData\Roaming\NetBeans\17
Cache directory: C:\Users\Enciso Suarez\AppData\Local\NetBeans\Cache\
```

Figura 1.Requerimientos Funcionales.Creación propia

I. INTRODUCCION

Este proyecto está pensado en el desarrollo de una aplicación para todas las personas amantes del futbol colombiano femenino y que permitirá mostrar a los usuarios una diversa y muy completa información la liga de una manera sencilla y agradable para todo el público, usaremos como base la programación orientada a objetos (POO), en la cual intentaremos abordar las diferentes temáticas vistas en clase principalmente sus 4 pilares como lo son:

¿Qué es Programación orientada a Objetos?

La Programación orientada a objetos (POO) es un paradigma de programación el cual se centra en la creación de clases y estas a su vez llevan objetos cuya finalidad es poder relacionarse entre si para lograr el objetivo de un aplicativo, se podría decir que la POO es la manera más cercana de llevar

las cosas de la vida real aun código que sea de fácil manejo y usabilidad.

1. Herencia:

Es la que nos permite heredar o extender atributos de una clase a otra o relacionar objetos entre sí esto con el fin de optimizar el código.

```
public class Arquera extends Jugadora {
    //Método constructor = new Arquera
    public Arquera(String nombre, String apellido, int edad, int dorsal, String nacionalidad) {
        super(nombre, apellido, edad, dorsal, posicion:"Arquera", nacionalidad);
    }
}
```

Figura 2. Herencia.Creación propia

2. Polimorfismo.

El polimorfismo permite a un método tener diferentes comportamientos y diferentes resultados gracias a que los objetos a los que pertenece comparten la superclase.

```
// Pintar las filas con los datos en la tabla
for (Jugadora jugadora : jugadoras) {
    tableModel.addRow(new Object[]{
        jugadora.getNombre(),
        jugadora.getApellido(),
        jugadora.getEdad(),
        jugadora.getDorsal(),
        jugadora.getPosicion(),
        jugadora.getNacionalidad()
    });
}
```

Figura 3. Polimorfismo.Creación propia

3. Encapsulamiento.

El encapsulamiento está encaminado a la protección o limitación en el manejo y control de los datos o elementos.

```
public class Estadistica {
    private int posicion, partidosJugados, partidosGanados, partidosEmpatados,
        partidosPerdidos, golesAFavor, golesEnContra, diferenciaDeGoles, puntos;
    private String equipo;

    public int getPosicion() {
        return posicion;
    }

    public void setPosicion(int posicion) {
        this.posicion = posicion;
    }

    public String getEquipo() {
        return equipo;
    }

    public void setEquipo(String equipo) {
        this.equipo = equipo;
    }

    public int getPartidosJugados() {
        return partidosJugados;
    }

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

Figura 4. Encapsulamiento.Creación propia

4. Abstracción.

El a representación de un objeto en clases que no puede ser instanciada por sí misma más si puede ser extendida a otras clases.

```
public class AdministradorDeDatos {
    // muestra al usuario opciones de equipos que desea consultar y/o tabla de posiciones
    public ArrayList<Estadistica> obtenerEstadisticasDelTorneo(int idEquipo) {
    }
}
```

Figura 5. Abstraccion.Creación propia

II. DICCIONARIO DE DATOS

DICCIONARIO DE DATOS		
NOMBRE	TIPO	DESCRIPCIÓN
USERNAME	String	constante para validar el usuario ingresado
PASSWORD	String	constante para validar la contraseña ingresada
Nombre	String	variable para almacenar el nombre de la jugadora
Apellido	String	variable para almacenar el apellido de la jugadora
edad	int	variable para almacenar la edad de la jugadora
dorsal	int	variable para almacenar el numero de la camiseta
Posicion	String	variable para almacenar la posicion de la jugadora
Nacionalidad	String	variable para almacenar la nacionalidad de la jugadora
dorsal	int	variable para almacenar el numero de la camiseta
AdministradorDeDatos	estadistica	Llama al objeto para mostrar las estadísticas
AdministradorJugadoras	Jugadora	Llama al objeto para mostrar las jugadoras
AdministradorCampeones	Equipos_Campeones	Llama al objeto para mostrar equipos campeones
opcionMenu	int	variable que almacena la eleccion del usuario en el menu ppal
tabla	int	variable que almacena la eleccion del usuario en el menu de estadísticas
pos	int	variable que indica la posicion del equipo
equipo	String	variable que indica el nombre del equipo
partidos jugados	int	variable que indica el # de partidos jugados por cada equipo
partidos ganados	int	variable que indica el # de partidos ganados por cada equipo
partidos empatados	int	variable que indica el # de partidos empatados por cada equipo
partidos perdidos	int	variable que indica el # de partidos perdidos por cada equipo
goles a favor	int	variable que indica el # de goles a favor por cada equipo
goles en contra	int	variable que indica el # de goles en contra por cada equipo
diferencia de goles	int	variable que indica el # de diferencia de goles por cada equipo
puntos	int	variable que indica el # de puntos por cada equipo
FramePrincipal	Jframe	ventana principal
BotonOpcion	Jbutton	Botones para cada una de las ventanas y sus acciones
comboBox	JcomboBox	opciones desplegables para ver estadísticas
etiquetas	Jlabel	etiquetas para mostrar texto en ventana
textFieldNombre	JtextField	espacio para escribir nombre de usuario
textFieldPassword	JPasswordField	espacio para escribir contraseña

Figure 6. Diccionario de datos.Creación propia

III. DIAGRAMA DE FLUJO

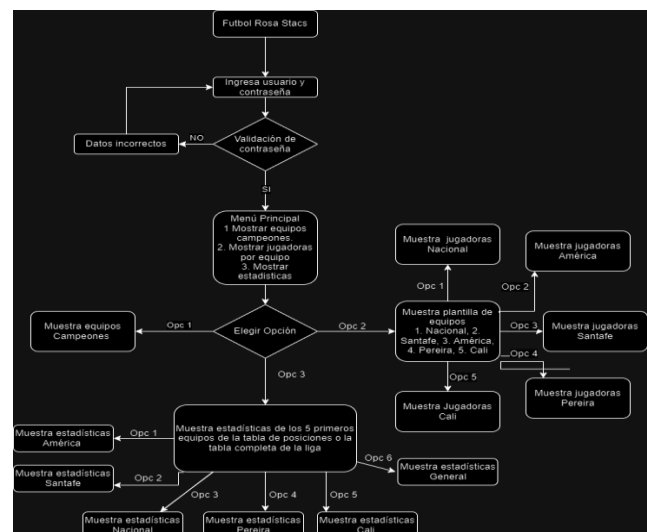


Figura 7. Diagrama de flujo.Creación propia

IV. Pseudo-Código:

1. Ingresar usuario y contraseña.
2. Si el login es correcto envía a menú principal.
3. Se muestran 4 opciones:
 - Mostrar equipos campeones.
 - Mostrar plantilla de primeros cinco equipos de la tabla de posiciones.
 - Mostrar estadísticas de la liga actual.
 - Salir.
4. Si selecciona la opción 1 mostrara los equipos campeones de manera descendente.
5. Se da opción de volver al menú principal.
6. Si selecciona la opción 2 nos mostrara un nuevo menú para seleccionar 1 equipo y consultar su plantilla actual (Nacional, América, Santafé, Pereira o Cali).
7. Se da opción de volver a menú principal.
8. Si selecciona la opción 3 lo llevará a un nuevo menú esta vez en “desplegable” donde podrá seleccionar 1 equipo para ver sus estadísticas o la tabla de posiciones general de la liga Betplay Femenina 2023 con sus 17 equipos participantes.
9. Finalmente se da opción de volver al menú principal y salir del programa.

V. DIAGRAMA DE CLASES

A continuación, se presenta el diagrama de clases correspondiente a la elaboración de la aplicación para consulta de estadísticas y datos relevantes de la Liga BetPlay Femenina colombiana.

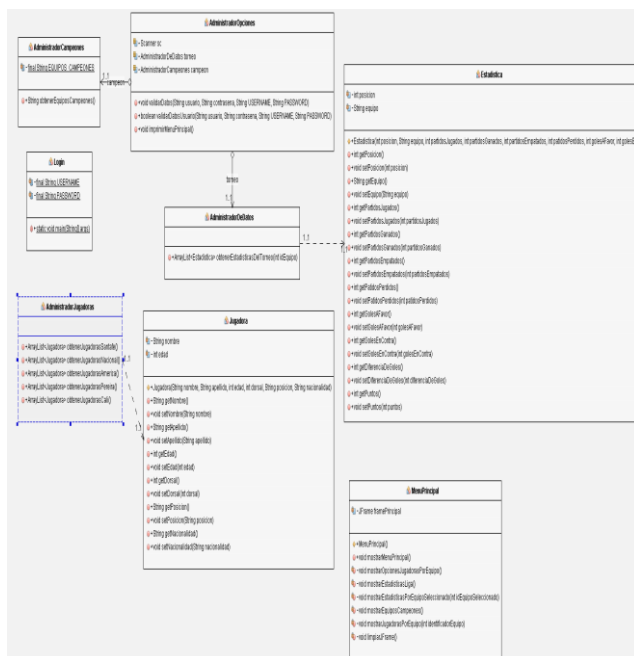


Figura 8. Diagrama de clases (creación propia)

VI. MODELO VISTA CONTROLADOR

El modelo vista controlador está pensado en la división de las partes que conforman un aplicativo, logrando la implantación de cada uno por separado y asegurando su buen mantenimiento, el modelo es el encargado de del manejo de datos, la vista es lo que finalmente puede ver el usuario por medio de la interfaz gráfica y el controlador es el encargado de la manipulación del modelo para poder mostrar la información por medio de la vista. (Avila, 2019)

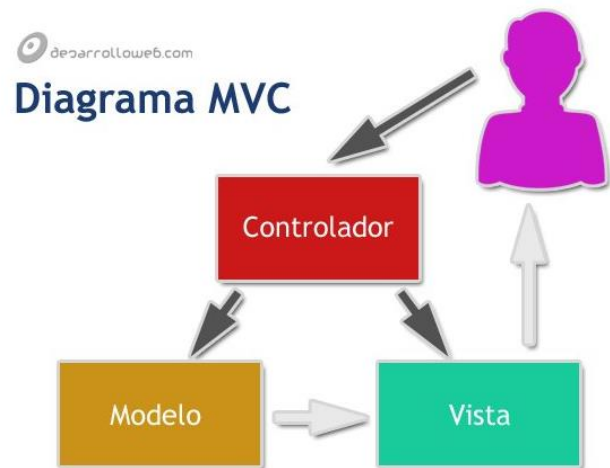


Figura 9. Modelo Vista Controlador

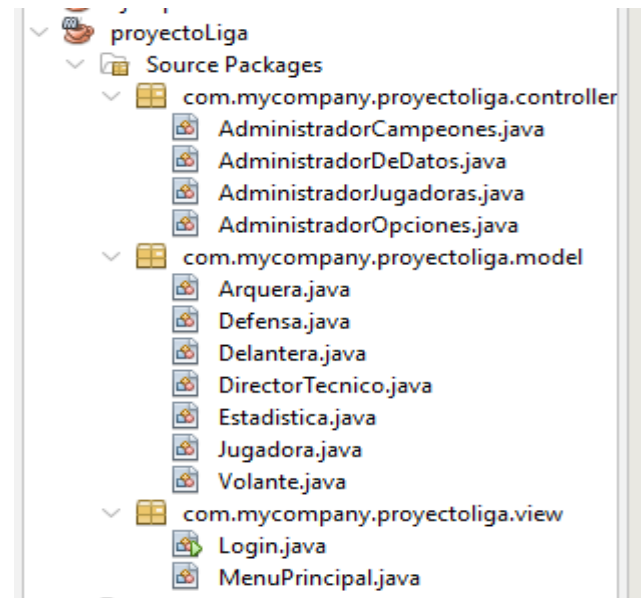


Figura 10. MVC Proyecto.Creación propia

VII. DIAGRAMA DE LISTAS SENCILLAS

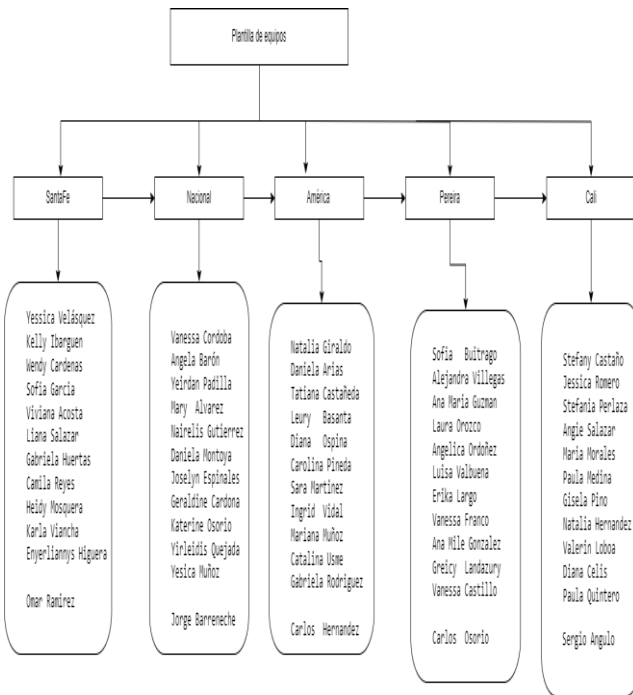


Figura 11. Diagrama de listas sencillas.Creación propia

VIII. DEFINICIONES ESTRUCTURA DE DATOS

1. **Listas Abstractas:** Representa una colección de datos en las que cada uno de ellos tiene una posición relativa, esto es útil en el manejo de datos de forma ágil y sencilla (Torres, 2023).
2. **Diagrama de clases sencilla:** Representación gráfica que ilustra cómo los elementos están organizados y conectados en una estructura de datos de lista enlazada, es una herramienta visual que representa gráficamente la organización y conexiones entre los elementos de una lista enlazada (Torres, 2023).
3. **Clase interna de enlace:** Es una clase que se define dentro de otra clase en el lenguaje de programación Java, También es conocida como clase anidada que se usa para encapsular y representar alguna funcionalidad relacionada con la clase externa Es una clase que se define dentro de otra clase en el lenguaje de programación Java., También es conocida como clase anidada que se usa para encapsular y representar alguna funcionalidad relacionada con la clase externa Es una clase que se define dentro de otra clase en el lenguaje de programación Java., también es conocida como clase anidada que se usa para encapsular y representar alguna funcionalidad relacionada con la clase externa (Torres, 2023).
4. **Lista Genérica:** Una lista genérica es una estructura de datos que puede almacenar elementos de cualquier tipo, las listas genéricas se implementan utilizando la interfaz List de la biblioteca estándar (java.util.List). Esta interfaz permite definir listas que pueden contener elementos de cualquier tipo (clases, tipos primitivos, etc.). El uso de estas listas

son útiles porque nos permite la reutilización del código, flexibilidad y seguridad en tipos de datos (Torres, 2023).

5. **Lista Tipada:** Es una estructura de datos que permite almacenar elementos de un tipo de dato específico. Ellos quieren decir que todos los datos almacenados en ella deben ser de un mismo tipo, el término "tipado" hace referencia a un tipo de dato específico para una variable (Torres, 2023).
6. **Lista Posición Ordinal:** Es una estructura de datos en la cual cada elemento tiene una posición numérica o refleja su lugar dentro de la secuencia. En pocas palabras estos datos tienen su posición dentro de la lista, este tipo de lista es muy útil en los casos en los cuales se necesita mostrar la posición de uno o más elementos. (Torres, 2023).
7. **Métodos de Iteración:** Son aquellos que nos permiten navegar por los datos de una lista, conjuntos, mapas, entre otras estructuras de datos similares, estos métodos son muy útiles en la programación, ya que permite realizar búsquedas, filtrado, transformación o cualquier otro tipo de operación en el cual se tenga que interactuar con los elementos de la colección (Torres, 2023)
8. **PILA (Push / Pop):** Es una estructura que tiene como base el "último en entrar es el primero en salir", esto quiere decir que el último dato ingresado es el primero en ser retirado. Push para ingresar y Pop para retirar son sus funciones básicas.

IX. DEFINICIONES DE PROGRAMACIÓN

1. **JTable:** Es una clase o componente visual que mediante el uso de filas y columnas permite mostrar información al usuario de una manera organizada y de fácil lectura, esta información puede ser ingresada directamente en el código u obtenida de una base de datos.
2. **Jlist:** Es una lista que muestra una serie de elementos en la que el usuario puede seleccionar una o más, la clase Jlist soporta listas de selección única y listas de selección múltiple.
3. **JProgressBar:** Es un componente que permite mostrar mediante una barra el progreso o avance en la ejecución de un programa o aplicativo.
4. **JTree:** Es uno de los componentes más complejos o de difícil entendimiento, muestra los datos de una manera jerárquica en la cual se van desplegando diversas ramas o nodos de acuerdo con nuestras necesidades.
5. **Jpanel:** Es quizás uno de los componentes más usados ya que es el contenedor donde podemos ubicar diferentes elementos como lo pueden ser botones, etiquetas, campos de texto entre otros, facilitándonos de este modo la ubicación de dichos elementos.
6. **JComboBox:** Es un componente compuesto que muestra una lista desplegable de opciones que

permanecen oculta hasta que se hace clic sobre ella y en la cual se puede elegir una de ellas y generando una acción previamente definida.

7. **JButton:** Es el componente que permite la creación de botones, que mediante la asignación de acciones puede ejecutar algún tipo de instrucción podemos encontrar diferentes tipos de botones como botones de comando, casillas de verificación botones de opciones o botones interruptores entre otros.
8. **JFrame:** Es una clase que permite la creación de ventanas en la que se pueden adicionar diferentes objetos con los que los usuarios pueden o no interactuar.
9. **JLabel:** Es un componente que permite mostrar un texto y/o una imagen, con los JLabel se generan las etiquetas que en pocas palabras lo que genera es una impresión o instrucción en pantalla para que el usuario la lea.

X. DISEÑO DE INTERFAZ GRAFICA

Proyecto Liga Betplay 2023

****Bienvenido a FemData Pro****

Usuario

Contraseña

Iniciar Sesión

Creado por:
David Sebastian Enciso Lopez
david.enciso@pi.edu.co

Figura 12. Diseño Interfaz.1.Creación propia

Proyecto Liga Betplay 2023

Menu Principal

Elige una Opción

1. Mostrar equipos campeones

2. Mostrar jugadoras por equipo

3. mostrar estadísticas torneo actual

Salir

Figura 13. Diseño Interfaz.2.Creación propia

Proyecto Liga Betplay 2023

Equipos campeones

2023 Independiente SantaFe
2022 América de Cali
2021 Deportivo Cali
2020 Independiente SantaFe
2019 América de Cali
2018 Atletico Huila
2017 Independiente SantaFe

Volver

Figura 14 Diseño Interfaz.3.Creación propia

Proyecto Liga Betplay 2023

¿Qué Equipo deseas consultar?

1. Nacional
2. América
3. SantaFe
4. Pereira
5. Cali

Volver

Figura 15. Diseño Interfaz.4.Creación propia

Proyecto Liga Betplay 2023

Plantilla de Nacional 2023

Nombre	Apellido	Edad	Dorsal	Posición	Nacionalidad
Vanesa	Cordoba	28	1	Arquera	Colombia
Angela	Barón	19	14	Defensa	Colombia
Yeirdan	Padilla	19	15	Defensa	Colombia
...

Volver

Figura 16. Diseño Interfaz.5.Creación propia

Proyecto Liga Betplay 2023

Tabla de posiciones

POS	EQUIPO	PJ	PG	PE	PP	GF	GC	DG	PUNTOS
1	America	16	13	1	2	43	8	35	40
2	SantaFe	16	10	5	1	33	12	21	35
3	Nacional	16	8	6	2	28	9	19	30
...

Volver

Figura 17. Diseño Interfaz.6.Creación propia

XI. EXPLICACIÓN DE CÓDIGO

1. Se importan librerías necesarias para el correcto funcionamiento del aplicativo.

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
import javax.swing.border.EmptyBorder;
import java.util.ArrayList;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableColumn;
import java.util.ArrayList;
```

2. Clase Login es la que contiene el metodo main y es acá donde arranca el aplicativo en el se generan las constantes para el usuario y contraseña con las que va a trabajar el programa.

```
public class Login {

    private final static String USERNAME = "David";
    private final static String PASSWORD = "Clave123";

    3. Dentro del método main se crea la interfaz para
    autenticar el usuario.
    public static void main(String[] args) {
        // Crear una instancia de JFrame
        JFrame frame = new JFrame("Proyecto Liga Betplay
        Femenina 2023");
        // Establecer el tamaño del JFrame
        frame.setSize(700, 650);
        // Establecer el color de fondo del JFrame (fucsia )
        frame.getContentPane().setBackground(new Color(253,
        121, 255));
        // Centrar la ventana en la pantalla
```

```
frame.setLocationRelativeTo(null);
frame.setLayout(new FlowLayout());
```

```
// Establecer la operación por defecto al cerrar la ventana
```

```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
// Crear un panel
```

```
JPanel panel = new JPanel();
```

```
// Establecer el diseño del panel como GridLayout
```

```
panel.setBorder(new EmptyBorder(30, 30, 30, 30));
```

```
panel.setLayout(new GridLayout(12, 1, 10, 10)); // 10
filas, 1 columna, espacio horizontal y vertical
```

```
// Crear etiquetas
```

```
JLabel etiquetaTitulo = new JLabel("**** Bienvenido a
Futbol Rosa Stacs ****");
```

```
JLabel etiquetaInstrucciones = new JLabel("Para iniciar
ingresa tu nombre de usuario y contraseña");
```

```
JLabel etiquetaLoginIncorrecto = new JLabel("Datos
incorrectos");
```

```
JLabel etiquetaVacía = new JLabel("-----
-----");
```

```
JLabel etiquetaVacía2 = new JLabel("-----
-----");
```

```
JLabel etiquetaVacía3 = new JLabel("-----
-----");
```

```
JLabel etiquetaCreadoPor = new JLabel("Creado por:");
```

```
JLabel etiquetaAutor = new JLabel("David Sebastián
Enciso Lopez");
```

```
JLabel etiquetaCorreo = new
```

```
JLabel("david.enciso@pi.edu.co");
```

```
JTextField textFieldNombre = new JTextField();
```

```
JPasswordField textFieldPassword = new
```

```
JPasswordField();
```

4. Se da estilo y organización a las etiquetas dentro del panel.

```
//dar estilo a las etiquetas
```

```
etiquetaTitulo.setFont(new Font("Comic Sans MS",
Font.BOLD, 20));
```

```
etiquetaInstrucciones.setFont(new Font("Comic Sans
MS", Font.BOLD, 14));
```

```
etiquetaLoginIncorrecto.setFont(new Font("Comic Sans
MS", Font.BOLD, 14));
```

```
etiquetaCreadoPor.setFont(new Font("Comic Sans MS",
Font.BOLD, 14));
```

```
etiquetaAutor.setFont(new Font("Comic Sans MS",
Font.BOLD, 14));
```

```
etiquetaCorreo.setFont(new Font("Comic Sans MS",
Font.BOLD, 14));
```

```
// Establecer alineación de las etiquetas al centro
```

```
etiquetaTitulo.setHorizontalAlignment(SwingConstants.CEN
TER);
```

```

    etiquetaInstrucciones.setHorizontalAlignment(SwingConstants.CENTER);

    etiquetaLoginIncorrecto.setHorizontalAlignment(SwingConstants.CENTER);

    etiquetaVacia.setHorizontalAlignment(SwingConstants.CENTER);

    etiquetaVacia2.setHorizontalAlignment(SwingConstants.CENTER);

    etiquetaVacia3.setHorizontalAlignment(SwingConstants.CENTER);

    etiquetaCreadoPor.setHorizontalAlignment(SwingConstants.CENTER);

    etiquetaAutor.setHorizontalAlignment(SwingConstants.CENTER);

    etiquetaCorreo.setHorizontalAlignment(SwingConstants.CENTER);

```

```

// Agregar las etiquetas al panel
panel.add(etiquetaTitulo);
panel.add(etiquetaInstrucciones);
panel.add(textFieldNombre);
panel.add(textFieldPassword);

```

5. Se crea el botón para validar el inicio de sesión.

```

// Crear un botón
JButton boton = new JButton("Iniciar sesión");
boton.setFont(new Font("Comic Sans MS", Font.BOLD, 14));

MenuPrincipal menuPrincipal = new MenuPrincipal();
// Agregar acción al botón para validar usuario y contraseña
boton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String usuario = textFieldNombre.getText();
        String contrasena = textFieldPassword.getText();

        AdministradorOpciones model = new AdministradorOpciones();
        boolean esLoginCorrecto =
            model.validarDatosUsuario(usuario, contrasena,
                USERNAME, PASSWORD);
        if (esLoginCorrecto) {
            frame.setVisible(false);
            menuPrincipal.mostrarMenuPrincipal();
        } else {
            panel.add(etiquetaLoginIncorrecto);
            panel.revalidate();
            panel.repaint();
        }
    }
});

```

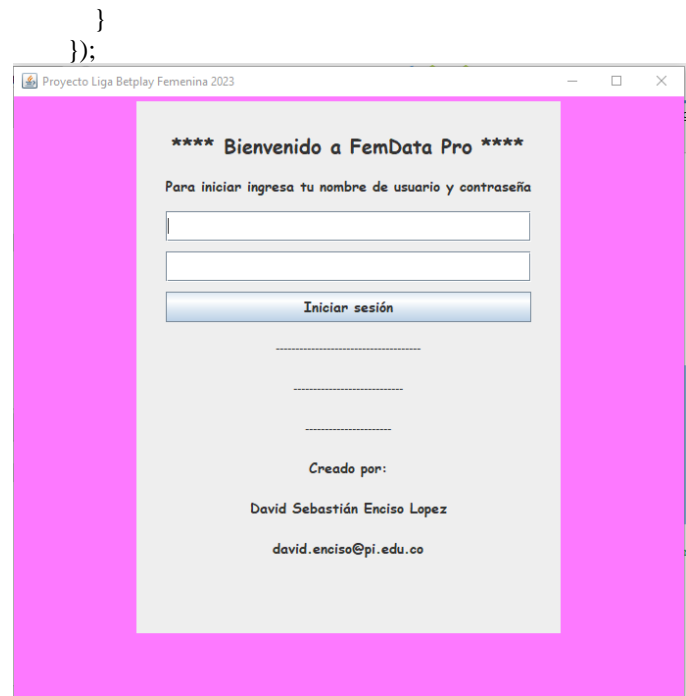


Figura 18. Pantalla de Login.Creación propia

6. Después de que la validación es correcta envía al menú principal.

```

public class MenuPrincipal {

    private JFrame framePrincipal;

    public MenuPrincipal() {
        framePrincipal = new JFrame("Proyecto Liga Betplay Femenina 2023");
        framePrincipal.setSize(600, 500);
        framePrincipal.setLocationRelativeTo(null);
        framePrincipal.getContentPane().setBackground(new Color(100, 150, 80));
        framePrincipal.setLayout(new FlowLayout());

        framePrincipal.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public void mostrarMenuPrincipal() {

        JPanel panel = new JPanel();
        panel.setBorder(new EmptyBorder(50, 50, 50, 50));
        panel.setLayout(new GridLayout(7, 1, 10, 10)); // 3
        // Crear etiqueta
        JLabel etiquetaTitulo = new JLabel("*** Menú principal ***");

        etiquetaTitulo.setHorizontalAlignment(SwingConstants.CENTER);

        panel.add(etiquetaTitulo);
    }
}

```

```
// Agrega etiqueta vacía para crear espacio entre el
titulo y los botones
panel.add(new JLabel(""));
```

7. Se crea la clase del menú principal y se hace el llamado correspondiente a cada clase y/o método de las funciones del aplicativo.

```
public class MenuPrincipal {

    private JFrame framePrincipal;

    public MenuPrincipal() {
        framePrincipal = new JFrame("Proyecto Liga Betplay
Femenina 2023");
        framePrincipal.setSize(600, 500);
        framePrincipal.setLocationRelativeTo(null);
        framePrincipal.getContentPane().setBackground(new
Color(100, 150, 80));
        framePrincipal.setLayout(new FlowLayout());

        framePrincipal.setDefaultCloseOperation(JFrame.EXIT_ON_
CLOSE);
    }

    public void mostrarMenuPrincipal() {

        JPanel panel = new JPanel();
        panel.setBorder(new EmptyBorder(50, 50, 50, 50));
        panel.setLayout(new GridLayout(7, 1, 10, 10)); // 3
filas, 1 columna, espacio horizontal y vertical
        // Crear etiqueta
        JLabel etiquetaTitulo = new JLabel("**** Menú
principal ****");

        etiquetaTitulo.setHorizontalAlignment(SwingConstants.CEN
TER);
        panel.add(etiquetaTitulo);
        // Agrega etiqueta vacía para crear espacio entre el
titulo y los botones
        panel.add(new JLabel(""));

        //Crear Botones en el menu principal
        JButton botonOpcion1 = new JButton("1. Mostrar
equipos campeones");
        JButton botonOpcion2 = new JButton("2. Mostrar
jugadoras por equipo");
        JButton botonOpcion3 = new JButton("3. Mostrar
estadísticas del torneo actual");
        JButton botonOpcion4 = new JButton("Salir");

        //crear acción del boton para mostrar equipos
campeones
        botonOpcion1.addActionListener(new
ActionListener() {
```

```
@Override
        public void actionPerformed(ActionEvent e) {
            mostrarEquiposCampeones();
        }
    });
    //crear acción del boton para mostrar jugadoras por
equipo
    botonOpcion2.addActionListener(new
ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            mostrarOpcionesJugadorasPorEquipo();
        }
    });
    //crear acción del boton para mostrar estadísticas y
tabla
    botonOpcion3.addActionListener(new
ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            mostrarEstadísticasLiga();
        }
    });
    //Crear boton para salir de la aplicación
    botonOpcion4.addActionListener(new
ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            System.exit(0);
        }
    });

    //Agregar botones al panel
    panel.add(botonOpcion1);
    panel.add(botonOpcion2);
    panel.add(botonOpcion3);
    panel.add(botonOpcion4);

    // Agregar el panel al JFrame
    framePrincipal.getContentPane().removeAll();
    framePrincipal.add(panel);
    framePrincipal.setVisible(true);
}
```




Figura 19. Pantalla menú principal.Creación propia

```
private void mostrarOpcionesJugadorasPorEquipo() {

    //Crear panel de mostrar equipos
    JPanel panel = new JPanel();
    panel.setBorder(new EmptyBorder(100, 100, 100,
100));
    panel.setLayout(new GridLayout(7, 1, 10, 10));
    JLabel etiquetaTitulo = new JLabel("¿Que equipo
deseas consultar?");

    etiquetaTitulo.setHorizontalAlignment(SwingConstants.CEN
TER);
    panel.add(etiquetaTitulo);

    //Crear botones para mostrar opciones disponibles
    JButton botonOpcion1 = new JButton("1. Nacional");
    JButton botonOpcion2 = new JButton("2. America");
    JButton botonOpcion3 = new JButton("3. SantaFe");
    JButton botonOpcion4 = new JButton("4. Pereira");
    JButton botonOpcion5 = new JButton("5. Cali");
    JButton botonOpcion6 = new JButton("6. Volver");

    //crear acción del boton
    botonOpcion1.addActionListener(new
ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            mostrarJugadorasPorEquipo(1);
        }
    });

    //crear acción del boton
    botonOpcion2.addActionListener(new
ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            mostrarJugadorasPorEquipo(2);
        }
    });
}
```

```
//crear acción del boton
botonOpcion3.addActionListener(new
ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        mostrarJugadorasPorEquipo(3);
    }
});

//crear acción del boton
botonOpcion4.addActionListener(new
ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        mostrarJugadorasPorEquipo(4);
    }
});

//crear acción del boton
botonOpcion5.addActionListener(new
ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        limpiarJFrame();
        mostrarMenuPrincipal();
    }
});

// Agregar el botón al panel
panel.add(botonOpcion1);
panel.add(botonOpcion2);
panel.add(botonOpcion3);
panel.add(botonOpcion4);
panel.add(botonOpcion5);
panel.add(botonOpcion6);

// Agregar el panel al JFrame
framePrincipal.getContentPane().removeAll();
framePrincipal.add(panel);
framePrincipal.revalidate();
framePrincipal.repaint();
framePrincipal.setVisible(true);
}
```



Figura 20. Menú plantilla jugadoras.Creación propia

NOMBRE	APELLIDO	EDAD	DORSAL	POSICIÓN	NACIONALIDAD
Vanessa	Cordoba	28	1	Arquera	Colombia
Angela	Barón	19	14	Defensa	Colombia
Yeirdan	Padilla	19	15	Defensa	Colombia
Mary	Alvarez	18	2	Defensa	Colombia
Nairelis	Gutierrez	28	18	Defensa	Venezuela
Daniela	Montoya	32	6	Volante	Colombia
Joselyn	Espinales	24	7	Volante	Ecuador
Geraldine	Cardona	29	8	Volante	Colombia
Katerine	Osorio	18	14	Volante	Colombia
Yirleidis	Quejada	20	19	Delantera	Colombia
Yesica	Muñoz	18	10	Delantera	Colombia
Jorge	Barreneche	47	0	D.T	Colombia

Figura 21.Plantilla jugadoras Nacional.Creación propia

```
private void mostrarEstadisticasLiga() {
    // se crea el titulo de la pestaña
    JLabel titulo = new JLabel("¿Que equipo deseas consultar?");
    titulo.setHorizontalAlignment(SwingConstants.CENTER);

    String[] options = {"America de Cali", "SantaFe", "Nacional", "Pereira", "Deportivo Cali", "Tabla de Posiciones"};

    // Crear una instancia de JComboBox y pasar el arreglo de opciones
    JComboBox<String> comboBox = new JComboBox<>(options);

    // Crear un botón para obtener la selección
```

```
JButton obtenerSeleccion = new JButton("Selecciona");

// Crear Boton para regresar al menu principal
JButton botonRegresarMenu = new JButton("Volver");
botonRegresarMenu.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        limpiarJFrame();
        mostrarMenuPrincipal();
    }
});

// Crear un panel para organizar los componentes
JPanel panel = new JPanel();
panel.setBorder(new EmptyBorder(100, 100, 100, 100));
panel.setLayout(new GridLayout(5, 1, 10, 10));
panel.add(titulo);
panel.add(comboBox);
panel.add(obtenerSeleccion);
panel.add(botonRegresarMenu);

obtenerSeleccion.addActionListener(e -> {
    int idEquipoSeleccionado = comboBox.getSelectedIndex();

    mostrarEstadisticasPorEquipoSeleccionado(idEquipoSeleccionado);
});

// Agregar el panel al JFrame
framePrincipal.getContentPane().removeAll();
framePrincipal.add(panel);
framePrincipal.revalidate();
framePrincipal.repaint();
framePrincipal.setVisible(true);
}

private void mostrarEstadisticasPorEquipoSeleccionado(int idEquipoSeleccionado) {
    AdministradorDeDatos obtenerEstadisticasDelTorneo = new AdministradorDeDatos();
    ArrayList<Estadistica> listaDeEstadisticas = obtenerEstadisticasDelTorneo.obtenerEstadisticasDelTorneo(idEquipoSeleccionado);

    // Crear un modelo de datos de tabla
    DefaultTableModel tableModel = new DefaultTableModel();

    // Crear una instancia de JTable con el modelo de datos
    JTable table = new JTable(tableModel);

    // Agregar la tabla a un JScrollPane
```

```

JScrollPane scrollPane = new JScrollPane(table);

JButton botonVolver = new JButton("Volver");
botonVolver.addActionListener(new ActionListener()
{
    @Override
    public void actionPerformed(ActionEvent e) {
        limpiarJFrame();
        mostrarEstadisticasLiga();
    }
});

// Agregar columnas al modelo de datos
tableModel.addColumn("POS.");
tableModel.addColumn("EQUIPO");
tableModel.addColumn("PJ");
tableModel.addColumn("PG");
tableModel.addColumn("PE");
tableModel.addColumn("PP");
tableModel.addColumn("GF");
tableModel.addColumn("GC");
tableModel.addColumn("GD");
tableModel.addColumn("PTOS");

for (int i = 0; i < listaDeEstadisticas.size(); i++) {
    Estadistica estadisticasEquipo =
listaDeEstadisticas.get(i);
    tableModel.addRow(new Object[]{
        estadisticasEquipo.getPosicion(),
        estadisticasEquipo.getEquipo(),
        estadisticasEquipo.getPartidosJugados(),
        estadisticasEquipo.getPartidosGanados(),
        estadisticasEquipo.getPartidosEmpatados(),
        estadisticasEquipo.getPartidosPerdidos(),
        estadisticasEquipo.getGolesAFavor(),
        estadisticasEquipo.getGolesEnContra(),
        estadisticasEquipo.getDiferenciaDeGoles(),
        estadisticasEquipo.getPuntos()
    });
}

// Cambiar el tamaño de la primer columna(Posición)
JTableColumn columnPosicion =
table.getColumnModel().getColumn(0);
columnPosicion.setPreferredWidth(75); // Establecer el
ancho deseado

// Cambiar el tamaño de la segunda columna (Equipo)
JTableColumn columnEquipo =
table.getColumnModel().getColumn(1);
columnEquipo.setPreferredWidth(200); // Establecer el
ancho deseado

JLabel etiquetaTabla = new JLabel ("Tabla de
Posiciones");

```

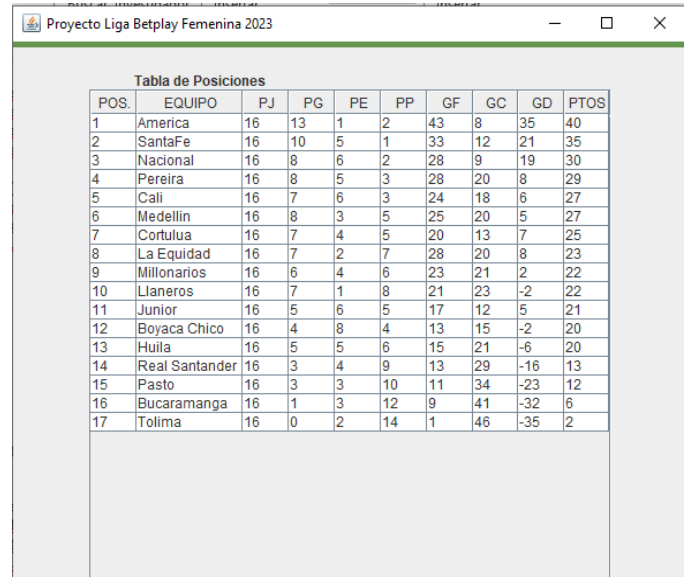
```

JPanel container = new JPanel();
container.setBorder(new EmptyBorder(20, 100, 20,
100));
container.setLayout(new BorderLayout(container,
BoxLayout.Y_AXIS));

container.add(etiquetaTabla);
container.add(scrollPane);
container.add(botonVolver);

// Agregar el panel al JFrame
framePrincipal.getContentPane().removeAll();
//framePrincipal.add(etiquetaTabla);
framePrincipal.add(container);
framePrincipal.revalidate();
framePrincipal.repaint();
framePrincipal.setVisible(true);
}

```



POS.	EQUIPO	PJ	PG	PE	PP	GF	GC	GD	PTOS
1	America	16	13	1	2	43	8	35	40
2	SantaFe	16	10	5	1	33	12	21	35
3	Nacional	16	8	6	2	28	9	19	30
4	Pereira	16	8	5	3	28	20	8	29
5	Cali	16	7	6	3	24	18	6	27
6	Medellin	16	8	3	5	25	20	5	27
7	Cortulua	16	7	4	5	20	13	7	25
8	La Equidad	16	7	2	7	28	20	8	23
9	Millonarios	16	6	4	6	23	21	2	22
10	Llaneros	16	7	1	8	21	23	-2	22
11	Junior	16	5	6	5	17	12	5	21
12	Boyaca Chico	16	4	8	4	13	15	-2	20
13	Huila	16	5	5	6	15	21	-6	20
14	Real Santander	16	3	4	9	13	29	-16	13
15	Pasto	16	3	3	10	11	34	-23	12
16	Bucaramanga	16	1	3	12	9	41	-32	6
17	Tolima	16	0	2	14	1	46	-35	2

Figura 22. Tabla de posiciones.Creación propia

```

private void mostrarEquiposCampeones() {
    JPanel panel = new JPanel();
    panel.setBorder(new EmptyBorder(100, 100, 100,
100));
    panel.setLayout(new BorderLayout(panel,
BoxLayout.Y_AXIS));

    JLabel etiquetaTitulo = new JLabel("*** Equipos
Campeones***");

    etiquetaTitulo.setHorizontalAlignment(SwingConstants.CEN
TER);

    panel.add(etiquetaTitulo);
    // Agrega etiqueta vacía para crear espacio entre el
titulo y los botones
    panel.add(new JLabel(" "));
}

```

```

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

AdministradorCampeones  campeon  =  new
AdministradorCampeones();
String  equiposCampeones  =
campeon.obtenerEquiposCampeones();

JLabel etiqueta1 = new JLabel(equiposCampeones);
etiqueta1.setHorizontalAlignment(SwingConstants.CENTER);
panel.add(etiqueta1);

//Crear boton para regresar al menu principal
JButton botonSalir = new JButton("Volver");
botonSalir.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        limpiarJFrame();
        mostrarMenuPrincipal();
    }
});
panel.add(new JLabel(""));
panel.add(new JLabel(""));
panel.add(botonSalir);

framePrincipal.getContentPane().removeAll();
framePrincipal.add(panel);
framePrincipal.revalidate();
framePrincipal.repaint();
framePrincipal.setVisible(true);
}

private void mostrarJugadorasPorEquipo(int
identificadorEquipo) {

    AdministradorJugadoras  administradorJugadoras  =
new AdministradorJugadoras();
    ArrayList<Jugadora> jugadoras = new ArrayList<>();

    if (identificadorEquipo == 1) {
        jugadoras
administradorJugadoras.obtenerJugadorasNacional();
    } else if (identificadorEquipo == 2) {
        jugadoras
administradorJugadoras.obtenerJugadorasAmerica();
    } else if (identificadorEquipo == 3) {
        jugadoras
administradorJugadoras.obtenerJugadorasSantaFe();
    } else if (identificadorEquipo == 4) {
        jugadoras
administradorJugadoras.obtenerJugadorasPereira();
    } else if (identificadorEquipo == 5) {
        jugadoras
administradorJugadoras.obtenerJugadorasCali();
    }
}

JPanel panel = new JPanel();

// Crear un modelo de datos de tabla
DefaultTableModel  tableModel  =  new
DefaultTableModel();

// Crear una instancia de JTable con el modelo de datos
JTable table = new JTable(tableModel);

// Agregar la tabla a un JScrollPane
JScrollPane scrollPane = new JScrollPane(table);

// Agregar columnas al modelo de datos
tableModel.addColumn("NOMBRE");
tableModel.addColumn("APELLIDO");
tableModel.addColumn("EDAD");
tableModel.addColumn("DORSAL");
tableModel.addColumn("POSICIÓN");
tableModel.addColumn("NACIONALIDAD");

// Pintar las filas con los datos en la tabla
for (Jugadora jugadora : jugadoras) {
    tableModel.addRow(new Object[]{
        jugadora.getNombre(),
        jugadora.getApellido(),
        jugadora.getEdad(),
        jugadora.getDorsal(),
        jugadora.getPosicion(),
        jugadora.getNacionalidad()
    });
}

// Cambiar el tamaño de la primera columna (Nombre)
TableColumn  columnNombre  =
table.getColumnModel().getColumn(0);
columnNombre.setPreferredWidth(100); // Establecer
el ancho deseado

// Cambiar el tamaño de la quinta columna (Apellido)
TableColumn  columnApellido  =
table.getColumnModel().getColumn(1);
columnApellido.setPreferredWidth(90); // Establecer
el ancho deseado

// Cambiar el tamaño de la quinta columna (Posición)
TableColumn  columnPosicion  =
table.getColumnModel().getColumn(4);
columnPosicion.setPreferredWidth(90); // Establecer el
ancho deseado

// Cambiar el tamaño de la sexta columna
(Nacionalidad)
TableColumn  columnNacionalidad  =
table.getColumnModel().getColumn(5);
columnNacionalidad.setPreferredWidth(120); //
Establecer el ancho deseado

JButton botonSalir = new JButton("Volver");

```

```

botonSalir.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        limpiarJFrame();
        mostrarOpcionesJugadorasPorEquipo();
    }
});
panel.add(botonSalir);

```

```

framePrincipal.getContentPane().removeAll();
framePrincipal.add(scrollPane);
framePrincipal.add(panel);
framePrincipal.revalidate();
framePrincipal.repaint();
framePrincipal.setVisible(true);
}

```

```

private void limpiarJFrame(){
    framePrincipal.getContentPane().removeAll();
    framePrincipal.revalidate();
    framePrincipal.repaint();
}
}

```

XII. CONCLUSIONES

1. Con la elaboración de este aplicativo profundice en los temas de programación orientada a objetos y sus 4 pilares quedándome mucho mas clara sus definiciones y aplicaciones
2. Logre comprender de mejor manera la arquitectura de modelo vista controlador y aplicarlo en el proyecto entendiendo sus definiciones sus ventajas en el mantenimiento y/o actualización del programa.
3. La programación orientada a objetos es la base en este largo camino que apenas inicio y con este trabajo aprendí a manejar tiempos y distribución del trabajo.
4. El conocimiento adquirido fue bastante enriquecedor y satisfactorio teniendo en cuenta que JAVA es uno de los lenguajes más populares en el mundo.
5. Debo continuar ampliando el conocimiento ya que es un lenguaje bastante complejo y extenso lo que nos invita a continuar en constante uso de este para mejorar la práctica.

XIII. COMPILACION DE CODIGO EN .JAR DESDE CMD, .EXE Y .MSI

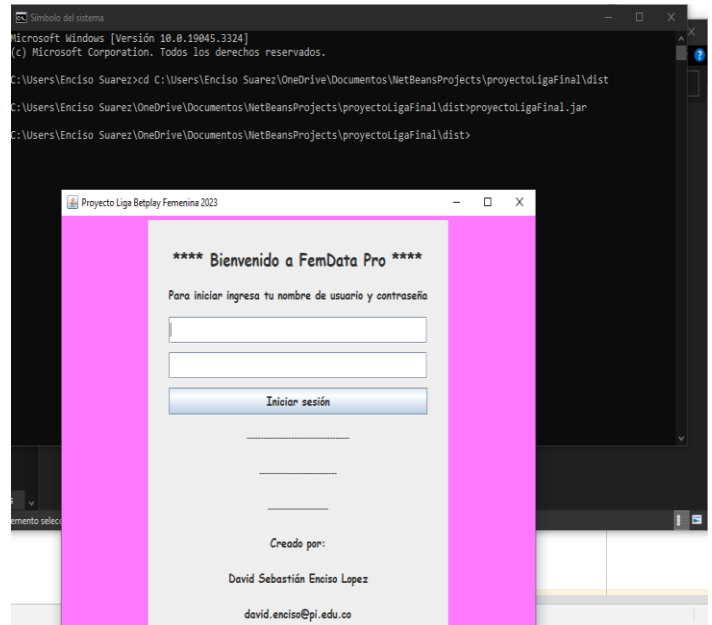


Figura 23. Ejecución .Jar desde CMD.Creación propia



Figura 24. Archivo.EXE.Creación propia



Figura 25. Instalador MSI.Creación propia

XIV. REFERENCIAS

Figura ModeloVistaCcontrolador: *Qué es MVC.* (2014, enero 2). Desarrolloweb.com. <https://desarrolloweb.com/articulos/que-es-mvc.html>

Tabla de posiciones. (n. d.). 365Scores. Consultado en septiembre 09, 2023, de <https://www.365scores.com/es/football/colombia/liga-aguila-women/league/7041/standings>

Plantilla Nacional 2023. (n. d.). 365Scores. Consultado en septiembre 09, 2023, de [https://www.365scores.com/es-mx/football/team/atl.-nacional-\(w\)-51588/squad](https://www.365scores.com/es-mx/football/team/atl.-nacional-(w)-51588/squad)

Plantilla Santa Fe 2023. (n. d.). 365Scores. Consultado en septiembre 09, 2023, de [https://www.365scores.com/es-mx/football/team/santa-fe-\(w\)-51597/squad](https://www.365scores.com/es-mx/football/team/santa-fe-(w)-51597/squad)

Plantilla América 2023. (n. d.). 365Scores. Consultado en septiembre 09, 2023, de [https://www.365scores.com/es-mx/football/team/america-de-cali-\(w\)-51600/squad](https://www.365scores.com/es-mx/football/team/america-de-cali-(w)-51600/squad)

Plantilla Pereira 2023. (n. d.). 365Scores. Consultado en septiembre 09, 2023, de [https://www.365scores.com/es-mx/football/team/deportivo-pereira-\(w\)-51587/squad](https://www.365scores.com/es-mx/football/team/deportivo-pereira-(w)-51587/squad)

Plantilla Cali 2023. (n. d.). 365Scores. Consultado en septiembre 09, 2023, de [https://www.365scores.com/es-mx/football/team/deportivo-cali-\(w\)-56751/squad](https://www.365scores.com/es-mx/football/team/deportivo-cali-(w)-56751/squad)

[Listado de imagenes](#)

Figura 1.Requerimientos Funcionales.Creación propia

Figura 2. Herencia.Creación propia

Figura 3. Polimorfismo.Creación propia

Figura 4. Encapsulamiento.Creación propia

Figura 5. Abstraccion.Creación propia

Figura 6. Diccionario de datos.Creación propia

Figura 7. Diagrama de flujo.Creación propia

Figura 8. Diagrama de clases (creacion propia)

Figura 9. ModeloVistaControlador

Figura 10. MVC Proyecto.Creación propia

Figura 11. Diagrama de listas sencillas. Creación propia

Figura 12. Diseño Interfaz.1.Creación propia

Figura 13. Diseño Interfaz.2.Creación propia

Figura 14 Diseño Interfaz.3.Creación propia

Figura 15. Diseño Interfaz.4.Creación propia

Figura 16. Diseño Interfaz.5.Creación propia

Figura 17. Diseño Interfaz.6.Creación propia

Figura 18. Pantalla de Login.Creación propia

Figura 19. Pantalla menú principal.Creación propia

Figura 20. Menú plantilla jugadoras.Creación propia

Figura 21.Plantilla jugadoras Nacional.Creación propia

Figura 22. Tabla de posiciones.Creación propia

Figura 23. Ejecución .Jar desde CMD.Creación propia

Figura 24. Archivo.EXE.Creación propia

Figura 25. Instalador MSI.Creación propia



David Sebastian Enciso Lopez

Desarrollador junior, apasionado por el futbol y estudiante del politécnico internacional, dentro de mis proyectos realizados a la fecha se pueden destacar la elaboración de código para el funcionamiento de un cajero electrónico mediante consola, programa de consulta para la Liga Betplay femenina con interfaz gráfica bajo el modelo-vista-controlador, este último es un maravillo y novedoso proyecto para los amantes del futbol femenino colombiano.