APÉNDICE VII

CÓDIGO FINAL DE LA SOLUCIÓN INFORMÁTICA

basededatos.pde

```
import processing.core.PApplet;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

String user     = "admin";
String pass     = "12345";
String database = "voley";
Connection con;
Statement query;

// Conexión a la BBDD
public void connectBBDD(){
    try {
        con= DriverManager.getConnection("jdbc:mysql://localhost:8889/"+database, user, pass);
        query = con.createStatement();
        System.out.println("Database \t-- connected");
    }
    catch(Exception e) {
        println("Database \t-- error");
        System.out.println(e);
    }
}

//getter para n filas de una tabla
public int getNumRowsTaula(String nomTaula){
    try {
        ResultSet rs = query.executeQuery( "SELECT COUNT(*) AS n FROM "+ nomTaula );
        rs.next();
        int numRows = rs.getInt("n");
        return numRows;
    }
    catch(Exception e) {
        System.out.println(e);
        return 0;
    }
}

// Getter datos tabla
// CANVIAR DADES, AFEGIR QUADRANT QUE POSI PRIMERA I SEGONA POSICIÓ, FER
QUERY.
public String[][] getInfoTablaJugadores(){
    int numFilas = getNumRowsTaula("jugador");
    int numCols  = 3;

    String[][] informacionJugadores = new String[numFilas][numCols];
    try {
```

```
        ResultSet rs = query.executeQuery( "SELECT j.nombre AS nombre, j.dorsal AS dorsal,
p.nombre AS posicion FROM jugador j, posicion p WHERE j.posicion_id=p.id");
        int nr = 0;
        while (rs.next()) {
            informacionJugadores[nr][1] = rs.getString("nombre");
            informacionJugadores[nr][0] = String.valueOf(rs.getInt("dorsal"));
            informacionJugadores[nr][2] = rs.getString("posicion");

            nr++;
        }
        println("Información JUGADORES \t-- COMPLETADO");
        //printArray2D(informacionJugadores);
        return informacionJugadores;
    }
    catch(Exception e) {
        println("Información JUGADORES \t-- ERROR");
        System.out.println(e);
        return null;
    }
}

public String[][] getInfoTablaEquipo(){
    int numFilas = getNumRowsTaula("equipo");
    int numCols  = 3;

    String[][] equipoInfo = new String[numFilas][numCols];
    try {
        ResultSet rs = query.executeQuery( "SELECT j.id AS id, j.nombre AS nombre, p.nombre AS
categoria FROM equipo j, categoria p WHERE j.categoria_id=p.id ORDER BY `fecha` ASC");
        int nr = 0;
        while (rs.next()) {
            equipoInfo[nr][0] = String.valueOf(rs.getInt("id"));
            equipoInfo[nr][1] = rs.getString("nombre");
            equipoInfo[nr][2] = rs.getString("categoria");
            nr++;
        }

        println("Información EQUIPO \t-- COMPLETADO");
        //printArray2D(equipoInfo);
        return equipoInfo;
    }
    catch(Exception e) {
        println("Información EQUIPO \t-- ERROR");

        System.out.println(e);
        return null;
    }
}

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 JUGADORES \t-- ERROR");
        System.out.println(e);
        return null;
    }
}

//imprimir contenido array en dos dimensiones
public void printArray2D(String[][] dades){
    println("\nTabla " + dades);
    for(int f=0; f<dades.length; f++){
        for(int c=0; c<dades[f].length; c++){
            print(dades[f][c]+" \t ");
        }
        println();
```

```
    }
}
public String[] getInfoPosicion(){
    int numFilas = getNumRowsTaula("posicion");
    int numCols  = 2;

    String[] informacionPosicion = new String[numFilas];
    try {
        ResultSet rs = query.executeQuery( "SELECT * FROM `posicion`");
        int nr = 0;
        while (rs.next()) {
            informacionPosicion[nr] = rs.getString("nombre");
            //informacionPosicion[nr] = String.valueOf(rs.getInt("id"));

            nr++;
        }
        println("Información POSICION \t-- COMPLETADO");
        //printArray2D(informacionPosicion);
        return informacionPosicion;
    }
    catch(Exception e) {
        println("Información Posicion \t-- ERROR");
        System.out.println(e);
        return null;
    }
}

// Insertar nuevos datos tabla posición
public void insertPosicion( String n){
    try {
        String q = "INSERT INTO posicion (id,nombre) VALUES (NULL, '"+n+"')";
        println("INSERT: "+q);
        query.execute( q);
        println("INSERT OK :)");
    }
    catch(Exception e) {
        System.out.println(e);
    }
}

String formataFechaEsp(String fechaEntrada){

  String y = fechaEntrada.split("-")[0];
  String m = fechaEntrada.split("-")[1];
  String d = fechaEntrada.split("-")[2];

  return d+"/"+m+"/"+y;
}
```

Button.pde

```
class Button {

 // Propiedades
 float x, y, w, h;  // Posición y dimensiones
 // Colores de contorno, de relleno...
 color fillColor, strokeColor;
 color fillColorOver, fillColorDisabled;
 String textBoton;  // Texto del botón
 boolean enabled;  // Habilitado o deshabilitado
 boolean opaco;    // Opaco o transparente
 int a;

 // Método Constructor
  Button(String text, float x, float y, float w, float h, boolean opaco){
   this.textBoton = text;
   this.x = x;
   this.y = y;

   this.w = w;
   this.h = h;
   this.enabled = true;
   this.opaco = opaco;

   fillColor = color(blackCoral);
   fillColorOver = color(blackCoral, 200);
   fillColorDisabled = color(50, 80, 100);
   strokeColor = color(0,0,0,0);
 }

 // Setters varios
 void setEnabled(boolean b){
   this.enabled = b;
 }
 boolean enability(){
   return enabled;
 }

 // Dibuja el botón
 void display(){
   pushStyle();

     if(opaco){
       a=255;
     } else{
       a=0;
     }

     if(!enabled){
```

```
      fill(fillColorDisabled,a);   // Color deshabilitado
    }
    else if(mouseOverButton()){
      fill(fillColorOver, a);      // Color cuando se ubica el ratón encima
    }
    else{
      fill(fillColor, a);          // Color botón activo pero sin el ratón encima
    }
    stroke(strokeColor, a); strokeWeight(2);        //Color y grosor del contorno
    float b=0;
    if(mouseX>hBanner){
      b=hBanner;
    }
    rect(this.x, this.y- hBanner, this.w, this.h, 10);    // Rectangulo del botón


    // Parámetros estéticos del texto
    fill(255); textAlign(CENTER); textSize(20);
    text(textBoton, this.x + this.w/2, this.y + this.h/2 + 10- hBanner);
   popStyle();
}


// Indica si el cursor esta sobre el botón
boolean mouseOverButton(){
  return (mouseX >= this.x) &&
         (mouseX<=this.x + this.w) &&
         (mouseY>= this.y) &&
         (mouseY<= this.y +this.h);
}
}
```

colors.pde

```
color prussianBlue = color(0, 36, 61);
color blackCoral =   color(63, 90, 110);
color ming =         color(88, 123, 127);
color royalBlueDark =color(10, 36, 99);
color redSalsa =     color(251, 54, 64);
color platinum =     color(226, 226, 226);
color celadonBlue =  color(#FB3640);

color c1;
color c2;

//pantalla paleta
void paleta(){
   pushStyle();
      noStroke();

      fill(prussianBlue);
      rect(0,0, width/7, height);


      fill(blackCoral);
      rect(width/7, 0, width/7, height);


      fill(ming);
      rect(2*width/7,0, width/7, height);


      fill(royalBlueDark);
      rect(3*width/7,0, width/7, height);


      fill(redSalsa);
      rect(4*width/7,0, width/7, height);

      fill(platinum);
      rect(5*width/7,0, width/7, height);


      fill(celadonBlue);
      rect(6*width/7,0, width/7, height);

      textFont(fuente3); stroke(255); textSize(15); textAlign(CENTER); fill(255);
      text("Prussian blue" , width/14, height - 100);
      text("Black coral" ,   3*width/14, height - 100);
      text("Ming" ,          5*width/14, height - 100);
      text("Royal Blue Dark",7*width/14, height - 100);
      text("Red Salsa" ,     9*width/14, height - 100);
      text("Platinum" ,      11*width/14, height - 100);
      text("Celadon blue" , 13*width/14, height - 100);
   popStyle();
```

```
}
```

Cone.pde

```
class Cone{
    //declaraciones de parámetros
    float x = width-45;
    float y = height - hBanner - 70;

    //método constructor
    Cone(float x, float y){
        this.x = x;
        this.y = y;
    }

    //setter posición
    void setPosition(float x, float y){
        this.x = x;
        this.y = y;
    }
    //método dibujar
    void display(){
        pushStyle();
            image(cono, x, y, 35, 45);
        popStyle();
    }

    //getter para saber si el ratón está encima del cono
    boolean mouseOverCono(){
        return (mouseX >= this.x) &&
          (mouseX<=this.x + 35) &&
          (mouseY>= this.y) &&
          (mouseY<= this.y + 45);
    }
}
```

declaracionesyDatos.pde

```
PImage img0, img1, img2, img3, cono, usuario;

PFont fuente1, fuente2, fuente3, fuente4;

Button goToMiEquipo;
Button goToPizarra;
Button goToResultados;
Button goToGuardados;
Button goToSobreNosotros;

Button save, newPlayer;
Button b1,b2;
Button close;
Button reset;

Pines p;
Table t;
Player p1;
Player p2;
Player p3;

Select entrenamiento,competicion;
Select mediaPista, pistaEntera;


Guardado guardado1;
Guardado guardado2;

int i=0;
int j=0;
int k=0;
int nConos=0;

float ancho= 500;
float alto = ancho*0.62;
float margenVertical = 110;

boolean logged = false;
TextField nombre, posicion1, dayn, monthn, yearn, ndorsal;
TextField nSaved, dSaved, mSaved, ySaved;

TextField s1_1, s1_2, s2_1, s2_2, s3_1, s3_2, s4_1, s4_2, s5_1, s5_2;

int longInfo;

String hability1, hability2, hability3, hability4, hability5;
String s;

int pantalla = 0;

boolean halfB = false;
boolean training = true;
```

```
// Número de files (capçalera inclosa) i columnes de la taula
int files = 20, columnes = 3;

// Títols de les columnes
String[] headers = {"Dorsal", "Nombre", "Posición"};
String[] jugadores = {"", "J1", "J2", "J3", "J4", "J5", "J6", "J7", "J8"};
int nPlayers =jugadores.length;

// Amplades de les columnes
//han de sumar 100
float[] colWidths = {10,70,20};
float[] colWidthsPlayers = {20,40,40};

String[][] informacionJugadores;
String [][] infoPartido;
String [] infoPosicion;

int xPosicion = width/2;
int yPosicion = 200;

Resultado r1, r2, r3, r4, r5, r6, r7;

// tu equipo - equipo contrario - set 1 equipo- set 1equipo contrario - set 2 equipo...
int [][] resultados  = {
    {0, 3, 25, 19, 25, 21, 22, 25, 24, 26, 0, 0},
    {1, 3, 25, 19, 22, 25, 24, 26, 15, 25, 0, 0},
    {2, 3, 25, 19, 25, 21, 22, 25, 24, 26,10,15},
    {1, 3, 25, 19, 22, 25, 24, 26, 22, 25, 0, 0},
    {0, 3, 25, 19, 25, 21, 22, 25, 24, 26, 0, 0},
    {1, 3, 25, 19, 22, 25, 24, 26, 15, 25, 0, 0},
    {2, 3, 25, 19, 25, 21, 22, 25, 24, 26,20,25},
    {1, 3, 25, 19, 22, 25, 24, 26, 22, 25, 0, 0},
    {0, 3, 25, 19, 25, 21, 22, 25, 24, 26, 0, 0},
    {1, 3, 25, 19, 22, 25, 24, 26, 15, 25, 0, 0},
    {2, 3, 25, 19, 25, 21, 22, 25, 24, 26,20,25},
    {1, 3, 25, 19, 22, 25, 24, 26, 22, 25, 0, 0},
};
String [] equipos = {"Equipo A", "Equipo B", "Equipo C","Equipo D", "Equipo E", "Equipo F","Equipo
G", "Equipo B", "Equipo C","Equipo A", "Equipo B", "Equipo C","Equipo A", "Equipo B", "Equipo
C","Equipo X", "Equipo Y", "Equipo Z"};

//formato DD/MM/AAAA
int[][] fecha={
    {10, 02, 2021},
    {13, 03, 2022},
    {10, 02, 2021},
    {13, 03, 2022},
    {10, 02, 2021},
    {13, 03, 2022},
    {10, 02, 2021},
    {13, 03, 2022},
```

```
    {10, 02, 2021},
    {13, 03, 2022},
    {10, 02, 2021},
    {13, 03, 2022},
};

Cone cono1;

color currentCol;

Button next, prev, guardarJugador;

ListaPaginada e;

boolean menuDisplayed = false;
PImage [] conos;

float logoW = 150 ;

float hBanner = 80;


String sets;
int h  = 314 - (int)hBanner;
int alturaCelda=40;

float marginW = 0;
float marginH = 0;

float wButton= 190;
float hButton= 50;

float xT = marginW +4 ;
float yT = marginH + hBanner + 20;

float wT = 950;
float hT = 550;

//pizarra
float wBlackboard = 650;
float hBlackboard = wBlackboard;

//toolbar
float wTB = 50;
float hTB = height - hBanner;


//tabla
float tableW = 1400, tableH = 700;

float wButtonTraining = 200;
float hButtonTraining = 40;
```

```
Select2 s1, s2;

// Valors dels Selects
String[] selectValues = {"RED", "GREEN", "BLUE"};

// Dimensions dels botons
float selectW = 300;
float selectH = 30;

// Color de fons de l'App
color bgColor = color(255);

// Valor numèric
int n = 0;
```

entrada.pde

```
void p0(int j){
    //video();
    pushStyle();
    textAlign(CENTER);
    textFont(fuente3);

    for(int i=0; i<2*width; i+=10){
        if(i%3==0){
            stroke(redSalsa,j);
        }else if(i%3==1){
            stroke(royalBlueDark,j);
        }else{
            stroke(ming,j);
        }
        line(i, 0, 0,i);
    }

    fill(royalBlueDark, j);
    text("keep the ball", width/2, 250);
    logoDrawing(width/2, height/2+50, 400, "FLYING", c3, c4);

    popStyle();

    stroke(2);
    fill(0);

}
```

Guardado.pde

```
class Guardado{

   //declaración de parámetros
   PImage img;
   String titulo;
   int day, month, year;
   float x, y, w, h;

   //método constructor
   Guardado(PImage img,String titulo, int day, int month, int year){

      this.day = day;
      this.month = month;
      this.year = year;
      this.titulo = titulo;
      this.x = 0;
      this.y = 0;
      this.img = img;
      w = 350;
      h = 250;

   }
   //setter posición
   void setXY(float a, float b){
      x = a;
      y = b;
   }

   //método dibujar
   void display(){
      pushStyle();
         fill(150);
         stroke(1);
         rect(x,y, w, h, 5);
         image(img, x+10,y+10, w- 20, h-50);
            //println(i+" "+j);
         textSize(20);
         fill(0);
         textAlign(LEFT);
         text(this.titulo, x+10, y+h-20);
         textSize(10);
         text(this.day +"/"+this.month+"/"+this.year, x+ 15, y+h-10);
      popStyle();
   }


}
```

guardados.pde

```
void guardados(){
   fondo();

   guardado1.setXY(30,155- hBanner);
   guardado1.display();
}

void fondo(){
   if(entrenamiento.getSelected()){
      fill(120);
   } else{
      fill(100);
   }

   stroke(0);
   for(int i=0; i<4; i+=1){
      for(int j=0; j<3; j+=1){
         rect( 30 + i*(width/4), 75 + j*(20 + height/4), 350, 250, 5);
         //println(i+" "+j);
      }
   }

}
```

inicio.pde

```
//animación pantalla de carga
void inicio(){
   background(blackCoral);
   pushMatrix();
      pushStyle();
      rotate(PI/5);
         textAlign(LEFT);
         fill(royalBlueDark);
         for(int i=-1000; i< 2000; i+=100){
            textFont(fuente3);
            logoDrawing(width/2-420, i-500+k, 100, "KEEP THE BALL FLYING", redSalsa,
royalBlueDark);
            textFont(fuente2);
```

```
            logoDrawing(width-40, i+350-k, 100, "   KEEP THE BALL FLYING", redSalsa,
royalBlueDark);
          }

          k+=5;
       popStyle();
    popMatrix();
}
```

listaPaginada.pde

```
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--;}
}
```

nuevosResultados.pde

```
//pantalla para introducir nuevos resultados
void nuevosResultados(){
    pushStyle();
        textAlign(CENTER);
        text("Nuevo resultado:", width/2, 50);

    popStyle();
}
```

p1.pde

```
//pantalla tabla
void p1(){

  pushStyle();
    //dibujar el objeto tabla
    t.display((width-tableW)/2, 100, tableW, tableH);

    //dibujar los botones correspondientes
    b1.display();
    b2.display();
```

```
    // Actualizar forma del cursor
    updateCursor();

    //Dibujar el nuevo jugador como objeto
    newPlayer.display();

  popStyle();
}

//pantalla para introducir un nuevo jugador
void newPlayer(){
  textAlign(LEFT);
  pushStyle();
    image(usuario, 130, 70, 50, 50);
    logoDrawing(450, 102.0, 70, "Nuevo jugador", redSalsa, royalBlueDark);
  popStyle();

  textSize(15); fill(0);

  pushStyle();
    textFont(fuente1);
    fill(blackCoral);
    textSize(30);

    text("Nombre:", 300, 200);
    text("Fecha de nacimiento:", 300, 300);
    text("Número dorsal: ", 300, 400);
    text("Primera posición: ", 300, 500);
    text("Segunda posición: ",  900, 500);


    s1.display();
    s2.display();
    nombre.display();
    dayn.display();
    monthn.display();
    yearn.display();
    ndorsal.display();
    guardarJugador.display();

  popStyle();

}
```

p2.pde

```
void p2(){

  if(mediaPista.getSelected()){
    halfBlackboard();
    p.displayPins1();

  } else{
    fullBlackboard();
    p.displayPins1();
  }

  p.checkPinsMotion();

  textAlign(CENTER);

  if (entrenamiento.getSelected()){
    text("TRAINING", width/2, height/2);

  } else{
    text("PLAYING", width/2, height/2);

  }

  if(competicion.getSelected()&&pistaEntera.getSelected()){
    p.displayPins2();
  }
  toolBar(entrenamiento.getSelected());
  menu();


}


//Dibuja sólo media pista
void halfBlackboard() {
  pushMatrix();
    pushStyle();
      rectMode(CENTER);
      fill(255);
      rect(width/2, height/2, wBlackboard, hBlackboard);
      stroke(0);
      strokeWeight(3);
      translate(width/2-wBlackboard/2,height/2-hBlackboard/2);

      beginShape();
        vertex(0, 0);
        vertex(0, hBlackboard);
        vertex(wBlackboard, hBlackboard);
        vertex(wBlackboard, 0);
      endShape(CLOSE);

      line(-30, 2*hBlackboard/3, wBlackboard + 30,2*hBlackboard/3);
```

```
    line(-40, hBlackboard, wBlackboard + 40, hBlackboard);

   popStyle();
  popMatrix();
}

//Dibuja la barra de herramientas en función de si está
//habilitado el modo entrenamiento o el modo competición
void toolBar(boolean training){

  pushStyle();

   fill(255);
   stroke(0);
   strokeWeight(0);

   rect(width-wTB-marginW,0, wTB, height-hBanner-2*marginH);
   strokeWeight(2);
   line(width-wTB-marginW, 0, width-wTB-marginW, height-hBanner-2*marginH);

   rectMode(CORNER);
   strokeWeight(1);

   save.display();
   colores(width-45, (height-hBanner-40)/2 - 60, color (255, 0,0));
   colores(width-45, (height-hBanner-40)/2, color (0, 255,0));
   colores(width-45, (height-hBanner-40)/2+60, color (0, 0,255));

   if(mouseOverObject(0,0,width- 40, height-hBanner)){

     if(mousePressed){
       fill(currentCol);
       ellipse(mouseX, mouseY-hBanner, 5, 5);
     }
   }else{
     cursor(ARROW);
   }

   if(training){
     image(cono,width-45, height - hBanner - 70, 35, 45);
     if (mouseOverObject(width-45, height - 70, 35, 45) && mousePressed){
         cono1.display();
         nConos++;
         println("número de conos: "+ nConos);
         delay(1000);
     }
     if(cono1.mouseOverCono()){
       cono1.setPosition(mouseX, mouseY);
     }
   }
  popStyle();
}
```

```
//Dibuja la pantalla entera
void fullBlackboard(){
  pushMatrix();
    pushStyle();

      rectMode(CENTER);
      fill(255);
      stroke(0);
      strokeWeight(3);
      translate(width/2-wBlackboard,height/2-hBlackboard/2-50);

      beginShape();
        vertex(0, 0);
        vertex(0, hBlackboard);
        vertex(wBlackboard*2, hBlackboard);
        vertex(wBlackboard*2, 0);
      endShape(CLOSE);

      line(wBlackboard, -30,wBlackboard, hBlackboard+30);

      line(2*wBlackboard/3, -20, 2*wBlackboard/3, hBlackboard +20);
      line(4*wBlackboard/3, -20, 4*wBlackboard/3, hBlackboard +20);

    popStyle();
  popMatrix();
}


void  trainingButton1(){
  if(training){
    c1 = color(150);
    c2 = color(100);
  } else {
    c1 = color(100);
    c2 = color(150);
  }

  pushStyle();

    fill(c1);
    stroke(0);
    strokeWeight(2);
    rect(width/2 - wButtonTraining/2, hButtonTraining-20, wButtonTraining/2, hButtonTraining,
10,0,0,10);

    fill(c2);
    rect(width/2, hButtonTraining-20, wButtonTraining/2, hButtonTraining, 0,10,10,0);

  popStyle();
}
```

```
//dibuja el menú
void menu(){
  pushStyle();

    stroke(0);
    strokeWeight(4);
    fill(0);
    textAlign(CENTER);


    //En caso de que el menú no esté desplegado, muestra tres rayas
    //en la parte superior izquierda de la pantalla.
    //De lo contrario, muestra el menú entero.
    if (!menuDisplayed){
      line(15, 15, 60, 15);
      line(15, 30, 60, 30);
      line(15, 45, 60, 45);
    } else{

      for(int i=0; i<2*height; i+=10){
        if(i%3==0){
            stroke(redSalsa,100);
        }else if(i%3==1){
            stroke(royalBlueDark,100);
        }else{
            stroke(ming,100);
        }

        if(i<300){
        line(i, 0, 0,i);
        }else{
          line(300, i-300, 0, i);
        }
      }

      fill(blackCoral, 170);
      strokeWeight(1);
      rect(25,50,250,250);

      stroke(0);
      strokeWeight(4);
      line(15, 15, 15+30, 15+30);
      line(15, 15+30, 15+30, 15);
      stroke(255);
      strokeWeight(1);


      textAlign(LEFT);
      textSize(40);
      fill(prussianBlue);
      pushStyle();
      textFont(fuente2);
```

```
    //Título:
    text("Menú:", 80, 45);
    popStyle();

    fill(255,255,255, 0);
    strokeWeight(1);

    //Selects:
    entrenamiento.display();
    competicion.display();
    mediaPista.display();
    pistaEntera.display();
    reset.display();
  }

  //Sólo es posible pulsar el boton cuando el framecount es un múltiple de 5
  //para mejorar experiencia del usuario y evitar posibles missclicks
  if(frameCount%5 == 0){
     if((mouseX >= 15) &&
       (mouseX<=120) &&
       (mouseY>= 100) &&
       (mouseY<=130)&& mousePressed){
       menuDisplayed= !menuDisplayed;
      }
  }
 popStyle();

}


//Método para devolver si el ratón está encima de un campo u objeto
boolean mouseOverObject(float x, float y, float w, float h){
  return (mouseX >= x) &&
        (mouseX<=x + w) &&
        (mouseY>= y) &&
        (mouseY<= y +h);
}
//??
void colores (float x, float y, color c){
  fill(c);
  rect(x, y, 40, 40, 5);

  if(mouseOverObject(x, y+hBanner, 40, 40)){
    cursor(HAND);
    if(mousePressed){
      currentCol = c;
    }
  }

}
```

Pin.pde

```
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;
```

```
  }
}
```

Pines.pde

```
class Pines {

  // Propietades de la Pizarra
  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;
      }
    }
  }
 }
}
```

Player.pde

```
class Player{
  // Declaración de datos
  String name;

  String firstPos, secPos;
  int age, numb, year, day, month;

  //Método constructor
  Player(String name, String firstPos, String secPos, int numb, int year, int day, int month){
    this.name = name;

    this.year = year;
    this.day = day;
    this.month = month;

    this.firstPos = firstPos;
    this.secPos = secPos;
    this.numb = numb;
  }


  void display(){
    textAlign(LEFT);

    pushStyle();
      image(usuario, 130, 70, 50, 50);
      logoDrawing(450, 102.0, 70, this.name, redSalsa, royalBlueDark);
    popStyle();

    textSize(15); fill(0);

    pushStyle();
      textFont(fuente1);
      fill(blackCoral);
      textSize(30);

      text("Fecha de nacimiento: "+ this.day+ "/"+ this.month+"/"+this.year, 300, 300);
      text("Número dorsal: "+ this.numb, 300, 400);
```

```
        text("Primera posición: " +this.firstPos, 300, 500);
        text("Segunda posición: " + this.secPos,  900, 500);
      popStyle();
   }


   String getName(){
      return this.name;
   }
}
```

Resultado.pde

```
class Resultado{
   float x,y;
   int nPartido;

   String contrario;
   String fecha;
   String tipo;

   // resultado del partido. equipos 1 y 2
   String r1;
   String r2;

   //set 1
   String s1_1;
   String s1_2;

   //set 2
   String s2_1;
   String s2_2;

   //set 3
   String s3_1;
   String s3_2;

   //set 4
   String s4_1;
   String s4_2;

   //set 5
   String s5_1;
   String s5_2;

   Resultado(float x, float y, int nPartido){
      this.x = x;
      this.y = y;

      this.nPartido = nPartido;
```

```
    this.contrario = infoPartido[nPartido][3];
    this.tipo = infoPartido[nPartido][2];
    this.fecha  = formataFechaEsp(infoPartido[nPartido][1]);

    this.r1 = infoPartido[nPartido][5];
    this.r2 = infoPartido[nPartido][6];

    this.s1_1 = infoPartido[nPartido][7];
    this.s1_2 = infoPartido[nPartido][8];
    this.s2_1 = infoPartido[nPartido][9];
    this.s2_2 = infoPartido[nPartido][10];
    this.s3_1 = infoPartido[nPartido][11];
    this.s3_2 = infoPartido[nPartido][12];
    this.s4_1 = infoPartido[nPartido][13];
    this.s4_2 = infoPartido[nPartido][14];
    this.s5_1 = infoPartido[nPartido][15];
    this.s5_2 = infoPartido[nPartido][16];
}



void setX (float x){
    this.x = x;
}

void setY (float y){
    this.y = y;
}

void checkContrario(){
    if(infoPartido[nPartido][4]!=infoPartido[4][4] ){
        this.contrario = infoPartido[nPartido][4];
    } else{
        this.contrario = infoPartido[nPartido][3];
    }
}

void display(){
    checkContrario();
    pushStyle();
        rectMode(CENTER);
        fill(150);
        rect(x, y, 450, 800, 5);
            pushStyle();
                fill(255);
                textFont(fuente1);
                textSize(180);
                textAlign(CENTER);
                fill(redSalsa);

                //resultado principal
                text(r1 +"-"+ r2, x-3, y-112, 380);
```

```
                fill(255);
                text(r1 +"-"+ r2, x, y-110, 380);
                fill(royalBlueDark);
                text(r1 +"-"+ r2, x+3, y-108, 380);
                textFont(fuente2);

                textSize(75);
                fill(redSalsa);

                text(contrario, x-2, y+ 40-2, 500, 300);
                fill(255);
                text(contrario, x, y+ 40, 500, 300);
                fill(royalBlueDark);
                text(contrario, x+2, y+ 40+2, 500, 300);
                textSize(15);
                fill(255);
                text(tipo, x,y+ 300, 500, 300);

                textFont(fuente1);
                textSize(15);
                fill(255);
                text(fecha,  x, y- 350, 380);
                textSize(12);
                textAlign(CENTER);

                // Encadenamiento de sets en funcion del numero de sets:
                sets = s1_1+"-"+s1_2+"/"+s2_1+"-"+s2_2+"/"+s3_1+"-"+s3_2+ "/" + s4_1 + "-" + s4_2+ "/"
+ s5_1 + "-" + s5_2;
                text(sets , x, y + 130, 380);
            popStyle();
        popStyle();
    }
}
```

Resultados.pde

```
void resultados(){
    e.display();
    prev.display();
    next.display();
}
```

Select.pde

```
class Select{
    float x, y, w;
```

```
    boolean selected;
    String txt;

    Select(String txt, float x, float y, boolean selected, float w){
        this.txt = txt;

        this.x =  x;
        this.y = y;
        this.w = w;
        this.selected = selected;
    }

    void display(){
        pushStyle();
            fill(255,255,255,0);
            stroke(0);
            strokeWeight(1);
            rect(x,y, w, w);

            if(selected){
                strokeWeight(2);
                line(x,y, x+w, y+w);
                line(x, y+w, x+w,y);
            }

            textAlign(LEFT);
            textSize(18);
            fill(255);
            text(this.txt, x+w+ 7, y+13);
        popStyle();
    }


    void setSelected (boolean select){
        selected = select;
    }

    boolean getSelected(){
        return selected;
    }

    boolean mouseOverSelected(){
        return (mouseX >= this.x) &&
                (mouseX<=this.x + this.w) &&
                (mouseY>= this.y +hBanner) &&
                (mouseY<= this.y + this.w+hBanner);
    }

}
```

Select2.pde

```
class Select2 {

  float x, y, w, h;          // Posició i dimensions
  String[] texts;            // Valors possibles
  String selectedValue;      // Valor Seleccionat

  boolean collapsed = true;  // Plegat / Desplegat
  boolean enabled;           // Abilitat / desabilitat

  float lineSpace = 15;      // Espai entre línies

  Select2(String[] texts, float x, float y, float w, float h){

    this.texts = texts;
    this.selectedValue = "";
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.enabled = true;
    this.collapsed = true;
  }

  void display(){
    pushStyle();
    stroke(0); strokeWeight(2); fill(255);
    rect(x, y, w, h);

    fill(100);
    rect(x + w - 30, y, 30, h);

    fill(0); stroke(0);
    triangle(x + w - 25, y+5, x + w - 15, y + 25, x + w - 5 , y+5);

    fill(0); textSize(14);
    text(selectedValue, x + 10, y + 20);

    if(!this.collapsed){

      fill(255); stroke(0);
      rect(x, y+h, w, (h + lineSpace)*texts.length);

      for(int i=0; i<texts.length; i++){

        if(i== clickedOption()){
          fill(200); noStroke();
          rect(x+4, y+4 + h + (h + lineSpace)*i - 2, w -8, h + lineSpace - 8);
        }
```

```
      fill(0);
      text(texts[i], x + 10, y + h + 25 + (h + lineSpace)*i);
    }
  }
  popStyle();
 }

 void setCollapsed(boolean b){
  this.collapsed = b;
 }

 void toggle(){
  this.collapsed = !this.collapsed;
 }


 void update(){
  int option = clickedOption();
  selectedValue = texts[option];
 }

// Indica si el cursor està sobre el select
boolean mouseOverSelect2(){
  if(this.collapsed){
    return (mouseX >= x) &&
           (mouseX <= x + w) &&
           ((mouseY-hBanner) >= y) &&
           ((mouseY-hBanner) <= y + h);
  }
  else {
    return (mouseX>= x) &&
           (mouseX<= (x+w)) &&
           ((mouseY-hBanner)>= y) &&
           ((mouseY-hBanner)<= y + h + (h + lineSpace)*texts.length);
  }
}

int clickedOption(){
  int i = (int)map(mouseY-hBanner, y + h, y + h + (h + lineSpace)*texts.length,
                   0, texts.length);
  return i;
}

}
```

RoundButton.pde

```
// Classe Botó
```

```
class RoundButton {
 // Propietats d'un botó:
float x, y, r;  // Posició i dimensions
// Colors de contorn, farciment, actiu i desactiu
color fillColor, strokeColor, fillColorOver, fillColorDisabled;
PImage icona;       // Icona del botó
boolean enabled;    // Abilitat / desabilitat
// Mètode Constructor
RoundButton(PImage icona, float x, float y, float r){
  this.icona = icona;
  this.x = x;
  this.y = y;
  this.r = r;
  this.enabled = true;
  fillColor = color(155);
  fillColorOver = color(155, 55, 55);
  fillColorDisabled = color(150);
  strokeColor = color(0);
}
// Setters
void setEnabled(boolean b){
  this.enabled = b;
}

boolean enability(){
  return enabled;
}
// Dibuixa el botó
void display(){
  pushStyle();
  if(!enabled){
    fill(fillColorDisabled);  // Color desabilitat
  }
  else if(mouseOverButton()){
    fill(fillColorOver);      // Color quan ratolí a sobre
  }
  else{
    fill(fillColor);          // Color actiu però ratolí fora
  }
  stroke(strokeColor); strokeWeight(3);          //Color i gruixa del contorn
  ellipse(this.x, this.y, 2*this.r, 2*this.r);   // Cercle del botó

  // Icona del botó
  imageMode(CENTER);
  if(icona!=null){
    image(icona, this.x, this.y, 2*this.r, 2*this.r);
  }
  popStyle();
}
// Indica si el cursor està sobre el botó
boolean mouseOverButton(){
  return dist(mouseX, mouseY, this.x, this.y)<= this.r;
```

```
}
 }
```

sobreNosotros.pde

```
void sobreNosotros(){
    pushStyle();
        fill(royalBlueDark, j);

        textMode(CENTER);
        textLeading(5);
        logoDrawing(width/2, height/2-250, 200, "KEEP THE BALL\nFLYING", c3, c4);

        fill(0);
        text("Finalizado el 6 de Abril de 2022", width/2 ,2*height/3);
    popStyle();
}
```

Table.pde

```
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: "+(this.numPage+1)+" / "+(this.numTotalPages+1), x, y + h + 50);
  popStyle();
 }
}
```

template.pde

```
color c3;
color c4;


//creació de la plantilla: banner i espai buit
void template(int j) {
 pushStyle();
   textSize(25);
   textAlign(LEFT);
   background(platinum);

   //banner
   fill(blackCoral, j);
   noStroke();
   rect(marginW, marginH, width-2*marginW, hBanner);

   c3 = color(251, 54, 64, j);
```

```
    c4 = color(10, 36, 99, j);

    if(pantalla!=0){
      logoDrawing(160, 44, 30, "keep the ball flying", c3, c4);
    }

    fill(255,255,255, j);

    text("Mi equipo", 250 +(width- 200)/6, hBanner/2+10);
    text("Pizarra", 250 +2*(width- 200)/6, hBanner/2+10);
    text("Resultados", 250 +3*(width- 200)/6, hBanner/2+10);
    text("Guardados", 250 +4*(width- 200)/6, hBanner/2+10);
    text("Sobre nosotros", 250 +5*(width- 200)/6, hBanner/2+10);

  popStyle();
  pushStyle();

  rectMode(CORNER);

  goToMiEquipo.display();
  goToPizarra.display();
  goToResultados.display();
  goToGuardados.display();
  goToSobreNosotros.display();
  popStyle();

  //zona libre

  fill(200, 200, 200,j);
  rect(marginW, hBanner+marginH, width-2*marginW, height - 2*marginH - hBanner);
}

//método para la rotulación de logotipo
void logoDrawing(float x, float y, int z, String texto, color c1, color c2){
  pushStyle();
    pushMatrix();

        textAlign(CENTER);
        textFont(fuente2);
        textSize(z);
        fill(c1);

        if(pantalla!=0){
          text(texto,x, y);
          fill(255);
          text(texto,x+1,y+1);
          fill(c2);
          text(texto,x+2,y+2);
        } else{
          text(texto,x, y);
          fill(255,j);
          text(texto,x+0.02*z,y+0.03*z);
```

```
        fill(c2);
        text(texto,x+2*0.02*z,y+2*0.02*z);
      }
   popMatrix();
  popStyle();
}
```

textField.pde

```
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-hBanner >= this.y  && mouseY-hBanner <= 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;
    }
  }
}
```

v06.pde

```
void setup() {

  connectBBDD();
  String [][] infoJugadores = getInfoTablaJugadores();
  String [][] infoEquipo    = getInfoTablaEquipo();
  infoPartido   = getInfoTablaPartido();
  infoPosicion = getInfoPosicion();

  // creacion de objetos
  p = new Pines((float)50, (float)280, (float)700,(float)700, (float)100, (float)width-100);

  img0 = loadImage("pizarra.jpg");
  img1 = loadImage("sobreNosotros1.jpg");
  img2 = loadImage("sobreNosotros2.jpg");
  img3 = loadImage("sobreNosotros3.jpg");
  usuario = loadImage("userIcon.png");

  s1 = new Select2(infoPosicion, 550, 475, (float)selectW, (float)selectH);
  s2 = new Select2(infoPosicion, 1200, 475, (float)selectW, (float)selectH);

  //botones
  goToMiEquipo      =  new Button("", 150 + 1*(width- 120)/6, hBanner/5, 150 + 2*(width- 120)/6,
hButton, false);
  goToPizarra       =  new Button("", 150 + 2*(width- 120)/6, hBanner/5, 150 + 3*(width- 120)/6,
hButton, false);
  goToResultados    =  new Button("", 150 + 3*(width- 120)/6, hBanner/5, 150 + 4*(width- 120)/6,
hButton, false);
  goToGuardados     =  new Button("", 150 + 4*(width- 120)/6, hBanner/5, 150 + 5*(width- 120)/6,
hButton, false);
  goToSobreNosotros =  new Button("", 150 + 5*(width- 120)/6, hBanner/5, 150 + 6*(width- 120)/6,
hButton, false);

  b1 = new Button("NEXT", 25 + tableW/2 + wButton/1.5, height - hButton - 20, wButton, hButton,
true);
  b2 = new Button("PREV", 25 + tableW/2 - wButton/1.5, height - hButton - 20, wButton, hButton,
true);

  next = new Button("NEXT", width- 150 - 25, height - 55, 150, 50, true);
```

```
prev = new Button("PREV", 25, height - 55, 150, 50, true);

save          =  new Button("S", width - 45,  10 + hBanner, 40 ,40, true );
newPlayer     =  new Button("NP", width - 45,  10 + hBanner, 40 ,40, true );
close         =  new Button("Cerrar", width/2, height - 200, 150, 50, true);

guardarJugador =  new Button("GUARDAR", width- 150 - 25, height - 55, 150, 50, true);
reset         =  new Button("Reset", 50, height - hButton - 20,wButton, hButton, true);
entrenamiento  =  new Select("Modo entrenamiento",50, 70 ,  true,   20);
competicion    =  new Select("Modo competición",  50, 100 , false,  20);
mediaPista     =  new Select("Media pista",       50, 150 , false,  20);
pistaEntera    =  new Select("Pista entera",      50, 180 , true,   20);

guardado1      =  new Guardado(img0, "k1", 17, 04, 2021);

// Creació de la taula -- no funciona
t = new Table(files, columnes);
t.setHeaders(headers);
t.setData(infoJugadores);
t.setColumnWidths(colWidths);

//creació jugadors
p1   = new Player("Pere Joan", "Libero", "Col", 10, 2001, 16, 2);
p2   = new Player("Pere Joan", "Central", "Punta", 10, 2001, 16, 2);
p3   = new Player("Pere Joan", "Opuesto", "Central", 10, 2001, 16, 2);

cono = loadImage("cono.png");

fuente1 = createFont("HKGrotesk-Bold.otf", 40);
fuente2 = createFont("Coco-Sharp-Extrabold-trial.ttf", 40);
fuente3 = createFont("Coco-Sharp-Heavy-Italic-trial.ttf", 40);
fuente4 = createFont("Keiner-SemiBold.ttf", 25);

nSaved = new TextField(550, 400, 200, 35);
dSaved = new TextField(800, 450, 50, 35);
mSaved = new TextField(870, 450, 50, 35);
ySaved = new TextField(940, 450, 50, 35);

nombre    = new TextField(450, 175, 700, alturaCelda-10);
dayn      = new TextField(520 + 120-50, 350 - (int)hBanner, 3/2*alturaCelda, alturaCelda);
monthn    = new TextField(520 + 120   , 350 - (int)hBanner, 3/2*alturaCelda, alturaCelda);
yearn     = new TextField(520 + 170   , 350 - (int)hBanner, 3/2*alturaCelda, alturaCelda);
ndorsal   = new TextField(520, 451 - (int)hBanner, alturaCelda, alturaCelda);

r1 = new Resultado((width+ 20)/6, height/2 - 50, 0);
r2 = new Resultado(3*(width+20)/6,height/2 - 50 , 1);
r3 = new Resultado(5*(width+20)/6, height/2 - 50, 2);
r4 = new Resultado((width+20)/6, height/2 - 50, 3);
r5 = new Resultado(3*(width+20)/6,height/2 - 50 , 4);
r6 = new Resultado(5*(width+20)/6, height/2 - 50, 5);

//array de resultados para el posterior recorrimiento
```

```
  Resultado [] arrResultados = {r1, r2, r3, r4, r5, r6};

  cono1 = new Cone (width-45, height - hBanner - 70);

  e = new ListaPaginada(arrResultados,10, 10, width- 100, height*9/10);

  //size(1200, 780);
  size(1300,900);
  fullScreen();
  noStroke();
  textAlign(CENTER);
  textSize(18);
  textFont(fuente4);
}


void draw() {

  if(frameCount<75){
    inicio();
  } else if(frameCount<125){
    fill(226,226, 240, i);
    rect(0,0,width, height);
    i++;
  } else{
    template(j);
    pushMatrix();
      translate(marginW, hBanner+marginH);
      pantallas(j);
    popMatrix();

    //println(pantalla);
    mousePointer();
    updateCursor();

    if(j<255){
      j+=25;
    }
  }
}

void pantallas (int j){
  //println(pantalla);
  if (pantalla==-1) { //paleta
    paleta();
  }
  if (pantalla==0) { //entrada
    p0(j);
  }
  if (pantalla==1) { //mi equipo
    p1();
  }
```

```
  if (pantalla==2) {
    p2();
  }
  if (pantalla==3) {
    resultados();
  }
  if (pantalla==4) {
    guardados();
  }
  if (pantalla==5) {
    sobreNosotros();
  }
  if (pantalla==6) {
    p1.display();
  }
  if (pantalla==7) {
    newPlayer();
  }
}

void keyPressed() {
   nSaved.keyPressed(key, (int)keyCode);
  dSaved.keyPressed(key, (int)keyCode);
  mSaved.keyPressed(key, (int)keyCode);
  ySaved.keyPressed(key, (int)keyCode);
  nombre.keyPressed(key, (int)keyCode);
  dayn.keyPressed(key, (int)keyCode);
  yearn.keyPressed(key, (int)keyCode);
  monthn.keyPressed(key, (int)keyCode);
  ndorsal.keyPressed(key, (int)keyCode);

}

void mousePointer(){
  if(mousePressed){
    println("mouse X: "+ mouseX);
    println("mouse Y: "+ mouseY);
  }
}

void updateCursor(){
  if((goToMiEquipo.mouseOverButton() && goToMiEquipo.enabled)||
    (goToGuardados.mouseOverButton() && goToGuardados.enabled)||
    (goToPizarra.mouseOverButton() && goToPizarra.enabled) ||
    mouseOverObject(0,0,0,0) || save.mouseOverButton() ||
    (b1.mouseOverButton() && b1.enabled)||
    (b2.mouseOverButton() && b2.enabled)||
    mouseOverObject(width-45, height - 70, 35, 45) ||
    (s1.mouseOverSelect2() && s1.enabled)||
    (s2.mouseOverSelect2() && s2.enabled) ||
    nombre.mouseOverTextField() ||
    nSaved.mouseOverTextField() ||
```

```
    dSaved.mouseOverTextField() ||
    mSaved.mouseOverTextField() ||
    ySaved.mouseOverTextField() ||
    nombre.mouseOverTextField() ||
    dayn.mouseOverTextField() ||
    yearn.mouseOverTextField() ||
    monthn.mouseOverTextField() ||
    ndorsal.mouseOverTextField()
    ){
      cursor(HAND);
  } else {
    cursor(ARROW);
  }
}

void mousePressed(){
  if(goToMiEquipo.mouseOverButton() && mousePressed){
    pantalla = 1;
  }
  if(goToPizarra.mouseOverButton()  && mousePressed){
    pantalla = 2;
  }
  if(goToResultados.mouseOverButton() && mousePressed){
    pantalla = 3;
  }
  if(goToGuardados.mouseOverButton() && mousePressed){
    pantalla = 4;
  }
  if(goToSobreNosotros.mouseOverButton() && mousePressed){
    pantalla = 5;
  }
  if(mouseOverLogo()&& mousePressed){
    pantalla = 0;
  }
  if(b1.mouseOverButton() && b1.enabled){
    t.nextPage();
  }
  else if(b2.mouseOverButton() && b2.enabled){
    t.prevPage();
  }
  nombre.isPressed();
  dayn.isPressed();
  monthn.isPressed();
  yearn.isPressed();
  ndorsal.isPressed();
  nSaved.isPressed();
  ySaved.isPressed();
  dSaved.isPressed();

  if(newPlayer.mouseOverButton() && mousePressed){
    pantalla=7;
    //nuevoJugador1.setDisp(true);
```

```
  //println("El pop up new player se encuentra en: "+ nuevoJugador1.getDisplay());
}
if(close.mouseOverButton() && mousePressed){
  //nuevoJugador1.setDisp(false);
}

if(prev.mouseOverButton() && mousePressed && prev.enability()){
  e.prev();
}
if(next.mouseOverButton() && mousePressed && next.enability()){
  println("mouse over next");
  e.next();
}
if(reset.mouseOverButton()&&mousePressed){
  p.resetPinPositions();
}

if(entrenamiento.mouseOverSelected()&&mousePressed){
  println("ENTRENAMIENTO PRESSED");
  entrenamiento.setSelected(true);
  competicion.setSelected(!true);
}
if(competicion.mouseOverSelected()&&mousePressed){
  competicion.setSelected(true);
  entrenamiento.setSelected(!true);
}

if(mediaPista.mouseOverSelected()&&mousePressed){
  println("mediaPista PRESSED");
  mediaPista.setSelected(true);
  pistaEntera.setSelected(!true);
}
if(pistaEntera.mouseOverSelected()&&mousePressed){
  pistaEntera.setSelected(true);
  mediaPista.setSelected(!true);
}
nombre.isPressed();

if(s1.mouseOverSelect2() && s1.enabled){
if(!s1.collapsed){
  s1.update();      // Actualitzar valor
}
s1.toggle();        // Plegar o desplegar
}

if(s2.mouseOverSelect2() && s2.enabled){
if(!s2.collapsed){
  s2.update();      // Actualitzar valor
}
s2.toggle();        // Plegar o desplegar
}
```

```
    if(mousePressed && guardarJugador.mouseOverButton()){

    if(s1.selectedValue=="Colocador"){
      s = "1";
    }
      insertaJugador(nombre.text, ndorsal.text, s);
    }
  }

  boolean mouseOverLogo(){
    boolean a = (mouseX >= marginW+10) &&
          (mouseX<=303) &&
          (mouseY>= 19) &&
          (mouseY<= 50);
    if(a){
      cursor(HAND);
    }else{cursor(ARROW);}
    return a ;
  }

// Modificar el número según Select 2
void updateNumber(){
  n = Integer.parseInt(s1.selectedValue);
}


void insertaJugador(String n, String d, String s){

  try {
       String q = " INSERT INTO `jugador` (`id`, `nombre`, `dorsal`, `equipo`, `posicion_id`) VALUES
(NULL, '"+n+"', '"+d+"', 1, '"+s+"')";
       println("INSERT: "+q);
       query.execute(q);
       println("INSERT OK :)");
    }
    catch(Exception e) {
       System.out.println(e);
    }
}
```