Cristo_de_la_SalApp

```
// Librería de MySQL i Processing
import de.bezier.data.sql.*;
import java.util.Random;


//enumeración de las pantallas de la aplicación
enum PANTALLA {
 INICIO, PRINCIPAL, CENSO, CONTABILIDAD, ARCHIVO, AVISOS, ENLACES, CENSO_DETALLE,
CENSO_NUEVOHERMANO,
   CONTABILIDAD_BALANCE, CONTABILIDAD_PRESUPUESTO, CONTABILIDAD_AÑADIRCONCEPTO,
CONTABILIDAD_DETALLEBALANCE,
   CONTABILIDAD_DETALLEMOVIMIENTO, ARCHIVO_NUEVO, ARCHIVO_DETALLE, AVISOS_NUEVOAVISO,
AVISOS_NUEVOEVENTO,
   AVISOS_DETALLEAVISO, AVISOS_DETALLEEVENTO;
};

///Pantalla actual
PANTALLA pantalla =PANTALLA.CONTABILIDAD_DETALLEBALANCE;

boolean logged= false;

String userNameAdmin = "admin";
String userNameUser = "user";

boolean admin= true;


float estadoDeCuentas=3.0;
int lastKeyCodePressed;


void setup() {
 size(1280, 800);
 connexionBBDD();
 setColors();
 setFonts();
 setMedias();
 setGUI();


 desktop = Desktop.getDesktop();
}

void draw() {
 //Establece una configuración por defecto
 textAlign(LEFT);
 fill(0);
 textFont(getFontAt(4));


 // Dibuja la pantalla correspondiente
 switch(pantalla) {
 case INICIO:
   dibujaPantallaInicio();
   break;
 case PRINCIPAL:
   dibujaPantallaPrincipal();
   break;
 case CENSO:
   dibujaPantallaCenso();
   break;
 case CONTABILIDAD:
   dibujaPantallaContabilidad();
   break;
```

```
 case ARCHIVO:
   dibujaPantallaArchivo();
   break;
 case AVISOS:
   dibujaPantallaAvisos();
   break;
 case ENLACES:
   dibujaPantallaEnlaces();
   break;
 case CENSO_DETALLE:
   dibujaPantallaCensoDetalle();
   break;
 case CENSO_NUEVOHERMANO:
   dibujaPantallaCensoNuevoHermano();
   break;
 case CONTABILIDAD_BALANCE:
   dibujaPantallaContabilidadBalance();
   break;
 case CONTABILIDAD_PRESUPUESTO:
   dibujaPantallaContabilidadPresupuesto();
   break;
 case CONTABILIDAD_AÑADIRCONCEPTO:
   dibujaPantallaContabilidadAñadirConcepto();
   break;
 case CONTABILIDAD_DETALLEBALANCE:
   dibujaPantallaContabilidadDetalleBalance();
   break;
 case CONTABILIDAD_DETALLEMOVIMIENTO:
   dibujaPantallaContabilidadDetalleMovimiento();
   break;
 case ARCHIVO_NUEVO:
   dibujaPantallaArchivoNuevo();
   break;
 case ARCHIVO_DETALLE:
   dibujaPantallaArchivoDetalle();
   break;
 case AVISOS_NUEVOAVISO:
   dibujaPantallaAvisosNuevoAviso();
   break;
 case AVISOS_NUEVOEVENTO:
   dibujaPantallaAvisosNuevoEvento();
   break;
 case AVISOS_DETALLEAVISO:
   dibujaPantallaAvisosDetalleAviso();
   break;
 case AVISOS_DETALLEEVENTO:
   dibujaPantallaAvisosDetalleEvento();
   break;
 }

 updateCursor();   // Modifica la apariencia del cursor
}
// Comprova si el login és correcte
boolean comprovaLogin() {
 return isValidated(userText.getValue(), passText.getValue() );
}

boolean comprovaAdmin() {
 return isAdmin(userText.getValue());
}
```

BarsDiagram

```
class BarsDiagram {
```

```
// Dimensiones del diagrama de Barras
float x, y, w, h;

// Información del diagrama (textos, valores y colores)
String[] texts;
float[] values;
float[] percentages;
color[] colors;

// Suma total de los valores
float total;

// Constructor

BarsDiagram(float x, float y, float w, float h) {
  this.x = x;
  this.y = y;
  this.w = w;
  this.h = h;
}

// Setters

void setTexts(String[] t) {
  this.texts = t;
}

void setValues(float[] v) {
  this.values = v;
  this.total = 0;
  for (int i=0; i<values.length; i++) {
    this.total += this.values[i];
  }

  this.percentages = new float[values.length];
  for (int i=0; i<percentages.length; i++) {
    this.percentages[i] = (this.values[i] / this.total)*100;
  }
}

void setColors(color[] c) {
  this.colors = c;
}

// Dibuja el Diagrama de Sectores

void display() {
  pushStyle();

  float widthBar = w / (float) this.values.length;

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

    float barValue = (this.values[i] / this.total)*h;
    float xBar = this.x + widthBar*i;

    fill(colors[i]);
    stroke(0);
    strokeWeight(5);
    rect(xBar, this.y + this.h - barValue, widthBar, barValue);

    float textX = xBar + widthBar/2;
    float textY = this.y + this.h + 50;
    fill(0);
```

```
    textAlign(CENTER);
    textSize(24);
    text(this.texts[i], textX, textY);

    float percX = xBar + widthBar/2;
    float percY = this.y + this.h - barValue - 50;
    String percentage = nf(this.percentages[i], 2, 2);
    fill(0);
    textAlign(CENTER);
    textSize(18);
    text(percentage+"%", percX, percY);

    textSize(24);
    text((int)this.values[i], percX, percY - 30);
  }
  popStyle();
 }
}
```

Button

```
// Clase Botón

public class Button {

 // propiedades de los botones
 float x, y, w, h; // posición y dimensión

 // Colores de contorno, fill, activo i desactivado
 color fillColor, strokeColor;
 color fillColorOver;

 String textBoton; // Texto
 boolean enabled; // Habilitado / deshabilitado

 int tf=4; // text font

 //Constructor
 Button(String text, float x, float y, float w, float h) {
   this.textBoton = text;
   this.x = x;
   this.y = y;
   this.w = w;
   this.h = h;
   this.enabled = true;
   fillColor = color(getColorAt(1));
   fillColorOver = color(getColorAt(0));
   strokeColor = color(0);
 }

 // Setters

 void setEnabled(boolean b) {
   this.enabled = b;
 }

 void setTextFont(int i) {
   this.tf= i;
 }

 // Dibujar el botón
 void display() {
   pushStyle();
   if (mouseOverButton()) {
     fill(fillColorOver); // Color cuando el mouse está encima
   } else {
```

```
      fill(fillColor); // Color activo sin mouse
    }
    stroke(strokeColor);
    strokeWeight(1); //Color i grosor del contorno
    rect(this.x, this.y, this.w, this.h, 10); // Rectangulo del botón

    // Texto (color, alineación i tamaño)
    fill(255);
    textAlign(CENTER);
    textFont(getFontAt(tf));
    text(textBoton, this.x + this.w/2, this.y + this.h/2+5);
    popStyle();
  }

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

CalendariPlus

```
import java.util.Calendar;

class CalendariPlus {

  // Textos representatius dels mesos
  String[] months = {"Jan", "Feb", "Mar", "Apr", "May", "Jun",
    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};

  // Informació del calendari
  int año, mes, dia;
  int numDaysMonth, numDaysPrevMonth;
  int dayOfWeek, firstDay;

  // Data seleccionada
  boolean dateSelected = false;
  int selectedDay=0, selectedMonth=0, selectedYear=0;

  // Calendari actual, i del mes anterior
  Calendar cal, cPrev;

  // Botons del calendari
  DayButton[] buttons;
  Button bNext, bPrev, bOK;

  // Dimensions del calendari
  int x, y, w, h;

  // Visibilitat del calendari
  boolean visible = false;


  // Constructor
  CalendariPlus(int x, int y, int w, int h) {


    this.buttons = new DayButton[37];

    this.cal = Calendar.getInstance();
    cal.set(Calendar.DAY_OF_MONTH, 1);

    this.año = cal.get(Calendar.YEAR);
```

```
  this.mes = cal.get(Calendar.MONTH) + 1;
  this.dia = cal.get(Calendar.DATE);

  this.numDaysMonth = cal.getActualMaximum(Calendar.DAY_OF_MONTH);

  this.dayOfWeek = cal.get(Calendar.DAY_OF_WEEK);
  if (dayOfWeek==Calendar.SUNDAY) {
    this.dayOfWeek = 6;
  } else {
    this.dayOfWeek = this.dayOfWeek - 2;
  }

  cal.set(Calendar.DAY_OF_WEEK, cal.getFirstDayOfWeek());
  this.firstDay = cal.get(Calendar.DATE);

  cPrev = Calendar.getInstance();
  setPrevCalendar(1, this.mes-2, this.año);

  this.numDaysPrevMonth = cPrev.getActualMaximum(Calendar.DAY_OF_MONTH);

  this.x = x;
  this.y = y;
  this.w = w;
  this.h = h;
  createCalendar(x, y, w, h);

  bNext = new Button("Siguiente", x+ w/3, y -70, 100, 50);
  bPrev = new Button("Anterior", x+w/3+100, y - 70, 100, 50);
  bOK   = new Button("OK", x+w/3+200, y - 70, 50, 50);
}

// Setters

void setCalendar(int d, int m, int y) {
  cal.set(Calendar.YEAR, y);
  cal.set(Calendar.MONTH, m);
  cal.set(Calendar.DATE, d);
}

void setPrevCalendar(int d, int m, int y) {
  cPrev.set(Calendar.YEAR, y);
  cPrev.set(Calendar.MONTH, m);
  cPrev.set(Calendar.DATE, d);
}

void setSelectedDate(int d, int m, int y) {
  this.selectedDay = d;
  this.selectedMonth = m;
  this.selectedYear = y;
}

// Va un mes enrera en el Calendari
void prevMonth() {

  this.buttons = new DayButton[37];

  this.mes --;
  if (this.mes==0) {
    this.mes = 12;
    this.año--;
  }
  setCalendar(this.dia, this.mes -1, this.año);

  this.numDaysMonth = cal.getActualMaximum(Calendar.DAY_OF_MONTH);
```

```
  this.dayOfWeek = cal.get(Calendar.DAY_OF_WEEK);
  if (dayOfWeek==Calendar.SUNDAY) {
    this.dayOfWeek = 6;
  } else {
    this.dayOfWeek = this.dayOfWeek - 2;
  }

  cal.set(Calendar.DAY_OF_WEEK, cal.getFirstDayOfWeek());
  this.firstDay = cal.get(Calendar.DATE);

  setPrevCalendar(1, this.mes -2, this.año);
  this.numDaysPrevMonth = cPrev.getActualMaximum(Calendar.DAY_OF_MONTH);

  createCalendar(x, y, w, h);
 }

 void createCalendar(int x, int y, int w, int h) {

  float dayWidth = w / 7;
  float dayHeight = h / 6;
  int numDia = 1;
  int f = 0, nb = 0;

  while (numDia<=numDaysMonth) {

    if (firstDay!=1 && f==0) {
      int cPrev=0;
      for (int p=firstDay, c=0; p<=numDaysPrevMonth; p++, c++) {
        buttons[nb] = new DayButton(x + c*dayWidth, y + f*dayHeight, dayWidth, dayHeight,
p, mes, año);
        buttons[nb].setEnabled(false);
        cPrev++;
        nb++;
      }
      for (int c=cPrev; c<7; c++) {
        buttons[nb] = new DayButton(x + c*dayWidth, y + f*dayHeight, dayWidth, dayHeight,
numDia, mes, año);
        numDia++;
        nb++;
      }
      f++;
    } else {
      for (int c=0; c<7; c++) {
        buttons[nb] = new DayButton(x + c*dayWidth, y + f*dayHeight, dayWidth, dayHeight,
numDia, mes, año);
        numDia++;
        nb++;
        if (numDia>numDaysMonth) {
          break;
        }
      }
      f++;
    }
  }
 }

 // Va un mes endavant en el calendari
 void nextMonth() {

  this.buttons = new DayButton[37];

  this.mes ++;
  if (this.mes==13) {
    this.mes = 1;
    this.año++;
```

```
  }
  setCalendar(this.dia, this.mes-1, this.año);

  this.numDaysMonth = cal.getActualMaximum(Calendar.DAY_OF_MONTH);

  this.dayOfWeek = cal.get(Calendar.DAY_OF_WEEK);
  if (dayOfWeek==Calendar.SUNDAY) {
    this.dayOfWeek = 6;
  } else {
    this.dayOfWeek = this.dayOfWeek - 2;
  }

  cal.set(Calendar.DAY_OF_WEEK, cal.getFirstDayOfWeek());
  this.firstDay = cal.get(Calendar.DATE);

  setPrevCalendar(1, this.mes-2, this.año);

  this.numDaysPrevMonth = cPrev.getActualMaximum(Calendar.DAY_OF_MONTH);

  createCalendar(x, y, w, h);
}



// Dibuixa el Calendari
void display() {
  if (visible) {
    pushStyle();

    fill(255); noStroke();
    rect(x, y-80, w, h);

    fill(0);
    textSize(36);
    textAlign(LEFT);
    text(months[mes-1]+"/"+año, x, y - 30);
    for (DayButton b : buttons) {
      if (b!=null) {
        b.display();
      }
    }

    if (dateSelected) {
      String dateText = this.selectedDay+"/"+this.selectedMonth+"/"+this.selectedYear;
      fill(0);
      textSize(24);
      textAlign(RIGHT);
      text(dateText, x+w, y - 30);
    }

    // Dibuixa els botons
    bNext.display();
    bPrev.display();
    bOK.display();
    popStyle();
  }


}


// Comprova si pitjam sobre els botons del Calendari
void checkButtons() {
  for (DayButton b : buttons) {
    if ((b!=null)&&(b.enabled)&&(b.mouseOver())) {
```

```
        boolean prevState = b.selected;
        deselectAll();
        b.setSelected(!prevState);
        if (b.selected) {
          dateSelected = true;
          setSelectedDate(b.dia, b.mes, b.año);
        } else {
          dateSelected = false;
        }
      }
    }
  }

  // Deselecciona tots els botons del Calendari
  void deselectAll() {
    for (DayButton b : buttons) {
      if (b!=null) {
        b.setSelected(false);
      }
    }
  }
}
```

Calendario

```
import java.util.Calendar;

class Calendario {

  // Textos representatius dels mesos
  String[] months = {"Jan", "Feb", "Mar", "Apr", "May", "Jun",
    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};

  // Informació del calendari
  int año, mes, dia;
  int numDaysMonth, numDaysPrevMonth;
  int dayOfWeek, firstDay;

  // Data seleccionada
  boolean dateSelected = false;
  int selectedDay=0, selectedMonth=0, selectedYear=0;

  // Calendari actual, i del mes anterior
  Calendar cal, cPrev;

  // Botons del calendari
  DayButton[] buttons;

  // Dimensions del calendari
  int x, y, w, h;

  // Special dates & info
  String[][] specialDates;


  // Constructor
  Calendario(int x, int y, int w, int h, String[][] fechasClave) {

    this.buttons = new DayButton[37];

    this.cal = Calendar.getInstance();
    cal.set(Calendar.DAY_OF_MONTH, 1);

    this.año = cal.get(Calendar.YEAR);
    this.mes = cal.get(Calendar.MONTH) + 1;
    this.dia = cal.get(Calendar.DATE);
```

```
  this.numDaysMonth = cal.getActualMaximum(Calendar.DAY_OF_MONTH);

  this.dayOfWeek = cal.get(Calendar.DAY_OF_WEEK);
  if (dayOfWeek==Calendar.SUNDAY) {
    this.dayOfWeek = 6;
  } else {
    this.dayOfWeek = this.dayOfWeek - 2;
  }

  cal.set(Calendar.DAY_OF_WEEK, cal.getFirstDayOfWeek());
  this.firstDay = cal.get(Calendar.DATE);

  cPrev = Calendar.getInstance();
  setPrevCalendar(1, this.mes-2, this.año);

  this.numDaysPrevMonth = cPrev.getActualMaximum(Calendar.DAY_OF_MONTH);

  this.x = x;
  this.y = y;
  this.w = w;
  this.h = h;

  setSpecialDates(fechasClave);
  createCalendar(x, y, w, h);
}

// Setters

void setCalendar(int d, int m, int y) {
  cal.set(Calendar.YEAR, y);
  cal.set(Calendar.MONTH, m);
  cal.set(Calendar.DATE, d);
}

void setPrevCalendar(int d, int m, int y) {
  cPrev.set(Calendar.YEAR, y);
  cPrev.set(Calendar.MONTH, m);
  cPrev.set(Calendar.DATE, d);
}

void setSelectedDate(int d, int m, int y) {
  this.selectedDay = d;
  this.selectedMonth = m;
  this.selectedYear = y;
}

// Assigna dies especials a mostrar en el calendari
void setSpecialDates(String[][] info) {
  this.specialDates = info;
  createCalendar(x, y, w, h);
}

String getSpecialDateInfo(String date) {
  for (int i=0; i<specialDates.length; i++) {
    if (this.specialDates[i][0].equals(date)) {
      return specialDates[i][1];
    }
  }
  return "";
}

boolean isSpecialDate(String date) {
  for (int i=0; i<specialDates.length; i++) {
    if (this.specialDates[i][0].equals(date)) {
```

```
        return true;
      }
    }
    return false;
  }

  // Va un mes enrera en el Calendari
  void prevMonth() {

    this.buttons = new DayButton[37];

    this.mes --;
    if (this.mes==0) {
      this.mes = 12;
      this.año--;
    }
    setCalendar(this.dia, this.mes -1, this.año);

    this.numDaysMonth = cal.getActualMaximum(Calendar.DAY_OF_MONTH);

    this.dayOfWeek = cal.get(Calendar.DAY_OF_WEEK);
    if (dayOfWeek==Calendar.SUNDAY) {
      this.dayOfWeek = 6;
    } else {
      this.dayOfWeek = this.dayOfWeek - 2;
    }

    cal.set(Calendar.DAY_OF_WEEK, cal.getFirstDayOfWeek());
    this.firstDay = cal.get(Calendar.DATE);

    setPrevCalendar(1, this.mes -2, this.año);
    this.numDaysPrevMonth = cPrev.getActualMaximum(Calendar.DAY_OF_MONTH);

    createCalendar(x, y, w, h);
  }

  void createCalendar(int x, int y, int w, int h) {

    float dayWidth = w / 7;
    float dayHeight = h / 6;
    int numDia = 1;
    int f = 0, nb = 0;

    this.buttons = new DayButton[37];

    while (numDia<=numDaysMonth) {

      if (firstDay!=1 && f==0) {
        int cPrev=0;
        for (int p=firstDay, c=0; p<=numDaysPrevMonth; p++, c++) {
          buttons[nb] = new DayButton(x + c*dayWidth, y + f*dayHeight, dayWidth, dayHeight,
p, mes, año);
          buttons[nb].setEnabled(false);
          cPrev++;
          nb++;
        }
        for (int c=cPrev; c<7; c++) {
          buttons[nb] = new DayButton(x + c*dayWidth, y + f*dayHeight, dayWidth, dayHeight,
numDia, mes, año);
          String d = (numDia<10) ? ("0"+numDia) : String.valueOf(numDia);
          String m = (mes<10) ? ("0"+mes) : String.valueOf(mes);
          if (isSpecialDate(año+"-"+m+"-"+d)) {
            buttons[nb].setSelected2(true);
            String info = getSpecialDateInfo(año+"-"+m+"-"+d);
            buttons[nb].setInfo(info);
```

```
        }
        numDia++;
        nb++;
      }
      f++;
    } else {
      for (int c=0; c<7; c++) {
        buttons[nb] = new DayButton(x + c*dayWidth, y + f*dayHeight, dayWidth, dayHeight,
numDia, mes, año);
        String d = (numDia<10) ? ("0"+numDia) : String.valueOf(numDia);
        String m = (mes<10) ? ("0"+mes) : String.valueOf(mes);
        if (isSpecialDate(año+"-"+m+"-"+d)) {
          buttons[nb].setSelected2(true);
          String info = getSpecialDateInfo(año+"-"+m+"-"+d);
          buttons[nb].setInfo(info);
        }
        numDia++;
        nb++;
        if (numDia>numDaysMonth) {
          break;
        }
      }
      f++;
    }
  }
}

// Va un mes endavant en el calendari
void nextMonth() {

  this.buttons = new DayButton[37];

  this.mes ++;
  if (this.mes==13) {
    this.mes = 1;
    this.año++;
  }
  setCalendar(this.dia, this.mes-1, this.año);

  this.numDaysMonth = cal.getActualMaximum(Calendar.DAY_OF_MONTH);

  this.dayOfWeek = cal.get(Calendar.DAY_OF_WEEK);
  if (dayOfWeek==Calendar.SUNDAY) {
    this.dayOfWeek = 6;
  } else {
    this.dayOfWeek = this.dayOfWeek - 2;
  }

  cal.set(Calendar.DAY_OF_WEEK, cal.getFirstDayOfWeek());
  this.firstDay = cal.get(Calendar.DATE);

  setPrevCalendar(1, this.mes-2, this.año);

  this.numDaysPrevMonth = cPrev.getActualMaximum(Calendar.DAY_OF_MONTH);

  createCalendar(x, y, w, h);
}



// Dibuixa el Calendari
void display() {
  pushMatrix();
  fill(0);
  textSize(36);
```

```
    textAlign(LEFT);
    text(months[mes-1]+"/"+año, x, y - 30);
    for (DayButton b : buttons) {
      if (b!=null) {
        b.display();
      }
    }

    if (dateSelected) {

      String dateText = this.selectedDay+"/"+this.selectedMonth+"/"+this.selectedYear;
      fill(0);
      textSize(24);
      textAlign(RIGHT);
      text(dateText, x+w, y - 30);
    }
    popMatrix();
  }


  // Comprova si pitjam sobre els botons del Calendari
  void checkButtons() {

    for (DayButton b : buttons) {
      if ((b!=null)&&(b.enabled)&&(b.mouseOver())) {
        boolean prevState = b.selected;
        deselectAll();
        b.setSelected(!prevState);
        if (b.selected) {
          dateSelected = true;
          setSelectedDate(b.dia, b.mes, b.año);
        } else {
          dateSelected = false;
        }
      }
    }
  }

  // Deselecciona tots els botons del Calendari
  void deselectAll() {
    for (DayButton b : buttons) {
      if (b!=null) {
        b.setSelected(false);
      }
    }
  }
}
```

Card

```
class Card {

// Propietats
String title;
String description;

// Dimensions
float x, y, w, h, b;

// Constructors

Card(){
}

Card(String title, String place, String date, String section, String description){
  this.title = title;
```

```
  this.description = description;
}

Card(String[] info){
  this.title = info[0];
  this.description = info[1];
}

//Setters

void setDimensions(float x, float y, float w, float h, float b){
  this.x = x; this.y = y;
  this.w = w; this.h = h;
  this.b = b;
}


// Dibuixa la Card

void display(boolean selectedCard){

  pushStyle();

  // Rectangle inferior
  stroke(0);
  if(selectedCard){
      fill(200, 100, 100);
  }
  else if(this.mouseOver()){
      fill(200);
  }
  else {
      fill(220);
  }
  rect(x, y, w, h, b/2);


  // Títol
  fill(0); textSize(24); textAlign(LEFT);
  text(title, x + 20, y + h/2 + 10);

  line(x + 145, y + 10, x + 145 , y + h - 10);

  // Descripció
  fill(0);textSize(20); textAlign(LEFT);
  text(description, x + 160, y + 10, w-170, h-20);


  popStyle();
}

boolean mouseOver(){
  return this.x < mouseX && mouseX < this.x + this.w &&
         this.y < mouseY && mouseY < this.y + this.h;
}


}
```

Colores

```
//Archivo con la información de los colores de la App.

// Array de colores
color[] colors;
```

```
// Establece los colores de la App
void setColors() {
 this.colors = new color[6];
 this.colors[0] = color(#401E3A);
 this.colors[1] = color(#610A0A);
 this.colors[2] = color(#960D0F);
 this.colors[3] = color(#D4AA7D);
 this.colors[4] = color(#F2E0C9);
 this.colors[5] = color(255);
}

// Getter del número de colores
int getNumColors() {
 return this.colors.length;
}

// Getter del color primario
color getFirstColor() {
 return this.colors[0];
}

// Getter del color secundario
color getSecondColor() {
 return this.colors[1];
}

// Getter del color terciario
color getThirdColor() {
 return this.colors[2];
}

// Getter del color i-éssimo
color getColorAt(int i) {
 return this.colors[i];
}
```

Copiar

```
import java.io.IOException;
import java.nio.file.*;

// Copia un fitxer a una altra ubicació
void copiar(String rutaOriginal, String rutaCopia, String titol) {
 Path original = Paths.get(rutaOriginal);
 Path copia    = Paths.get(rutaCopia+"/"+titol);
 try {
   Files.copy(original, copia);
   println("OK: archivo copiado en la carpeta.");
 }
 catch (IOException e) {
   println("ERROR: No s&apos;ha pogut copiar el fitxer.");
 }
}
```

DataBase

```
// Objecte de connexió a la BBDD
MySQL msql;

// Paràmetres de la connexió
String user = "admin2";
String pass = "12345";
String database = "cristo";


// Connexió
```

```
void connexionBBDD() {

 msql = new MySQL( this, "localhost:8889", database, user, pass );

 // Si la connexió s&apos;ha establert
 if (msql.connect()) {
   // La connexió s&apos;ha establert correctament
   println("Connexió establerta :)");
 } else {
   // La connexió ha fallat!!!
   println("Error de Connexió :(");
 }
}

boolean isValidated(String usuario, String password) {
 msql.query( "SELECT count(*) AS n FROM `user` WHERE `numhermano` = %s AND `password` LIKE
&apos;%s&apos;", usuario, password );
 msql.next();
 int numRows = msql.getInt("n");
 if (numRows == 1) {
   return true;
 } else {
   return false;
 }
}

boolean isAdmin(String usuario) {
 msql.query("SELECT count(*) AS n FROM `user` WHERE `role_id` = 1 AND `numhermano` = %s",
usuario );
 msql.next();
 int role = msql.getInt("n");
 if (role == 1) {
   return true;
 } else {
   return false;
 }
}

// Obté el número de files de la taula
int getNumRowsTabla(String nombreTabla) {
 msql.query( "SELECT COUNT(*) AS n FROM %s", nombreTabla );
 msql.next();
 int numRows = msql.getInt("n");
 return numRows;
}

// Obté el número de files de la query
int getNumRowsQuery(String q) {
 msql.query( q);
 msql.next();
 int numRows = msql.getInt("n");
 return numRows;
}

// Obté informació de la taula Unitat
String[][] getInfoTablaCenso() {

 int numRows = getNumRowsTabla("hermano");
 String[][] data = new String[numRows][5];

 int nr=0;
 msql.query( "SELECT * FROM hermano" );
 while (msql.next()) {
   data[nr][0] = String.valueOf(msql.getInt("user_numhermano"));
   data[nr][1] = msql.getString("nombre");
```

```
   data[nr][2] = msql.getString("apellidos");
   data[nr][3] = formataFecha(String.valueOf(msql.getDate("fechaalta")));
   data[nr][4] = String.valueOf(msql.getInt("telefono"));
   nr++;
 }
 return data;
}

// Obté informació de la taula Unitat
String[][] getInfoTablaCensoBuscar(String buscar) {

 String q2 = "SELECT COUNT(*) AS n FROM hermano WHERE apellidos LIKE &apos;%"+buscar+"%
&apos;";
 int numRows = getNumRowsQuery(q2);
 println("NR:"+numRows);

 if (numRows>0) {
   String[][] data = new String[numRows][5];

   String q = "SELECT * FROM hermano WHERE apellidos LIKE &apos;%"+buscar+"%&apos;";
   int nr=0;
   msql.query( q );
   while (msql.next()) {
     data[nr][0] = String.valueOf(msql.getInt("user_numhermano"));
     data[nr][1] = msql.getString("nombre");
     data[nr][2] = msql.getString("apellidos");
     data[nr][3] = formataFecha(String.valueOf(msql.getDate("fechaalta")));
     data[nr][4] = String.valueOf(msql.getInt("telefono"));
     nr++;
   }
   return data;
 } else {
   String[][] array = new String[5][5];

   // Rellenar el array con strings vacíos
   for (int i = 0; i < 5; i++) {
     for (int j = 0; j < 5; j++) {
       array[i][j] = "";
     }
   }

   return array;
 }
}

// Obté informació de la taula Hermano
String[] getInfoTablaHermano(String idHermano) {

 String[] data = new String[22];

 msql.query( "SELECT * FROM hermano WHERE user_numhermano=&apos;"+idHermano+"&apos;" );
 msql.next();
 data[0] = String.valueOf(msql.getInt("user_numhermano"));
 titulo1 = data[0]+".pdf";
 data[1] = String.valueOf(msql.getInt("user_role_id"));
 data[2] = msql.getString("nombre");
 data[3] = msql.getString("apellidos");
 data[4] = formataFecha(String.valueOf(msql.getDate("fechanacimiento")));
 data[5] = msql.getString("dni");
 data[6] = msql.getString("calle");
 data[7] = String.valueOf(msql.getInt("numerodireccion"));
 data[8] = msql.getString("piso");
 data[9] = msql.getString("localidad");
 data[10] = msql.getString("provincia");
 data[11] = msql.getString("telefono");
```

```
 data[12] = msql.getString("correoelectronico");
 data[13] = msql.getString("banco");
 data[14] = msql.getString("titular");
 data[15] = msql.getString("dnititular");
 data[16] = msql.getString("iban");
 data[17] = msql.getString("entidad");
 data[18] = msql.getString("oficina");
 data[19] = msql.getString("digitocontrol");
 data[20] = msql.getString("numerocuenta");
 data[21] = formataFecha(String.valueOf(msql.getDate("fechaalta")));
 return data;
}


String formataFecha2(String fechaEntrada) {

 String dia = fechaEntrada.split("/")[0];
 String mes = fechaEntrada.split("/")[1];
 String ano = fechaEntrada.split("/")[2];

 return ano+"-"+mes+"-"+dia;
}

String formataFecha(String fechaEntrada) {

 String ano = fechaEntrada.split("-")[0];
 String mes = fechaEntrada.split("-")[1];
 String dia = fechaEntrada.split("-")[2];

 return dia+"/"+mes+"/"+ano;
}


void insertInfoTablaHermano(String nombre, String apellidos, String fechanacimiento, String
dni, String calle, String numerodireccion, String piso, String localidad, String provincia,
String telefono, String correoelectronico, String banco, String titular, String dnititular,
String iban, String entidad, String oficina, String digitocontrol, String numerocuenta,
String fechaalta) {
 String numHermano = String.valueOf(getNumeroUltimoHermano()+1);
 // Insertar el nuevo registro en la tabla &apos;user&apos;
 String password = generatePassword(6);
 String q2 = "INSERT INTO user (numhermano, password, role_id) VALUES (&apos;"+numHermano+"
&apos;,&apos;"+password+"&apos;,&apos;2&apos;);";
 // Insertar el nuevo registro en la tabla &apos;hermano&apos;
 String sNombre = nombre.replace("&apos;", "\&apos;");
 String sApellidos = apellidos.replace("&apos;", "\&apos;");
 String q3 = "INSERT INTO hermano (user_numhermano, user_role_id, nombre, apellidos,
fechanacimiento, dni, calle, numerodireccion, piso, localidad, provincia, telefono,
correoelectronico, banco, titular, dnititular, iban, entidad, oficina, digitocontrol,
numerocuenta, fechaalta) VALUES (&apos;"+numHermano+"&apos;,&apos;2&apos;, &apos;"+sNombre+"
&apos;,&apos;"+sApellidos+"&apos;,&apos;"+fechanacimiento+"&apos;,&apos;"+dni+"&apos;,&apos;"
+calle+"&apos;,&apos;"+numerodireccion+"&apos;,&apos;"+piso+"&apos;,&apos;"+localidad+"
&apos;,&apos;"+provincia+"&apos;,&apos;"+telefono+"&apos;,&apos;"+correoelectronico+"&apos;,
&apos;"+banco+"&apos;,&apos;"+titular+"&apos;,&apos;"+dnititular+"&apos;,&apos;"+iban+"
&apos;,&apos;"+entidad+"&apos;,&apos;"+oficina+"&apos;,&apos;"+digitocontrol+"&apos;,&apos;"
+numerocuenta+"&apos;,&apos;"+fechaalta+"&apos;);";
 println(q2);
 println(q3);
 msql.query(q3);
 msql.query(q2);
}


String generatePassword(int length) {
 String capitalCaseLetters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
```

```
 String lowerCaseLetters = "abcdefghijklmnopqrstuvwxyz";

 String numbers = "1234567890";
 String combinedChars = capitalCaseLetters + lowerCaseLetters + numbers;
 Random random = new Random();
 char[] password = new char[length];

 password[0] = lowerCaseLetters.charAt(random.nextInt(lowerCaseLetters.length()));
 password[1] = capitalCaseLetters.charAt(random.nextInt(capitalCaseLetters.length()));
 password[2] = numbers.charAt(random.nextInt(numbers.length()));

 for (int i = 3; i< length; i++) {
   password[i] = combinedChars.charAt(random.nextInt(combinedChars.length()));
 }
 return new String(password);
}

int getNumeroUltimoHermano() {
 String q = "SELECT MAX( user_numhermano ) AS n FROM hermano";
 println(q);
 msql.query(q);
 msql.next();
 return (msql.getInt("n"));
}



// Obté informació de la taula archivo
String[][] getInfoTablaArchivo() {

 int numRows = getNumRowsTabla("archivo");
 String[][] data = new String[numRows][3];

 int nr=0;
 msql.query( "SELECT archivo.titulo AS titulo, archivo.datacion AS datacion, archivo.file AS
file, tipo_arch.tipo FROM archivo, tipo_arch WHERE archivo.tipo_arch_idtipo_arch=tipo_arch.
idtipo_arch ORDER BY datacion ASC;" );
 while (msql.next()) {
   data[nr][0] = msql.getString("titulo");
   data[nr][1] = String.valueOf(msql.getInt("datacion"));
   data[nr][2] = msql.getString("tipo");
   nr++;
 }
 return data;
}

void insertInfoTablaArchivo(String titulo, String datacion, String file, String tipo) {
 String q4 = "INSERT INTO `archivo` (`id`, `titulo`, `datacion`, `file`,
`tipo_arch_idtipo_arch`) VALUES (NULL, &apos;"+titulo+"&apos;, &apos;"+datacion+"&apos;,
&apos;"+file+"&apos;, &apos;"+tipo+"&apos;);";
 println(q4);
 msql.query(q4);
}

String[][] getInfoTablaMovimientos(String tipoMov, int numFilas) {
 String q = "SELECT CONCAT(UPPER(SUBSTRING(c.nombre, 1, 1)) ,&apos;.&apos;, t.idtipo_mov) AS
codigo, t.nombre AS concepto, SUM(m.cantidad) AS cantidad FROM movimiento m, tipo_mov t,
categoria_mov c WHERE m.tipo_mov_idtipo_mov=t.idtipo_mov AND t.categoria=c.idcategoria_mov
AND c.nombre=&apos;"+tipoMov+"&apos; GROUP BY m.tipo_mov_idtipo_mov, t.categoria ORDER BY t.
idtipo_mov ASC;";
 String[][] data = new String[numFilas][3];

 int nr=0;
 msql.query(q );
 while (msql.next()) {
```

```
    data[nr][0] = msql.getString("codigo");
    data[nr][1] = msql.getString("concepto");
    data[nr][2] = String.valueOf(msql.getFloat("cantidad"));
    nr++;
  }
  return data;
}

float getTotalIngresos() {
  String q ="SELECT SUM(m.cantidad) AS total FROM movimiento m, categoria_mov c, tipo_mov t
WHERE m.tipo_mov_idtipo_mov=t.idtipo_mov AND t.categoria=c.idcategoria_mov AND c.
idcategoria_mov=&apos;1&apos;";
  msql.query(q);
  msql.next();
  return msql.getFloat("total");
}

float getTotalGastos() {
  String q ="SELECT SUM(m.cantidad) AS total FROM movimiento m, categoria_mov c, tipo_mov t
WHERE m.tipo_mov_idtipo_mov=t.idtipo_mov AND t.categoria=c.idcategoria_mov AND c.
idcategoria_mov=&apos;2&apos;";
  msql.query(q);
  msql.next();
  return msql.getFloat("total");
}

float getEstadoCuentas() {
  String qCantIng = "SELECT SUM(m.cantidad) AS cantidad FROM movimiento m, tipo_mov t,
categoria_mov c WHERE m.tipo_mov_idtipo_mov=t.idtipo_mov AND t.categoria=c.idcategoria_mov
AND c.nombre=&apos;Ingresos&apos; GROUP BY c.nombre;";
  msql.query(qCantIng);
  msql.next();
  float CantIng= msql.getFloat("cantidad");
  String qCantGast = "SELECT SUM(m.cantidad) AS cantidad FROM movimiento m, tipo_mov t,
categoria_mov c WHERE m.tipo_mov_idtipo_mov=t.idtipo_mov AND t.categoria=c.idcategoria_mov
AND c.nombre=&apos;Gastos&apos; GROUP BY c.nombre;";
  msql.query(qCantGast);
  msql.next();
  float CantGast= msql.getFloat("cantidad");
  float CantTot = CantIng - CantGast;
  return CantTot;
}
```

DayButton

```
class DayButton {

  // Dimensions del botó
  float x, y, w, h;

  // Data representativa
  int dia, mes, año;

  // Estats del botó
  boolean selected, selected2, enabled;

  String info;

  // Constructor
  DayButton(float x, float y, float w, float h, int d, int m, int a) {
    this.x = x;
    this.y=y;
    this.w = w;
    this.h = h;
    this.dia = d;
```

```
    this.mes = m;
    this.año = a;
    this.selected = false;
    this.selected2 = false;
    this.enabled = true;
  }

  // Setters

  void setEnabled(boolean b) {
    this.enabled = b;
  }

  void setSelected(boolean b) {
    this.selected = b;
  }

  void setSelected2(boolean b) {
    this.selected2 = b;
  }

  void setInfo(String info) {
    this.info = info;
  }


  // Dibuixa el botó
  void display() {
pushMatrix();
    pushStyle();
    if (enabled) {
      fill(255);
    } else {
      fill(100);
    }
    stroke(0);
    strokeWeight(1);
    rect(x, y, w, h, 5);
    if (selected) {
      fill(200, 100, 100);
      noStroke();
      ellipse(x + w/2, y+h/2, 50,50);
    }
    if (selected2) {
      fill(10, 200, 100);
      noStroke();
      ellipse(x + w/2, y+h/2, 50,50);
      fill(200);
      textSize(10);
      textAlign(CENTER);
      text(info, x + w/2, y+h/1.25);
    }
    fill(0);
    textSize(24);
    textAlign(CENTER);
    text(dia, x + w/2, y + h/2 + 10);

popMatrix();
    popStyle();
  }

  // Ratolí sobre el botó
  boolean mouseOver() {
    return mouseX>=this.x && mouseX<=this.x+this.w &&
      mouseY>=this.y && mouseY<=this.y+this.h;
```

```
 }
}
```

DibujaPantallas

```
// Funciones de dibujo de las pantallas

void dibujaPantallaInicio() {

 pushMatrix();
 pushStyle();
 background(colors[1]);


 //habilitar y deshabilitar botones
 disableButtons();
 bInicioSesion.setEnabled(comprovaLogin());



 //dibujar elementos de la pantalla
 fill(0);
 inicioSesion();
 //display elementos GUI
 bInicioSesion.display();
 displayInicioSesiontf();
 PopUpinicioSesion.setVisible(false);

 if (logged == true) {
   PopUpinicioSesion.setTexts("Bienvenido, " + userText.getValue(), "Usuario y contraseña
correctos");
   PopUpinicioSesion.setVisible(true);
   PopUpinicioSesion.bAceptar.setEnabled(true);
 }

 PopUpinicioSesion.display();
 popStyle();
 popMatrix();
}

void dibujaPantallaPrincipal() {
 pushMatrix();
 pushStyle();
 //imagen de fondo
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
 enableButtonsMenu();


 //dibujar elementos de la pantalla
 menu();
 tituloCarruselFotos();
 pcAvisosPrincipal.display();
 //fila2();

 //display elementos GUI
 displayButtonsMenu();

 translate(menuWidth, bannerHeight);
 cristo.display();

 popStyle();
 popMatrix();
}
```

```
void dibujaPantallaCenso() {
 pushMatrix();
 pushStyle();
 //imagen de fondo
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
 enableButtonsMenu();
 enableButtonsTabla();
 enableButtonsPagedTable();

 //dibujar elementos de la pantalla
 menu();


 //display elementos GUI
 displayButtonsMenu();
 displayButtonsTabla();
 buscar.display();
 stCenso.display();
 displayButtonsPagedTableCenso();
 popStyle();
 popMatrix();
}

void dibujaPantallaContabilidad() {
 pushMatrix();
 pushStyle();
 //imagen de fondo
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
 enableButtonsMenu();
 enableButtonsContabilidad();

 //dibujar elementos de la pantalla
 menu();
 pushStyle();
 textFont(getFontAt(1));
 fill(0);
 text("Estado de cuentas: " + estadoDeCuentas+ " €", 250, 450);
 popStyle();

 //display elementos GUI
 displayButtonsMenu();
 //ldIngresos.display();
 gastos.display();
 bBalance.display();
 bPresupuesto.display();
 popStyle();
 popMatrix();
}

void dibujaPantallaArchivo() {
 pushMatrix();
 pushStyle();
 //imagen de fondo
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
 enableButtonsMenu();
```

```
 enableButtonsTabla();
 bPrevArchivo.setEnabled(true);
 bNextArchivo.setEnabled(true);
 //dibujar elementos de la pantalla
 menu();
 stArchivo.display();

 //display elementos GUI
 displayButtonsMenu();
 displayButtonsTabla();
 displayButtonsPagedTableArchivo();
 popStyle();
 popMatrix();
}

void dibujaPantallaAvisos() {
 pushMatrix();
 pushStyle();
 //imagen de fondo
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
 enableButtonsMenu();
 bPrevAviso.setEnabled(true);
 bNextAviso.setEnabled(true);
 bAñadirAviso.setEnabled(true) ;
 bModificarAviso.setEnabled(true);
 bDetalleAviso.setEnabled(true);
 bAñadirEvento.setEnabled(true);
 bModificarEvento.setEnabled(true);
 bDetalleEvento.setEnabled(true);
 bMesAnteriorAviso.setEnabled(true);
 bMesPosteriorAviso.setEnabled(true);

 //dibujar elementos de la pantalla
 menu();

 //display elementos GUI
 displayButtonsMenu();
 displayCalendarioEventos();
 pcAvisos.display();
 bPrevAviso.display();
 bNextAviso.display();
 bAñadirAviso.display();
 bModificarAviso.display();
 bDetalleAviso.display();
 bAñadirEvento.display();
 bModificarEvento.display();
 bDetalleEvento.display();
 bMesAnteriorAviso.display();
 bMesPosteriorAviso.display();

 popStyle();
 popMatrix();
}

void dibujaPantallaEnlaces() {
 pushMatrix();
 pushStyle();
 //imagen de fondo
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
```

```
 enableButtonsMenu();
 enableButtonsEnlaces();


 //dibujar elementos de la pantalla
 menu();
 cuadroEnlacesRRSS();
 cuadroEnlacesVarios();

 //display elementos GUI
 displayButtonsMenu();
 displayButtonsEnlaces();
 popStyle();
 popMatrix();
}

void dibujaPantallaCensoDetalle() {
 pushMatrix();
 pushStyle();
 //imagen de fondo
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
 enableButtonsMenu();
 if (admin==true) {
   bAceptarCenso.setEnabled(true);
 }
 bFicha.setEnabled(true);


 //dibujar elementos de la pantalla
 menu();

 //display elementos GUI
 displayButtonsMenu();
 detalleHermano();
 if (admin==true) {
   bAceptarCenso.display();
 }
 bFicha.display();
 pushMatrix();
 translate(menuWidth, bannerHeight);
 if (admin== true) {
   titDetallePersonal.display();
 } else {
   titDetallePersonalUser.display();
 }
 popMatrix();

 popStyle();
 popMatrix();
}

void dibujaPantallaCensoNuevoHermano() {
 pushMatrix();
 pushStyle();
 //imagen de fondo
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
 enableButtonsMenu();
 bAceptarCenso.setEnabled(true);
 bFicha.setEnabled(true);
```

```
 bCalendario.setEnabled(true);
 bCalendarioAlta.setEnabled(true);

 //dibujar elementos de la pantalla
 menu();

 //display elementos GUI
 displayButtonsMenu();
 bAceptarCenso.display();
 bFicha.display();
 nuevoHermano();
 displaytfNuevoHermano();
 displaycpFechaNacimiento();
 displaycpFechaAlta();
 popStyle();
 popMatrix();
}

void dibujaPantallaContabilidadBalance() {
 pushMatrix();
 pushStyle();
 //imagen de fondo
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
 enableButtonsMenu();
 bPrevGastos.setEnabled(true);
 bNextGastos.setEnabled(true);
 bAñadirConcepto.setEnabled(true);
 bDetalleBalance.setEnabled(true);

 //dibujar elementos de la pantalla
 menu();


 //display elementos GUI
 displayButtonsMenu();
 titIngresos.display();
 titGastos.display();
 bDetalleBalance.display();
 stBalanceIngresos.display();
 stGastos.display();
 bPrevGastos.display();
 bNextGastos.display();
 bAñadirConcepto.display();
 popStyle();
 popMatrix();
}


void dibujaPantallaContabilidadPresupuesto() {
 pushMatrix();
 pushStyle();
 //imagen de fondo
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
 enableButtonsMenu();
 bPrevGastos.setEnabled(true);
 bNextGastos.setEnabled(true);
 bAñadirConcepto.setEnabled(true);

 //dibujar elementos de la pantalla
```

```
 menu();


 //display elementos GUI
 displayButtonsMenu();
 titIngresos.display();
 titGastos.display();
 stBalanceIngresos.display();
 stGastosPresupuesto.display();
 bPrevGastos.display();
 bNextGastos.display();
 bAñadirConcepto.display();
 popStyle();
 popMatrix();
}

void dibujaPantallaContabilidadAñadirConcepto() {
 pushStyle();

 pushMatrix();
 //imagen de fondo
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
 enableButtonsMenu();

 bAceptarConcepto.setEnabled(true);
 bCalendarioMovimiento.setEnabled(true);
 bAñadirRecibo.setEnabled(true);
 if (cpFechaMovimiento.visible == true) {
   disableButtons();
   bAceptarConcepto.setEnabled(false);
   bCalendarioMovimiento.setEnabled(false);
   bAñadirRecibo.setEnabled(false);
 }

 //dibujar elementos de la pantalla
 menu();
 textFont(getFontAt(8));
 text("Fecha de movimiento: ", 230, 440);


 //display elementos GUI
 displayButtonsMenu();
 titConcepto.display();
 tfTitulo.display();
 tfCantidad.display();
 bAceptarConcepto.display();
 bAñadirRecibo.display();
 pushStyle();
 textFont(getFontAt(7));
 stlTipoConcepto.display();
 popStyle();
 displaycpFechaMovimiento();
 popMatrix();
}

void dibujaPantallaContabilidadDetalleBalance() {
 pushMatrix();
 pushStyle();
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
```

```
 enableButtonsMenu();
 bAceptarConcepto.setEnabled(true);
 bPrevDetalle.setEnabled(true);
 bNextDetalle.setEnabled(true);
 bDetalleConcepto.setEnabled(true);

 //dibujar elementos de la pantalla
 menu();

 //display elementos GUI
 displayButtonsMenu();
 tDetalleItem.display(45+menuWidth, 130+bannerHeight, 1000, 65);
 stDetalleItem.display();
 bPrevDetalle.display();
 bNextDetalle.display();
 bDetalleConcepto.display();
 popStyle();
 popMatrix();
}

void dibujaPantallaContabilidadDetalleMovimiento() {
 pushMatrix();
 pushStyle();
 //imagen de fondo
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
 enableButtonsMenu();
 bAceptarConcepto.setEnabled(true);

 //dibujar elementos de la pantalla
 menu();
 textFont(getFontAt(8));
 text("Fecha de movimiento: ", 230, 440);


 //display elementos GUI
 displayButtonsMenu();
 titConcepto.display();
 bAceptarConcepto.display();
 tiTitulo.display();
 tiCantidad.display();
 tiFechaMovimiento.display();
 tiTipo.display();
 popStyle();
 popMatrix();
}

void dibujaPantallaArchivoNuevo() {
 pushMatrix();
 pushStyle();
 //imagen de fondo
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
 enableButtonsMenu();

 bCalendarioArchivo.setEnabled(true);
 itbInsertarArchivo.setEnabled(true);
 bAceptarArchivo.setEnabled(true);
 if (sCategoriaArchivo.collapsed == false) {
   itbInsertarArchivo.setEnabled(false);
 }
```

```
 //dibujar elementos de la pantalla
 menu();
 textFont(getFontAt(8));
 text("Año de datación: ", 230, 440);


 //display elementos GUI
 displayButtonsMenu();
 titArchivo.display();
 bAceptarArchivo.display();
 tfTituloArchivo.display();
 itbInsertarArchivo.display();
 sCategoriaArchivo.display();
 tfAñoDatacion.display();
 popStyle();
 popMatrix();
}

void dibujaPantallaArchivoDetalle() {
 pushMatrix();
 pushStyle();
 //imagen de fondo
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
 enableButtonsMenu();
 bAceptarArchivo.setEnabled(true);

 //dibujar elementos de la pantalla
 menu();
 text("Fecha de datación: ", 230, 440);


 //display elementos GUI
 displayButtonsMenu();
 titArchivo.display();
 bAceptarArchivo.display();
 tiTituloArchivo.display();
 itbInsertarArchivo.display();
 tiCategoriaArchivo.display();
 tiAñoDatacion.display();
 popStyle();
 popMatrix();
}

void dibujaPantallaAvisosNuevoAviso() {
 pushMatrix();
 pushStyle();
 //imagen de fondo
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
 enableButtonsMenu();
 itbInsertarArchivoAvisos.setEnabled(true);
 bAceptarAvisosAlertas.setEnabled(true);


 //dibujar elementos de la pantalla
 menu();

 //display elementos GUI
 displayButtonsMenu();
```

```
 titNuevoAviso.display();
 bAceptarAvisosAlertas.display();
 tfTituloAviso.display();
 taNuevoAviso.display();
 itbInsertarArchivoAvisos.display();
 popStyle();
 popMatrix();
}

void dibujaPantallaAvisosDetalleAviso() {
 pushMatrix();
 pushStyle();
 //imagen de fondo
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
 enableButtonsMenu();
 itbVerArchivoAvisos.setEnabled(true);
 bAceptarAvisosAlertas.setEnabled(true);


 //dibujar elementos de la pantalla
 menu();

 //display elementos GUI
 displayButtonsMenu();
 titDetalleAviso.display();
 bAceptarAvisosAlertas.display();
 tiTituloDetalleAviso.display();
 tiDetalleAviso.display();
 itbVerArchivoAvisos.display();
 popStyle();
 popMatrix();
}

void dibujaPantallaAvisosNuevoEvento() {
 pushMatrix();
 pushStyle();
 //imagen de fondo
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
 enableButtonsMenu();
 itbInsertarArchivoEventos.setEnabled(true);
 bAceptarAvisosAlertas.setEnabled(true);
 bCalendarioEvento.setEnabled(true);

 //dibujar elementos de la pantalla
 menu();
 pushStyle();
 textFont(getFontAt(5));
 text("Fecha:", 600+menuWidth, 600+bannerHeight);
 popStyle();

 //display elementos GUI
 displayButtonsMenu();
 titNuevoEvento.display();
 bAceptarAvisosAlertas.display();
 tfTituloEvento.display();
 taNuevoEvento.display();
 itbInsertarArchivoEventos.display();
 displaycpNuevoEvento();
 //tiDetalleEvento.display();
```

```
 popStyle();
 popMatrix();
}

void dibujaPantallaAvisosDetalleEvento() {
 pushMatrix();
 pushStyle();
 //imagen de fondo
 background(255);

 //habilitar y deshabilitar botones
 disableButtons();
 enableButtonsMenu();
 itbVerArchivoEventos.setEnabled(true);
 bAceptarAvisosAlertas.setEnabled(true);
 bCalendarioEvento.setEnabled(true);

 //dibujar elementos de la pantalla
 menu();
 pushStyle();
 textFont(getFontAt(5));
 text("Fecha:", 600+menuWidth, 600+bannerHeight);
 fill(255);
 rect(680+menuWidth, 575+bannerHeight, 200, 45);
 popStyle();

 //display elementos GUI
 displayButtonsMenu();
 titDetalleEvento.display();
 bAceptarAvisosAlertas.display();
 tiDetalleEvento.display();
 tiTituloDetalleEvento.display();
 itbVerArchivoEventos.display();
 //displaycpNuevoEvento();

 popStyle();
 popMatrix();
}
```

DibujaZonas

```
// Dibujo de las zonas de la pantalla

void inicioSesion() {
 pushMatrix();
 pushStyle();
 translate(inicioSesionX, inicioSesionY);

 fill(getColorAt(4));
 rect(0, 0, marcoWidth, marcoHeight);


 fill(getColorAt(3));
 rect(0, 0, marcoWidth, 30);

 displayImg(8, (marcoWidth/2)-37.5, 60, 75, 75);

 popMatrix();
 usuarioInicioSesion();
 fill(0);
 textFont(getFontAt(7));
 text("Inicio de sesión", 328, 177);
 popStyle();
}

void usuarioInicioSesion() {
```

```
 pushMatrix();
 pushStyle();
 translate(inicioSesionX, inicioSesionY);

 fill(getColorAt(3));
 rect((marcoWidth/2)-(marcoCuentaWidth/2), (marcoHeight/2)-(marcoCuentaHeight/2)+50,
marcoCuentaWidth, marcoCuentaHeight);

 fill(0);
 textAlign(LEFT);
 textSize(24);
 text("Username: ", (marcoWidth/2)-(marcoCuentaWidth/2)+20, (marcoHeight/2)-
(marcoCuentaHeight/2)+110);
 text("Password: ", (marcoWidth/2)-(marcoCuentaWidth/2)+20, (marcoHeight/2)-
(marcoCuentaHeight/2)+210);

 popStyle();
 popMatrix();
}

void logo() {
 pushStyle();
 fill(getColorAt(3));
 rect(30, 20, logoWidth, logoHeight);
 image(getImgAt(0), 30, 20, logoWidth, logoHeight);
 popStyle();
}

void menuBar() {
 pushStyle();
 strokeWeight(1);
 line(menuWidth, 1280, menuWidth, bannerHeight);
 strokeWeight(0);
 fill(getColorAt(1));
 rect(0, 0, menuWidth, menuHeight);
 fill(0);
 text("MENU", menuWidth/2, menuHeight/2);
 logo();
 displayButtonsMenu();
 popStyle();
}

void banner() {
 pushStyle();
 pushMatrix();
 textAlign(LEFT);
 translate(menuWidth, 0);
 fill(getColorAt(1));
 stroke(0);
 strokeWeight(1);
 line(0, bannerHeight, 1280, bannerHeight);
 strokeWeight(0);
 rect(0, 0, bannerWidth, bannerHeight);
 fill(255);
 textFont(getFontAt(3));
 text("Hermandad del Stmo. Cristo de la Sala", 20, 70);
 image(getIconUser(), 830,(bannerHeight/2)-25, 50,50);
 textFont(getFontAt(5));
 text(userNameAdmin, 890,(bannerHeight/2)+7);
 popStyle();
 popMatrix();
}

void columnaEntera() {
 pushMatrix();
```

```
 pushStyle();
 translate(menuWidth, bannerHeight);
 fill(getColorAt(3));
 rect(20, 20, 1040, 660);
 popMatrix();
 popStyle();
}

void columna1() {
 pushMatrix();
 pushStyle();
 translate(menuWidth, bannerHeight);
 fill(getColorAt(2));
 rect(20, 20, 500, 660);
 popMatrix();
 popStyle();
}

void columna2() {
 pushMatrix();
 pushStyle();
 translate(menuWidth+520, bannerHeight);
 fill(getColorAt(3));
 rect(20, 20, 500, 660);
 popMatrix();
 popStyle();
}

void columna1y2() {
 columna1();
 columna2();
}

void fila1() {
 pushMatrix();
 pushStyle();
 translate(menuWidth+520, bannerHeight);
 fill(getColorAt(3));
 rect(menuWidth+540, bannerHeight+20, 520, 310);
 popMatrix();
 popStyle();
}

void fila2() {
 pushMatrix();
 pushStyle();
 translate(menuWidth+520, bannerHeight+350);
 fill(getColorAt(3));
 rect(20, 0, 520, 330);
 popMatrix();
 popStyle();
}

void menu() {
 menuBar();
 banner();
 bPrincipal.display();
}



void avisos() {
 pushStyle();
 pushMatrix();
 fill(getColorAt(3));
```

```
 rect(menuWidth+20, primerIconY+iconHeight, ((1280-menuWidth)/2)-10, iconHeight*3);
 popStyle();
 popMatrix();
}



void detalleHermano() {
 pushStyle();
 pushMatrix();

 translate(menuWidth, bannerHeight);

 displayDetalleHermano();
 strokeWeight(3);
 line(50, 375, 1035, 375);
 image(getImgAt(8), 33, 75, 118, 135);
 textFont(getFontAt(6));
 text("Fecha de nacimiento:", 167, 177+12);
 text("Domicilio:", 40, 250+12);
 text("Teléfono:", 40, 311+12);
 text("Correo electrónico:", 413, 311+12);
 text("Nombre del banco:", 40, 420+12);
 text("Titular de la cuenta:", 40, 480+12);
 text("Fecha de alta:", 630, 625+12);


 popMatrix();
 popStyle();
}

void nuevoHermano() {
 pushStyle();
 pushMatrix();

 translate(menuWidth, bannerHeight);

 strokeWeight(3);
 line(50, 375, 1035, 375);
 image(getImgAt(8), 33, 75, 118, 135);
 textFont(getFontAt(6));
 text("Fecha de nacimiento:", 167, 177+12);
 text("Domicilio:", 40, 250+12);
 text("Teléfono:", 40, 311+12);
 text("Correo electrónico:", 413, 311+12);
 text("Nombre del banco:", 40, 420+12);
 text("Titular de la cuenta:", 40, 480+12);
 text("Fecha de alta:", 630, 625+12);
 titDetallePersonal.display();

 popMatrix();
 popStyle();
}

void tituloCarruselFotos() {
 pushStyle();
 pushMatrix();

 translate(menuWidth, bannerHeight);

 fill(getColorAt(3));
 rect(30, 35, 495, 35);
 textFont(getFontAt(7));
 fill(0);
 textAlign(LEFT);
```

```
 text("Bienvenido" + userText.getValue() + " a tu perfil de la hermandad", 40, 57);

 popStyle();
 popMatrix();
}

void cuadroEnlacesRRSS() {
 pushMatrix();
 pushStyle();
 translate(menuWidth, bannerHeight);
 strokeJoin(ROUND);
 strokeWeight(2);
 fill(255);
 rect(70, 140, 445, 410);
 fill(getColorAt(3));
 rect(70, 140, 445, 35);
 fill(0);
 textFont(getFontAt(7));
 text("Redes Sociales", 80, 165);
 popStyle();
 popMatrix();
}

void cuadroEnlacesVarios() {
 pushMatrix();
 pushStyle();
 translate(menuWidth, bannerHeight);
 strokeJoin(ROUND);
 strokeWeight(2);
 fill(255);
 rect(580, 140, 445, 410);
 fill(getColorAt(3));
 rect(580, 140, 445, 35);
 fill(0);
 textFont(getFontAt(7));
 text("Otros enlaces de interés", 590, 165);
 popStyle();
 popMatrix();
}
```

Folder

```
// Ruta del fitxer
String rutaFitxer ="";

// Nom del fitxer
String titol="";

// Carpeta on copiar els fitxers
String rutaCopia =
"/Users/xiscopolgonzalez/Desktop/Cristo_de_la_SalApp/Cristo_de_la_SalApp/data";

// Obri finestra per seleccionar fitxer
void fileSelected(File selection) {
 if (selection == null) {
   println("No s&apos;ha seleccionat cap fitxer.");
 } else {

   // Obtenim la ruta del fitxer seleccionat
   rutaFitxer = selection.getAbsolutePath();
   println("Ruta del Fitxer:"+rutaFitxer);

   titol = selection.getName();
   println("Nom del Fitxer:"+titol);

   copiar(rutaFitxer, rutaCopia, titol);
```

```
 }
}
```

Fonts

```
// Archivo con la información de las fuentes de la App

// URL de la carpeta donde se encuentran los medias
String URL_FONTS = "fuentes/";

// Array de tipografias
PFont[] fonts;

// Establece las fuentes de la App
void setFonts() {
 this.fonts = new PFont[9];
 this.fonts[0] = createFont(URL_FONTS+"Sacred Valley.ttf", medidaTitulo);
 this.fonts[1] = createFont(URL_FONTS+"LANENAR_.ttf", medidaSubtitulo);
 this.fonts[2] = createFont(URL_FONTS+"Sono-ExtraLight.ttf", medidaParrafo);
 this.fonts[3] = createFont(URL_FONTS+"La Estroma.ttf", medidaTitulo);
 this.fonts[4] = createFont(URL_FONTS+"LANENAR_.ttf", medidaParrafo);
 this.fonts[5] = createFont(URL_FONTS+"LANENAR_.ttf", 24);
 this.fonts[6] = createFont(URL_FONTS+"Sacred Valley.ttf", 24);
 this.fonts[7] = createFont(URL_FONTS+"LANENAR_.ttf", 18);
 this.fonts[8] = createFont(URL_FONTS+"LANENAR_.ttf", 20);
}

// Getter del número de fuentes
int getNumFonts() {
 return this.fonts.length;
}

// Getter de la fuente primaria
PFont getFirstFont() {
 return this.fonts[0];
}

// Getter de la fuente secundaria
PFont getSecondFont() {
 return this.fonts[1];
}

// Getter de la fuente terciaria
PFont getThirdFont() {
 return this.fonts[2];
}

// Getter de la fuente i-esima
PFont getFontAt(int i) {
 return this.fonts[i];
}
```

GUI

```
//Configuración de los elementos de la GUI

// Creación de los elementos de la GUI
void setGUI() {
 initButtons();
 initTextField();
 initTextInfo();
 initImgTextButton();
 initCalendar();
 initLinesDiagram();
 initBarsDiagram();
 initShowImage();
 initTitulo();
```

```
 initSelect();
 initSelectTable();
 initCalendariPlus();
 initTable();
 initSelectTextList();
 initTextArea();
 initPagedCard();
 initPopUp();
}

// Botones
Button[] buttons;

Button bInicioSesion, bPrincipal;

// Creación de los botones de la GUI
Button bAñadir, bModificar, bDetalle, bAceptarCenso, bFicha, bPrevCenso, bNextCenso,
bPrevGastos, bNextGastos, bFacebook, bTwitter, bInstagram, bYoutube, bAyuntamiento,
bArzobispado, bWebCofrade, bOtrasHermandades, bBalance, bPresupuesto, bAñadirConcepto;
Button bAceptarConcepto, bCalendario, bCalendarioAlta, bCalendarioMovimiento, bAñadirRecibo,
bDetalleBalance;
Button bPrevDetalle, bNextDetalle, bPrevArchivo, bNextArchivo, bAceptarArchivo,
bCalendarioArchivo, bAceptarAvisosAlertas;
Button bDetalleConcepto, bCalendarioEvento, bPrevAviso, bNextAviso, bAñadirAviso,
bModificarAviso, bDetalleAviso;
Button bAñadirEvento, bModificarEvento, bDetalleEvento, bRecuerdos, bMesAnteriorAviso,
bMesPosteriorAviso;

void initButtons() {
 buttons = new Button[48];
 buttons[0] = new Button("Principal", 850, (bannerHeight/2)-13.5, 100, 25);
 buttons[1] = new Button("Iniciar sesión", 320+(marcoWidth/2)-75, 600, 150, 30);
 buttons[2] = new Button("Añadir", menuWidth+20, primerIconY+20, 200, 50);
 buttons[3] = new Button("Buscar", (3*menuWidth)+20, primerIconY+20, 200, 50);
 buttons[4] = new Button("Aceptar", 641+menuWidth, 20+bannerHeight, 403, 40);
 buttons[5] = new Button("Ficha Inscripción", 40+menuWidth, 605+bannerHeight, 574, 60);
 buttons[6] = new Button("PREV", 950, 715, 60, 60);
 buttons[7] = new Button("NEXT", 1050, 715, 60, 60);
 buttons[8] = new Button("Facebook", 325, bannerHeight+200, buttonEnlaceW, buttonEnlaceH);
 buttons[9] = new Button("Twitter", 325, bannerHeight+200+(20+buttonEnlaceH), buttonEnlaceW,
buttonEnlaceH);
 buttons[10] = new Button("Instagram", 325, bannerHeight+200+2*(20+buttonEnlaceH),
buttonEnlaceW, buttonEnlaceH);
 buttons[11] = new Button("Youtube", 325, bannerHeight+200+3*(20+buttonEnlaceH),
buttonEnlaceW, buttonEnlaceH);
 buttons[12] = new Button("Arzobispado de Toledo", 835, bannerHeight+200, buttonEnlaceW,
buttonEnlaceH);
 buttons[13] = new Button("Ayuntamiento de Bargas", 835, bannerHeight+200+
(20+buttonEnlaceH), buttonEnlaceW, buttonEnlaceH);
 buttons[14] = new Button("Web Cofrade", 835, bannerHeight+200+2*(20+buttonEnlaceH),
buttonEnlaceW, buttonEnlaceH);
 buttons[15] = new Button("Otras hermandades", 835, bannerHeight+200+3*(20+buttonEnlaceH),
buttonEnlaceW, buttonEnlaceH);
 buttons[16] = new Button("Balance de ingresos y gastos", 825, 200, 400, 50);
 buttons[17] = new Button("Presupuesto", 825, 275, 400, 50);
 buttons[18] = new Button("PREV", 1080, 680, 150, 30);
 buttons[19] = new Button("NEXT", 1080, 720, 150, 30);
 buttons[20] = new Button("Añadir concepto", 1080, 170, 150, 40);
 buttons[21] = new Button("Aceptar", 645+menuWidth, 175+bannerHeight, 405, 40);
 buttons[22] = new Button("Detalle", (2*menuWidth)+20, primerIconY+20, 200, 50); //
(2*menuWidth)+20, primerIconY+20, 200, 50)
 buttons[23] = new Button("Calendario", 575, 165+bannerHeight, 100, 45);
 buttons[24] = new Button("Calendario", 970, 605+bannerHeight, 100, 45);
 buttons[25] = new Button("Calendario", 420, 410, 100, 45);
 buttons[26] = new Button("Añadir recibo", 780, 560, 465, 45);
```

```
buttons[27] = new Button("Detalle", 1080, 220, 150, 40);
buttons[28] = new Button("PREV", 1080, 580, 150, 30);
buttons[29] = new Button("NEXT", 1080, 620, 150, 30);
buttons[30] = new Button("Detalle", 45+menuWidth, 580, 150, 80);
buttons[31] = new Button("PREV", 950, 715, 60, 60);
buttons[32] = new Button("NEXT", 1050, 715, 60, 60);
buttons[33] = new Button("Aceptar", 643+menuWidth, 175+bannerHeight, 400, 40);
buttons[34] = new Button("Calendario", 400, 310+bannerHeight, 100, 45);
buttons[35] = new Button("Aceptar", 643+menuWidth, 120+bannerHeight, 400, 40);
buttons[36] = new Button("Calendario", 680+menuWidth+225, 570+bannerHeight, 130, 45);
buttons[37] = new Button("PREV", menuWidth+20, 680, 60, 60);
buttons[38] = new Button("NEXT", menuWidth+90, 680, 60, 60);
buttons[39] = new Button("Añadir", menuWidth+20, primerIconY+10, 100, 50);
buttons[40] = new Button("Modificar", (2*menuWidth)-80, primerIconY+10, 100, 50);
buttons[41] = new Button("Detalle", menuWidth+220, primerIconY+10, 100, 50);
buttons[42] = new Button("Añadir", (2*menuWidth)+350, primerIconY+10, 100, 50);
buttons[43] = new Button("Modificar", (2*menuWidth)+450, primerIconY+10, 100, 50);
buttons[44] = new Button("Detalle", (2*menuWidth)+550, primerIconY+10, 100, 50);
buttons[45] = new Button("Recuerdos de la hermandad", 325, 680, 850, buttonEnlaceH);
buttons[46] = new Button("Siguiente", (2*menuWidth)+780, primerIconY+10, 80, 50);
buttons[47] = new Button("Anterior", (2*menuWidth)+700, primerIconY+10, 80, 50);

bPrincipal = buttons[0];
bInicioSesion = buttons[1];
bAñadir = buttons[2];
bModificar = buttons[3];
bAceptarCenso = buttons[4];
bFicha = buttons[5];
bPrevCenso = buttons[6];
bNextCenso = buttons[7];
bFacebook = buttons[8];
bTwitter =buttons[9];
bInstagram= buttons[10];
bYoutube= buttons[11];
bArzobispado= buttons[12];
bAyuntamiento= buttons[13];
bWebCofrade= buttons[14];
bOtrasHermandades= buttons[15];
bBalance = buttons[16];
bPresupuesto = buttons[17];
bPrevGastos = buttons[18];
bNextGastos = buttons[19];
bAñadirConcepto = buttons[20];
bAceptarConcepto = buttons[21];
bDetalle = buttons[22];
bCalendario = buttons[23];
bCalendarioAlta = buttons[24];
bCalendarioMovimiento = buttons[25];
bAñadirRecibo = buttons[26];
bDetalleBalance = buttons[27];
bPrevDetalle= buttons[28];
bNextDetalle= buttons[29];
bDetalleConcepto = buttons[30];
bPrevArchivo = buttons[31];
bNextArchivo = buttons[32];
bAceptarArchivo = buttons [33];
bCalendarioArchivo = buttons[34];
bAceptarAvisosAlertas = buttons[35];
bCalendarioEvento = buttons[36];
bPrevAviso = buttons [37];
bNextAviso = buttons [38];
bAñadirAviso = buttons [39];
bModificarAviso = buttons [40];
bDetalleAviso = buttons [41];
bAñadirEvento = buttons [42];
```

```
 bModificarEvento = buttons [43];
 bDetalleEvento = buttons [44];
 bRecuerdos = buttons[45];
 bMesAnteriorAviso = buttons[46];
 bMesPosteriorAviso = buttons[47];
}

//Desactivar todos los botones
void disableButtons() {
 for (int i = 0; i<buttons.length; i++) {
   buttons[i].setEnabled(false);
 }
 itbCenso.setEnabled(false);
 itbContabilidad.setEnabled(false);
 itbArchivo.setEnabled(false);
 itbAvisos.setEnabled(false);
 itbEnlaces.setEnabled(false);
 itbPerfilPersonal.setEnabled(false);
 itbInsertarArchivo .setEnabled(false);
 itbInsertarArchivoAvisos .setEnabled(false);
 itbInsertarArchivoEventos.setEnabled(false);
 itbVerArchivoAvisos.setEnabled(false);
 itbVerArchivoEventos.setEnabled(false);
 PopUpinicioSesion.bAceptar.setEnabled(false);
}

void enableButtonsTabla() {
 bAñadir.setEnabled(true);
 bModificar.setEnabled(true);
 bDetalle.setEnabled(true);
}

void enableButtonsPagedTable() {
 bNextCenso.setEnabled(true);
 bPrevCenso.setEnabled(true);
}

void enableButtonsContabilidad() {
 bBalance.setEnabled(true);
 bPresupuesto.setEnabled(true);
}

void displayButtonsTabla() {
 bAñadir.display();
 bModificar.display();
 bDetalle.display();
}

// Dibuixa els botons
void displayButtonsPagedTableCenso() {
 bNextCenso.display();
 bPrevCenso.display();
}

void displayButtonsPagedTableArchivo() {
 bNextArchivo.display();
 bPrevArchivo.display();
}

void enableButtonsEnlaces() {
 bFacebook.setEnabled(true);
 bTwitter.setEnabled(true);
 bInstagram.setEnabled(true);
 bYoutube.setEnabled(true);
 bArzobispado.setEnabled(true);
```

```
 bAyuntamiento.setEnabled(true);
 bWebCofrade.setEnabled(true);
 bOtrasHermandades.setEnabled(true);
 bRecuerdos.setEnabled(true);
}
void displayButtonsEnlaces() {
 bFacebook.setTextFont(8);
 bTwitter.setTextFont(8);
 bInstagram.setTextFont(8);
 bYoutube.setTextFont(8);
 bArzobispado.setTextFont(8);
 bAyuntamiento.setTextFont(8);
 bWebCofrade.setTextFont(8);
 bOtrasHermandades.setTextFont(8);
 bFacebook.display();
 bTwitter.display();
 bInstagram.display();
 bYoutube.display();
 bArzobispado.display();
 bAyuntamiento.display();
 bWebCofrade.display();
 bOtrasHermandades.display();
 bRecuerdos.display();
}


//TextField

// Declaració de les variables
TextField userText, passText;
TextField buscar;
TextField tfNombre, tfApellidos, tfDNI, tfCalle, tfNumero, tfPiso, tfLocalidad, tfProvincia,
tfTelefono, tfCorreoElectronico;
TextField tfBanco, tfTitular, tfDNITitular, tfIBAN, tfEntidad, tfOficina, tfDigitoControl,
tfNumeroCuenta;
TextField tfTitulo, tfCantidad;
TextField tfTituloArchivo, tfTituloAviso, tfTituloEvento, tfAñoDatacion;

void initTextField() {
 userText = new TextField("usuario", (marcoWidth/2)-(marcoCuentaWidth/2)+20+inicioSesionX,
(marcoHeight/2)-(marcoCuentaHeight/2)+130+inicioSesionY, 350, 35);
 passText = new TextField("contraseña", (marcoWidth/2)-(marcoCuentaWidth/2)
+20+inicioSesionX, (marcoHeight/2)-(marcoCuentaHeight/2)+230+inicioSesionY, 350, 35);
 buscar = new TextField("", 850, primerIconY+25, 410, 35);
 tfNombre = new TextField ("Nombre", 170+menuWidth, 95+bannerHeight, 305, 45);
 tfApellidos = new TextField ("Apellidos", 500+menuWidth, 95+bannerHeight, 535, 45);
 tfDNI = new TextField ("DNI", 695+menuWidth, 165+bannerHeight, 340, 45);
 tfCalle = new TextField ("Calle", 148+menuWidth, 235+bannerHeight, 147, 45);
 tfNumero = new TextField ("N°", 313+menuWidth, 235+bannerHeight, 89, 45);
 tfPiso = new TextField ("Piso", 417+menuWidth, 235+bannerHeight, 89, 45);
 tfLocalidad = new TextField ("Localidad", 521+menuWidth, 235+bannerHeight, 198, 45);
 tfProvincia = new TextField ("Provincia", 735+menuWidth, 235+bannerHeight, 301, 45);
 tfTelefono = new TextField ("Teléfono", 148+menuWidth, 297+bannerHeight, 255, 45);
 tfCorreoElectronico = new TextField ("Correo Electrónico", 615+menuWidth, 297+bannerHeight,
420, 45);
 tfBanco = new TextField ("Banco", 235+menuWidth, 405+bannerHeight, 800, 45);
 tfTitular= new TextField ("Titular", 235+menuWidth, 462+bannerHeight, 380, 45);
 tfDNITitular = new TextField ("DNI del titular", 649+menuWidth, 462+bannerHeight, 386, 45);
 tfIBAN = new TextField ("IBAN", 40+menuWidth, 522+bannerHeight, 165, 45);
 tfEntidad = new TextField ("Entidad", 225+menuWidth, 522+bannerHeight, 100, 45);
 tfOficina = new TextField ("Oficina", 341+menuWidth, 522+bannerHeight, 100, 45);
 tfDigitoControl = new TextField ("Dígito Control", 460+menuWidth, 522+bannerHeight, 202,
45);
 tfNumeroCuenta = new TextField ("Número de cuenta", 677+menuWidth, 522+bannerHeight, 357,
45);
```

```
 tfTitulo = new TextField ("Titulo", 230, 350, 1020, 40);
 tfCantidad = new TextField ("Cantidad", 780, 490, 465, 40);
 tfTituloArchivo = new TextField("Título", 27+menuWidth, 247+bannerHeight, 1020, 45);
 tfTituloAviso = new TextField("Título", 27+menuWidth, 190+bannerHeight, 1020, 45);
 tfTituloEvento = new TextField("Título", 27+menuWidth, 190+bannerHeight, 1020, 45);
 tfAñoDatacion  = new TextField("Año", 400, 310+bannerHeight, 100, 45);
}

void displayInicioSesiontf() {
 userText.display();
 passText.display();
}

void displaytfNuevoHermano() {
 tfNombre.display();
 tfApellidos.display();
 tfDNI.display();
 tfCalle.display();
 tfNumero.display();
 tfPiso.display();
 tfLocalidad.display();
 tfProvincia.display();
 tfTelefono.display();
 tfCorreoElectronico.display();
 tfBanco.display();
 tfTitular.display();
 tfDNITitular.display();
 tfIBAN.display();
 tfEntidad.display();
 tfOficina.display();
 tfDigitoControl.display();
 tfNumeroCuenta.display();
}

//TextInfo
TextInfo tiNombre, tiApellidos, tiDNI, tiCalle, tiFechaNacimiento, tiFechaAlta, tiNumero,
tiPiso, tiLocalidad, tiProvincia, tiTelefono, tiCorreoElectronico;
TextInfo tiBanco, tiTitular, tiDNITitular, tiIBAN, tiEntidad, tiOficina, tiDigitoControl,
tiNumeroCuenta, tiFechaNacimientoAñadir;
TextInfo tiTitulo, tiCantidad, tiFechaMovimiento, tiTipo, tiDetalleEvento, tiDetalleAviso,
tiTituloDetalleEvento, tiTituloDetalleAviso;
TextInfo tiTituloArchivo, tiCategoriaArchivo, tiAñoDatacion;

void initTextInfo() {
 pushMatrix();
 tiNombre = new TextInfo ("Nombre", 170, 95, 305, 45);
 tiApellidos = new TextInfo ("Apellidos", 500, 95, 535, 45);
 tiDNI = new TextInfo ("DNI", 695, 165, 340, 45);
 tiCalle = new TextInfo ("Calle", 148, 235, 147, 45);
 tiNumero = new TextInfo ("N°", 313, 235, 89, 45);
 tiPiso = new TextInfo ("Piso", 417, 235, 89, 45);
 tiLocalidad = new TextInfo ("Localidad", 521, 235, 198, 45);
 tiProvincia = new TextInfo ("Provincia", 735, 235, 301, 45);
 tiTelefono = new TextInfo ("Teléfono", 148, 297, 255, 45);
 tiCorreoElectronico = new TextInfo ("Correo Electrónico", 615, 297, 420, 45);
 tiBanco = new TextInfo ("Banco", 235, 405, 800, 45);
 tiTitular= new TextInfo ("Titular", 235, 462, 380, 45);
 tiDNITitular = new TextInfo ("DNI del titular", 649, 462, 386, 45);
 tiIBAN = new TextInfo ("IBAN", 40, 522, 165, 45);
 tiEntidad = new TextInfo ("Entidad", 225, 522, 100, 45);
 tiOficina = new TextInfo ("Oficina", 341, 522, 100, 45);
 tiDigitoControl = new TextInfo ("Dígito Control", 460, 522, 202, 45);
 tiNumeroCuenta = new TextInfo ("Número de cuenta", 677, 522, 357, 45);
 tiFechaNacimiento = new TextInfo("Fecha Nacimiento", 380, 165, 300, 45);
 tiFechaAlta = new TextInfo("Fecha Alta", 770, 608, 268, 45);
```

```
 tiTitulo = new TextInfo ("Titulo", 230, 350, 1020, 40);
 tiCantidad = new TextInfo ("Cantidad", 780, 490, 465, 40);
 tiFechaMovimiento = new TextInfo ("Fecha de movimiento", 420, 413, 350, 40);
 tiTipo = new TextInfo ("Tipo", 780, 413, 465, 40);
 tiTituloDetalleEvento = new TextInfo ("Título", 27+menuWidth, 190+bannerHeight, 1020, 45);
 tiTituloDetalleAviso = new TextInfo("Título", 27+menuWidth, 190+bannerHeight, 1020, 45);
 tiDetalleEvento = new TextInfo ("Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation", 27+menuWidth, 250+bannerHeight, 1020, 260);
 tiDetalleAviso = new TextInfo ("Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation", 27+menuWidth, 250+bannerHeight, 1020, 260);
 tiTituloArchivo  = new TextInfo("Título", 27+menuWidth, 247+bannerHeight, 1020, 45);
 tiCategoriaArchivo  = new TextInfo("Título", 580+menuWidth, 310+bannerHeight,
selectArchivoW, selectArchivoH);
 tiAñoDatacion = new TextInfo("Título", 380, 310+bannerHeight, 100, 45);

 popMatrix();
}


void displayDetalleHermano() {
 tiNombre.display();
 tiApellidos.display();
 tiDNI.display();
 tiCalle.display();
 tiNumero.display();
 tiPiso.display();
 tiLocalidad.display();
 tiProvincia.display();
 tiTelefono.display();
 tiCorreoElectronico.display();
 tiBanco.display();
 tiTitular.display();
 tiDNITitular.display();
 tiIBAN.display();
 tiEntidad.display();
 tiOficina.display();
 tiDigitoControl.display();
 tiNumeroCuenta.display();
 tiFechaNacimiento.display();
 tiFechaAlta.display();
}

//ImgTextButton

ImgTextButton[] imgtextbuttons;
ImgTextButton itbCenso, itbContabilidad, itbArchivo, itbAvisos, itbEnlaces,
itbPerfilPersonal;
ImgTextButton itbInsertarArchivo, itbInsertarArchivoAvisos, itbInsertarArchivoEventos,
itbVerArchivoAvisos, itbVerArchivoEventos;

void initImgTextButton() {
 imgtextbuttons = new ImgTextButton[11];
 imgtextbuttons[0] = new ImgTextButton(getIconCenso(), "Censo", 0, primerIconY, iconWidth,
iconHeight);
 imgtextbuttons[1] = new ImgTextButton(getIconContabilidad(), "Contabilidad", 0,
segundoIconY, iconWidth, iconHeight);
 imgtextbuttons[2] = new ImgTextButton(getIconArchivo(), "Archivo", 0, tercerIconY,
iconWidth, iconHeight);
 imgtextbuttons[3] = new ImgTextButton(getIconAvisos(), "Avisos y alertas", 0, cuartoIconY,
iconWidth, iconHeight );
 imgtextbuttons[4] = new ImgTextButton(getIconEnlaces(), "Enlaces", 0, quintoIconY,
iconWidth, iconHeight);
 imgtextbuttons[5] = new ImgTextButton(getIconCenso(), "Perfil Personal", 0, primerIconY,
```

```
iconWidth, iconHeight);
 imgtextbuttons[6] = new ImgTextButton(getIconFile(), "Insertar Archivo", 730+menuWidth,
380+bannerHeight, 190, 100);
 imgtextbuttons[7] = new ImgTextButton(getIconFile(), "Insertar Archivo", 440+menuWidth,
550+bannerHeight, 160, 100);
 imgtextbuttons[8] = new ImgTextButton(getIconFile(), "Insertar Archivo", 200+menuWidth,
550+bannerHeight, 160, 100);
 imgtextbuttons[9] = new ImgTextButton(getIconFile(), "Ver Archivo", 440+menuWidth,
550+bannerHeight, 160, 100);
 imgtextbuttons[10] = new ImgTextButton(getIconFile(), "Ver Archivo", 200+menuWidth,
550+bannerHeight, 160, 100);

 itbCenso = imgtextbuttons[0];
 itbContabilidad = imgtextbuttons[1];
 itbArchivo = imgtextbuttons[2];
 itbAvisos = imgtextbuttons[3];
 itbEnlaces = imgtextbuttons[4];
 itbPerfilPersonal = imgtextbuttons[5];
 itbInsertarArchivo = imgtextbuttons[6];
 itbInsertarArchivoAvisos = imgtextbuttons[7];
 itbInsertarArchivoEventos = imgtextbuttons[8];
 itbVerArchivoAvisos = imgtextbuttons[9];
 itbVerArchivoEventos = imgtextbuttons[10];
}

// Activar los botones del menú
void enableButtonsMenu() {
 if (admin == true) {
   itbCenso.setEnabled(true);
   itbPerfilPersonal.setEnabled(false);
 } else {
   itbCenso.setEnabled(false);
   itbPerfilPersonal.setEnabled(true);
 }
 itbContabilidad.setEnabled(true);
 itbArchivo.setEnabled(true);
 itbAvisos.setEnabled(true);
 itbEnlaces.setEnabled(true);
 bPrincipal.setEnabled(true);
}

void displayButtonsMenu() {
 if (admin == true) {
   itbCenso.display();
 } else {
   itbPerfilPersonal.display();
 }
 itbContabilidad.display();
 itbArchivo.display();
 itbAvisos.display();
 itbEnlaces.display();
}

//calendario


// Variable de Calendari
Calendario cEventos;

String[][] fechasClave = {{"2023-03-15", "CUMPLE"}, {"2023-03-28", "FESTA"}};

void initCalendar() {
 cEventos = new Calendario(menuWidth+20+((1280-menuWidth)/2), primerIconY+iconHeight+20,
((1280-menuWidth)/2)-35, (iconHeight*3)-50, fechasClave);
}
```

```
void displayCalendarioEventos() {
 cEventos.display();
}

//LinesDiagram

LinesDiagram ldIngresos;

// Dades del Diagrama (etiquetes)
String[] textos = {"Jan", "Feb", "Mar", "Apr", "May", "Jun",
 "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };

// Dades del Diagrama (valors)
float[] values = {400, 600, 100, 300, 55, 100, 90, 220, 186, 400, 600, 10 };

// Color de la línia
color colorLine = color(#401E3A);

void initLinesDiagram() {
 ldIngresos = new LinesDiagram(menuWidth+50, bannerHeight+50, (width/2)-100, (height/2)-100);

 // Configuració de Dades (textos, valors, colors)
 ldIngresos.setTexts(textos);
 ldIngresos.setValues(values);
 ldIngresos.setColors(colorLine);
}

//BarsDiagram

BarsDiagram gastos;

String[] textosbd = {"INGRESOS", "GASTOS"};
float[] valuesbd = {0, 0};
color[] colorsbd = {color (64, 30, 58), color (150, 13, 15),
 color(255, 0, 0), color(0, 255, 0)};

void initBarsDiagram() {
 gastos = new BarsDiagram(870, 520, 350, 200);

 // Configuración de datis (textos, valores, colores)
 gastos.setTexts(textosbd);
 valuesbd[0] = getTotalIngresos();
 valuesbd[1] = getTotalGastos();
 gastos.setValues(valuesbd);
 gastos.setColors(colorsbd);
}

//ShowImage

ShowImage cristo;

// Nombre de las imágenes
String[] noms = {"cristo1.jpg", "cristo2.jpeg", "cristo3.jpeg",
 "cristo4.jpeg"};

void initShowImage() {

 cristo = new ShowImage(35, 70, 485, 575);
 cristo.setImages(noms);
}

//Titulo

Titulo titIngresos, titGastos, titConcepto, titDetallePersonal, titDetallePersonalUser,
```

```
titArchivo, titNuevoAviso, titNuevoEvento;
Titulo titDetalleAviso, titDetalleEvento;
void initTitulo() {
 titIngresos = new Titulo("Ingresos", 250, 170, 800, 40);
 titGastos = new Titulo("Gastos", 250, 470, 800, 40);
 titConcepto = new Titulo ("Concepto", 230, 275, 610, 40);
 titDetallePersonal = new Titulo ("Detalle personal", 20, 25, 605, 35);
 titDetallePersonalUser = new Titulo("Detalle personal", 20, 25, 1080-20-10, 35);
 titArchivo = new Titulo("Archivo", 27+menuWidth, 175+bannerHeight, 610, 40);
 titNuevoAviso = new Titulo("Nuevo aviso", 27+menuWidth, 120+bannerHeight, 610, 40);
 titNuevoEvento = new Titulo("Nuevo evento", 27+menuWidth, 120+bannerHeight, 610, 40);
 titDetalleAviso = new Titulo("Detalle aviso", 27+menuWidth, 120+bannerHeight, 610, 40);
 titDetalleEvento = new Titulo("Detalle evento", 27+menuWidth, 120+bannerHeight, 610, 40);
}

//Select

Select sCategoriaArchivo;



String[] selectValuesCategoriaArchivo = {"Imágen", "Documento", "Programa", "Vídeo"};

// Dimensions dels botons
float selectConceptoW = 465;
float selectConceptoH = 40;
float selectArchivoW = 470;
float selectArchivoH = 45;

void initSelect() {
 sCategoriaArchivo = new Select(selectValuesCategoriaArchivo, 580+menuWidth,
310+bannerHeight, selectArchivoW, selectArchivoH);
}

//SelectTextList

SelectTextList stlTipoConcepto;

String selectedCountry;
String[][] selectValuesConcepto = {{"0", "I.1"}, {"1", "I.2"}, {"2", "I.3"}, {"3", "G.1"},
{"4", "G.2"}, {"5", "G.3"},
 {"6", "G.4"}, {"7", "G.5"}, {"8", "G.6"}, {"9", "G.7"}, {"10", "G.8"}, {"11", "G.9"},
{"12", "G.10"},
 {"13", "G.11"}, {"14", "G.12"}, {"15", "G.13"}, {"16", "G.14"}, {"17", "G.15"}, {"18", "G.
16"}, {"19", "G.17"}, {"20", "G.18"} };

void initSelectTextList() {
 stlTipoConcepto = new SelectTextList(selectValuesConcepto, 780, 420, selectConceptoW-50,
selectConceptoH);
}

//SelectTable

SelectTable stCenso, stGastos, stGastosPresupuesto, stArchivo;

int filasCenso = 6, columnasCenso = 5;

String[] headersCenso = {"N°", "Nombre", "Apellidos", "Fecha Alta", "Teléfono"};

float[] colWidthsCenso = {10, 20, 30, 20, 20};

int[] maxCharsCenso = {5, 10, 20, 15, 10};

// Dades de la taula
String[][] infoCenso = {
```

```
 {"1", "Pere", "Soler Miralles De las Mercedes", "33", "Home"},
 {"2", "Maria", "Garcia Lopez", "25", "Dona"},
 {"3", "Joan Jose Maria", "Melis Cabrer", "47", "Home"},
 {"4", "Bel", "Riera Mates", "52", "Dona"},
 {"5", "Jose", "Perez Galdós", "37", "Home"},
 {"6", "Pere", "Soler Miralles", "33", "Home"},
 {"7", "Maria", "Garcia Lopez", "25", "Dona"},
 {"8", "Joan", "Melis Cabrer", "47", "Home"},
 {"9", "Bel", "Riera Mates", "52", "Dona"},
 {"10", "Jose", "Perez Galdós", "37", "Home"},
 {"11", "Pere", "Soler Miralles", "33", "Home"},
 {"12", "Maria", "Garcia Lopez", "25", "Dona"},
 {"13", "Joan", "Melis Cabrer", "47", "Home"},
 {"14", "Bel", "Riera Mates", "52", "Dona"},
 {"15", "Jose", "Perez Galdós", "37", "Home"},
};

int filasArchivo = 5, columnasArchivo = 3;
String[] headersArchivo = {"Título", "Fecha", "Categoría"};

float[] colWidthsArchivo = {40, 30, 30};

int[] maxCharsArchivo = {30, 20, 20};

// Dades de la taula
String[][] infoArchivo = {
 {"Cristo Antiguo", "1920", "Imágen"},
 {"Programa de fiestas", "2020", "Programa"},
 {"Boletín de inscripción", "1620", "Documento"},
 {"Cristo Antiguo", "1920", "Imágen"},
 {"Procesión COVID-19", "2021", "Vídeo"},
 {"Cristo Antiguo", "1920", "Imágen"},

};

int filasGastos = 4, columnasGastos = 3;

String[] headersGastos = {"Código", "Concepto", "Cantidad"};

float[] colWidthsGastos = {20, 60, 20};

int[] maxCharsGastos = {10, 30, 15};

// Dades de la taula
String[][] infoGastos = {
 {"G.1", "Servicios y mantenimiento ermita", "1200.00€"},
 {"G.2", "Aportación radio Santa Maria", "1200.00€"},
 {"G.3", "Caridad", "1200.00€"},
 {"G.4", "Misas hermanos difuntos", "1200.00€"},
 {"G.5", "Mantenimiento y adquisición patrimonio", "1200.00€"},
 {"G.6", "Fuegos artificiales", "1200.00€"},
 {"G.7", "Bandas de música y coros", "1200.00€"},
 {"G.8", "Flores", "1200.00€"},
 {"G.9", "Programa de fiestas y más imprenta", "1200.00€"},
 {"G.10", "Carne cena de hermandad", "1200.00€"},
 {"G.11", "Luz y sonido miserere", "1200.00€"},
 {"G.12", "Limonada y migas", "1200.00€"},
 {"G.13", "Gastos para organización de otros actos", "1200.00€"},
 {"G.14", "Premio carrozas y colaboraciones civiles", "1200.00€"},
 {"G.15", "Gastos Bancarios", "1200.00€"},
 {"G.16", "Alojamineto página web", "1200.00€"},
 {"G.17", "Varios y gastos extraordinarios", "1200.00€"},
 {"G.18", "Adquisición artículos devoción", "1200.00€"},
};
```

```
String[][] infoGastosPresupuesto = {
 {"G.1", "Servicios y mantenimiento ermita", "1200.00€"},
 {"G.2", "Aportación radio Santa Maria", "1200.00€"},
 {"G.3", "Caridad", "1200.00€"},
 {"G.4", "Misas hermanos difuntos", "1200.00€"},
 {"G.5", "Mantenimiento y adquisición patrimonio", "1200.00€"},
 {"G.6", "Fuegos artificiales", "1200.00€"},
 {"G.7", "Bandas de música y coros", "1200.00€"},
 {"G.8", "Flores", "1200.00€"},
 {"G.9", "Programa de fiestas y más imprenta", "1200.00€"},
 {"G.10", "Carne cena de hermandad", "1200.00€"},
 {"G.11", "Luz y sonido miserere", "1200.00€"},
 {"G.12", "Limonada y migas", "1200.00€"},
 {"G.13", "Gastos para organización de otros actos", "1200.00€"},
 {"G.14", "Premio carrozas y colaboraciones civiles", "1200.00€"},
 {"G.15", "Gastos Bancarios", "1200.00€"},
 {"G.16", "Alojamineto página web", "1200.00€"},
 {"G.17", "Varios y gastos extraordinarios", "1200.00€"},
 {"G.18", "Adquisición artículos devoción", "1200.00€"},
};
SelectTable stBalanceIngresos, stDetalleItem;

int filasBalanceIngresos = 4, columnasBalanceIngresos=3;
int filasDetalleItem = 4, columnasDetalleItem = 3;


String[] headersBalance = {"Código", "Concepto", "Cantidad"};
String[] headersDetalleItem = {"G.1", "GASTOS SERVICIOS Y MANTENIMIENTO ERMITA", "-1321.06"};

// Amplades de les columnes
float[] colWidthsBalance = {20, 50, 30};
float[] colWidthsDetalleItem = {10, 70, 20};

// Dades de la taula
String[][] infoBalanceIngresos = {
 {"I.1", "Donativos", "3907,35€"},
 {"I.2", "Cuotas hermanos", "3907,35€"},
 {"I.3", "Subvención ayuntamiento", "3907,35€"},
};
String[][] infoDetalleItem = {
 {"", "Rcbo. Iberdrola", "-234,25€"},
 {"", "Rcbo. Iberdrola", "-686,25€"},
 {"", "Rcbo. Iberdrola", "-123,25€"},
 {"", "Rcbo. Iberdrola", "-432,25€"},
 {"", "Rcbo. Iberdrola", "-369,25€"},
 {"", "Rcbo. Iberdrola", "-234,25€"},
};

int[] maxCharsBalanceIngresos = {10, 35, 15};
int[] maxCharsDetalleItem = {10, 50, 20};

void initSelectTable() {
 stBalanceIngresos = new SelectTable(filasBalanceIngresos, columnasBalanceIngresos, 250,
210, 800, 240);
 stBalanceIngresos.setHeaders(headersBalance);
 String[][] infoIngresos = getInfoTablaMovimientos("ingresos", 3);
 stBalanceIngresos.setData(infoIngresos);
 stBalanceIngresos.setColumnWidths(colWidthsBalance);
 stBalanceIngresos.setColumnMaxChars(maxCharsBalanceIngresos);
 stGastos = new SelectTable(filasGastos, columnasGastos, 250, 510, 800, 240);
 stGastos.setHeaders(headersGastos);
 stGastos.setData(getInfoTablaMovimientos("gastos", 18));
 //stGastos.setData(infoGastos);
 stGastos.setColumnWidths(colWidthsGastos);
 stGastos.setColumnMaxChars(maxCharsGastos);
```

```
 stCenso = new SelectTable(filasCenso, columnasCenso, 20+menuWidth, 285, 1280-menuWidth-40,
410);
 stCenso.setHeaders(headersCenso);
 stCenso.setData(getInfoTablaCenso());
 stCenso.setColumnWidths(colWidthsCenso);
 stCenso.setColumnMaxChars(maxCharsCenso);
 stGastosPresupuesto = new SelectTable(filasGastos, columnasGastos, 250, 510, 800, 240);
 stGastosPresupuesto.setHeaders(headersGastos);
 stGastosPresupuesto.setData(infoGastosPresupuesto);
 stGastosPresupuesto.setColumnWidths(colWidthsGastos);
 stGastosPresupuesto.setColumnMaxChars(maxCharsGastos);
 stDetalleItem = new SelectTable(filasDetalleItem, columnasDetalleItem, 45+menuWidth,
220+bannerHeight, 1000, 240);
 stDetalleItem.setHeaders(headersDetalleItem);
 stDetalleItem.setData(infoDetalleItem);
 stDetalleItem.setColumnWidths(colWidthsDetalleItem);
 stDetalleItem.setColumnMaxChars(maxCharsDetalleItem);
 stArchivo = new SelectTable(filasArchivo, columnasArchivo, 20+menuWidth, 285, 1280-
menuWidth-40, 410);
 stArchivo.setHeaders(headersArchivo);
 stArchivo.setData(getInfoTablaArchivo());
 stArchivo.setColumnWidths(colWidthsArchivo);
 stArchivo.setColumnMaxChars(maxCharsArchivo);
}

//CalendariPlus

CalendariPlus cpFechaNacimiento, cpFechaAlta, cpFechaMovimiento, cpFechaArchivo,
cpNuevoEvento;
String dataCalendariNacimiento=" ";
String dataCalendariAlta="";
String dataCalendariMovimiento="";
String dataCalendarioArchivo = "";
String dataCalendarioEvento = "";

void initCalendariPlus() {
 cpFechaNacimiento = new CalendariPlus(680, 250, 600, 380);
 cpFechaAlta = new CalendariPlus(680, 300, 600, 380);
 cpFechaMovimiento = new CalendariPlus(550, 410, 600, 380);
 //cpFechaArchivo = new CalendariPlus(300+menuWidth, 400+bannerHeight, 600, 300);
 cpNuevoEvento = new CalendariPlus (400, 300, 600, 380);
}

void displaycpFechaNacimiento() {
 pushStyle();
 // Rectangle
 fill(255);
 rect(680, 165+bannerHeight, 200, 45);

 // Text amb data seleccionada
 fill(0);
 textAlign(LEFT);
 textSize(24);
 text(dataCalendariNacimiento, 690, 165+bannerHeight+30);


 cpFechaNacimiento.display();
 bCalendario.display();

 popStyle();
}

void displaycpNuevoEvento() {
 pushStyle();
 // Rectangle
```

```
 fill(255);
 rect(680+menuWidth, 570+bannerHeight, 220, 45);

 // Text amb data seleccionada
 fill(0);
 textAlign(LEFT);
 textSize(24);
 text(dataCalendarioEvento, 685+menuWidth, 570+bannerHeight+30);

 cpNuevoEvento.display();
 bCalendarioEvento.display();

 popStyle();
}

void displaycpFechaAlta() {
 pushStyle();
 // Rectangle
 fill(255);
 rect(1080, 605+bannerHeight, 180, 45);

 // Text amb data seleccionada
 fill(0);
 textAlign(LEFT);
 textSize(24);
 text(dataCalendariAlta, 1082, 605+bannerHeight+30); //SE PINTA EL MATEIX QUE A
dataCalendari!!
 cpFechaAlta.display();
 bCalendarioAlta.display();
 popStyle();
}

void displaycpFechaMovimiento() {
 pushStyle();
 // Rectangle
 fill(255);
 rect(530, 410, 180, 45);

 // Text amb data seleccionada
 fill(0);
 textAlign(LEFT);
 textSize(24);
 text(dataCalendariMovimiento, 535, 410+30); //SE PINTA EL MATEIX QUE A dataCalendari!!
 popStyle();
}

//Table
Table tDetalleItem;

// Número de files (capçalera inclosa) i columnes de la taula
int filas = 1, columnas = 3;

// Títols de les columnes
String[] headers = {"G.1", "GASTOS SERVICIOS Y MANTENIMIENTO ERMITA", "-1332,05€"};

// Amplades de les columnes
float[] colWidths = {10, 70, 20};

// Dades de la taula
String[][] info = {};

void initTable() {
 tDetalleItem = new Table(1, 3);
 tDetalleItem.setHeaders(headers);
 tDetalleItem.setData(info);
```

```
 tDetalleItem.setColumnWidths(colWidths);
}


//TextArea

TextArea taNuevoAviso, taNuevoEvento;

void initTextArea() {
 taNuevoAviso = new TextArea (27+menuWidth, 250+bannerHeight, 1020, 260, 70, 10);
 taNuevoEvento = new TextArea (27+menuWidth, 250+bannerHeight, 1020, 260, 70, 10);
}

//pagedCard

PagedCard pcAvisos, pcAvisosPrincipal;

// Número de files (capçalera inclosa) i columnes de la taula
int numCardsPage = 4;

// Dades de la taula
String[][] infoCards = {
 {"Títol 1", "Descripció 1 Lorem Ipsum is simply dummy text of the printing and typesetting
industry. Lorem Ipsum has been the industry&amp;apos;s standard dummy text ever since the
1500s"},
 {"Títol 2", "Descripció 2"},
 {"Títol 3", "Descripció 3"},
 {"Títol 4", "Descripció 4"},
 {"Títol 5", "Descripció 5"},
 {"Títol 6", "Descripció 6"},
 {"Títol 7", "Descripció 7"},
 {"Títol 8", "Descripció 8"},
 {"Títol 9", "Descripció 9"},
 {"Títol 0", "Descripció 0"},
};


void initPagedCard() {
 pcAvisos = new PagedCard(numCardsPage);
 pcAvisos.setDimensions(menuWidth+20, primerIconY+(iconHeight-50), ((1280-menuWidth)/2)-10,
iconHeight*4-iconHeight);
 pcAvisos.setData(infoCards);
 pcAvisos.setCards();
 pcAvisosPrincipal = new PagedCard(8);
 pcAvisosPrincipal.setDimensions(menuWidth+540, bannerHeight+30, 520, 550);
 pcAvisosPrincipal.setData(infoCards);
 pcAvisosPrincipal.setCards();
}

PopUp PopUpinicioSesion;

void initPopUp() {
 PopUpinicioSesion  = new PopUp("Bienvenido, "+ userText.getValue(), "Usuario y contraseña
correctos", width/2-250, height/2-125, 500, 300);
}
```

ImgTextButton

```
class ImgTextButton {

 // Propietats d&amp;apos;un botó:
 float x, y, w, h; // Posició i dimensions
```

```
 PImage img;

 // Colores de contorno, fill, activo i desactivado
 color fillColor, strokeColor;
 color fillColorOver;

 boolean enabled; // Abilitat / desabilitat

 String textBoton;

 // Mètode Constructor
 ImgTextButton(PImage imgs, String textBoton, float x, float y, float w, float h) {
   this.img = imgs;
   this.x = x;
   this.y = y;
   this.w = w;
   this.h = h;
   this.textBoton = textBoton;
   this.enabled = true;
   fillColor = color(getColorAt(1));
   fillColorOver = color(getColorAt(0));
   strokeColor = color(0);
 }

 // Setters

 void setEnabled(boolean b) {
   this.enabled = b;
 }

 // Dibuixa el botó
 void display() {

   pushStyle();
   if (mouseOverButton()) {
     fill(fillColorOver); // Color cuando el mouse está encima
   } else {
     fill(fillColor); // Color activo sin mouse
   }
   stroke(strokeColor);
   strokeWeight(1); //Color i grosor del contorno
   rect(this.x, this.y, this.w, this.h, 10); // Rectangulo del botón

   // Texto (color, alineación i tamaño)
   fill(255);
   textAlign(CENTER);
   textFont(getFontAt(4));
   text(textBoton, this.x + this.w/2, this.y + this.h/2+30);
   imageMode(CENTER);
   image(img, this.x + this.w/2, this.y + this.h/2-20, this.w/3, this.h/3+10);
   popStyle();
 }

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

LinesDiagram

```
class LinesDiagram {

 // Dimensions del diagrama de Barres
 float x, y, w, h;

 // Informació del diagrama (textos, valors i colors)
 String[] texts;
 float[] values;
 color colorLines;
 float maxValue;

 // Constructor

 LinesDiagram(float x, float y, float w, float h) {
   this.x = x;
   this.y = y;
   this.w = w;
   this.h = h;
 }

 // Setters

 void setTexts(String[] t) {
   this.texts = t;
 }

 void setValues(float[] v) {
   this.values = v;
   this.maxValue = this.values[0];
   for (int i=0; i<values.length; i++) {
     if (this.values[i]>this.maxValue) {
       maxValue = this.values[i];
     }
   }
 }

 void setColors(color c) {
   this.colorLines = c;
 }

 // Dibuixa el Diagrama de Sectors

 void display() {
   pushStyle();
   textFont(getFontAt(4));

   stroke(0);
   strokeWeight(2);
   line(this.x, this.y, this.x, this.y + this.h);
   line(this.x, this.y + this.h, this.x + this.w, this.y + this.h);

   float widthBar = w / (float) this.values.length;

   for (int i=0; i<this.values.length-1; i++) {

     // Posició Mes i
     float barValue1 = this.y + this.h -map(this.values[i], 0, maxValue, 0, h-50);
     float xBar1 = this.x + widthBar*i + widthBar/2;

     // Posició Mes i+1
     float barValue2 = this.y + this.h - map(this.values[i+1], 0, maxValue, 0, h-50);
     float xBar2 = this.x + widthBar*(i+1) + widthBar/2;

     // Linia de mesos i a i+1
     stroke(colorLines);
```

```
      strokeWeight(2);
      line(xBar1, barValue1, xBar2, barValue2);

      // Quadradet del mes i
      noStroke();
      fill(colorLines);
      rectMode(CENTER);
      rect(xBar1, barValue1, 10, 10);

      //
      float textY = this.y + this.h + 50;
      textFont(getFontAt(4));
      fill(0);
      textAlign(CENTER);
      textSize(24);
      text(this.texts[i], xBar1, textY);

      textSize(24);
      text((int)this.values[i], xBar1, barValue1 - 20);

      if (i+1==this.values.length-1) {
        text(this.texts[i+1], xBar2, textY);
        text((int)this.values[i+1], xBar2, barValue2 - 20);

        noStroke();
        fill(colorLines);
        rectMode(CENTER);
        rect(xBar2, barValue2, 10, 10);
      }
    }
    popStyle();
  }
}
```

Media

```
// Archivo con la información de los medias de la App

// URL de la carpeta donde se encuentran los medias
String URL_IMGS = "imgs/";
String URL_VECTORS = "vectors/";

// Array de imágenes
PImage[] imgs;
// Array de imágenes vectoriales  (SVG)
PShape[] shapes;

// Establece las imágenes de la App
void setMedias() {
 this.imgs = new PImage[13];
 this.imgs[0] = loadImage(URL_IMGS+"logo.png");
 this.imgs[1] = loadImage(URL_IMGS+"mantonmanila.jpeg");
 this.imgs[2] = loadImage(URL_IMGS+"archivo.png");
 this.imgs[3] = loadImage(URL_IMGS+"avisos.png");
 this.imgs[4] = loadImage(URL_IMGS+"censo.png");
 this.imgs[5] = loadImage(URL_IMGS+"contabilidad.png");
 this.imgs[6] = loadImage(URL_IMGS+"enlaces.png");
 this.imgs[7] = loadImage(URL_IMGS+"file.png");
 this.imgs[8] = loadImage(URL_IMGS+"user.png");
 this.imgs[9] = loadImage(URL_IMGS+"cristo1.jpg");
 this.imgs[10] = loadImage(URL_IMGS+"cristo2.jpeg");
 this.imgs[11] = loadImage(URL_IMGS+"cristo3.jpeg");
 this.imgs[12] = loadImage(URL_IMGS+"cristo4.jpeg");
}
void setVectors() {
```

```
 this.shapes = new PShape[2];
 this.shapes[0] = loadShape(URL_VECTORS+"logoNegro.svg");
 this.shapes[1] = loadShape(URL_VECTORS+"logoBlanco.svg");
}

// Getter del número d&apos;imatges
int getNumImatges() {
 return this.imgs.length;
}

//getter de la imagen del logo (.png)
PImage getLogo() {
 return this.imgs[0];
}

// Getter de la imagen del fondo (mantón de manila)
PImage getFondoManton() {
 return this.imgs[1];
}

// Getter del icono de la pestaña archivo
PImage getIconArchivo() {
 return this.imgs[2];
}

// Getter del icono de la pestaña Avisos
PImage getIconAvisos() {
 return this.imgs[3];
}

// Getter del icono de la pestaña Censo
PImage getIconCenso() {
 return this.imgs[4];
}

// Getter del icono de la pestaña Contabilidad
PImage getIconContabilidad() {
 return this.imgs[5];
}

// Getter del icono de la pestaña archivo
PImage getIconEnlaces() {
 return this.imgs[6];
}

// Getter del icono file
PImage getIconFile() {
 return this.imgs[7];
}

// Getter del icono user
PImage getIconUser() {
 return this.imgs[8];
}


// Getter de la imagen i-ésima
PImage getImgAt(int i) {
 return this.imgs[i];
}

// Getter del número d&apos;imatges vectorials
int getNumVectors() {
 return this.shapes.length;
}
```

```
// Getter de la imatge del logo
PShape getSVGLogo() {
 return this.shapes[1];
}


// Dibuja el logo
void displayLogo() {
 image(getLogo(), 30, 20, logoWidth, logoHeight);
}

// Dibuja la imágen i-ésima
void displayImg(int i, float x, float y, float w, float h) {
 image(getImgAt(i), x, y, w, h);
}
```

OpenPdf

```
//Nombre de los archivos PDF
String titulo1="";
String titulo2="CATALOGO.pdf";

// carpeta donde se ubican los archivos
String ruta =
"/Users/xiscopolgonzalez/Desktop/Cristo_de_la_SalApp/Cristo_de_la_SalApp/data/";
```

Option

```
// Classe Option

class Option {

// Propietats d&apos;un option:
float x, y, w, h; // Posició i dimensions
// Colors de contorn, farciment, actiu i desactiu
color fillColor, strokeColor;
color fillColorOver, fillColorDisabled;
String textBoto; // Text
boolean enabled; // Abilitat / desabilitat

// Mètode Constructor
Option(String text, float x, float y, float w, float h){
  this.textBoto = text;
  this.x = x;
  this.y = y;
  this.w = w;
  this.h = h;
  this.enabled = true;
  fillColor = color(255);
  fillColorOver = color(155, 155, 155);
  fillColorDisabled = color(150);
  strokeColor = color(0);
}

// Setters

void setEnabled(boolean b){
  this.enabled = b;
}

// 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
  }
  noStroke();
  rect(this.x, this.y, this.w, this.h); // Rectangle de l&apos;option

  // Text (color, alineació i mida)
  fill(0); textAlign(CENTER); textSize(34);
  text(textBoto, this.x + this.w/2, this.y + this.h/2 + 10);
  popStyle();
}

// Indica si el cursor està sobre l&apos;option
boolean mouseOverButton(){
  return (mouseX >= this.x) &&
         (mouseX<=this.x + this.w) &&
         (mouseY>= this.y) &&
         (mouseY<= this.y + this.h);
}

}
```

PagedCard

```
class PagedCard {


 String[][] cardsData; // Dades de les Cards
 Card[] cards;             // Cards
 int numCards; // Número total de Cards
 int numCardsPage; // Número de Cards en 1 Pàgina

 int numPage;
 int numTotalPages;

 float x, y, w, h;
 int selectedCard = -1;

 // Constructor
 PagedCard(int ncp) {
   this.numCardsPage = ncp;
   this.numPage = 0;
 }

 // Setters

 void setDimensions(float x, float y, float w, float h) {
   this.x = x;
   this.y = y;
   this.w = w;
   this.h = h;
 }

 void setData(String[][] d) {
   this.cardsData = d;
   this.numTotalPages = d.length / this.numCardsPage;
 }

 void setCards() {

   cards = new Card[this.cardsData.length];
```

```
  for (int np=0; np<=numTotalPages; np++) {

    int firstCardPage = numCardsPage*np;
    int lastCardPage = numCardsPage*(np+1) - 1;
    float hCard = h / (float) numCardsPage;
    float yCard = y;
    float b = 10;

    for (int i = firstCardPage; i <= lastCardPage; i++) {
      if (i<cards.length) {
        cards[i] = new Card(cardsData[i]);
        cards[i].setDimensions(x, yCard, w, hCard, b);
        yCard += hCard + b;
      }
    }
  }
}

void nextPage() {
  if (this.numPage<this.numTotalPages) {
    this.numPage++;
  }
}

void prevPage() {
  if (this.numPage>0) {
    this.numPage--;
  }
}

// Dibuixa taula
void display() {
  pushStyle();

  // Dibuixa Cards corresponent a la Pàgina
  int firstCardPage = numCardsPage*numPage;
  int lastCardPage = numCardsPage*(numPage+1) - 1;
  for (int i = firstCardPage; i <= lastCardPage; i++) {
    if (i<cards.length && cards[i]!=null) {
      cards[i].display(i==this.selectedCard);
    }
  }
  popStyle();
}

void checkCardSelection(){

  boolean selected = false;
  int firstCardPage = numCardsPage*numPage;
  int lastCardPage = numCardsPage*(numPage+1) - 1;
  for (int i = firstCardPage; i <= lastCardPage; i++) {
    if (i<cards.length && cards[i]!=null && cards[i].mouseOver()) {
      selectedCard = i;
      println("Selected Card: "+i);
      selected = true;
      break;
    }
  }
  if(!selected){
    selectedCard = -1;
  }
}

boolean checkMouseOver(){
```

```
    int firstCardPage = numCardsPage*numPage;
    int lastCardPage = numCardsPage*(numPage+1) - 1;
    for (int i = firstCardPage; i <= lastCardPage; i++) {
      if (i<cards.length && cards[i]!=null && cards[i].mouseOver()) {
        return true;
      }
    }
    return false;
  }

  void printSelectedCard(){
    if(selectedCard !=-1){
      Card cSelected = cards[selectedCard];
      fill(0); textSize(18);
      text("Seleccionada: ", 900, 300);
      textSize(24);
      text(cSelected.title, 900, 340);
    }
  }
}
```

PagedTable

```
class PagedTable {

  String[] tableHeaders; // Títols de les columnes
  String[][] tableData; // Dades de la taula
  float[] columnWidths; // Amplades de les columnes
  int[] maxCaracters; // Maxim de lletres de les columnes

  int numCols, numRows; // Número de files i columnes

  int numPage;
  int numTotalPages;

  // Constructor
  PagedTable(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 = d.length / (this.numRows-1);
  }

  void setValueAt(String value, int nr, int nc){
    this.tableData[nr][nc] = value;
  }

  void setColumnWidths(float[] w){
    this.columnWidths = w;
  }

  void setColumnMaxChars(int[] c){
    this.maxCaracters = c;
  }
```

```
void nextPage(){
  if(this.numPage<this.numTotalPages){
    this.numPage++;
  }
}

void prevPage(){
  if(this.numPage>0){
    this.numPage--;
  }
}

// Dibuixa taula
void display(float x, float y, float w, float h){

  pushStyle();

  fill(200, 50); stroke(0);strokeWeight(3);
  rect(x, y, w, h);

  float rowHeight = h / numRows;
  fill(getColorAt(0)); stroke(0);strokeWeight(3);
  rect(x, y, w, rowHeight);

  // Dibuixa files
  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);
  }

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

  // Dibuixa textos
  fill(0); textSize(24);
  for(int r = 0; r < numRows; r++){
    xCol = x;
    for(int c = 0; c< numCols; c++){
      if(r==0){
       pushStyle();
        fill(255);
        text(tableHeaders[c], xCol + 10, y + (r+1)*rowHeight - 10);
        popStyle();
      }
      else{
        int k = (numRows-1)*numPage + (r-1);
        if(k<tableData.length){
          String t = retallaText(tableData[k][c], maxCaracters[c]);
          text(t, xCol + 10, y + (r+1)*rowHeight - 10);
        }
      }
      xCol += w*columnWidths[c]/100.0;
    }
  }

  // Informació de la Pàgina
  fill(0);
  //text("Pag: "+(this.numPage+1)+" / "+(this.numTotalPages+1), x, y + h + 50);
```

```
    popStyle();
  }


  String retallaText(String allText, int maxLength){
    String t = allText.substring(0, min(maxLength, allText.length()));
    if(!allText.equals(t)){
       t+="...";
    }
    return t;
  }


}
```

PopUp

```
class PopUp {

  // Dimensions
  float x, y, w, h;

// Propietats
PImage img;
String title;
String message;

Button bAceptar;
float buttonW = 200;
float buttonH = 40;

boolean visible = true;

// Constructor

PopUp(String title, String message, float x, float y, float w, float h){
  this.title = title;
  this.message = message;
  this.x = x; this.y = y;
  this.w = w; this.h = h;
  this.bAceptar = new Button("Aceptar", x + w/2 - buttonW/2,
                                        y + h - buttonH*1.5,
                                        buttonW, buttonH);
}

//Setters

void setImage(PImage img){
  this.img = img;
}

void setTexts(String title, String message){
  this.title = title;
  this.message = message;
}

void setVisible(boolean b){
  this.visible = b;
  if(!this.visible){
    this.bAceptar.setEnabled(false);
  }
  else {
    this.bAceptar.setEnabled(true);
  }
}
```

```
// Dibuixa el PopUp

void display(){

  if(this.visible){
    float b = 40;

    pushStyle();

    // Rectangle
    stroke(0); strokeWeight(10);fill(colors[1]);
    rect(x, y, w, h, b/2);

    line(x, y + 2*b , x+w, y + 2*b);

    // Títol
    fill(255); textSize(38); textAlign(LEFT);
    text(title, x + b, y + 1.4*b);

    // Missatge
    fill(255);textSize(24); textAlign(CENTER);
    text(message, x + w/2, y + 4*b);

    // Botó d&apos;Acceptar
    bAceptar.display();
    popStyle();
  }
}


}
```

Select

```
class Select {

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

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

 float lineSpace = 5; // Espai entre línies

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

   this.texts = texts;
   this.filteredTexts = 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(getColorAt(1));
    rect(x + w - 30, y, 30, h);

    fill(255); stroke(0);
    triangle(x + w - 25, y+5, x + w - 15, y + 25, x + w - 5 , y+5);

    fill(0); textSize(14);
    textFont(getFontAt(7));
    text(selectedValue, x + 10, y + 25);

    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);
        textFont(getFontAt(7));
        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 mouseOverSelect(){
  if(this.collapsed){
    return (mouseX >= x) &&
           (mouseX <= x + w) &&
           (mouseY >= y) &&
           (mouseY <= y + h);
  }
  else {
    return (mouseX>= x) &&
           (mouseX<= x + w) &&
           (mouseY>= y) &&
           (mouseY<= y + h + (h + lineSpace)*texts.length);
  }
}

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

```
}
```

SelectTable

```
class SelectTable {

 String[] tableHeaders; // Títols de les columnes
 String[][] tableData; // Dades de la taula
 float[] columnWidths; // Amplades de les columnes
 int[] maxCaracters; // Maxim de lletres de les columnes

 int numCols, numRows; // Número de files i columnes

 int numPage;
 int numTotalPages;

 int selectedRow = -1;

 float x, y, w, h;
 float rowHeight;

 // Constructor
 SelectTable(int nr, int nc, float x, float y, float w, float h) {
   this.numRows = nr;
   this.numCols = nc;
   this.numPage = 0;
   this.x = x;
   this.y = y;
   this.h = h;
   this.w = w;
   this.rowHeight = h / nr;
 }

 // Setters

 void setHeaders(String[] h) {
   this.tableHeaders = h;
 }

 void setData(String[][] d) {
   this.tableData = d;
   if (d.length % (this.numRows-1)==0) {
     this.numTotalPages = (d.length / (this.numRows-1)) -1;
   } else {
     this.numTotalPages = (d.length / (this.numRows-1)) ;
   }
 }

 void setValueAt(String value, int nr, int nc) {
   this.tableData[nr][nc] = value;
 }

 void setColumnWidths(float[] w) {
   this.columnWidths = w;
 }

 void setColumnMaxChars(int[] c) {
   this.maxCaracters = c;
 }

 void nextPage() {
   if (this.numPage<this.numTotalPages) {
     this.numPage++;
   }
```

```
}

void prevPage() {
  if (this.numPage>0) {
    this.numPage--;
  }
}

// Dibuixa taula
void display() {

  pushStyle();

  fill(200, 50);
  stroke(0);
  strokeWeight(3);
  rect(x, y, w, h);

  fill(getColorAt(0));
  stroke(0);
  strokeWeight(3);
  rect(x, y, w, rowHeight);

  // Dibuixa files
  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);
  }

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

  // Dibuixa textos
  fill(0);
  textSize(24);
  for (int r = 0; r < numRows; r++) {
    xCol = x;
    for (int c = 0; c< numCols; c++) {
      if (r==0) {
        pushStyle();
        fill(255);
        text(tableHeaders[c], xCol + 10, y + (r+1)*rowHeight - 10);
        popStyle();
      } else {
        int k = (numRows-1)*numPage + (r-1);
        if (k<tableData.length) {
          String t = retallaText(tableData[k][c], maxCaracters[c]);
          if (k == selectedRow) {
            fill(255, 0, 0, 50);
            rect(xCol, y + (r)*rowHeight, columnWidths[c]*w/100, rowHeight);
            fill(255, 0, 0);
          } else {
            fill(0);
          }
          text(t, xCol + 10, y + (r+1)*rowHeight - 10);
        }
```

```
        }
        xCol += w*columnWidths[c]/100.0;
      }
    }

    // Informació de la Pàgina
    fill(0);
    //text("Pag: "+(this.numPage+1)+" / "+(this.numTotalPages+1), x, y + h + 50);

    popStyle();
  }


  String retallaText(String allText, int maxLength) {
    String t = allText.substring(0, min(maxLength, allText.length()));
    if (!allText.equals(t)) {
      t+="...";
    }
    return t;
  }

  boolean clickOnTableRow(int nr) {
    return mouseX>= x && mouseX<= x+this.w &&
      mouseY>= y + nr*rowHeight && mouseY<= y + (nr+1)*rowHeight;
  }

  void checkSelections() {
    for (int r = 0; r < numRows; r++) {
      if (clickOnTableRow(r)) {
        if (selectedRow==-1) {
          selectedRow = (r-1) + (numRows-1)*numPage;
        } else {
          selectedRow=-1;
        }
      }
    }
  }

  String getSelectedInfoId() {
    return this.tableData[selectedRow][1];
  }

  String[] getSelectedInfo() {
    return this.tableData[selectedRow];
  }

  String stringSelected() {
    String txt="";
    if (selectedRow!= -1) {
      String[] info = getSelectedInfo();
      for (int i=0; i<info.length; i++) {
        txt+= info[i]+"\n";
      }
    }
    return txt;
  }
}
```

SelectTextList

```
class SelectTextList {

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

  TextField textField;       // Camp de text
```

```
String selectedId; // Id Seleccionat
String selectedValue; // Valor Seleccionat

boolean enabled; // Abilitat / desabilitat

int numMatchs = 0;
ArrayList<Option> options;

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

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

  this.textField = new TextField((int)x, (int)y, (int)w, (int)h);
  this.options = new ArrayList<Option>();
}

void display() {
  pushStyle();
  textField.display();

  fill(100);
  rect(x + w - 50, y, 50, h);

  fill(0);
  stroke(0);
  triangle(x + w - 45, y+5, x + w - 25, y + 35, x + w - 5, y+5);

  for (Option b : this.options) {
    b.display();
  }
  popStyle();
}

void update() {

  String searchFor = this.textField.text;

  this.numMatchs = 0;
  this.options = new ArrayList<Option>();

  if (this.textField.selected) {

    if (searchFor.length() > 0) {
      for (int i=0; i<texts.length; i++) {
        if (texts[i][1].startsWith(searchFor)) {
          Option b = new Option(texts[i][1], x, y + h + 5 + (h + 5)*numMatchs, w, h);
          options.add(b);
          this.numMatchs++;
          if (this.numMatchs==5) {
            break;
          }
        }
      }
    } else {
      for (int i=0; i<texts.length; i++) {
        Option b = new Option(texts[i][1], x, y + h + 5 + (h + 5)*i, w, h);
        options.add(b);
```

```
          if (i==3) {
            break;
          }
        }
      }
    } else {
      options.clear();
    }
  }

  boolean mouseOverButtons() {
    for (Option b : options) {
      if (b.mouseOverButton()) {
        return true;
      }
    }
    return false;
  }

  void buttonPressed() {
    boolean pressed = false;
    for (Option b : options) {
      if (b.mouseOverButton()) {
        this.textField.text = b.textBoto;
        this.selectedValue = b.textBoto;
        for (int i=0; i<texts.length; i++) {
          if (texts[i][1].equals(this.selectedValue)) {
            this.selectedId = texts[i][0];
            break;
          }
        }
        pressed = true;
      }
    }
    if (pressed) {
      options.clear();
    }
  }


  void mouseOn() {
    // Pitjam damunt el textList
    this.textField.isPressed();
    this.buttonPressed();

    if (this.textField.selected) {
      this.update();
    } else {
      this.options.clear();
    }
  }

  void keyOn() {
    if (this.textField.selected) {
      this.textField.keyPressed(key, (int)keyCode);
      this.update();
    }
  }
}
```

ShowImage

```
class ShowImage {

  // Dimensions
```

```
 int x, y, w, h;

 // Index imatge actual
 int currentImage;

 // Títols de les imatges
 String[] noms;

 // Imatges
 PImage[] imgs;

 // Temps
 int tempsImg = 100;

 // Constructor
 ShowImage(int x, int y, int w, int h){
   this.x = x; this.y = y; this.w = w; this.h = h;
   this.currentImage = 0;
 }

 // Setters

 void setImages(String[] noms){
   this.noms = noms;
   this.imgs = new PImage[noms.length];
   for(int i=0; i<imgs.length; i++){
     imgs[i] = loadImage(URL_IMGS+noms[i]);
   }
 }


 void next(){
   if(this.currentImage<this.imgs.length-1){
     this.currentImage++;
   }
   else {
     this.currentImage=0;
   }
 }


 // Dibuixa la Imatge
 void display(){
   pushStyle();
   fill(150); stroke(0);
   rect(x-5, y-5, w+10, h+10);

   // Imatge a mostrar
   PImage img = imgs[currentImage];
   image(img, x, y, w, h);


  // Passa a la següent imatge
  if(frameCount%tempsImg==0){
    next();
  }
 }

}
```

Sizes

```
// Archivo con la información de medidas de objetos de la GUI

float medidaTitulo = 60;
float medidaSubtitulo = 40;
```

```
float medidaParrafo = 16;

// Dimensiones Barra Menú Lateral
int menuWidth = 200,
 menuHeight = 800;

// Dimensiones Banner
int bannerWidth = 1280,
 bannerHeight = 100;

// Dimensiones logo
int logoWidth = 140,
 logoHeight = 140;

//dimensiones iconos del menú
int iconWidth = 200,
 iconHeight = ((menuHeight - (logoHeight+40))/5);

//posición incial de los iconos del menú
int primerIconY = 180,
 segundoIconY = iconHeight+primerIconY,
 tercerIconY = iconHeight+segundoIconY,
 cuartoIconY = iconHeight+tercerIconY,
 quintoIconY = iconHeight+cuartoIconY;

//dimensiones marco inicio de sesión
int marcoWidth = 650, marcoHeight= 490;

//posicion marco inicio de sesión
int inicioSesionX=(1280/2)-(marcoWidth/2);
int inicioSesionY= (800/2)-(marcoHeight/2);

//dimensiones marco insertar usuario y contraseña
int marcoCuentaWidth = 500;
int marcoCuentaHeight = 250;

// Dimensiones de los botones de Enlaces
float buttonEnlaceW = 335;
float buttonEnlaceH = 67;
```

Table

```
class Table {

 String[] tableHeaders; // Títols de les columnes
 String[][] tableData; // Dades de la taula
 float[] columnWidths; // Amplades de les columnes

 int numCols, numRows; // Número de files i columnes

 // Constructor
 Table(int nr, int nc){
   this.numRows = nr;
   this.numCols = nc;
 }

 // Setters

 void setHeaders(String[] h){
   this.tableHeaders = h;
 }

 void setData(String[][] d){
   this.tableData = d;
 }
```

```
 void setValueAt(String value, int nr, int nc){
   this.tableData[nr][nc] = value;
 }

 void setColumnWidths(float[] w){
   this.columnWidths = w;
 }

 // Dibuixa taula
 void display(float x, float y, float w, float h){

   fill(200, 50); stroke(0);strokeWeight(3);
   rect(x, y, w, h);

   float rowHeight = h / numRows;
   fill(getColorAt(0)); stroke(0);strokeWeight(3);
   rect(x, y, w, rowHeight);

   // Dibuixa files
   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);
   }

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

   // Dibuixa textos
   fill(0); textSize(24);
   for(int r = 0; r< numRows; r++){
     xCol = x;
     for(int c = 0; c< numCols; c++){
       if(r==0){
         pushStyle();
         fill(255);
         text(tableHeaders[c], xCol + 10, y + (r+1)*rowHeight - 10);
         popStyle();
       }
       else{
         text(tableData[r-1][c], xCol + 10, y + (r+1)*rowHeight - 10);
       }
       xCol += w*columnWidths[c]/100.0;
     }
   }
 }


}
```

TextArea

```
// Component Camp de Text

class TextArea {

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

```
// Colors
color bgColor = color(getColorAt(5));
color fgColor =color(0, 0, 0);
color selectedColor = color(getColorAt(4));
color borderColor = color(30, 30, 30);
int borderWeight = 1;

// Text del camp
String text = "";
String[] lines;
int textSize = 24;

boolean selected = false;

// Constructor
TextArea(int x, int y, int w, int h, int nc, int nr) {
  this.x = x;
  this.y = y;
  this.w = w;
  this.h = h;
  this.numCols = nc;
  this.numRows = nr;

  this.lines = new String[nr];
}

// Dibuixa el Camp de Text
void display() {
  pushStyle();
  if (selected) {
    fill(selectedColor);
  } else {
    fill(bgColor);
  }

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

  fill(fgColor);
  textSize(textSize);
  for (int i=0; i<lines.length; i++) {
    if (lines[i]!=null) {
      text(lines[i], x + 5, y + (i+1)*textSize);
    }
  }
  popStyle();
}

void updateLines() {
  if (text.length()>0) {
    int numLines = constrain(text.length() / numCols, 0, this.numRows-1);
    println("NUM LINES: "+numLines);
    for (int i=0; i<=numLines; i++) {
      int start = i*numCols;
      int end = min(text.length(), (i+1)*numCols);
      lines[i] = text.substring(start, end);
    }
  } else {
    for (int i=0; i<lines.length; i++) {
      lines[i] ="";
    }
  }
  printArray(lines);
}
```

```
 // 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(&amp;apos; &amp;apos;); // SPACE
     } else if (keyCode == ENTER) {
     } else {
       addText(key);
     }
   }
 }

 // Afegeix la lletra c al final del text
 void addText(char c) {
   if (this.text.length() < this.numCols*this.numRows) {
     this.text += c;
     println("TEXT:"+this.text);
   }
   updateLines();
 }

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

 // 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;
   }
 }
}
```

TextField

```
// Component Camp de Text

class TextField {

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

 // Colors
 color bgColor = color(getColorAt(5));
 color fgColor = color(0, 0, 0);
 color selectedColor = color(getColorAt(4));
```

```
color borderColor = color(30, 30, 30);
int borderWeight = 1;

// Text del camp
String text = "";
String textDefault = "";
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;
}

TextField(String t, int x, int y, int w, int h) {
  this.x = x;
  this.y = y;
  this.w = w;
  this.h = h;
  this.text=t;
  this.textDefault = t;
}

// Getters

String getValue() {
  return this.text;
}


// 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);
  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(&apos; &apos;); // SPACE
    } else {

      boolean isKeyCapitalLetter = (key >= &apos;A&apos; && key <= &apos;Z&apos;);
      boolean isKeySmallLetter = (key >= &apos;a&apos; && key <= &apos;z&apos;);
      boolean isKeyNumber = (key >= &apos;0&apos; && key <= &apos;9&apos;) || (key==&apos;.
```

```
&apos;);
       boolean isSpecialKey = (keyCode==50);
       boolean isKeyAcento = (keyCode==65 || keyCode==69 || keyCode==73 || keyCode==79 ||
         keyCode==85);
       if (isKeyCapitalLetter || isKeySmallLetter || isKeyNumber || isSpecialKey ||
isKeyAcento) {
         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--;
   }
 }

 // Lleva tot el text
  void removeAllText(){
    this.text = "";
  }

 // 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;
     if (this.text==this.textDefault) {
       this.text="";
     }
   } else {
     selected = false;
     if (this.text=="") {
       this.text=this.textDefault;
     }
   }
 }
}
```

TextInfo

```
class TextInfo {

 float x, y, w, h;
 String text;
```

```
 int c = 5;

 TextInfo(String t, float x, float y, float w, float h) {

   this.text = t;
   this.x = x;
   this.y = y;
   this.w = w;
   this.h = h;
 }

 void setColor(int c) {
   this.c = c;
 }

 void display() {
   pushStyle();
   pushMatrix();
   stroke(0);
   strokeWeight(2);
   fill(getColorAt(c));
   rect(x, y, w, h);
   textFont(getFontAt(5));
   fill(0);
   text(this.text, x + 5, y + 10, w, h);
   popStyle();
   popMatrix();
 }
}
```

Titulo

```
class Titulo {

 float x, y, w, h;
 String text;


 Titulo(String t, float x, float y, float w, float h) {

   this.text = t;
   this.x = x;
   this.y = y;
   this.w = w;
   this.h = h;
 }


 void display() {
   pushStyle();
   pushMatrix();
   stroke(0);
   strokeWeight(2);
   fill(getColorAt(3));
   rect(x, y, w, h);
   textFont(getFontAt(5));
   fill(0);
   text(this.text, x + 10, y + 27);
   popStyle();
   popMatrix();
 }
}
```

Web

```
import java.awt.Desktop;
import java.net.URI;
```

```
import java.io.IOException;
import java.net.URISyntaxException;

Desktop desktop;

void openWebPage(String siteUrl) {
 try {
   URI site = new URI(siteUrl);
   if (Desktop.isDesktopSupported() && desktop.isSupported(Desktop.Action.BROWSE)) {
     desktop.browse(site);
   } else {
     println("App no suporta el navegador");
   }
 }
 catch(URISyntaxException e) {
   e.printStackTrace();
 }
 catch(IOException e) {
   e.printStackTrace();
 }
}
```

keyPressed

```
// Quan pitjam tecla
void keyPressed() {
 //INICIO DE SESIÓN
 if (pantalla==PANTALLA.INICIO) {
   userText.keyPressed(key, (int)keyCode);
   passText.keyPressed(key, (int)keyCode);
 }
 //CENSO
 else if (pantalla==PANTALLA.CENSO) {
   buscar.keyPressed(key, (int)keyCode);
 } else if (pantalla==PANTALLA.CENSO_NUEVOHERMANO) {
   tfNombre.keyPressed(key, (int)keyCode);
   tfApellidos.keyPressed(key, (int)keyCode);
   tfDNI.keyPressed(key, (int)keyCode);
   tfCalle.keyPressed(key, (int)keyCode);
   tfNumero.keyPressed(key, (int)keyCode);
   tfPiso.keyPressed(key, (int)keyCode);
   tfLocalidad.keyPressed(key, (int)keyCode);
   tfProvincia.keyPressed(key, (int)keyCode);
   tfTelefono.keyPressed(key, (int)keyCode);
   tfCorreoElectronico.keyPressed(key, (int)keyCode);
   tfBanco.keyPressed(key, (int)keyCode);
   tfTitular.keyPressed(key, (int)keyCode);
   tfDNITitular.keyPressed(key, (int)keyCode);
   tfIBAN.keyPressed(key, (int)keyCode);
   tfEntidad.keyPressed(key, (int)keyCode);
   tfOficina.keyPressed(key, (int)keyCode);
   tfDigitoControl.keyPressed(key, (int)keyCode);
   tfNumeroCuenta.keyPressed(key, (int)keyCode);

   //CONTABILIDAD Añadir Concepto
 } else if (pantalla == PANTALLA.CONTABILIDAD_AÑADIRCONCEPTO) {
   tfTitulo.keyPressed(key, (int)keyCode);
 }
 //Archivo Nuevo
 else if (pantalla == PANTALLA.ARCHIVO_NUEVO) {
   tfTituloArchivo.keyPressed(key, (int)keyCode);
   tfAñoDatacion.keyPressed(key, (int)keyCode);
 }
 //Nuevo Aviso
 else if (pantalla == PANTALLA.AVISOS_NUEVOAVISO) {
   tfTituloAviso.keyPressed(key, (int)keyCode);
```

```
    taNuevoAviso.keyPressed(key, (int)keyCode);
  }
  //Nuevo Evento
  else if (pantalla == PANTALLA.AVISOS_NUEVOEVENTO) {
    tfTituloEvento.keyPressed(key, (int)keyCode);
    taNuevoEvento.keyPressed(key, (int)keyCode);
  }
  comprovaLogin();
  lastKeyCodePressed= (int)keyCode;

  // Mes anterior
  if (keyCode==LEFT) {
    cEventos.prevMonth();
    println("PREV MONTH");
  }
  // Mes calendario posterior
  else if (keyCode==RIGHT) {
    cEventos.nextMonth();
    println("PREV MONTH");
  }
  stlTipoConcepto.keyOn(); // Tipo de Concepto
}
```

mousePressed

```
void mousePressed() {
  if (bNextCenso.mouseOverButton() && bNextCenso.enabled) {
    stCenso.nextPage();
  } else if (bPrevCenso.mouseOverButton() && bPrevCenso.enabled) {
    stCenso.prevPage();
  } else if (bNextGastos.mouseOverButton() && bNextGastos.enabled && pantalla == PANTALLA.
CONTABILIDAD_BALANCE) {
    stGastos.nextPage();
  } else if (bPrevGastos.mouseOverButton() && bPrevGastos.enabled && pantalla == PANTALLA.
CONTABILIDAD_BALANCE) {
    stGastos.prevPage();
  } else if (bNextGastos.mouseOverButton() && bNextGastos.enabled && pantalla == PANTALLA.
CONTABILIDAD_PRESUPUESTO) {
    stGastosPresupuesto.nextPage();
  } else if (bPrevGastos.mouseOverButton() && bPrevGastos.enabled && pantalla == PANTALLA.
CONTABILIDAD_PRESUPUESTO) {
    stGastosPresupuesto.prevPage();
  } else if (bPrevDetalle.mouseOverButton() && bPrevDetalle.enabled) {
    stDetalleItem.prevPage();
  } else if (bNextDetalle.mouseOverButton() && bNextDetalle.enabled) {
    stDetalleItem.nextPage();
  } else if (itbCenso.mouseOverButton() && itbCenso.enabled) {
    pantalla = PANTALLA.CENSO;
  } else if (itbContabilidad.mouseOverButton() && itbContabilidad.enabled) {
    estadoDeCuentas = getEstadoCuentas();
    pantalla = PANTALLA.CONTABILIDAD;
  } else if (itbArchivo.mouseOverButton() && itbArchivo.enabled) {
    pantalla = PANTALLA.ARCHIVO;
  } else if (itbAvisos.mouseOverButton() && itbAvisos.enabled) {
    pantalla = PANTALLA.AVISOS;
  } else if (itbEnlaces.mouseOverButton() && itbEnlaces.enabled) {
    pantalla = PANTALLA.ENLACES;
  } else if (bPrincipal.mouseOverButton() && bPrincipal.enabled) {
    pantalla = PANTALLA.PRINCIPAL;
  } else if (bBalance.mouseOverButton() && bBalance.enabled) {
    pantalla = PANTALLA.CONTABILIDAD_BALANCE;
  } else if (bPresupuesto.mouseOverButton() && bPresupuesto.enabled) {
    pantalla = PANTALLA.CONTABILIDAD_PRESUPUESTO;
  } else if (bFacebook.mouseOverButton() && bFacebook.enabled) {
    openWebPage("https://www.facebook.com/stmocristodelasalabargas/");
  } else if (bTwitter.mouseOverButton() && bTwitter.enabled) {
```

```
    openWebPage("https://twitter.com/cristo_sala");
  } else if (bInstagram.mouseOverButton() && bInstagram.enabled) {
    openWebPage("https://www.instagram.com/hermandadcristodelasala/?hl=es");
  } else if (bYoutube.mouseOverButton() && bYoutube.enabled) {
    openWebPage("https://www.youtube.com/channel/UCri0gUrGJPZ23ZlmgmGa9Yg");
  } else if (bArzobispado.mouseOverButton() && bArzobispado.enabled) {
    openWebPage("https://www.architoledo.org/");
  } else if (bAyuntamiento.mouseOverButton() && bAyuntamiento.enabled) {
    openWebPage("https://www.bargas.es/");
  } else if (bWebCofrade.mouseOverButton() && bWebCofrade.enabled) {
    openWebPage("http://www.semanasantatoledo.com/");
  } else if (bOtrasHermandades.mouseOverButton() && bOtrasHermandades.enabled) {
    openWebPage("https://www.humildadtoledo.com/enlaces-de-interes");
  } else if (bAñadir.mouseOverButton() && bAñadir.enabled && pantalla == PANTALLA.CENSO) {
    pantalla = PANTALLA.CENSO_NUEVOHERMANO;
  } else if (bAñadirConcepto.mouseOverButton() && bAñadirConcepto.enabled && admin == true) {
    pantalla = PANTALLA.CONTABILIDAD_AÑADIRCONCEPTO;
  } else if (bAceptarConcepto.mouseOverButton() && bAceptarConcepto.enabled) {
    pantalla = PANTALLA.CONTABILIDAD_BALANCE;
  } else if (bDetalle.mouseOverButton() && bDetalle.enabled && pantalla == PANTALLA.CENSO) {
    String[] info = stCenso.getSelectedInfo();
    String [] infoH = getInfoTablaHermano(info[0]);
    printArray(infoH);
    tiNombre.text= infoH[2];
    tiApellidos.text = infoH[3];
    tiDNI.text = infoH[5];
    tiCalle.text = infoH[6];
    tiFechaNacimiento.text = infoH[4];
    tiFechaAlta.text = infoH[21];
    tiNumero.text = infoH[7];
    tiPiso.text = infoH[8];
    tiLocalidad.text = infoH[9];
    tiProvincia.text = infoH[10];
    tiTelefono.text = infoH[11];
    tiBanco.text = infoH[13];
    tiTitular.text = infoH[14];
    tiDNITitular.text = infoH[15];
    tiIBAN.text = infoH[16];
    tiEntidad.text = infoH[17];
    tiOficina.text = infoH[18];
    tiDigitoControl.text = infoH[19];
    tiNumeroCuenta.text = infoH[20];
    tiCorreoElectronico.text = infoH[12];
    pantalla = PANTALLA.CENSO_DETALLE;
  } else if (bFicha.mouseOverButton() && bFicha.enabled && pantalla == PANTALLA.
CENSO_DETALLE) {
    launch(ruta+titulo1);
  } else if ( bAceptarCenso.mouseOverButton() && bAceptarCenso.enabled && pantalla ==
PANTALLA.CENSO_NUEVOHERMANO) {
    pantalla = PANTALLA.CENSO;
  } else if (itbPerfilPersonal.mouseOverButton() && itbPerfilPersonal.enabled) {
    pantalla = PANTALLA.CENSO_DETALLE;
  } else if (bDetalleBalance.mouseOverButton() && bDetalleBalance.enabled) {
    pantalla = PANTALLA.CONTABILIDAD_DETALLEBALANCE;
  } else if (bFicha.mouseOverButton() && bFicha.enabled && pantalla == PANTALLA.
CENSO_NUEVOHERMANO) {
    selectInput("Selecciona un fitxer ...", "fileSelected");
  } else if (bAñadirRecibo.mouseOverButton() && bAñadirRecibo.enabled && pantalla == PANTALLA.
CONTABILIDAD_AÑADIRCONCEPTO) {
    selectInput("Selecciona un fitxer ...", "fileSelected");
  } else if (bRecuerdos.mouseOverButton() && bRecuerdos.enabled && pantalla == PANTALLA.
ENLACES) {
    launch(ruta+titulo2);
  } else if (bDetalleConcepto.mouseOverButton() && bDetalleConcepto.enabled) {
    pantalla =  PANTALLA.CONTABILIDAD_DETALLEMOVIMIENTO;
```

```
 } else if (bPrevArchivo.mouseOverButton() && bPrevArchivo.enabled) {
   stArchivo.prevPage();
 } else if (bNextArchivo.mouseOverButton() && bNextArchivo.enabled) {
   stArchivo.nextPage();
 } else if (bAñadir.mouseOverButton() && bAñadir.enabled && pantalla == PANTALLA.ARCHIVO) {
   pantalla = PANTALLA.ARCHIVO_NUEVO;
 } else if (bDetalle.mouseOverButton() && bDetalle.enabled && pantalla == PANTALLA.ARCHIVO) {
   pantalla = PANTALLA.ARCHIVO_DETALLE;
 } else if (itbInsertarArchivo.mouseOverButton() && itbInsertarArchivo.enabled && pantalla
== PANTALLA.ARCHIVO_NUEVO) {
   selectInput("Selecciona un fitxer ...", "fileSelected");
 } else if (itbInsertarArchivoAvisos.mouseOverButton() && itbInsertarArchivoAvisos.enabled) {
   selectInput("Selecciona un fitxer ...", "fileSelected");
 } else if (bAceptarArchivo.mouseOverButton() && bAceptarArchivo.enabled && pantalla ==
PANTALLA.ARCHIVO_NUEVO) {
   pantalla = PANTALLA.ARCHIVO;
 } else if (bAceptarAvisosAlertas.mouseOverButton() && bAceptarAvisosAlertas.enabled) {
   pantalla = PANTALLA.AVISOS;
 } else if (bAñadirAviso.mouseOverButton() && bAñadirAviso.enabled) {
   pantalla = PANTALLA.AVISOS_NUEVOAVISO;
 } else if (bModificarAviso.mouseOverButton() && bModificarAviso.enabled) {
   pantalla = PANTALLA.AVISOS_NUEVOAVISO;
 } else if (bDetalleAviso.mouseOverButton() && bDetalleAviso.enabled) {
   pantalla = PANTALLA.AVISOS_DETALLEAVISO;
 } else if (bAñadirEvento.mouseOverButton() && bAñadirEvento.enabled) {
   pantalla = PANTALLA.AVISOS_NUEVOEVENTO;
 } else if (bModificarEvento.mouseOverButton() && bModificarEvento.enabled) {
   pantalla = PANTALLA.AVISOS_NUEVOEVENTO;
 } else if (bDetalleEvento.mouseOverButton() && bDetalleEvento.enabled) {
   pantalla = PANTALLA.AVISOS_DETALLEEVENTO;
 } else if (bMesAnteriorAviso.mouseOverButton() && bMesAnteriorAviso.enabled) {
   cEventos.prevMonth();
 } else if (bMesPosteriorAviso.mouseOverButton() && bMesPosteriorAviso.enabled) {
   cEventos.nextMonth();
 } else if (bModificar.mouseOverButton()&& bModificar.enabled) {
   String [][] info = getInfoTablaCensoBuscar(buscar.getValue());
   stCenso = new SelectTable(filasCenso, columnasCenso, 20+menuWidth, 285, 1280-menuWidth-
40, 410);
   stCenso.setHeaders(headersCenso);
   stCenso.setData(info);
   stCenso.setColumnWidths(colWidthsCenso);
   stCenso.setColumnMaxChars(maxCharsCenso);
 }

 userText.isPressed();
 passText.isPressed();
 buscar.isPressed();
 tfNombre.isPressed();
 tfApellidos.isPressed();
 tfDNI.isPressed();
 tfCalle.isPressed();
 tfNumero.isPressed();
 tfPiso.isPressed();
 tfLocalidad.isPressed();
 tfProvincia.isPressed();
 tfTelefono.isPressed();
 tfCorreoElectronico.isPressed();
 tfBanco.isPressed();
 tfTitular.isPressed();
 tfDNITitular.isPressed();
 tfIBAN.isPressed();
 tfEntidad.isPressed();
 tfOficina.isPressed();
 tfDigitoControl.isPressed();
 tfNumeroCuenta.isPressed();
```

```
 tfTitulo.isPressed();
 tfCantidad.isPressed();
 tfTituloArchivo.isPressed();
 tfTituloAviso.isPressed();
 tfTituloEvento.isPressed();
 tfAñoDatacion.isPressed();
 taNuevoAviso.isPressed();
 taNuevoEvento.isPressed();
 cEventos.checkButtons();

 if (sCategoriaArchivo.mouseOverSelect() && sCategoriaArchivo.enabled) {
   if (!sCategoriaArchivo.collapsed) {
     sCategoriaArchivo.update();      // Actualitzar valor
   }
   sCategoriaArchivo.toggle();        // Plegar o desplegar
 }
 stCenso.checkSelections();
 stGastos.checkSelections();
 stGastosPresupuesto.checkSelections();
 stBalanceIngresos.checkSelections();
 stDetalleItem.checkSelections();
 stArchivo.checkSelections();




 if (bAceptarCenso.mouseOverButton() && bAceptarCenso.enabled) {
   // Agafar els valors dels camps del formulari
   String nombre = String.valueOf(tfNombre.getValue());
   String apellidos = String.valueOf(tfApellidos.getValue());
   String fechanacimiento = formataFecha2(String.valueOf(dataCalendariNacimiento));
   String dni = String.valueOf(tfDNI.getValue());
   String calle = String.valueOf(tfCalle.getValue());
   String numero = String.valueOf(tfNumero.getValue());
   String piso = String.valueOf(tfPiso.getValue());
   String localidad = String.valueOf(tfLocalidad.getValue());
   String provincia = String.valueOf(tfProvincia.getValue());
   String telefono = String.valueOf(tfTelefono.getValue());
   String correoelectronico = String.valueOf(tfCorreoElectronico.getValue());
   String banco = String.valueOf(tfBanco.getValue());
   String titular = String.valueOf(tfTitular.getValue());
   String dnititular = String.valueOf(tfDNITitular.getValue());
   String iban = String.valueOf(tfIBAN.getValue());
   String entidad = String.valueOf(tfEntidad.getValue());
   String oficina = String.valueOf(tfOficina.getValue());
   String digitocontrol = String.valueOf(tfDigitoControl.getValue());
   String numerocuenta = String.valueOf(tfNumeroCuenta.getValue());
   String fechaalta = formataFecha2(String.valueOf(dataCalendariAlta));
   // Inserir a la BBDD
   insertInfoTablaHermano(nombre, apellidos, fechanacimiento, dni, calle, numero, piso,
localidad, provincia, telefono, correoelectronico, banco, titular, dnititular, iban,
entidad, oficina, digitocontrol, numerocuenta, fechaalta);
   // Resetear camps del formulari
   resetFormularioCenso();
   stCenso = new SelectTable(filasCenso, columnasCenso, 20+menuWidth, 285, 1280-menuWidth-
40, 410);
   stCenso.setHeaders(headersCenso);
   stCenso.setData(getInfoTablaCenso());
   stCenso.setColumnWidths(colWidthsCenso);
   stCenso.setColumnMaxChars(maxCharsCenso);
 }

 if (bAceptarArchivo.mouseOverButton() && bAceptarArchivo.enabled) {
   // Agafar els valors dels camps del formulari
   String titulo = String.valueOf(tfTituloArchivo.getValue());
```

```
   String datacion = String.valueOf(tfAñoDatacion.getValue());
   String file = titol;
   String tipo = String.valueOf(sCategoriaArchivo.selectedValue);
   String idTipoArchivo = String.valueOf(obtenerIdTipoArchivo(tipo));
   insertInfoTablaArchivo( titulo, datacion, file, idTipoArchivo);
   // Inserir a la BBDD
   // Resetear camps del formulari
   resetFormularioArchivo();
   stArchivo = new SelectTable(filasArchivo, columnasArchivo, 20+menuWidth, 285, 1280-
menuWidth-40, 410);
   stArchivo.setHeaders(headersArchivo);
   stArchivo.setData(getInfoTablaArchivo());
   stArchivo.setColumnWidths(colWidthsArchivo);
   stArchivo.setColumnMaxChars(maxCharsArchivo);
 }




 // Si pitja el botó, canvia la visibilitat del calendari.
 if (bCalendarioAlta.mouseOverButton()&& bCalendarioAlta.enabled) {
   cpFechaAlta.visible = !cpFechaAlta.visible;
 }

 if (cpFechaAlta.bNext.mouseOverButton()) {
   cpFechaAlta.nextMonth();
 }

 if (cpFechaAlta.bPrev.mouseOverButton()) {
   cpFechaAlta.prevMonth();
 }

 if (cpFechaAlta.bOK.mouseOverButton() && cpFechaAlta.dateSelected) {
   dataCalendariAlta = cpFechaAlta.selectedDay +"/"+ cpFechaAlta.selectedMonth + "/"+
cpFechaAlta.selectedYear;
   cpFechaAlta.visible = false;
 }

 cpFechaAlta.checkButtons();


 cpFechaMovimiento.checkButtons();

 // Si pitja el botó, canvia la visibilitat del calendari.
 if (bCalendarioMovimiento.mouseOverButton()&& bCalendarioMovimiento.enabled) {
   cpFechaMovimiento.visible = !cpFechaMovimiento.visible;
 }

 if (cpFechaMovimiento.bNext.mouseOverButton()) {
   cpFechaMovimiento.nextMonth();
 }

 if (cpFechaMovimiento.bPrev.mouseOverButton()) {
   cpFechaMovimiento.prevMonth();
 }

 if (cpFechaMovimiento.bOK.mouseOverButton() && cpFechaMovimiento.dateSelected) {
   dataCalendariMovimiento = cpFechaMovimiento.selectedDay +"/"+ cpFechaMovimiento.
selectedMonth + "/"+ cpFechaMovimiento.selectedYear;
   cpFechaMovimiento.visible = false;
 }
 cpFechaNacimiento.checkButtons();
```

```
 // Si pitja el botó, canvia la visibilitat del calendari.
 if (bCalendario.mouseOverButton()&& bCalendario.enabled) {
   cpFechaNacimiento.visible = !cpFechaNacimiento.visible;
 }

 if (cpFechaNacimiento.bNext.mouseOverButton()) {
   cpFechaNacimiento.nextMonth();
 }

 if (cpFechaNacimiento.bPrev.mouseOverButton()) {
   cpFechaNacimiento.prevMonth();
 }

 if (cpFechaNacimiento.bOK.mouseOverButton() && cpFechaNacimiento.dateSelected) {
   dataCalendariNacimiento= cpFechaNacimiento.selectedDay +"/"+ cpFechaNacimiento.
selectedMonth + "/"+ cpFechaNacimiento.selectedYear;
   cpFechaNacimiento.visible = false;
 }


 cpNuevoEvento.checkButtons();

 // Si pitja el botó, canvia la visibilitat del calendari.
 if (bCalendarioEvento.mouseOverButton()&& bCalendarioEvento.enabled) {
   cpNuevoEvento.visible = !cpNuevoEvento.visible;
 }

 if (cpNuevoEvento.bNext.mouseOverButton()) {
   cpNuevoEvento.nextMonth();
 }

 if (cpNuevoEvento.bPrev.mouseOverButton()) {
   cpNuevoEvento.prevMonth();
 }

 if (cpNuevoEvento.bOK.mouseOverButton() && cpNuevoEvento.dateSelected) {
   dataCalendarioEvento= cpNuevoEvento.selectedDay +"/"+ cpNuevoEvento.selectedMonth + "/"+
cpNuevoEvento.selectedYear;
   cpNuevoEvento.visible = false;
 }

 if (bInicioSesion.mouseOverButton() && bInicioSesion.enabled && comprovaLogin()) {
   logged = true;
   admin = comprovaAdmin();
 }


 stlTipoConcepto.mouseOn();

 if (pantalla == PANTALLA.PRINCIPAL) {
   pcAvisosPrincipal.checkCardSelection();
 }

 if (PopUpinicioSesion.bAceptar.mouseOverButton() && PopUpinicioSesion.bAceptar.enabled) {
   pantalla = PANTALLA.PRINCIPAL;
 }

 if (bNextAviso.mouseOverButton() && bNextAviso.enabled) {
   pcAvisos.nextPage();
 } else if (bPrevAviso.mouseOverButton() && bPrevAviso.enabled) {
   pcAvisos.prevPage();
 } else if (pantalla == PANTALLA.AVISOS)
   pcAvisos.checkCardSelection();
}
```

```
// Modifica el cursor
void updateCursor() {

 boolean mouseOnOneButton = false;

 for (int i=0; i<buttons.length; i++) {
   if (buttons[i].mouseOverButton() && buttons[i].enabled) {
     mouseOnOneButton = true;
   }
 }
 for (int i=0; i<imgtextbuttons.length; i++) {
   if (imgtextbuttons[i].mouseOverButton() && imgtextbuttons[i].enabled) {
     mouseOnOneButton = true;
   }
 }
 if (mouseOnOneButton) {
   cursor(HAND);
 } else {
   cursor(ARROW);
 }
}

// Reset del Formulari
void resetFormularioCenso() {
 tfNombre.removeAllText();
 tfApellidos.removeAllText();
 tfDNI.removeAllText();
 tfCalle.removeAllText();
 tfNumero.removeAllText();
 tfPiso.removeAllText();
 tfLocalidad.removeAllText();
 tfProvincia.removeAllText();
 tfTelefono.removeAllText();
 tfCorreoElectronico.removeAllText();
 tfBanco.removeAllText();
 tfTitular.removeAllText();
 tfDNITitular.removeAllText();
 tfIBAN.removeAllText();
 tfEntidad.removeAllText();
 tfOficina.removeAllText();
 tfDigitoControl.removeAllText();
 tfNumeroCuenta.removeAllText();
}

// Reset del Formulari
void resetFormularioArchivo() {
 tfTituloArchivo.removeAllText();
 dataCalendarioArchivo = "";
 titol="";
 tfAñoDatacion.removeAllText();
}

int obtenerIdTipoArchivo(String tipo) {
 int idTipoArchivo = -1;
 msql.query("SELECT idtipo_arch FROM tipo_arch WHERE tipo = &apos;"+tipo+"&apos;");
 if (msql.next()) {
   idTipoArchivo = msql.getInt("idtipo_arch");
 }
 return idTipoArchivo;
}
```