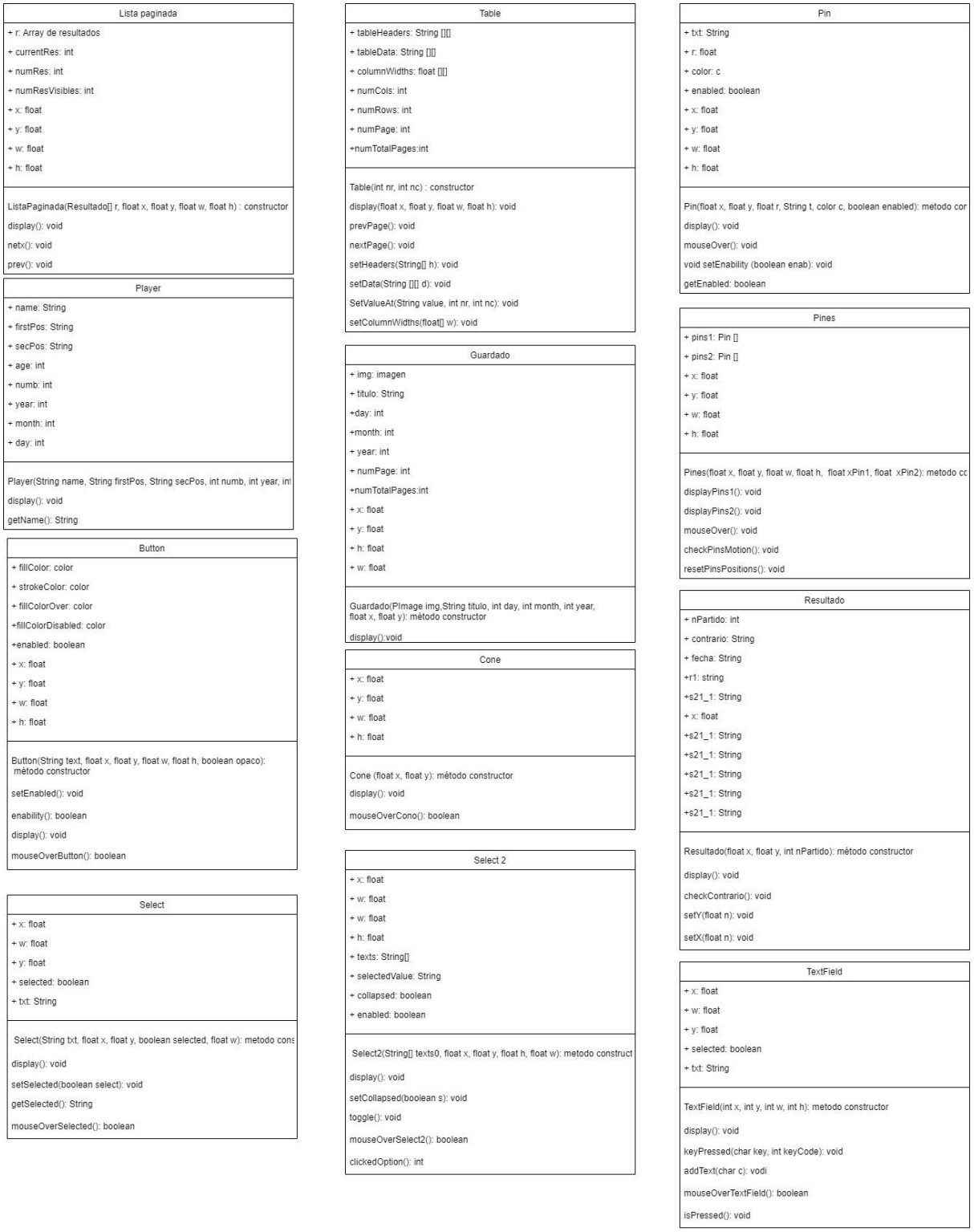
APÉNDICE III

DOCUMENTACIÓN DEL CÓDIGO

1. Diagrama de Clases.

A continuación se detalla el diagrama de clases perteneciente a la aplicación



2. Código de la aplicación.

2.1. Clase Tabla.

Esta clase crea tablas paginadas:

```
class Table {

 String[] tableHeaders;    // Títulos de las columnas
 String[][] tableData;     // Datos de la tabla
 float[] columnWidths;     // Anchura de las columnas
 int numCols, numRows;  // Número de filas y columnas
 int numPage;
 int numTotalPages;

 // Constructor
 Table(int nr, int nc){
   this.numRows = nr;
   this.numCols = nc;
   this.numPage = 0;
 }
  // Setters
 void setHeaders(String[] h){
   this.tableHeaders = h;
 }

  void setData(String[][] d){
   this.tableData = d;
   this.numTotalPages = longInfo / (this.numRows-1);
  }
  void setValueAt(String value, int nr, int nc){
   this.tableData[nr][nc] = value;
  }
  void setColumnWidths(float[] w){
   this.columnWidths = w;
  }

 //cambio de página
 void nextPage(){
   if(this.numPage<this.numTotalPages){
     this.numPage++;
   }
 }
  void prevPage(){
   if(this.numPage>0){
     this.numPage--;
   }
 }

 // Dibujar tabla
 void display(float x, float y, float w, float h){

   pushStyle();

     fill(255); stroke(0);strokeWeight(3);
     rect(x, y, w, h);
```

```
      float rowHeight = h / numRows;
      fill(blackCoral); stroke(0);strokeWeight(1);
      rect(x, y, w, rowHeight);

      // Dibujar filas
      stroke(0);

      for(int r = 1; r <numRows; r++){
        if(r==1){
          strokeWeight(3);
        } else {
          strokeWeight(1);
        }

        line(x, y + r*rowHeight, x + w, y + r*rowHeight);
      }

      // Dibujar columnas
      float xCol = x;

      for(int c = 0; c<numCols; c++){
        xCol += w*columnWidths[c]/100.0;
        line(xCol, y, xCol, y + h);
      }

      // Dibujar textos
      fill(0); textSize(24); textAlign(LEFT);

      for(int r = 0; r< numRows; r++){
        xCol = x;

        for(int c = 0; c< numCols; c++){

          if(r==0){
            text(tableHeaders[c], xCol + 10, y + (r+1)*rowHeight - 10);
          }
          else{
            int k = (numRows-1)*numPage + (r-1);
            if(k<this.tableData.length){
              text(tableData[k][c], xCol + 10, y + (r+1)*rowHeight - 10);
            }
          }
          xCol += w*columnWidths[c]/100.0;
        }
      }

      fill(0);
      text("Pag: "+(numPage+1)+" / "+(numTotalPages+1), x, y+h +50);
    popStyle();
  }
}
```

2.2. Clase ListaPaginada.

La clase ListaPaginada se usa para mostrar los resultados de los partidos anteriores de una forma más dinámica y visual.

```
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--;}
}
```

2.3. Classe TextField

Esta clase se usa a modo de introducir cadenas de texto a la aplicación.

```
class TextField {

  // Propietats del camp de text
  int x, y, h, w;

  // Colors
  color bgColor = color(140, 140, 140);
  color fgColor = color(0, 0, 0);
  color selectedColor = color(190, 190, 60);
  color borderColor = color(30, 30, 30);
  int borderWeight = 1;

  // Text del camp
  String text = "";
  int textLength = 0;
  int textSize = 24;

  boolean selected = false;

  // Constructor
  TextField(int x, int y, int w, int h) {
      this.x = x; this.y = y; this.w = w; this.h = h;
   }

  // Dibuixa el Camp de Text
  void display() {

      if (selected) {
         fill(selectedColor);
      } else {
         fill(bgColor);
      }

      strokeWeight(borderWeight);
      stroke(borderColor);
      rect(x, y, w, h, 5);

      fill(fgColor);
      textSize(textSize);
      textAlign(LEFT);
      text(text, x + 5, y + textSize);
   }
```

```
    // Afegeix, lleva el text que es tecleja
    void keyPressed(char key, int keyCode) {
       if (selected) {
          if (keyCode == (int)BACKSPACE) {
             removeText();
          } else if (keyCode == 32) {
             addText(' '); // SPACE
          } else {

            boolean isKeyCapitalLetter = (key >= 'A' && key <= 'Z');
            boolean isKeySmallLetter = (key >= 'a' && key <= 'z');
            boolean isKeyNumber = (key >= '0' && key <= '9');

            if (isKeyCapitalLetter || isKeySmallLetter || isKeyNumber) {
                addText(key);
            }
          }
       }
    }

    // Afegeix la lletra c al final del text
    void addText(char c) {
       if (textWidth(this.text + c) < w) {
          this.text += c;
          textLength++;
       }
    }

    // Lleva la darrera lletra del text
    void removeText() {
       if (textLength - 1 >= 0) {
          text = text.substring(0, textLength - 1);
          textLength--;
       }
    }

    // Indica si el ratolí està sobre el camp de text
    boolean mouseOverTextField() {
       if (mouseX >= this.x && mouseX <= this.x + this.w) {
          if (mouseY >= this.y && mouseY <= this.y + this.h) {
             return true;
          }
       }
       return false;
    }

    // Selecciona el camp de text si pitjam a sobre
    // Deselecciona el camp de text si pitjam a fora
    void isPressed() {
       if (mouseOverTextField()) {
          selected = true;
       } else {
          selected = false;
       }
    }
}
```

2.4. Clase Pines

Esta clase se usa para crear la colección de pines que se va a usar en la pantalla pizarra

```
class Pines {
 // Propiedades del conjunto de pines
 float x, y, w, h, xPin1, xPin2;
 // Arrays de Pins
 Pin[] pins1, pins2;

 // Constructor
 Pines(float x, float y, float w, float h,  float xPin1, float  xPin2){
   this.x = x;
   this.y = y;
   this.w = w;
   this.h = h;
   this.xPin1= xPin1;
   this.xPin2 = xPin2;
   // Crea 5 pins (Equipo 1)
   pins1 = new Pin[6];
   String txt;
   for(int i=0; i<pins1.length; i++){
     if(i<4){
         txt = (i+1)+"";
         pins1[i] = new Pin( xPin1  , y + 80*i, 30, txt, color(blackCoral),
true);
     } else if(i==4){
         txt = "C";
         pins1[i] = new Pin( xPin1, y + 80*i, 30, txt, color(prussianBlue),
true);
     } else if(i>4){
         txt = "L";
         pins1[i] = new Pin( xPin1 , y + 80*i, 30, txt, color(redSalsa),
true);
     }
   }

   // Crea 5 pins (Equipo 2)
   pins2 = new Pin[6];
   for(int i=0; i<pins2.length; i++){
      if(i<4){
         txt = (i+1)+"";
         pins2[i] = new Pin( xPin2 , y + 80*i, 30, txt, color(ming), true);
     } else if(i==4){
         txt = "C";
         pins2[i] = new Pin( xPin2 , y + 80*i, 30, txt, color(royalBlueDark),
true);
     } else if(i>4){
         txt = "L";
         pins2[i] = new Pin( xPin2 , y + 80*i, 30, txt, color(celadonBlue),
true);
         }
      txt = (i+1)+"";
   }
 }
```

```
 // Resetea la posición de todos los Pins
 void resetPinPositions(){
   for(int i=0; i<pins2.length; i++){
     pins1[i].setPosition( 100 , y + 80*i);
     pins2[i].setPosition( width-100 , y+ 80*i);
   }
 }

 // Dibuja los Pins
 void displayPins1(){
   for(Pin p : pins1){
     p.display();
   }
 }

 void displayPins2(){
   for(Pin p : pins2){
     p.display();
   }
 }

 // Comprueba si el cursor está sobre la Pizarra
 boolean mouseOver(){
   return mouseX >= this.x && mouseX <= this.x + this.w &&
          mouseY >= this.y && mouseY <= this.y + this.h &&
          mousePressed ;
 }

// Comprueba si es necesario mover algún pin
 void checkPinsMotion(){
   if(mousePressed){

     // Comprueba los pins del equipo 1
     for(Pin p : pins1){
       if(p.mouseOver()){
         p.setPosition(mouseX, mouseY);
         break;
       }
     }

     // Comprueba los pins del equipo 2
     for(Pin p : pins2){
       if(p.mouseOver()){
         p.setPosition(mouseX, mouseY);
         break;
       }
     }
   }
 }
}
```

2.5. Clase Pin

La clase pin crea un círculo a modo de representar donde se sitúa un jugador en la pizarra.

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

  // Constructor
es 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();
 }

// Devuelve 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;
 }
}
```

3. Repositorio GITHUB.

Se puede encontrar el código entero en el siguiente enlace:
https://github.com/catalinafullana/Solucion