DOCUMENTACIÓN DE PROGRAMA QUE SUMA DOS NÚMEROS COMPLEJOS CON PROGRAMACIÓN ORIENTADA A EVENTOS

1. **Análisis de requisitos y elementos de interés**
   Se requiere una suma de dos números complejos, los *números complejos* se componen por una *parte real* y una *parte imaginaria,* por lo que podemos observar que ambas partes son elementos de interés y que nuestro objeto de estudio es un número complejo.
2. **MODELADO DE CLASE MODELO**
   2.1. Clase CL_NUM_COMLEJO

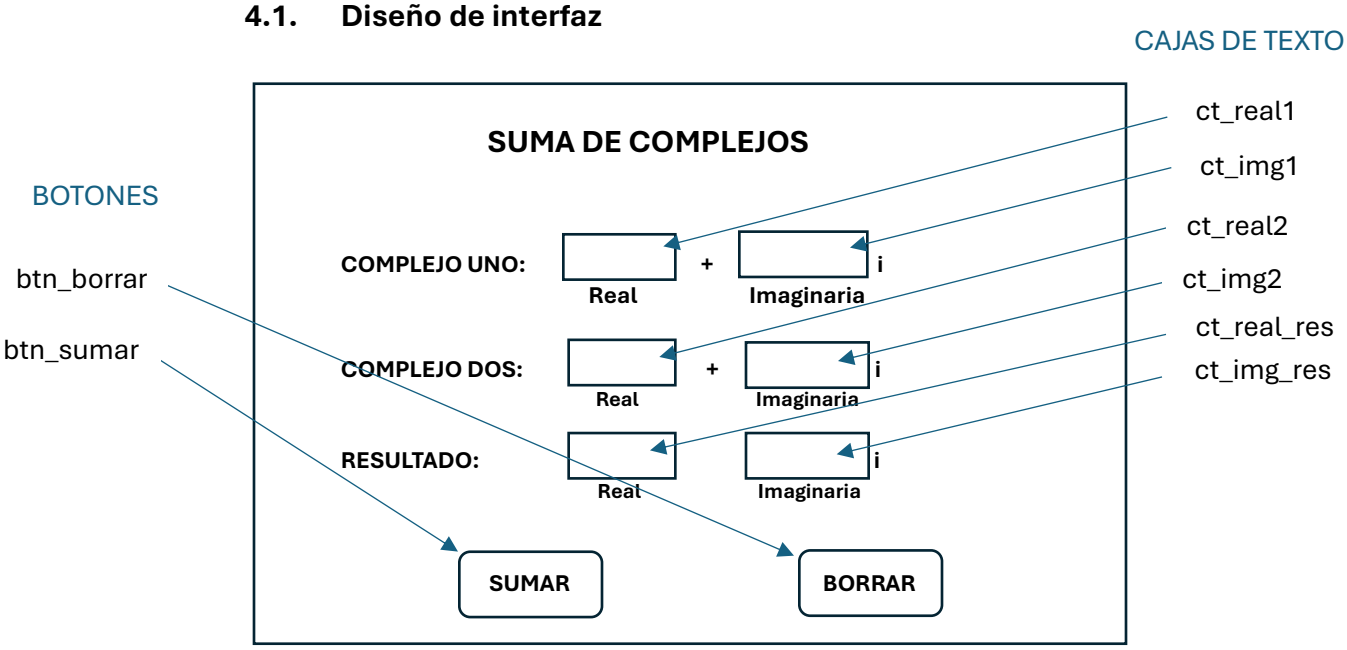
   2.2. Método para la funcionalidad de CL_NUM_COMPLEJO
   El método “sumar_complejo” funciona recibiendo como parámetro dos objetos de tipo CL_NUM_COMPLEJO, no tiene caso modelar dos veces a un número complejo por lo que se puede instanciar el modelo que ya tienes para que lo uses las veces que lo necesites. En este caso se instancia un primer objeto que represente al primer número complejo, la segunda instancia representaría el segundo numero complejo y el tercer objeto instanciado representa el resultado de nuestra suma de números complejos. Por lo tanto el método recibe un parámetro por valor (segundo número complejo) y un parámetro por referencia (guarda el resultado de la suma).

3. PSEUDOCÓDIGO DE CLASE MODELO “CL_NUM_COMPLEJO”

```
CLASE CL_NUM_COMPLEJO
/*
Nombre: Rodrigo Díaz Salguero
Fecha: 28/10/2024
*/
INICIO
SECCIÓN DE ATRIBUTOS
      parte_real: REAL, PRIVADO
      parte_imaginaria: REAL, PRIVADO
SECCIÓN DE MÉTODOS

      MÉTODO set_parte_real(x:REAL)
      INICIO
            parte_real ← x
      FIN MÉTODO set_parte_real

      MÉTODO set_parte_imaginaria(x:REAL)
      INICIO
            parte_imaginaria ← x
      FIN MÉTODO set_parte_imaginaria

      MÉTODO get_parte_real:REAL
      INICIO
            REGRESAR parte_real
      FIN MÉTODO get_parte_real

      MÉTODO get_parte_imaginaria:REAL
      INICIO
            REGRESAR parte_imaginaria
      FIN MÉTODO get_parte_imaginaria

      MÉTODO sumar_complejo(x:CL_NUM_COMPLEJO, y:CL_NUM_COMPLEJO)
      //Parametro x: Contiene los atributos a sumar
      //Parametro y: Donde se guarda el resultado
      INICIO
            y.set_parte_real(parte_real + x.get_parte_real)
            y.set_parte_imaginaria(parte_imaginaria +
x.get_parte_imaginaria)
      FIN MÉTODO sumar_complejo

FIN CLASE CL_NUM_COMPLEJO
```

4. MODELADO DE CLASE VISTA

4.1. Diseño de interfaz

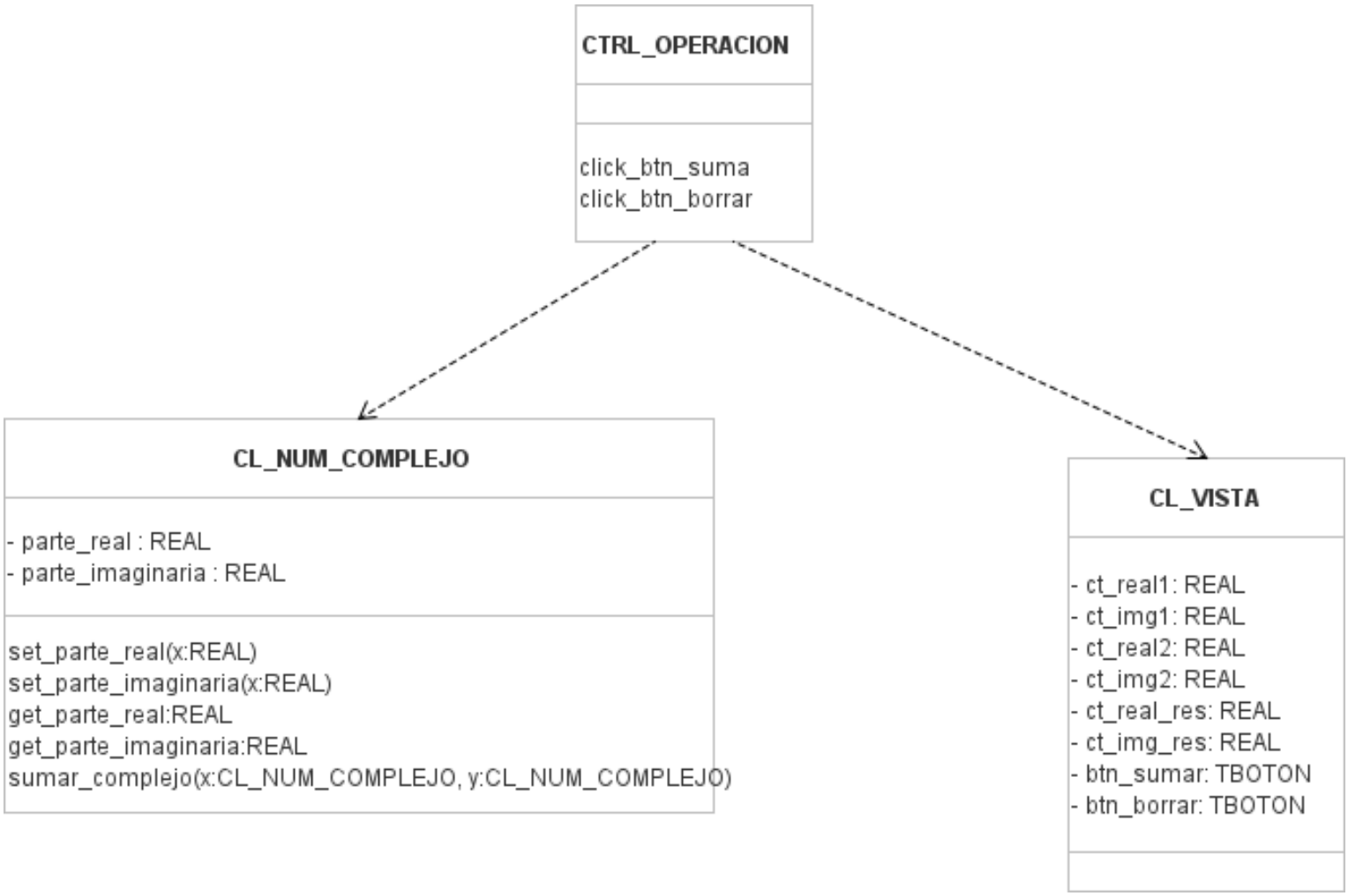


4.2. Modelado de clase CL_VISTA

4.3. Determinación de eventos

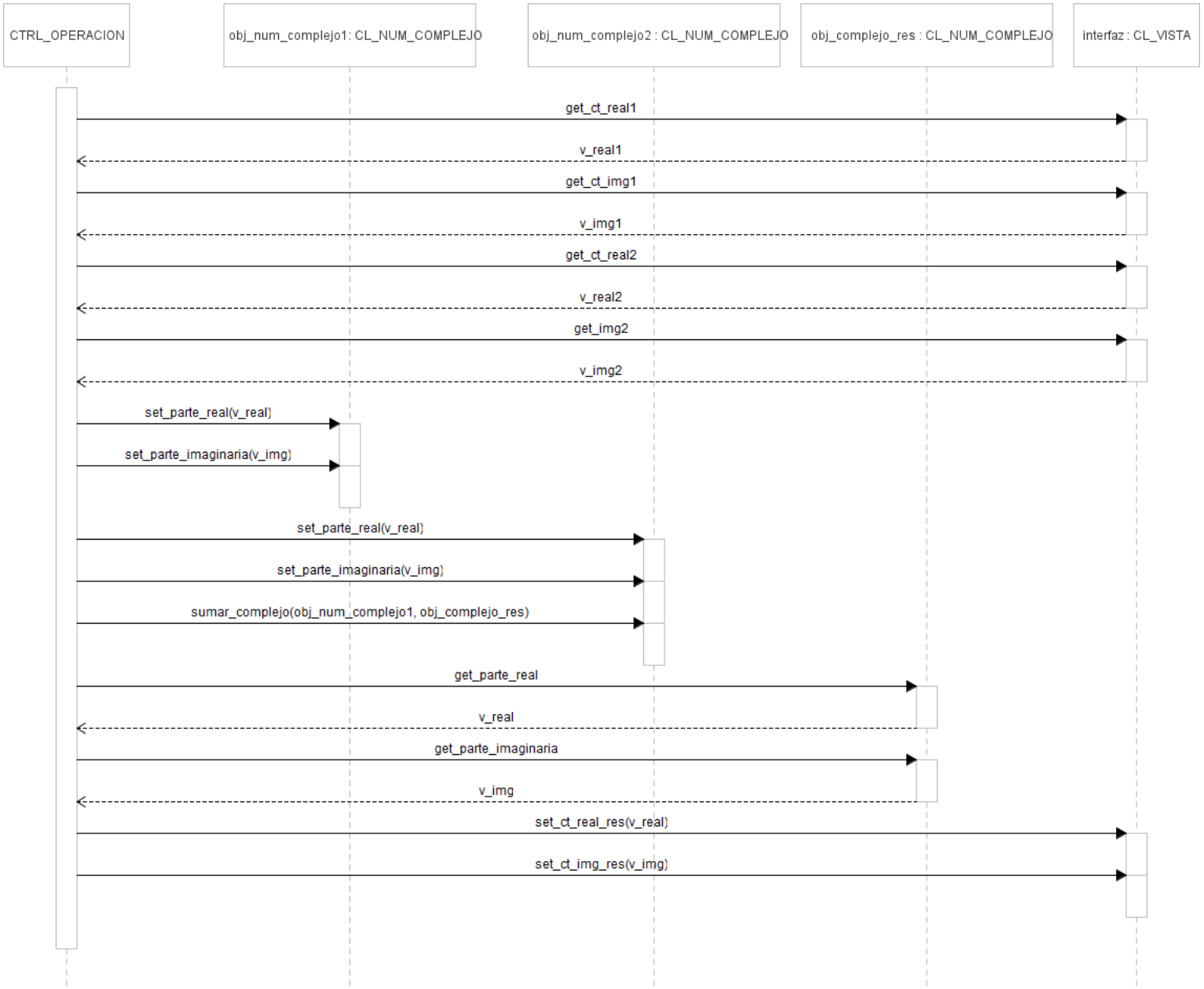
1. Evento click en btn_sumar debe generar las acciones necesarias para que CL_NUM_COMPLEJO realice la suma
2. Evento click en btn_borrar debe generar las acciones necesarias para que la clase CL_VISTA “borre” el contenido de todas las cajas de texto

5. DIAGRAMA DE CLASES



6. DIAGRAMAS DE SECUENCIAS

6.1. Diagrama de secuencia de método “click_btn_suma” de Control

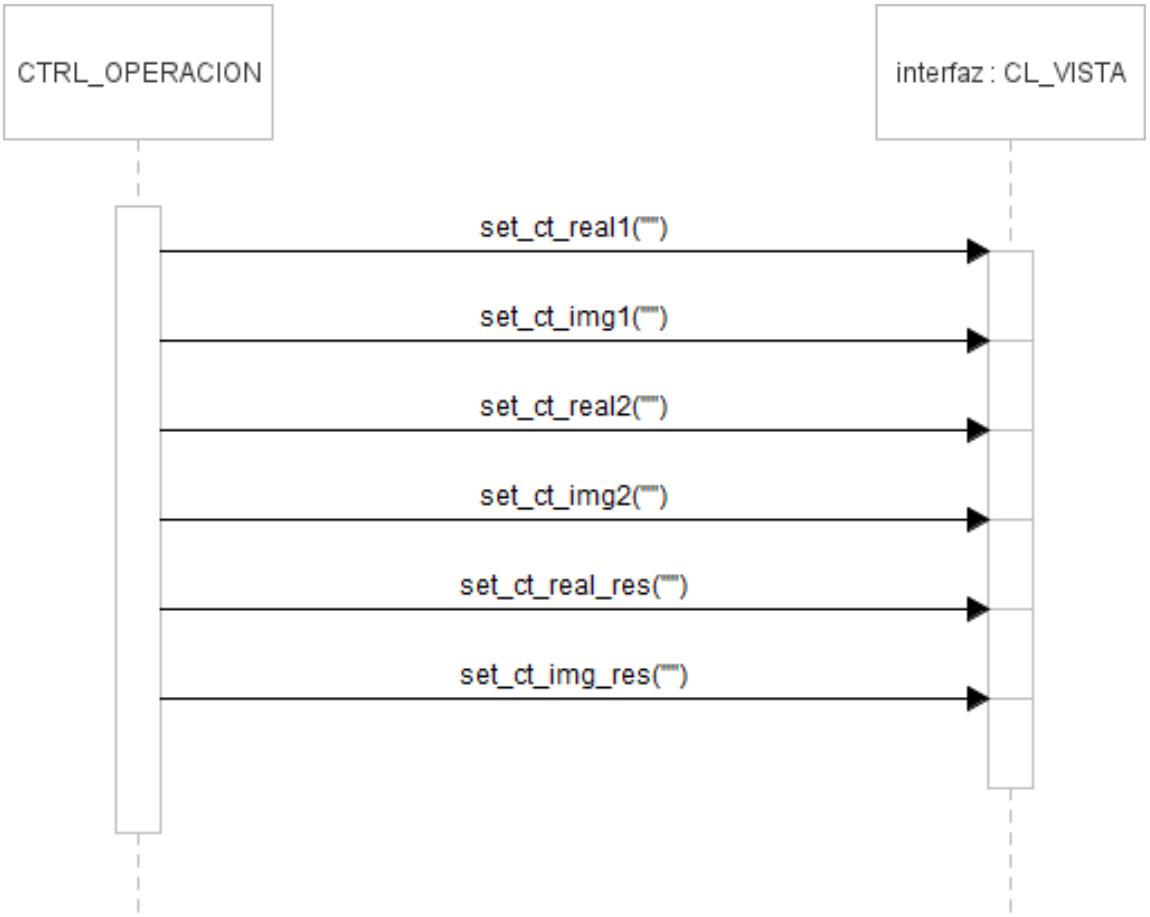


6.1.2. Pseudocódigo del evento “click_btn_suma”

```
MÉTODO click_btn_sumar
SECCIÓN DE VARIABLES
      v_img1, v_img2, v_real1, v_real2: REAL
INICIO
      Obj_num_complejo1:NUEVO OBJETO DE CL_NUM_COMPLEJO
      Obj_num_complejo2:NUEVO OBJETO DE CL_CUM_COMPLEJO
```

```
      Obj_complejo_res:NUEVO OBJETO DE CL_NUM_COMPLEJO
      //OBTENER VALORES REALES E IMAGINARIOS DE LOS NUMEROS
      v_real1 <- interfaz.get_ct_real1
      v_img1 <- interfaz.get_ct_img1
      v_real2 <- interfaz.get_ct_real2
      v_img2 <- interfaz.get_ct_img2

      //SETEAR VALORES
      obj_num_complejo1.set_parte_real(v_real1)
      obj_num_complejo1.set_parte_imaginaria(v_img1)

      obj_num_complejo2.set_parte_real(v_real2)
      obj_num_complejo2.set_parte_imaginaria(v_img2)

      //SUMAR COMPLEJOS
      obj_num_complejo2.sumar_complejo(obj_num_complejo1,obj_complejo_
res)

      //OBTENER RESULTADOS DE obj_complejo_res
      v_real1 <- obj_complejo_res.get_parte_real
      v_img1 <- obj_complejo_res.get_parte_imaginaria

      //SETEAR EL RESULTADO A UNA CAJA DE TEXTO
      interfaz.set_ct_real_res(v_real1)
      interfaz.set_ct_img_res(v_img1)
      //DESTRUIR OBJETOS DESPUES DE USARLOS
      Obj_num_complejo1.destructor
      Obj_num_complejo2.destructor
      Obj_complejo_res.destructor

FIN MÉTODO click_btn_sumar
```

6.2. Diagrama de secuencia de método “click_btn_borrar” de Control



6.2.1. Pseudocódigo de evento “click_btn_borrar”

```
MÉTODO click_btn_borrar
INICIO
      interfaz.set_ct_real1("")
      interfaz.set_ct_img1("")
      interfaz.set_ct_real2("")
      interfaz.set_ct_img2("")
      interfaz.set_ct_real_res("")
      interfaz.set_ct_img_res("")
FIN MÉTODO click_btn_borrar
```

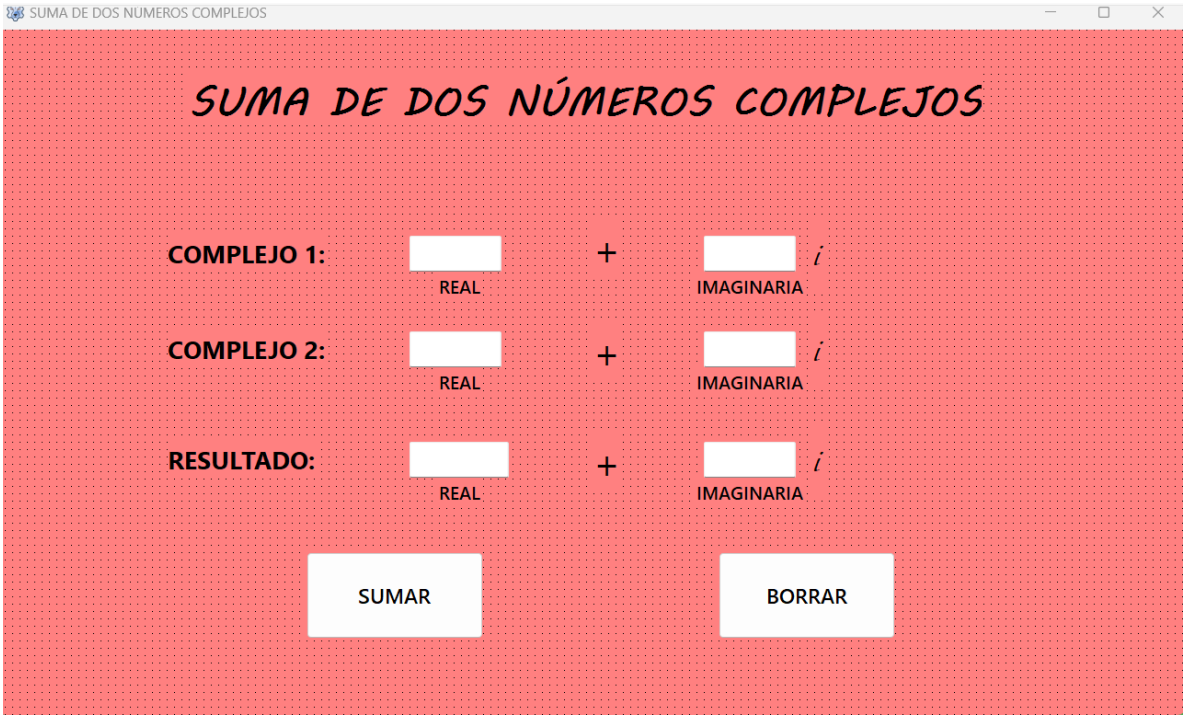
7. IMPLEMENTACIÓN CON PASCAL EN IDE LAZARUS

7.1. Codificación en PASCAL de CL_NUM_COMPLEJO

```
{$mode objfpc}{$H+}

unit U_CL_NUM_COMPLEJO;

interface
```

```
type
  CL_NUM_COMPLEJO = class
  private
    parte_real: Real;
    parte_imaginaria: Real;
  public
    procedure set_parte_real(x: Real);
    procedure set_parte_imaginaria(x: Real);
    function get_parte_real: Real;
    function get_parte_imaginaria: Real;
    procedure sumar_complejo(x: CL_NUM_COMPLEJO; y: CL_NUM_COMPLEJO);
  end;

implementation

procedure CL_NUM_COMPLEJO.set_parte_real(x: Real);
begin
  parte_real := x;
end;

procedure CL_NUM_COMPLEJO.set_parte_imaginaria(x: Real);
begin
  parte_imaginaria := x;
end;

function CL_NUM_COMPLEJO.get_parte_real: Real;
begin
  Result := parte_real;
end;

function CL_NUM_COMPLEJO.get_parte_imaginaria: Real;
begin
  Result := parte_imaginaria;
end;

procedure CL_NUM_COMPLEJO.sumar_complejo(x: CL_NUM_COMPLEJO; y:
CL_NUM_COMPLEJO);
begin
  y.set_parte_real(parte_real + x.get_parte_real);
  y.set_parte_imaginaria(parte_imaginaria + x.get_parte_imaginaria);
end;

end.
```

7.2. Diseño de interfaz con IDE Lazarus



7.3. Codificación en PASCAL de CL_VISTA

```
unit U_CL_VISTA;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls,
Buttons;

type

  { CL_VISTA }

  CL_VISTA = class(TForm)
    btn_sumar: TButton;
    btn_borrar: TButton;
    ct_img1: TEdit;
    ct_img2: TEdit;
    ct_img_res: TEdit;
    ct_real1: TEdit;
    ct_real2: TEdit;
    ct_real_res: TEdit;
    Label1: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
```

```
    Label14: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure btn_borrarClick(Sender: TObject);
    procedure btn_sumarClick(Sender: TObject);
  private

  public
    procedure set_ct_real_res(txt:String);
    procedure set_ct_img_res(txt:String);
    function get_ct_real1:String;
    function get_ct_img1:String;
    function get_ct_real2:String;
    function get_ct_img2:String;
    function get_ct_real_res:String;
    function get_ct_img_res:String;
  end;

var
  interfaz: CL_VISTA;

implementation

uses
  U_CTRL_OPERACION;

{$R *.lfm}

{ CL_VISTA }

procedure CL_VISTA.FormCreate(Sender: TObject);
begin
    obj_control:=CTRL_OPERACION.Create;
end;

procedure CL_VISTA.btn_sumarClick(Sender: TObject);
begin
    obj_control.click_btn_sumar;
end;

procedure CL_VISTA.btn_borrarClick(Sender: TObject);
begin
    obj_control.click_btn_borrar;
end;
```

```
procedure CL_VISTA.set_ct_real_res(txt:String);
begin
  ct_real_res.Text:= txt;
end;

procedure CL_VISTA.set_ct_img_res(txt:String);
begin
  ct_img_res.Text:= txt;
end;

function CL_VISTA.get_ct_real1:String;
begin
  Result:= ct_real1.Text;
end;

function CL_VISTA.get_ct_img1:String;
begin
  Result:= ct_img1.Text;
end;

function CL_VISTA.get_ct_real2:String;
begin
  Result:= ct_real2.Text;
end;

function CL_VISTA.get_ct_img2:String;
begin
  Result:= ct_img2.Text;
end;

function CL_VISTA.get_ct_real_res:String;
begin
  Result:= ct_real_res.Text;
end;

function CL_VISTA.get_ct_img_res:String;
begin
  Result:= ct_img_res.Text;
end;
end.
```

7.4. Codificación de CTRL_OPERACION

```
  unit U_CTRL_OPERACION;
{$mode objfpc}{$H+}

interface

uses
  U_CL_NUM_COMPLEJO, U_CL_VISTA, SysUtils, Dialogs;

type
  CTRL_OPERACION = class
  public
    procedure click_btn_sumar;
    procedure click_btn_borrar;
```

```
  private
    obj_num_complejo1: CL_NUM_COMPLEJO;
    obj_num_complejo2: CL_NUM_COMPLEJO;
    obj_complejo_res: CL_NUM_COMPLEJO;
  end;

var
   obj_control: CTRL_OPERACION;

implementation

procedure CTRL_OPERACION.click_btn_sumar;
var
  v_img1, v_img2, v_real1, v_real2: Real;
begin
  // OBTENER VALORES REALES E IMAGINARIOS DE LOS NUMEROS
  v_real1 := StrToFloat(interfaz.get_ct_real1);
  v_img1 := StrToFloat(interfaz.get_ct_img1);
  v_real2 := StrToFloat(interfaz.get_ct_real2);
  v_img2 := StrToFloat(interfaz.get_ct_img2);

  // SETEAR VALORES
  obj_num_complejo1.set_parte_real(v_real1);
  obj_num_complejo1.set_parte_imaginaria(v_img1);

  obj_num_complejo2.set_parte_real(v_real2);
  obj_num_complejo2.set_parte_imaginaria(v_img2);

  // SUMAR COMPLEJOS
  obj_num_complejo2.sumar_complejo(obj_num_complejo1,
obj_complejo_res);

  // OBTENER RESULTADOS DE obj_complejo_res
  v_real1 := obj_complejo_res.get_parte_real;
  v_img1 := obj_complejo_res.get_parte_imaginaria;

  // SETEAR EL RESULTADO A UNA CAJA DE TEXTO
  interfaz.ct_real_res.Text := FloatToStr(v_real1);
  interfaz.ct_img_res.Text := FloatToStr(v_img1);
end;

procedure CTRL_OPERACION.click_btn_borrar;
begin
  interfaz.ct_real1.Text := '';
  interfaz.ct_img1.Text := '';
  interfaz.ct_real2.Text := '';
  interfaz.ct_img2.Text := '';
  interfaz.ct_real_res.Text := '';
  interfaz.ct_img_res.Text := '';
end;

end.
```

7.5. Codificación de CL_NUM_COMPLEJO

```
 unit U_CL_NUM_COMPLEJO;
{$mode objfpc}{$H+}

interface

type
  CL_NUM_COMPLEJO = class
  private
    parte_real: Real;
    parte_imaginaria: Real;
  public
    procedure set_parte_real(x: Real);
    procedure set_parte_imaginaria(x: Real);
    function get_parte_real: Real;
    function get_parte_imaginaria: Real;
    procedure sumar_complejo(x: CL_NUM_COMPLEJO; y: CL_NUM_COMPLEJO);
  end;

implementation

procedure CL_NUM_COMPLEJO.set_parte_real(x: Real);
begin
  parte_real := x;
end;

procedure CL_NUM_COMPLEJO.set_parte_imaginaria(x: Real);
begin
  parte_imaginaria := x;
end;

function CL_NUM_COMPLEJO.get_parte_real: Real;
begin
  Result := parte_real;
end;

function CL_NUM_COMPLEJO.get_parte_imaginaria: Real;
begin
  Result := parte_imaginaria;
end;

procedure CL_NUM_COMPLEJO.sumar_complejo(x: CL_NUM_COMPLEJO; y:
CL_NUM_COMPLEJO);
begin
  y.set_parte_real(parte_real + x.get_parte_real);
  y.set_parte_imaginaria(parte_imaginaria + x.get_parte_imaginaria);
end;

end.
```

8. EJECUCIÓN DEL PROGRAMA