Criterio C: Desarrollo del producto

Técnicas usadas para el desarrollo del producto:

Programación Orientada a Objetos – POO

Para aplicar esta técnica, se ha desarrollado la aplicación mediante Processing. Por ello es que se ha recurrido a la creación y al uso de varias clases:

- **Button**: Para la creación y el uso de botones para facilitar la interacción del usuario con la aplicación.
- **Cone**: Es simplemente un cono que indica si la pizarra está en modo entrenamiento o modo partido
- **Guardado**: Es un entrenamiento o estratégia de partido guardado
- **ListaPaginada**: Se usa a modo de mostrar los resultados.
- **Pin**: un círculo con un indicador (1-5/C/L) usado en la pizarra a modo de representar a cada jugador en la pizarra
- **Pines**: El conjunto de pines de ambos equipos
- **Player**: Un jugador del equipo
- **Resultado**: Indica los resultados de los partidos pasados
- **Select y Select2:** para elegir entre diferentes opciones. Son diferentes estéticamente
- **Table**: Creación de una tabla que muestra los datos proporcionados por un array
- **TextField**: Área de texto para introducir datos

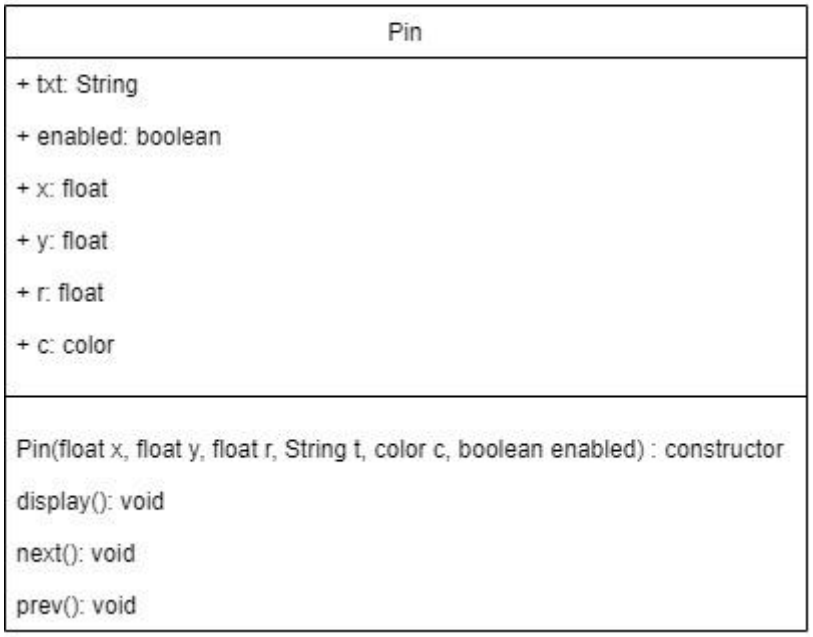
En caso de la tabla **Pin**, en primer lugar, se definen las variables que necesita ésta: x, y, r, txt c, y enabled, las cuales se usan para indicar: posición (x e y), radio del círculo (r), el texto que van a contener en ellos, el color y si están habilitados o no. A continuación, se muestra el método constructor usando las variables mencionadas anteriormente. Por otro lado, se crean los métodos setPosition, display, mouseOver, setEnability y getEnabled, de manera que llevan a cabo las siguientes tareas:
setPosition: Cambia la posición de los pines.
display: Muestra los pines mediante el uso de las variables indicadas anteriormente
mouseOver: Indica si el ratón se ubica encima del pin
setEnability: Cambia el valor booleano de la habilitación del pin
getEnabled: Devuelve el valor booleano de la habilitación del pin

Esta es la expresión escrita:

```
class Pin {

  // Propietades de un Pin
  float x, y, r;
  String txt;
  color c;
  boolean enabled;
```

```
  // Constructor
  Pin(float x, float y, float r, String t, color c, boolean enabled){
    this.x = x;
    this.y = y;
    this.r = r;
    this.txt = t;
    this.c = c;
    this.enabled = enabled;
  }


  // Setter de posición
  void setPosition(float x, float y){
    this.x = x;
    this.y = y;
  }

  // Dibuja el Pin
  void display(){
    pushStyle();
      stroke(0); strokeWeight(3); fill(c);
      ellipse(x, y-hBanner, 2*r, 2*r);
      fill(255); textAlign(CENTER); textSize(r);
      text(txt, x, y-hBanner + r/4);
    popStyle();
  }

  // Indica si el cursor está sobre el Pin
  boolean mouseOver(){
    return dist(mouseX, mouseY, this.x, this.y)<=this.r;
  }

  //Setter habilitado
  void setEnability (boolean enab){
    this.enabled = enab;
  }

  // Getter habilitado
  boolean getEnabled(){
    return enabled;
  }
}
```

Y esta la expresión visual:



Más tarde, en cuanto a la clase **ListaPaginada**, se definen las siguientes variables: r, currentRes, numRes, numResVisibles, x, y, w, h. Las cuales indican los siguientes valores:

- **r:** El array de resultados
- **currentRes:** El elemento que se está visualizando en ese momento
- **numRes:** El número total de elementos visualizados
- **numResVisibles:** el número de resultados a visualizar por pantalla
- **x e y:** Posición donde se muestra la lista
- **w y h:** Anchura y altura de la tabla, respectivamente.

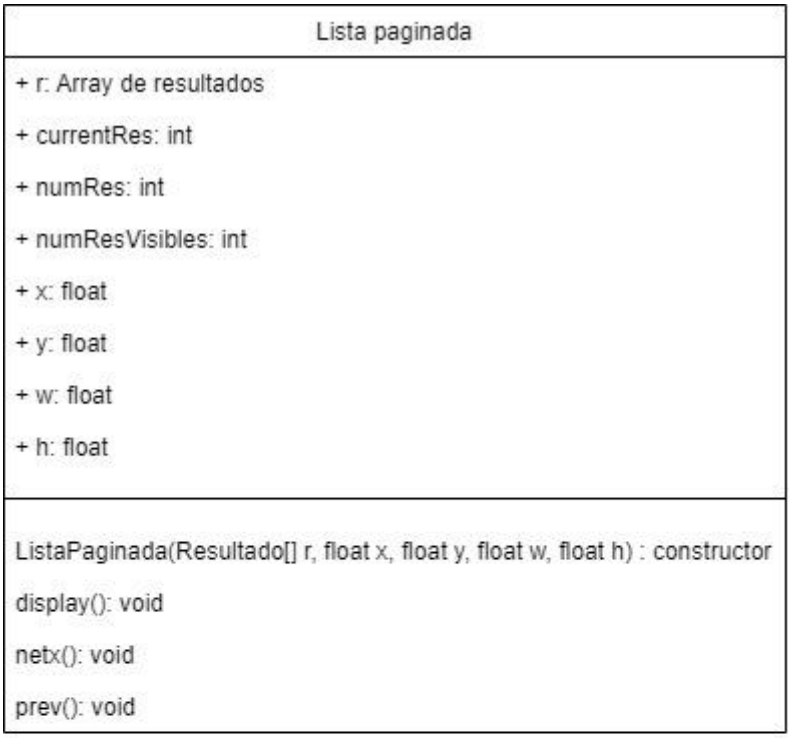
De esta manera, también se usan 3 métodos que son los siguientes:

- **Display**: enseña los
- **Next y prev:** Se usan para cambiar de página en la lista

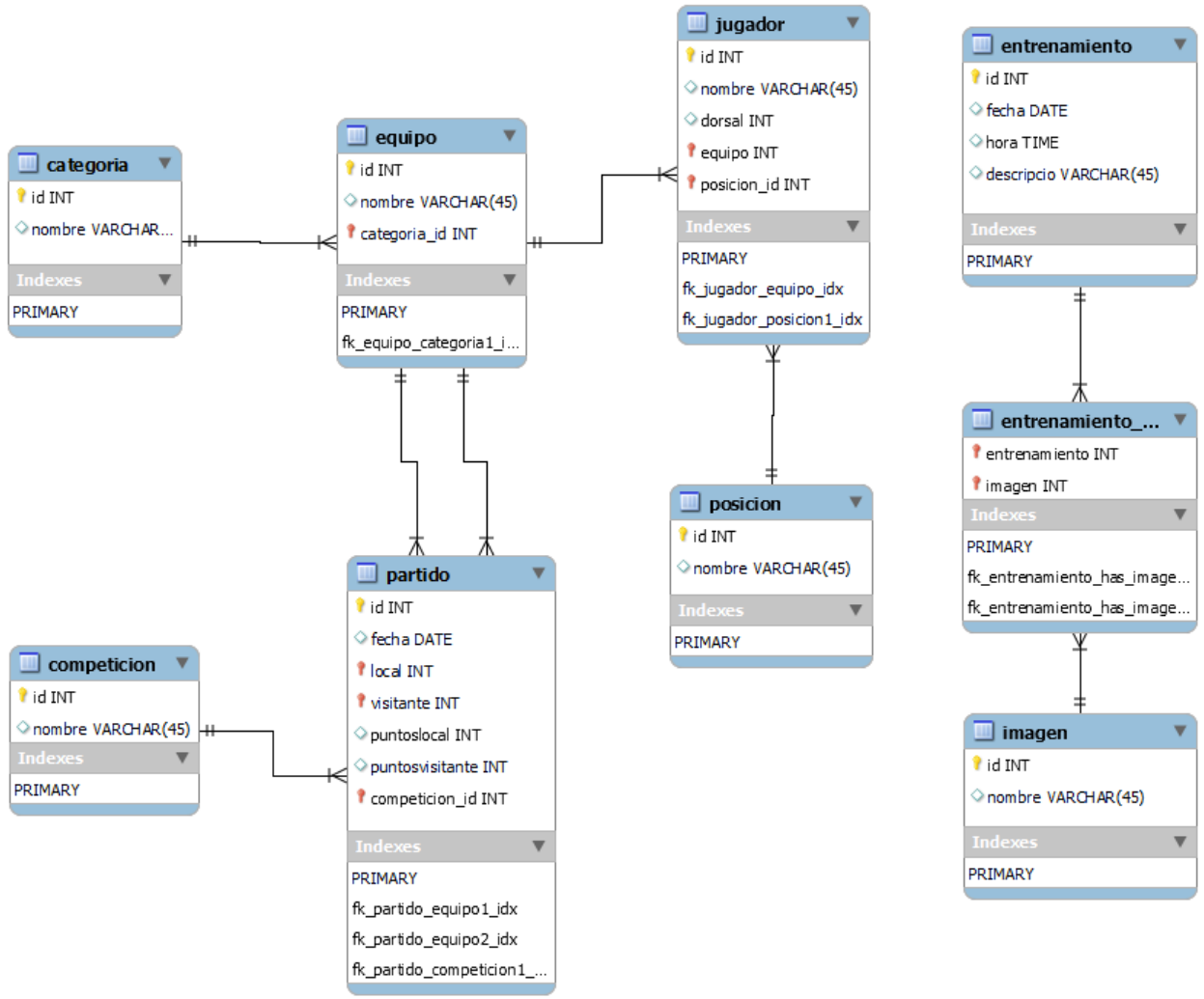
```
class ListaPaginada{
   //declaración de variables a usar
   Resultado[] r;
   int currentRes=0;
   int numRes;
   int numResVisibles = 3;
   float x, y, w, h;

   //método constructor
   ListaPaginada(Resultado[] r, float x, float y, float w, float h){
      this.r = r;
      this.x = x;
      this.y = y;
      this.w = w;
      this.h = h;
      numRes = r.length;
   }
```

```
    // método para dibujar la lista
    void display(){
        //set positions
        for(int i=0; i<numResVisibles; i++){
            r[i].setX(((i+0.5)*width)/(numResVisibles));
        }

        for (int i=numResVisibles+1, k = 0; i<2*numResVisibles+2 && i<r.length; i++, k++){
            r[i].setX(((k+0.5)*width)/numResVisibles);
        }

        //display results
        for(int i=currentRes*numResVisibles;i<numResVisibles+currentRes; i++){
            if(r[i] != null){
                r[i].display();
            }
        }

        // Activar y desactivar botones
        // para evitar un error index out of bounds
        if(currentRes*numResVisibles>r.length){
            next.setEnabled(false);
        } else{
            next.setEnabled(true);
        }
        if(currentRes*numResVisibles==0){
            prev.setEnabled(false);
        } else{
            prev.setEnabled(true);
        }
    }

    //métodos para pasar de página en la lista
    void next() { this.currentRes++;}
    void prev() { this.currentRes--;}
}
```

Y de forma visual:


Lista paginada
+ r: Array de resultados
+ currentRes: int
+ numRes: int
+ numResVisibles: int
+ x: float
+ y: float
+ w: float
+ h: float
ListaPaginada(Resultado[] r, float x, float y, float w, float h) : constructor
display(): void
netx(): void
prev(): void

Base de datos relacional



En el diagrama se muestran un total de 9 tablas:

- Competicion: tipo de competición: campeonato, liga regular, amistoso…
- Categoria: detalla a que categoría pertenece un jugador
- Equipo: indica el club contra el cual se juega el partido
- Partido: detalla la fecha, el equipo contrario, los puntos del local y los del equipo visitante.
- Jugador: incluye la información perteneciente a un jugador
- Posicion: Central/libero/colocador…
- Entrenamiento: detalla todos los datos pertenecientes al entrenamiento
- Entrenamiento_has_imagen: enlaza los entrenamientos con sus respectivas imagenes
- Imagen

Todas estas tablas de MySQL han sido gestionadas con PhPMyAdmin además de llevar a cabo la conexión Processing-Base de datos. Un ejemplo de ello es la tabla jugador:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id	int(11)			No	*Ninguna*		AUTO_INCREMENT
2	nombre	varchar(45)	utf8_general_ci		Sí	*NULL*		
3	dorsal	int(11)			Sí	*NULL*		
4	equipo	int(11)			No	1		
5	posicion_id	int(11)			No	*Ninguna*		

Los campos son los siguientes: id, nombre, dorsal, equipo y posicion_id. La llave primaria corresponde a id por otro lado, equipo es una llave foránea que se usa para conectar las tablas jugador y equipo.

Asimismo, es necesario destacar algunas query usadas en la aplicación:

- SELECT p.id AS id, p.fecha AS fecha, c.nombre AS competicion, e1.nombre AS local, e2.nombre AS visitante, p.setslocal AS setslocal, p.setsvisitante AS setsvisitante, p.s1local AS s1local, p.s1visitante AS s1visitante, p.s2local AS s2local, p.s2visitante AS s2visitante, p.s3local AS s3local, p.s3visitante AS s3visitante, p.s4local AS s4local, p.s4visitante AS s4visitante, p.s5local AS s5local, p.s5visitante AS s5visitante FROM partido p , equipo e1, equipo e2, competicion c WHERE p.local=e1.id AND p.visitante=e2.id AND p.competicion_id=c.id ORDER BY `fecha` DESC
- INSERT INTO posicion (id,nombre) VALUES (NULL, '"+n+"')

Estructuras de datos

En cuanto a las estructuras de datos usadas para el desarrollo de la aplicación, cabe destacar el abundante uso de Arrays bidimensionales en el código para almacenar los datos extraídos de la base de datos. Como por ejemplo:

- String[][] informacionJugadores
- String [][] infoPartido
- String[] headers
- String[] jugadores
- int [][] resultados
- int[][] fecha
- String[][] tableData

Una de las funcionalidades que se han podido implementar en la aplicación gracias al uso de los arrays bidimensionales es la visualización de los datos de los partidos anteriores, es decir la extraccion de los datos desde la base de datos mediante el siguiente metodo:

```
public String[][] getInfoTablaPartido(){
    int numFilas = getNumRowsTaula("partido");
```

```
    int numCols  = 17;

    String[][] infoPartido = new String[numFilas][numCols];
    try {
            ResultSet rs = query.executeQuery( "SELECT p.id AS id, p.fecha AS
                 fecha, c.nombre AS competicion, e1.nombre AS local, e2.nombre AS
                 visitante, p.setslocal AS setslocal, p.setsvisitante AS
                 setsvisitante, p.s1local AS s1local, p.s1visitante AS
                 s1visitante, p.s2local AS s2local, p.s2visitante AS s2visitante,
                 p.s3local AS s3local, p.s3visitante AS s3visitante, p.s4local AS
                 s4local, p.s4visitante AS s4visitante, p.s5local AS s5local,
                 p.s5visitante AS s5visitante FROM partido p , equipo e1, equipo
                 e2, competicion c WHERE p.local=e1.id AND p.visitante=e2.id AND
                 p.competicion_id=c.id ORDER BY `fecha` DESC");
        int nr = 0;

        while (rs.next()) {
            infoPartido[nr][0]  = String.valueOf(rs.getInt("id"));
            infoPartido[nr][1]  = String.valueOf(rs.getDate("fecha"));
            infoPartido[nr][2]  = rs.getString("competicion");

            infoPartido[nr][3]  = rs.getString("local");
            infoPartido[nr][4]  = rs.getString("visitante");

            infoPartido[nr][5]  = String.valueOf(rs.getInt("setslocal"));
            infoPartido[nr][6]  = String.valueOf(rs.getInt("setsvisitante"));
            infoPartido[nr][7]  = String.valueOf(rs.getInt("s1local"));
            infoPartido[nr][8]  = String.valueOf(rs.getInt("s1visitante"));
            infoPartido[nr][9]  = String.valueOf(rs.getInt("s2local"));
            infoPartido[nr][10] = String.valueOf(rs.getInt("s2visitante"));
            infoPartido[nr][11] = String.valueOf(rs.getInt("s3local"));
            infoPartido[nr][12] = String.valueOf(rs.getInt("s3visitante"));
            infoPartido[nr][13] = String.valueOf(rs.getInt("s4local"));
            infoPartido[nr][14] = String.valueOf(rs.getInt("s4visitante"));
            infoPartido[nr][15] = String.valueOf(rs.getInt("s5local"));
            infoPartido[nr][16] = String.valueOf(rs.getInt("s5visitante"));
            nr++;
        }
        println("Información PARTIDO \t-- COMPLETADO");
        //printArray2D(infoPartido);
        return infoPartido;
    }
    catch(Exception e) {
        println("Información PARTIDO \t-- ERROR");
        System.out.println(e);
        return null;
    }
}
```

En cuanto a las clases y funciones destacar la clase Table y la función getInfoTablaPartido() (mencionada anteriormente)

La primera permite la representación gráfica del array bidimensional que es el equipo y tiene la siguiente estructura:


Table
+ tableHeaders: String [][]
+ tableData: String [][]
+ columnWidths: float [][]
+ numCols: int
+ numRows: int
+ numPage: int
+numTotalPages:int
Table(int nr, int nc) : constructor
display(float x, float y, float w, float h): void
prevPage(): void
nextPage(): void
setHeaders(String[] h): void
setData(String [][] d): void
SetValueAt(String value, int nr, int nc): void
setColumnWidths(float[] w): void