HISTORICO LIGA BETPLAY FEMENINO

De Los Rios Rodriguez, Cristian Camilo
Correo: cristian.de.los.rios@pi.edu.co
Programación de Software y aplicativos móviles, Politécnico Internacional
Autopista Sur No. 67 - 71

Resumen - "Histórico BetPlay femenino" es una aplicación para los fanáticos del futbol profesional colombiano femenino y los que quieran conocer más sobre el mismo. La persona contará con un usuario y contraseña para el acceso con la que podrá acceder y seleccionar diferentes años entre el 2000 y el 2023, los cuales mostrarán en pantalla detalles de las finales de esos años seleccionados, así como el nombre de la goleadora y cantidad de goles anotados por la misma en ese año. Para esto usaremos Netbeans y modelo vista controlador (MVC).

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

Abstract – "Historico BetPlay Femenino" is an application for fans of Colombian professional women's soccer and those who want to know more about it. The person will have a username and password for access with which they can access and select different years between 2000 and 2023, which will show details of the finals of those selected years on the screen, as well as the name of the scorer and amount of goals scored by it in that year. For this we will use Netbeans and a Model View Controller (MVC).

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

I. INTRODUCCIÓN

Este proyecto se realizará mediante el aplicativo Apache Netbens IDE 17, mediante arquitectura de modelo vista controlador (MVC), enfocando sus procesos en "Histórico BetPlay femenino", que permitirá a sus usuarios tener la visualización del histórico de la Liga BetPlay - Futbol Femenino Colombiano durante los últimos 4 años, en cuanto a finales jugadas y estadísticas de las mismas.
"Histórico BetPlay femenino" contará con un entorno gráfico que permitirá seleccionar diferentes opciones y funciones asociadas al año que el usuario seleccione como ejemplo:

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

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

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

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

Se podrá acceder fácilmente al histórico de finales de cada año seleccionado y retornar para elegir cualquier otro año sin problemas, de esta forma podrán recopilar ágilmente los datos del año que deseen, con un entorno gráfico amigable para cualquier usuario.

II. CÓDIGO DE HISTORICO BETPLAY FEMENINO.

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

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

```
import java.util.Scanner;
```

2**. Se crea el Main que contiene el menú principal, donde se crea un Switch para ese propósito, además de llamar el método "mostrarInfoAnio", para mostrar toda la información de los años.**

```
public class BetPlayLigaFemenina { // EL main imprime el menú principal.
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in); // Se crea variable para scanner y que el usuario ingrece opción por teclado.
      int opcion; //Opción se declara como Int, que sea leido un número. Switch

      do {
         // Mmenú para que el usuario seleccione el año
         System.out.println("***Bienvenido al histórico de BetPlay femenino***\n");
         System.out.println("Seleccione el año de su interés:");
         System.out.println("1. 2023");
         System.out.println("2. 2022");
         System.out.println("3. 2021");
         System.out.println("4. 2020");
         System.out.println("5. Salir");
         opcion = scanner.nextInt();

         System.out.println();

         // Se abre el menú en el Switch
         switch (opcion) {
            case 1:
               mostrarInfoAnio(new Anio2023()); // cada case llama un método. Para este caso el método "mostrar info" declarado en cada Anio.
               break;
            case 2:
               mostrarInfoAnio(new Anio2022());
               break;
            case 3:
               mostrarInfoAnio(new Anio2021());
               break;
            case 4:
               mostrarInfoAnio(new Anio2020());
               break;
            case 5:
               System.out.println("Saliendo...");
               break;
            default:
               System.out.println("Opción inválida."); // Si no eligen una opción del 1 al 5 del menú, dirá que la opción no es valida y repetirá el menú.
               break;
         }

         System.out.println(); //Genera un espacio extra para que quede visiblemente mejor.
      } while (opcion != 5); // Se cierra el ciclo al seleccionar 5.
   }

   public static void mostrarInfoAnio(Anio anio) { // declara el método para ser llamado.
      System.out.println(anio.obtenerInfoAnio());
   }
}
```

3. Se crea la clase equipo, que contiene el método para que pueda ser llamado por otras clases y mostrar los datos del nombre del equipo.

```
class Equipo { // Se crea clase equipo que representa cada equipo que participa.
   private String nombre; // Atributo nombre que identifica al equipo.

   public Equipo(String nombre) { // Constructor para crear un equipo con su nombre
      this.nombre = nombre; // Inicializar el nombre con el valor proporcionado
   }

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

// La clase "Equipo" permite encapsular la información relevante de un equipo, como su nombre, para ser utilizada en otras partes del programa, como en la representación de las finales.
```

4. Se crea la clase jugadora, esta contiene 2 métodos, para mostrar el nombre de la jugadora y la cantidad de goles anotados, dependiendo del año seleccionado.

```
class Jugadora {
   private String nombre; // El nombre de la jugadora
   private int goles; // La cantidad de goles que ha marcado la jugadora

   public Jugadora(String nombre, int goles) { // Constructor para crear una jugadora con su nombre y la cantidad de goles
      this.nombre = nombre; // Inicializar el nombre con el valor proporcionado
      this.goles = goles; // Inicializar la cantidad de goles con el valor proporcionado
   }

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

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

5. Se crea la clase final, esta encapsula los resultados de las finales década año y estadísticas de la misma, como campeón, subcampeón, estadio, goleadora y demás. Crea métodos para llamar esas subclases.

```
class Final { // representa un partido final entre dos equipos en el torneo de la Liga Femenina de BetPlay.
    private Equipo equipoLocal;// El equipo local en la final
    private Equipo equipoVisitante;// El equipo visitante en la final
    private int golesLocal;// Los goles del equipo local
    private int golesVisitante; // Los goles del equipo visitante
    private String estadio;// El estadio donde se jugó la final
    private String ganador;// El equipo ganador de la final


    // Constructor para crear una instancia de "Final" con los detalles de la final
    public Final(Equipo equipoLocal, Equipo equipoVisitante, int golesLocal, int golesVisitante, String estadio, String ganador) {
        this.equipoLocal = equipoLocal; // Inicializar el equipo local con el valor proporcionado
        this.equipoVisitante = equipoVisitante;// Inicializar el equipo visitante con el valor proporcionado
        this.golesLocal = golesLocal; // Inicializar los goles del equipo local con el valor proporcionado
        this.golesVisitante = golesVisitante; // Inicializar los goles del equipo visitante con el valor proporcionado
        this.estadio = estadio; // Inicializar el estadio con el valor proporcionado
        this.ganador = ganador; // Inicializar el ganador con el valor proporcionado
    }

    // Método para obtener información detallada sobre la final
    public String obtenerInfo() {
        return "Partido:\n" +
                equipoLocal.getNombre() + " " + golesLocal + " - " +
                golesVisitante + " " + equipoVisitante.getNombre() + "\n" +
                "Estadio: " + estadio + "\n" +
                "Ganador: " + ganador + "\n";
    }
}

//Esta clase permite encapsular la información sobre una final específica,
//incluidos los detalles sobre los equipos participantes, los goles marcados y el ganador.
//Se utiliza en el programa para representar las finales de cada año en el torneo y mostrar detalles sobre los partidos finales.
```

6. Se crea la super clase "Anio" que contendrá subclases especificas para cada año y genera un constructor con los detalles del año elegido.

```
//SUPER CLASE

// Clase abstracta para representar un año específico en el torneo
abstract class Anio {
    // Atributos privados para almacenar los detalles del año
    private int anio; // El año del torneo
    private Equipo equipoCampeon; // El equipo campeón del torneo en ese año
    private Equipo equipoSubcampeon; // El equipo subcampeón del torneo en ese año
    private Final finalPartidoIda; // Los detalles del partido de ida de la final en ese año
    private Final finalPartidoVuelta; // Los detalles del partido de vuelta de la final en ese año
    private Jugadora goleadora; // La jugadora goleadora del torneo en ese año

    // Constructor para crear un objeto de tipo Anio con los detalles proporcionados
    public Anio(int anio, Equipo equipoCampeon, Equipo equipoSubcampeon, Final finalPartidoIda, Final finalPartidoVuelta, Jugadora goleadora) {
        this.anio = anio; // Inicializar el año con el valor proporcionado
        this.equipoCampeon = equipoCampeon; // Inicializar el equipo campeón con el valor proporcionado
        this.equipoSubcampeon = equipoSubcampeon; // Inicializar el equipo subcampeón con el valor proporcionado
        this.finalPartidoIda = finalPartidoIda; // Inicializar los detalles del partido de ida con el valor proporcionado
        this.finalPartidoVuelta = finalPartidoVuelta; // Inicializar los detalles del partido de vuelta con el valor proporcionado
        this.goleadora = goleadora; // Inicializar la jugadora goleadora con el valor proporcionado
    }

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

    // Métodos para acceder a los detalles del año
    public int getAnio() {
        return anio;
    }

    public Equipo getEquipoCampeon() {
        return equipoCampeon;
    }

    public Equipo getEquipoSubcampeon() {
        return equipoSubcampeon;
    }

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

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

    public Jugadora getGoleadora() {
        return goleadora;
    }

    // Sobrescribir el método toString para obtener información del año al convertir el objeto a cadena de texto
    @Override
    public String toString() {
        return obtenerInfoAnio();
    }
}
```

//La clase "Anio" proporciona la estructura base y los métodos necesarios para representar y acceder a los detalles de un año específico en el torneo de la
//Liga Femenina de BetPlay. Las subclases que representan años específicos deben implementar el método obtenerInfoAnio() para ofrecer información detallada
//sobre ese año en particular.

7. Se crean las diferentes subclases Anio2020, Anio2021, Anio2022 y Anio2023, que continene la información detallada de las estadísticas de cada año. Estas subclases al ser llamadas, mostrarán la información detallada de las finales de cada año.

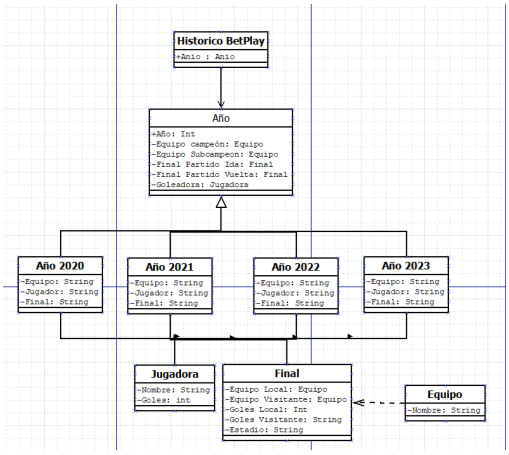
```
// Subclase de Anio que representa el año 2020 en el torneo de la Liga Femenina de BetPlay
class Anio2020 extends Anio {

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

    // Implementación del método abstracto obtenerInfoAnio() para mostrar información detallada del año 2020
    @Override
    public String obtenerInfoAnio() {
        // Crear una cadena con la información detallada del año 2020
        return "Año 2020:\n" +
            "Campeón: " + getEquipoCampeon().getNombre() + "\n" +// Obtener información detallada del campeón
            "Subcampeón: " + getEquipoSubcampeon().getNombre() + "\n" + // Obtener información detallada del subcampeón
            "Goleadora: " + getGoleadora().getNombre() + " (" + getGoleadora().getGoles() + " goles)\n" +// Obtener información detallada del nombre y goles de la goleadora
            "Detalles del partido ida:\n" +
            getFinalPartidoIda().obtenerInfo() + "\n" + // Obtener información detallada del partido de ida
            "Detalles del partido vuelta:\n" +
            getFinalPartidoVuelta().obtenerInfo(); // Obtener información detallada del partido de vuelta
    }
}
```

//La subclase Anio2020 proporciona una implementación específica para representar el año 2020 en el torneo,
//mostrando información detallada sobre ese año, incluyendo los equipos,
//los resultados de los partidos de ida y vuelta de la final y la goleadora.

```
//No se comenta, pues es el mismo proceso que en la clase Anio2020
class Anio2021 extends Anio {
    public Anio2021() {
        super(
            2021,
            new Equipo("Deportivo Cali"),
            new Equipo("Santa Fe"),
            new Final(
                new Equipo("Santa Fe"),
                new Equipo("Deportivo Cali"),
                1, 4,
                "Estadio El Campín, Bogotá",
                "Santa Fe"
            ),
            new Final(
                new Equipo("Deportivo Cali"),
                new Equipo("Santa Fe"),
                2, 2,
                "Estadio Deportivo Cali, Palmira",
                "Deportivo Cali"
            ),
            new Jugadora("Catalina Usme", 12)
        );
    }

    @Override
```

```
    public String obtenerInfoAnio() {
        return "Año 2021:\n" +
                "Campeón: " + getEquipoCampeon().getNombre() +
"\n" +
                "Subcampeón: " +
getEquipoSubcampeon().getNombre() + "\n" +
                "Goleadora: " + getGoleadora().getNombre() + " (" +
getGoleadora().getGoles() + " goles)\n" +
                "Detalles del partido ida:\n" +
                getFinalPartidoIda().obtenerInfo() +
                "Detalles del partido vuelta:\n" +
                getFinalPartidoVuelta().obtenerInfo();
    }
}

//No se comenta, pues es el mismo proceso que en la clase
Anio2020
class Anio2022 extends Anio {
    public Anio2022() {
        super(
                2022,
                new Equipo("América de Cali"),
                new Equipo("Deportivo Cali"),
                new Final(
                        new Equipo("Deportivo Cali"),
                        new Equipo("América de Cali"),
                        2, 1,
                        "Estadio Deportivo Cali, Palmira",
                        "Deportivo Cali"
                ),
                new Final(
                        new Equipo("América de Cali"),
                        new Equipo("Deportivo Cali"),
                        3, 1,
                        "Estadio Pascual Guerrero, Cali",
                        "América de Cali"
                ),
                new Jugadora("Catalina Usme", 15)
        );
    }

    @Override
    public String obtenerInfoAnio() {
        return "Año 2022:\n" +
                "Campeón: " + getEquipoCampeon().getNombre() +
"\n" +
                "Subcampeón: " +
getEquipoSubcampeon().getNombre() + "\n" +
                "Goleadora: " + getGoleadora().getNombre() + " (" +
getGoleadora().getGoles() + " goles)\n" +
                "Detalles del partido ida:\n" +
                getFinalPartidoIda().obtenerInfo() +
                "Detalles del partido vuelta:\n" +
                getFinalPartidoVuelta().obtenerInfo();
    }
}


//No se comenta, pues es el mismo proceso que en la clase
Anio2020
class Anio2023 extends Anio {
    public Anio2023() {
        super(
                2023,
                new Equipo("Santa Fe"),
                new Equipo("América de Cali"),
                new Final(
                        new Equipo("Santa Fe"),
                        new Equipo("América de Cali"),
                        2, 0,
                        "Estadio El Campín, Bogotá",
                        "Santa Fe"
                ),
                new Final(
                        new Equipo("América de Cali"),
                        new Equipo("Santa Fe"),
                        0, 0,
                        "Estadio Pascual Guerrero, Cali",
                        "América de Cali"
                ),
                new Jugadora("Catalina Usme", 11)
        );
    }

    @Override
    public String obtenerInfoAnio() {
        return "Año 2023:\n" +
                "Campeón: " + getEquipoCampeon().getNombre() +
"\n" +
                "Subcampeón: " +
getEquipoSubcampeon().getNombre() + "\n" +
                "Goleadora: " + getGoleadora().getNombre() + " (" +
getGoleadora().getGoles() + " goles)\n" +
                "Detalles del partido ida:\n" +
                getFinalPartidoIda().obtenerInfo() +
                "Detalles del partido vuelta:\n" +
                getFinalPartidoVuelta().obtenerInfo();
    }
}
```

III. DIAGRAMA UML



IV. Diccionario de datos

Diccionario de datos		
Nombre	**Tipo de dato**	**Descripción**
Scanner	Scanner	Para leer el dato ingresado por el usuario
opcion	Int	Entero para las opciones del menú
do	do	Ciclo para el menú
opcion	switch	Crea el menú
opcion	while	Cierra el ciclo del menú
nombre	String	Nombre del equipo
nombre	String	Nombre de la goleadora
goles	Int	Cantidad de goles de la goleadora
golesLocal	Int	Cantidad de goles anotados por el equipo local en la final
golesVisitante	Int	Cantidad de goles anotados por el equipo visitante en la final
estadio	String	Estadio donde se jugó la final
ganador	String	Equipo que ganó la final
anio	Int	Represente el año en el que se jugó la final
Equipo	clase	Representa un equipo participante en la final del torneo.
Jugadora	clase	Representa una jugadora destacada en el torneo, con nombre y cantidad de goles.
Final	clase	Representa los detalles de una final, incluyendo los equipos participantes, los resultados de los partidos de ida y vuelta, el lugar y el equipo ganador.
obtenerInfoAnio	Método	Método abstracto en la clase "Anio" que devuelve una cadena con información detallada del año.
obtenerInfo	Método	Método en la clase "Final" que devuelve una cadena con información detallada sobre una final.
BetPlayLigaFemenina	Clase (principal)	Representa el punto de entrada de la aplicación y contiene el bucle principal para mostrar el menú de años y detalles.
mostrarInfoAnio	Método	Método en la clase "BetPlayLigaFemenina" que muestra información detallada sobre un año específico.

V. Conceptos de la POO y MVC.

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

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

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

La clase abstracta **Anio** es un ejemplo de abstracción, ya que encapsula los detalles generales de cada año en el torneo, como el equipo campeón, el equipo subcampeón, los detalles de las finales y la jugadora goleadora. Aunque cada año tiene detalles únicos, la clase **Anio** proporciona una representación abstracta de estos detalles comunes.

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

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

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

heredan de **Anio** y extienden su funcionalidad específica para cada año.

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

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

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

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

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

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

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

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

Cada uno de estos componentes se centra en tareas especificas para facilitar la comprensión, el mantenimiento y las actualizaciones del código.
Ya que todo está separado, favorece la re utilización del código en diferentes partes de la aplicación. Adicional los componentes, al estar separados, son más fáciles de probar y esto mejora enormemente la calidad del Software.

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

VI. Tecnologías utilizadas.

Este proyecto fue desarrollado en:

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

Para el diagrama se ha utilizado:

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

VII. Referencias.

Colaboradores de los proyectos Wikimedia. "Liga Profesional Femenina de Fútbol 2020 (Colombia) - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Liga_Profesional_Femenina_de_Fútbol_2020_(Colombia) (accedido el 15 de agosto de 2023).

Colaboradores de los proyectos Wikimedia. "Liga Profesional Femenina de Fútbol 2021 (Colombia) - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Liga_Profesional_Femenina_de_Fútbol_2021_(Colombia) (accedido el 15 de agosto de 2023).

Colaboradores de los proyectos Wikimedia. "Liga Profesional Femenina de Fútbol 2022 (Colombia) - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Liga_Profesional_Femenina_de_Fútbol_2022_(Colombia) (accedido el 15 de agosto de 2023).

Colaboradores de los proyectos Wikimedia. "Liga Profesional Femenina de Fútbol 2023 (Colombia) - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Liga_Profesional_Femenina_de_Fútbol_2023_(Colombia) (accedido el 15 de agosto de 2023).