DOCUMENTACIÓN PARA EL DESARROLLO DE UN SISTEMA DE CONTROL DE VENTAS CON ALMACENAMIENTO VOLÁTIL

1. DEFINICIÓN DEL PROBLEMA Y ANÁLISIS DE REQUISITOS

1.1. Antecedentes

Un tendero de una tienda común vende cualquier tipo de productos, la información de dichos productos la tiene almacenada en una tabla hecha a mano por el tendero en la cual registra nombre y precio de cada producto.

De este modo, al realizar una venta tiene la necesidad de revisar su lista de productos y hacer los cálculos necesarios para poder cobrar al cliente el total de venta, así mismo al tendero le gustaría saber el número de venta y la fecha en la que se realizó la misma.

1.2. Problemática

Una tienda común tiene la necesidad de agilizar el proceso de venta de productos, el sistema que maneja la tienda es hacer los cálculos del total de venta a mano, haciéndolo un proceso lento y además con mucha facilidad de que alguna equivocación pueda perjudicar a la tienda.

1.3 Propuesta de solución

Se propone desarrollar un sistema de control de ventas que tenga dos módulos, el primer módulo deberá permitir tener almacenados temporalmente los productos en memoria principal, el segundo módulo deberá permitir realizar el proceso de venta de una forma ágil y satisfacer las necesidades del tendero.

1.4 Análisis

Primero podemos lograr identificar nuestras clases modelo y las relaciones entre estas.

Fácilmente se puede identificar un elemento significativo como el producto, por lo tanto podemos pensar en una clase CL_PRODUCTO, esta clase nos podría ayudar a almacenar la información pertinente de nuestros productos, la cuál es el nombre, clave y precio del producto, para así poder satisfacer la necesidad del cliente de poder registrar los productos para que con base en esa información se puedan generar líneas de detalle, aquí podemos entonces identificar otro elemento significativo, entonces podemos pensar en una clase CL_LINEA_DETALLE de la cuál se puede identificar las unidades adquiridas, el nombre del producto, el precio del producto, clave y subtotal.

De igual manera podemos identificar que una venta se compone por un número de venta, una fecha de venta, por las líneas de detalle y por un total. Entonces podemos identificar la clase CL_VENTA.

2. MODELADO DE CLASES

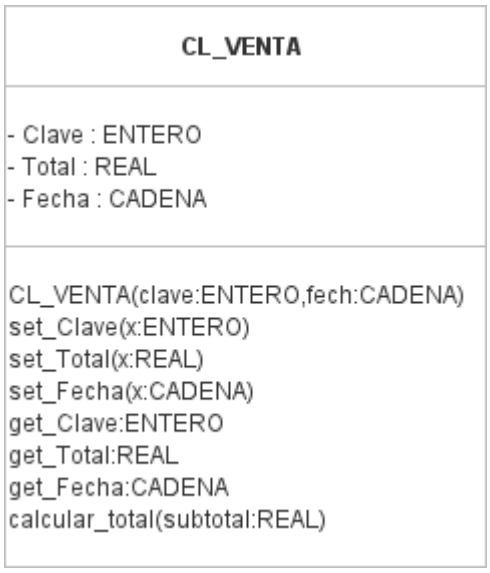
2.1. Modelado de CL_PRODUCTO

CL_PRODUCTO
- Clave : ENTERO - Nombre : CADENA - Precio : REAL
CL_PRODUCTO(clave:ENTERO,nom:CADENA,precio:REAL) set_Clave(x:ENTERO) set_Nombre(x:CADENA) set_Precio(x:REAL) get_Clave:ENTERO get_Nombre:CADENA get_Precio:REAL

2.2. Modelado de CL_LINEA_DETALLE

CL_LINEA_DETALLE
- Clave : ENTERO - Unidades_Adquiridas : ENTERO - Subtotal : REAL
CL_LINEA_DETALLE(clave:ENTERO,ua:ENTERO,prod:CL_PRODUCTO) set_Clave(x:ENTERO) set_Unidades_Adquiridas(x:ENTERO) set_Subtotal(x:REAL) get_Clave:ENTERO get_Unidades_Adquiridas:ENTERO get_Subtotal:REAL calcular_subtotal

2.3. Modelado de CL_VENTA



3. PSEUDOCÓDIGO DE LAS CLASES MODELO

3.1. Pseudocódigo de clase CL_PRODUCTO

```
CLASE CL_PRODUCTO
/*
NOMBRE: RODRIGO DÍAZ SALGUERO
FECHA: 18/02/2025
ACTUALIZACIÓN: ---
*/
INICIO
      SECCIÓN DE ATRIBUTOS
            Clave : ENTERO, PRIVADO
            Nombre : CADENA, PRIVADO
            Precio : REAL, PRIVADO

      SECCIÓN DE MÉTODOS

      CL_PRODUCTO(clave:ENTERO, nom:CADENA, precio:REAL)
      INICIO
            Clave <- clave
            Nombre <- nom
            Precio <- precio
      FIN MÉTODO CL_PRODUCTO

      set_Clave(x:ENTERO)
      INICIO
            Clave <- x
      FIN MÉTODO set_Clave

      set_Nombre(x:CADENA)
      INICIO
```

```
            Nombre <- x
      FIN MÉTODO set_Nombre

      set_Precio(x:REAL)
      INICIO
            Precio <- x
      FIN MÉTODO set_Precio

      get_Clave:ENTERO
      INICIO
            REGRESAR Clave
      FIN MÉTODO get_Clave

      get_Nombre:CADENA
      INICIO
            REGRESAR Nombre
      FIN MÉTODO get_Nombre

      get_Precio:REAL
      INICIO
            REGRESAR Precio
      FIN MÉTODO get_Precio

FIN CLASE CL_PRODUCTO
```

3.2. Pseudocódigo de clase CL_LINEA_DETALLE

```
CLASE CL_LINEA_DETALLE
/*
NOMBRE: RODRIGO DÍAZ SALGUERO
FECHA: 18/02/2025
ACTUALIZACIÓN: ---
*/
INICIO
      SECCIÓN DE ATRIBUTOS
            Clave : ENTERO, PRIVADO
            Unidades_Adquiridas : ENTERO, PRIVADO
            Subtotal : REAL, PRIVADO
            Producto : CL_PRODUCTO

      SECCION DE METODOS

            CL_LINEA_DETALLE(clave:ENTERO,ua:ENTERO,prod:CL_PRODUCTO)
            INICIO
                  Clave <- clave
                  Unidades_Adquiridas <- ua
                  Subtotal <- 0
                  Producto <- prod
            FIN MÉTODO CL_LINEA_DETALLE

            set_Clave(x: ENTERO)
            INICIO
                  Clave <-- x
```

```
            FIN METODO set_Clave

            set_Unidades_Adquiridas(x: ENTERO)
            INICIO
                  Unidades_Adquiridas <-- X
            FIN METODO set_Unidades_Adquiridas

            set_Subtotal(x: REAL)
            INICIO
                  Subtotal <-- x
            FIN METODO set_Subtotal

            set_Producto(x: CL_PRODUCTO)
            INICIO
                  Producto <-- x
            FIN METODO set_Producto

            get_Clave: ENTERO
            INICIO
                  REGRESAR Clave
            FIN METODO Clave

            get_Unidades_Adquiridas: ENTERO
            INICIO
                  REGRESAR Unidades_Adquiridas
            FIN METODO get_Unidades_Adquiridas

            get_Subtotal: REAL
            INICIO
                  REGRESAR Subtotal
            FIN METODO get_Subtotal

            get_Producto: CL_PRODUCTO
            INICIO
                  REGRESAR Producto
            FIN METODO Producto

            calcular_subtotal
            INICIO
                  subtotal <- Producto.get_precio * unidades_adquiridas
            FIN METODO calcular_subtotal

FIN CLASE CL_LINEA_DETALLE
```

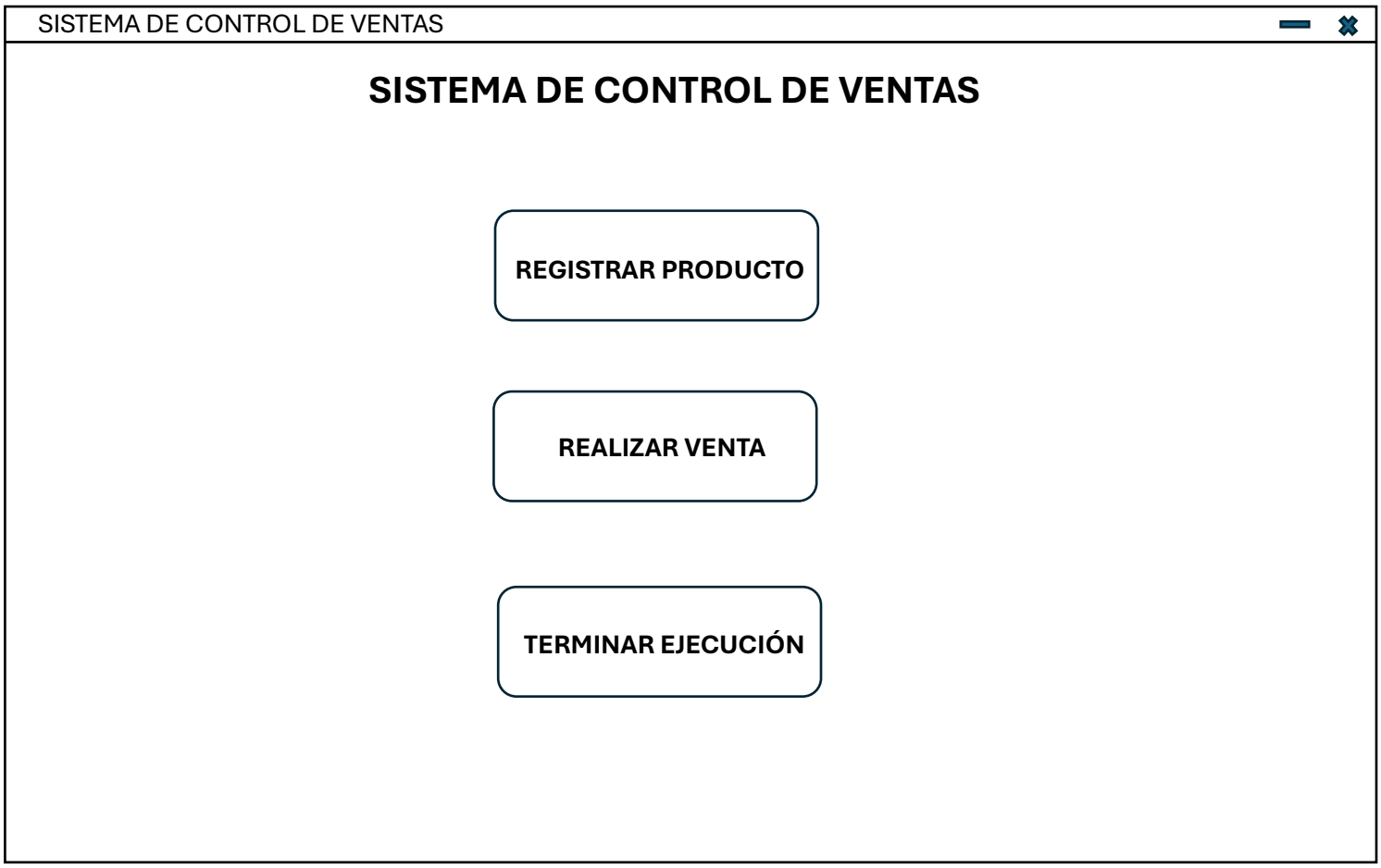
3.3. Pseudocódigo de clase CL_VENTA

```
CLASE CL_VENTA
/*
NOMBRE: RODRIGO DÍAZ SALGUERO
FECHA: 18/02/2025
ACTUALIZACIÓN: ---
*/
INICIO
      SECCIÓN DE ATRIBUTOS
```

```
        Clave : ENTERO, PRIVADO
        Total : REAL, PRIVADO
        Fecha : CADENA, PRIVADO

    SECCIÓN DE MÉTODOS

        CL_VENTA(clave:ENTERO,fech:CADENA)
        INICIO
            Clave <- clave
            Total <- 0
            Fecha <- fech
        FIN MÉTODO CL_VENTA

        set_Clave(x:ENTERO)
        INICIO
            Clave <- x
        FIN MÉTODO set_Clave

        set_Total(x:REAL)
        INICIO
            Total <- x
        FIN MÉTODO set_Total

        set_Fecha(x:CADENA)
        INICIO
            Fecha <- x
        FIN MÉTODO set_Fecha

        get_Clave:ENTERO
        INICIO
            REGRESAR Clave
        FIN MÉTODO get_Clave

        get_Total:REAL
        INICIO
            REGRESAR Total
        FIN MÉTODO get_Total

        get_Fecha:CADENA
        INICIO
            REGRESAR Fecha
        FIN MÉTODO get_Fecha

        calcular_total(subtotal:REAL)
        INICIO
            Total <- Total+subtotal
        FIN MÉTODO calcular_total

FIN CLASE CL_VENTA
```

4. MODELADO DE LA CLASE VISTA

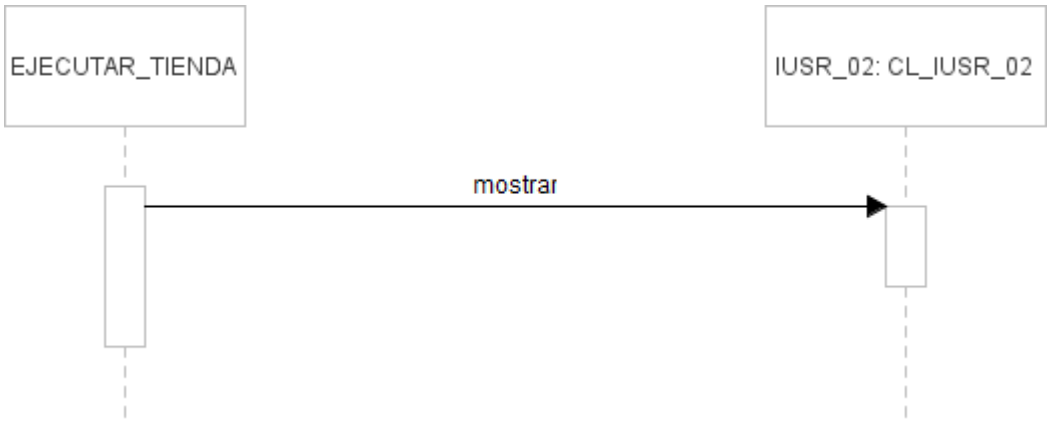
4.1. Diseño de la interfaz IUSR_01



4.2. Modelado de la clase CL_IUSR_01

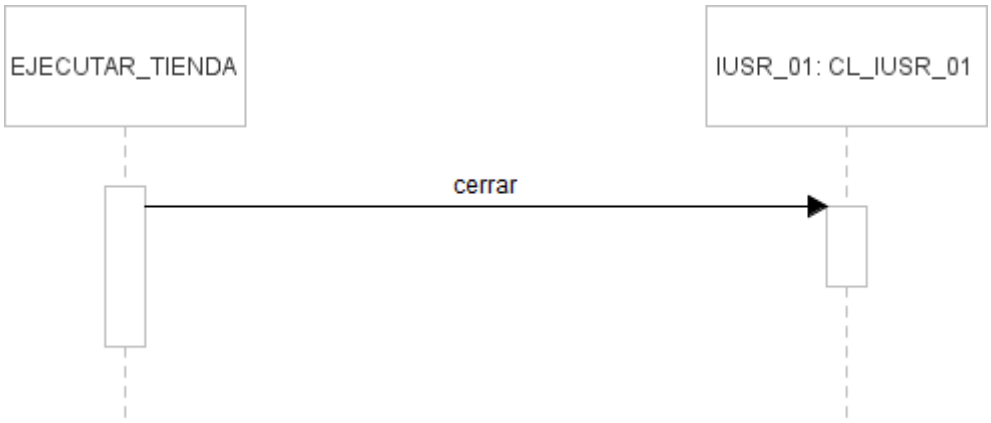
4.3. Determinación de eventos de la interfaz IUSR_01

- evt_mostrar_registrar: Evento CLICK en btn_registrar, el programa deberá generar las acciones necesarias para que se muestre la interfaz IUSR_02



- evt_mostrar_venta: Evento CLICK en btn_venta, el programa deberá generar las acciones necesarias para que se muestre la interfaz IUSR_03

- evt_terminar_ejecucion: Evento CLICK en btn_salir, el programa deberá generar las acciones necesarias para que se cierre la interfaz IUSR_01



4.4. Pseudocódigos de los eventos de IUSR_01

- **Pseudocódigo de evt_mostrar_registrar**

```
INICIO
      IUSR_02.mostrar
FIN
```

- **Pseudocódigo de evt_mostrar_venta**

```
INICIO
      IUSR_03.mostrar
FIN
```

- **Pseudocódigo de evt_terminar_ejecucion**

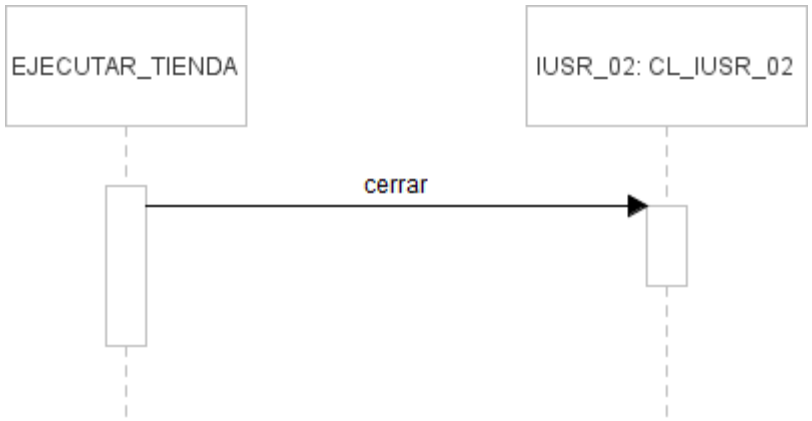
```
INICIO
      IUSR_01.cerrar
FIN
```

4.5. Diseño de interfaz IUSR_02

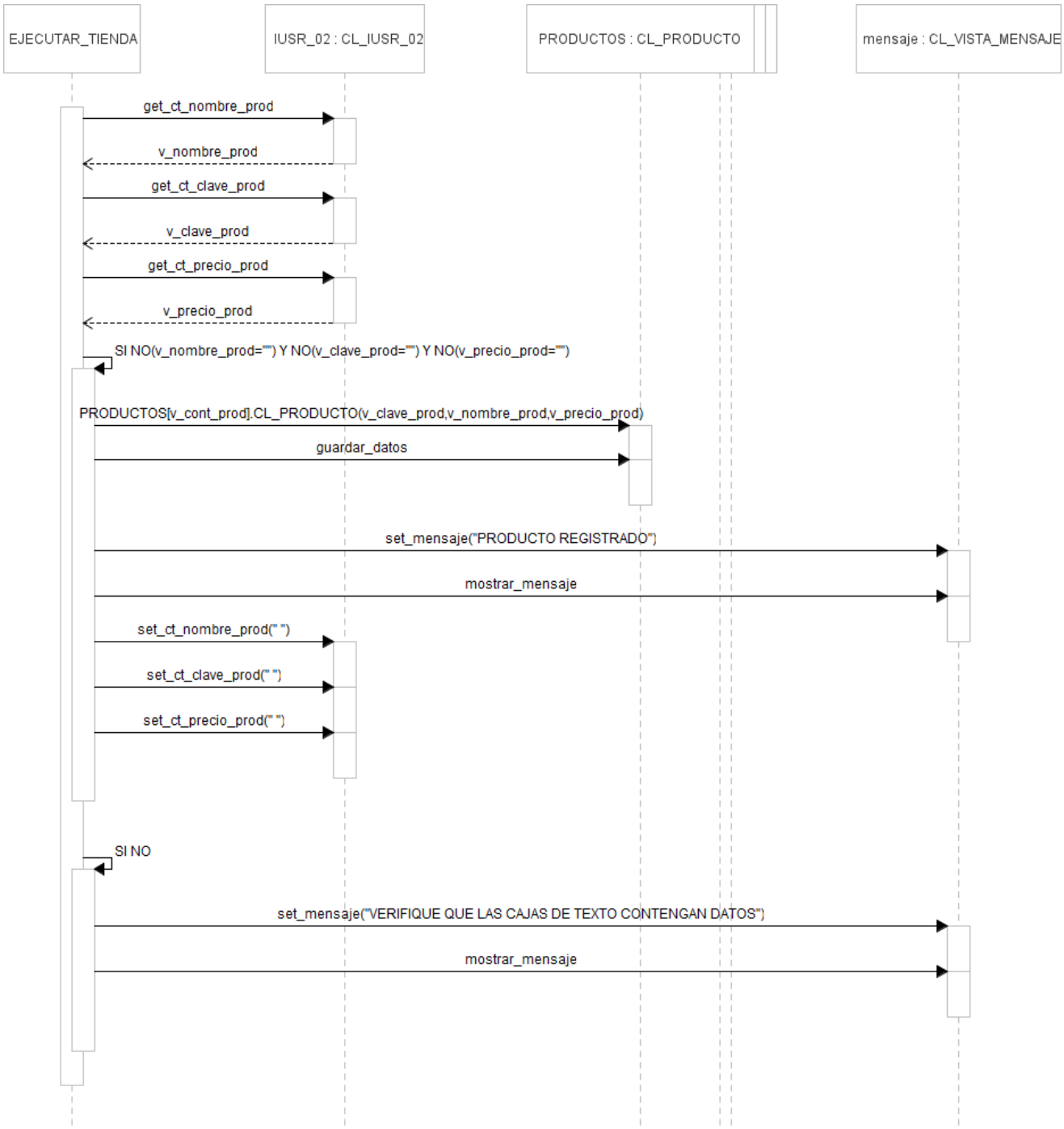
4.6. Modelado de la clase CL_IUSR_02

4.7. Determinación de eventos de la interfaz IUSR_02

- evt_cerrar_iusr_02: Evento CLICK en btn_cerrar, el programa deberá generar las acciones necesarias para que se cierre la interfaz IUSR_02



- evt_registrar_prod: Evento CLICK en btn_registrar_prod, el programa deberá generar las acciones necesarias para que se verifique que las cajas de texto ct_clave_prod, ct_nombre_prod, ct_precio_prod contengan datos, si así lo es, se obtiene el contenido, se realiza la instanciación de un objeto de la colección PRODUCTOS y se mandan como parámetros los datos obtenidos al constructor por último se muestra un mensaje “PRODUCTO REGISTRADO”, de no contener datos alguna de en las cajas de texto se manda un mensaje “VERIFIQUE QUE LAS CAJAS DE TEXTO CONTENGAN DATOS”



4.8. Pseudocódigos de los eventos de IUSR_02

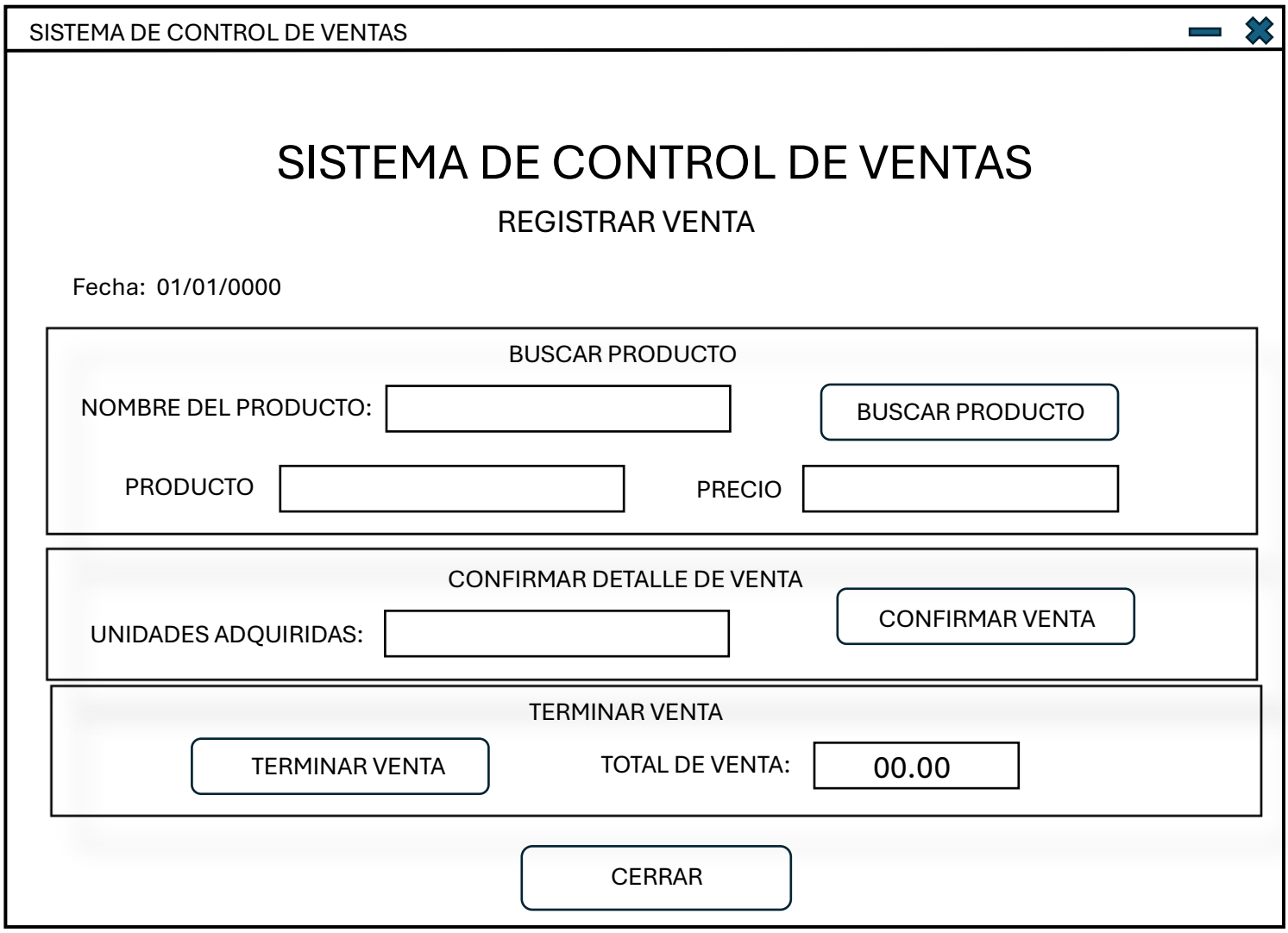
- **Pseudocódigo de evt_cerrar_iusr_02**

```
INICIO
      IUSR_02.cerrar
FIN
```

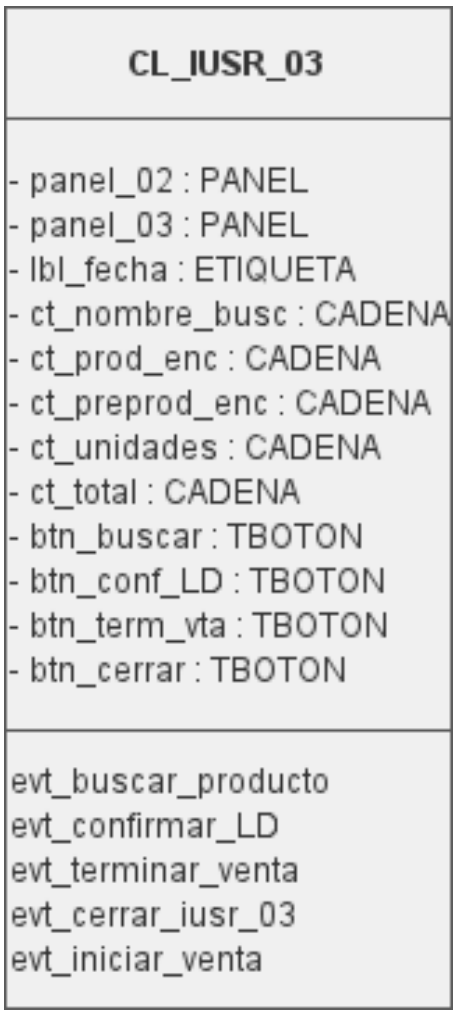
- **Pseudocódigo de evt_registrar_prod**

```
INICIO

      v_nombre_prod <- IUSR_02.get_ct_nombre_prod
      v_clave_prod <- IUSR_02.get_ct_clave_prod
      v_precio_prod <- IUSR_02.get_ct_precio_prod

      SI NO(v_nombre_prod="") Y NO(v_clave_prod="") Y
NO(v_precio_prod="")

      PRODUCTOS[v_cont_prod].CL_PRODUCTO(v_clave_prod,v_nombre_prod,v_
precio_prod)
            mensaje.set_mensaje("PRODUCTO REGISTRADO")
            mensaje.mostrar_mensaje
            IUSR_02.set_ct_nombre_prod(" ")
            IUSR_02.set_ct_clave_prod(" ")
            IUSR_02.set_ct_precio_prod(" ")
      SI NO
            mensaje.set_mensaje("VERIFIQUE QUE LAS CAJAS DE TEXTO
CONTENGAN DATOS")
            mensaje.mostrar_mensaje
      FIN SI
FIN
```

4.9. Diseño de la clase IUSR_03

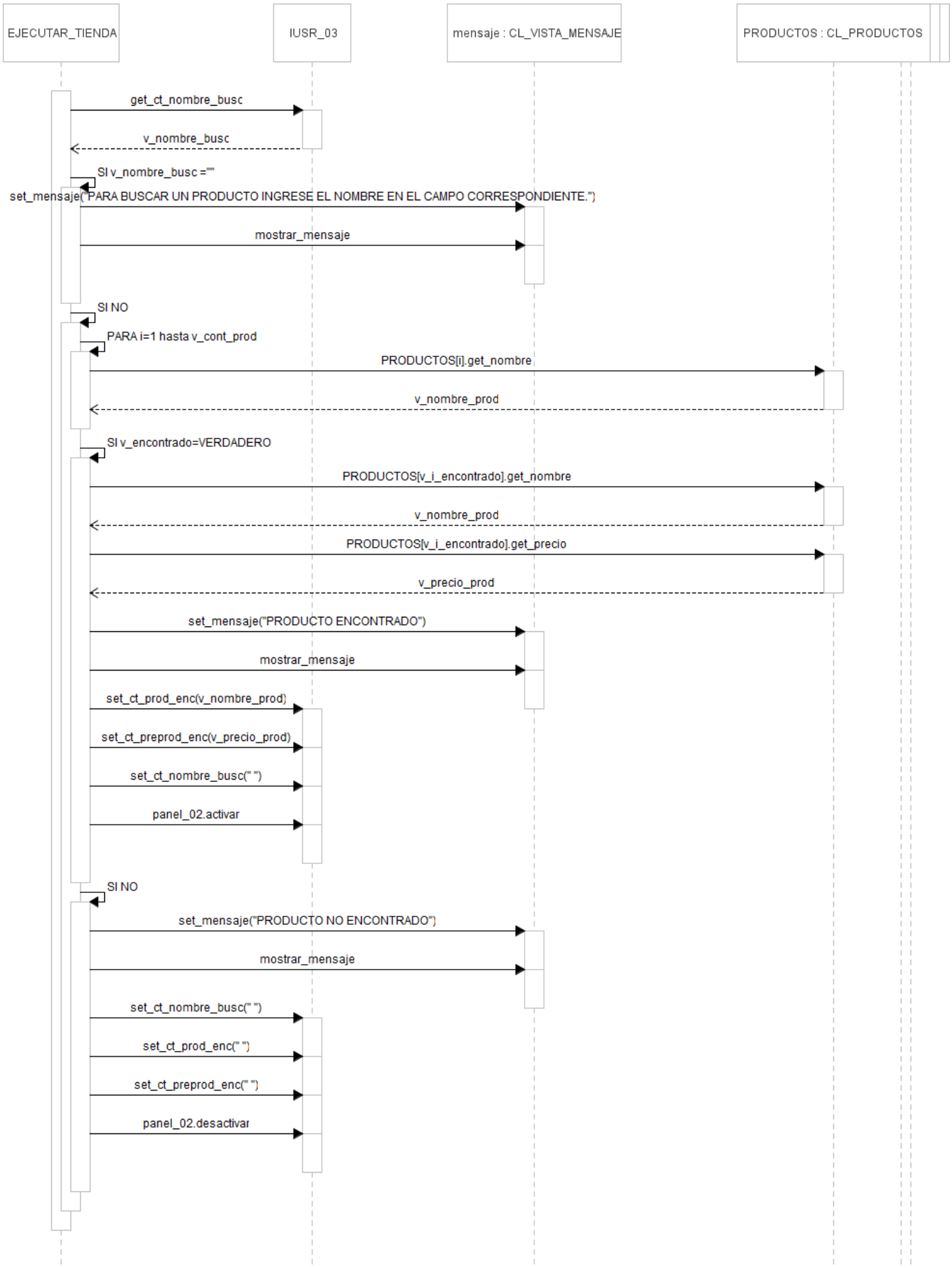


4.10. Modelado de la clase CL_IUSR_03

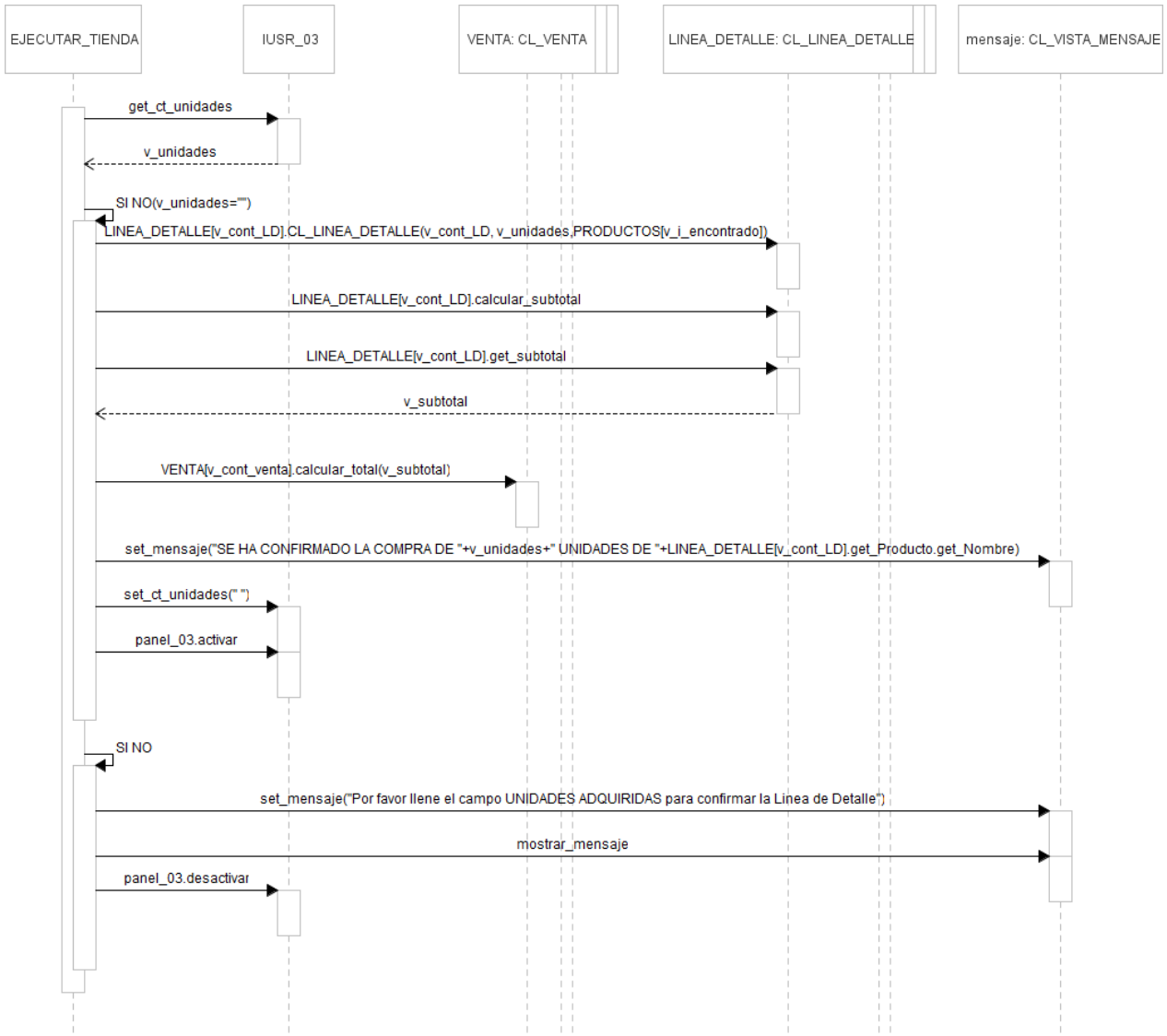


4.11. Determinación de eventos de la interfaz IUSR_03

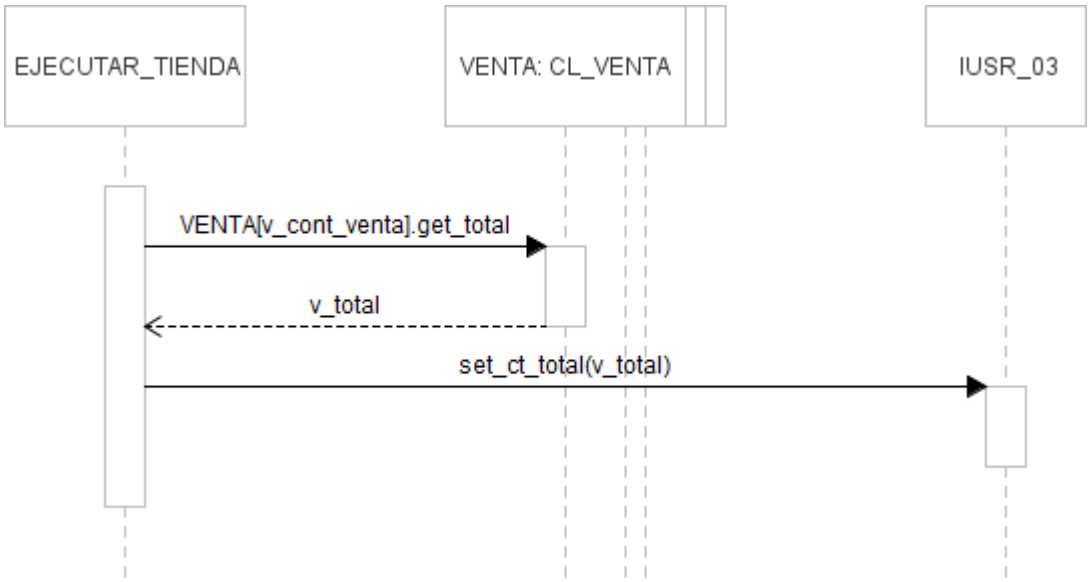
- Evento click en btn_buscar, el programa debe generar las acciones necesarias para verificar si existe algún dato en ct_nombre_busc, si es así, obtener el valor de ct_nombre_busc y buscar dentro de la colección de objetos de CL_PRODUCTO, si el producto es encontrado se seteara el nombre y precio del producto en las cajas de texto correspondientes y si no se encuentra se dará un mensaje de producto no encontrado.


EJECUTAR_TIENDA
IUSR_03
mensaje : CL_VISTA_MENSAJE
PRODUCTOS : CL_PRODUCTOS
get_ct_nombre_busc
v_nombre_busc
SI v_nombre_busc =""
set_mensaje("PARA BUSCAR UN PRODUCTO INGRESE EL NOMBRE EN EL CAMPO CORRESPONDIENTE.")
mostrar_mensaje
SI NO
PARA i=1 hasta v_cont_prod
PRODUCTOS[i].get_nombre
v_nombre_prod
SI v_encontrado=VERDADERO
PRODUCTOS[v_i_encontrado].get_nombre
v_nombre_prod
PRODUCTOS[v_i_encontrado].get_precio
v_precio_prod
set_mensaje("PRODUCTO ENCONTRADO")
mostrar_mensaje
set_ct_prod_enc(v_nombre_prod)
set_ct_preprod_enc(v_precio_prod)
set_ct_nombre_busc(" ")
panel_02.activar
SI NO
set_mensaje("PRODUCTO NO ENCONTRADO")
mostrar_mensaje
set_ct_nombre_busc(" ")
set_ct_prod_enc(" ")
set_ct_preprod_enc(" ")
panel_02.desactivar

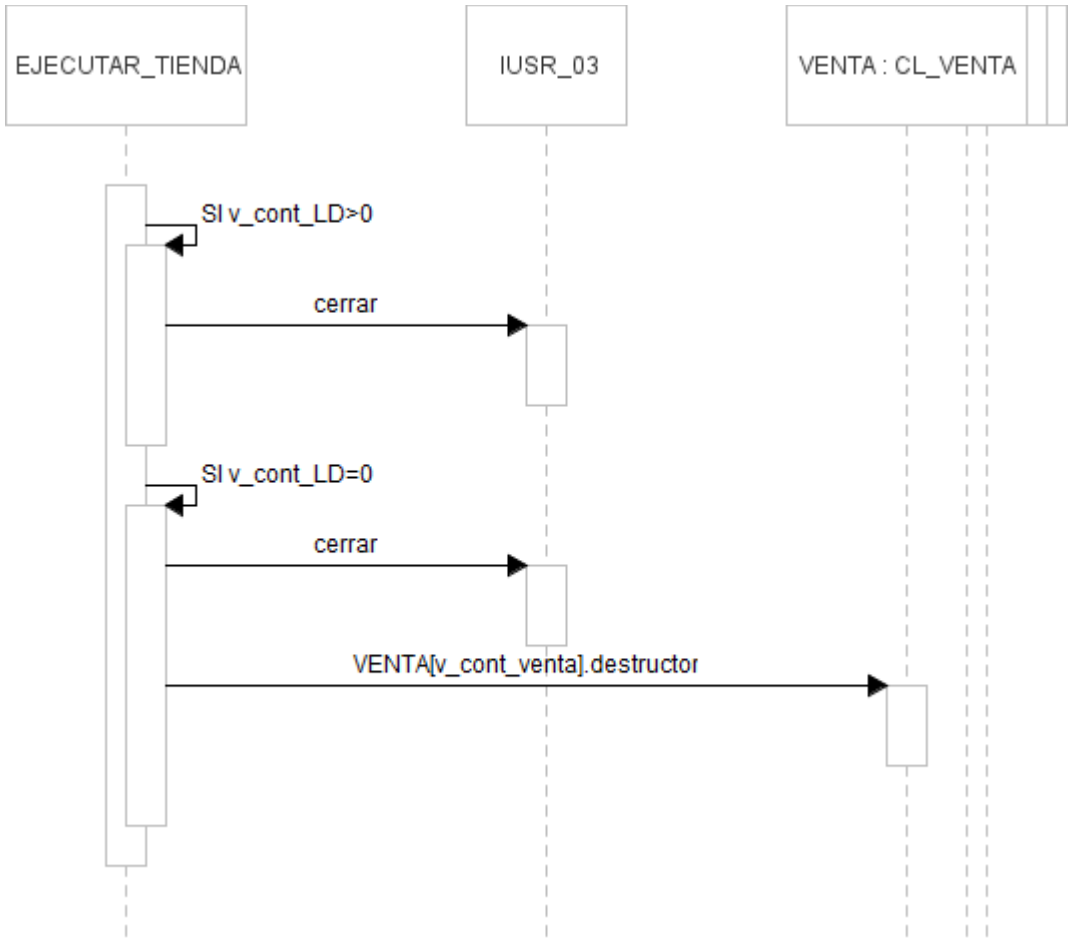
- Evento click en btn_conf_LD, el programa deberá generar las acciones necesarias para verificar que el contenido de ct_unidades exista y si es así obtener el valor de la caja de texto e instanciar un objeto de clase CL_LINEA_DETALLE y ejecutar su método “calcular_subtotal”, posteriormente ejecutar el método “calcular_total” del ultimo objeto instanciado de CL_VENTA.



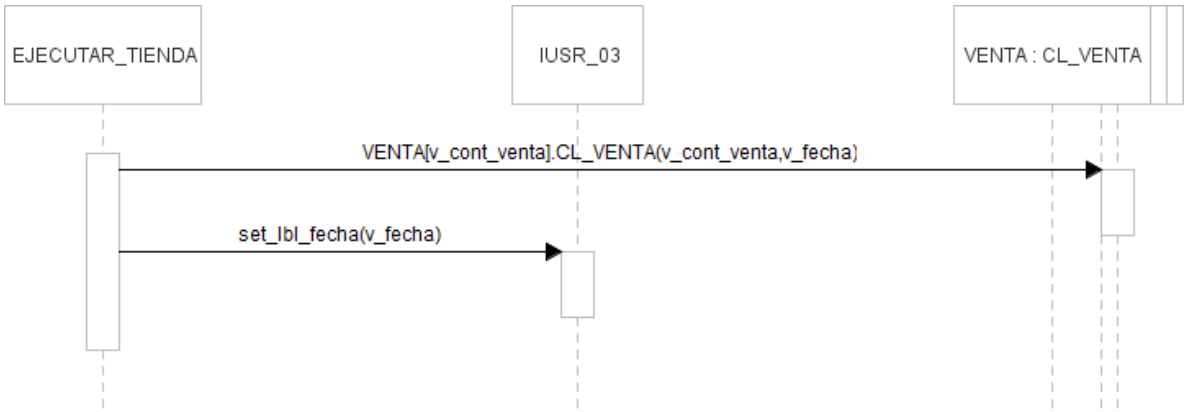
- Evento click en btn_term_vta, debe generar las acciones necsarias para obtener el total del ultimo objeto VENTA y setearlo en la caja de texto ct_total de la interfaz IUSR_03.



- Evento evt_cerrar_iusr_03, el programa debe generar las acciones necesarias para que se cierre la interfaz IUSR_03 y se elimine el objeto venta si no se realizo una compra.



- Evento evt_iniciar_venta, el programa debe realizar las acciones necesarias para instanciar un objeto VENTA y colocar la fecha del día de la venta.



4.12. Pseudocódigos de los eventos de IUSR_03

- **Pseudocódigo evt_buscar_producto**

```
INICIO
      v_nombre_busc.IUSR_03.get_ct_nombre_busc
      SI v_nombre_busc="" HACER
            mensaje.set_mensaje("Ingrese un producto a buscar")
            mensaje.mostrar_mensaje
      SI NO
            //MÉTODO DE BÚSQUEDA
            v_encontrado <- FALSO
            PARA i=1 HASTA v_cont_prod
                  v_nombre_p <- PRODUCTOS[i].get_Nombre
                  SI v_nombre_p = v_nom_busc
                        v_i_encontrado <- i
                        v_encontrado <- VERDADERO
                  FIN SI
            FIN PARA

            SI v_encontrado=VERDADERO
                  v_nombre_prod <- PRODUCTOS[v_i_encontrado].get_nombre
                  v_precio_prod <- PRODUCTOS[v_i_encontrado].get_precio
                  mensaje.set_mensaje(“PRODUCTO ENCONRADO”)
                  mensaje.mostrar_mensaje
                  IUSR_03.set_ct_prod_enc(v_nombre_prod)
                  IUSR_03.set_ct_preprod_enc(v_precio_prod)
                  IUSR_03.set_ct_nombre_busc(“ “)
                  IUSR_03.panel_02.activar
            SI NO
                  mensaje.set_mensaje(“PRODUCTO NO ENCONTRADO”)
                  mensaje.mostrar_mensaje
                  IUSR_03.set_ct_prod_enc(“ “)
                  IUSR_03.set_ct_preprod_enc(“ “)
                  IUSR_03.set_ct_nombre_busc(“ “)
                  IUSR_03.panel_02.desactivar
```

```
          FIN SI

     FIN SI
FIN
```

- **Pseudocódigo evt_conf_LD**

```
INICIO
     v_unidades ← IUSR_03.get_ct_unidades

     SI NO(v_unidades = "")
       LINEA_DETALLE[v_cont_LD].CL_LINEA_DETALLE(v_cont_LD,
v_unidades,PRODUCTOS[v_i_encontrado])
       LINEA_DETALLE[v_cont_LD].calcular_subtotal
       v_subtotal ← LINEA_DETALLE[v_cont_LD].get_subtotal
       VENTA[v_cont_venta].calcular_total(v_subtotal)
     SI NO
          mensaje.set_mensaje("LLENE EL CAMPO DE UNIDADES PARA
CONFIRMAR LA LINEA DE DETALLE")
          mensaje.mostrar_mensaje
     FIN SI
FIN
```

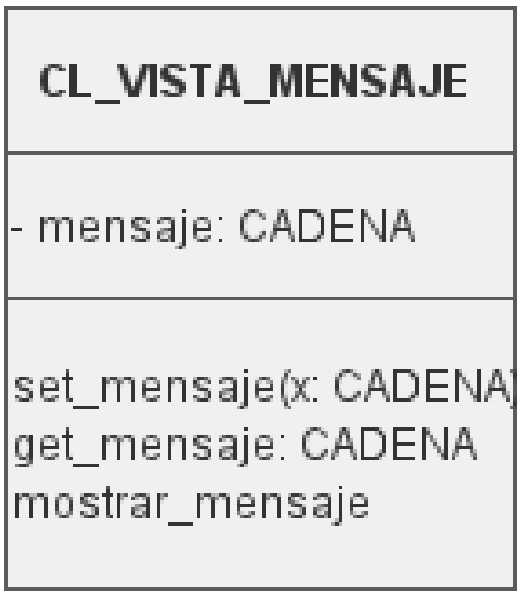
- **Pseudocódigo evt_term_vta**

```
INICIO
     v_total ← VENTA[v_cont_venta].get_total
     IUSR_03.set_ct_total(v_total)
FIN
```

- **Pseudocódigo evt_cerrar_iusr_03**

```
INICIO
     IUSR_03.set_ct_total(" ")
     IUSR_03.cerrar
FIN
```

4.13. Modelado de clase CL_VISTA_MENSAJE

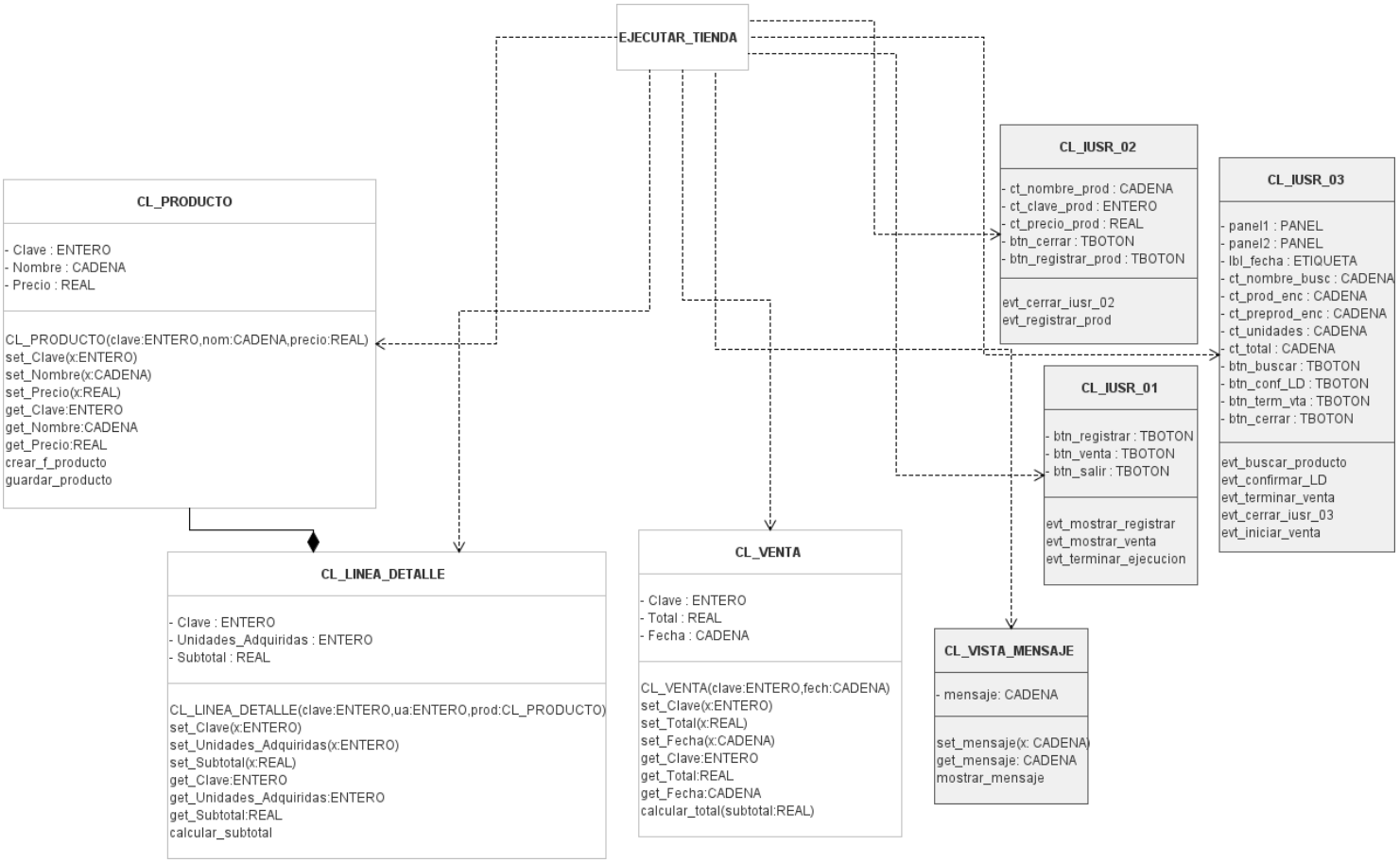


4.14. Pseudocódigo de clase CL_VISTA_MENSAJE

```
CLASE CL_VISTA_MENSAJE
/*
Autor: Rodrigo Díaz Salguero
Fecha: 16/10/2024
Actualización: 25/03/2025
COMENTARIO: Se actualizó para poder mostrar
mensajes en una interfaz GUI
*/
INICIO
      SECCIÓN DE ATRIBUTOS
            mensaje : CADENA, PRIVADO
      SECCIÓN DE MÉTODOS
            MÉTODO set_mensaje(x:CADENA)
            INICIO
                  mensaje <-- x
            FIN MÉTODO set_mensaje

            MÉTODO get_mensaje:CADENA
            INICIO
                  REGRESAR mensaje
            FIN MÉTODO get_mensaje

            MÉTODO mostrar_mensaje
            INICIO
                  MENSAJE(mensaje)
            FIN MÉTODO mostrar_mensaje
```

FIN CLASE CL_VISTA_MENSAJE

5. DIAGRAMA DE CLASES



6. IMPLEMENTACIÓN CON PASCAL EN LAZARUS

6.1. CL_PRODUCTO.pas

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

interface

type
  CL_PRODUCTO = class
  private
    Clave: Integer;
    Nombre: String;
    Precio: Real;
  public
    constructor Create(clv: Integer; nom: String; prc: Real);
    procedure set_Clave(x: Integer);
    procedure set_Nombre(x: String);
    procedure set_Precio(x: Real);
    function get_Clave: Integer;
    function get_Nombre: String;
    function get_Precio: Real;
  end;

implementation

constructor CL_PRODUCTO.Create(clv: Integer; nom: String; prc: Real);
begin
  Self.Clave := clv;
  Self.Nombre := nom;
  Self.Precio := prc;
end;

procedure CL_PRODUCTO.set_Clave(x: Integer);
begin
  Clave := x;
end;

procedure CL_PRODUCTO.set_Nombre(x: String);
begin
  Nombre := x;
end;

procedure CL_PRODUCTO.set_Precio(x: Real);
begin
  Precio := x;
end;

function CL_PRODUCTO.get_Clave: Integer;
begin
  Result := Clave;
end;
```

```
function CL_PRODUCTO.get_Nombre: String;
begin
  Result := Nombre;
end;

function CL_PRODUCTO.get_Precio: Real;
begin
  Result := Precio;
end;

end.
```

6.2. CL_LINEA_DETALLE.pas

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

interface

uses U_CL_PRODUCTO;

type
  CL_LINEA_DETALLE = class
  private
    Clave: Integer;
    Unidades_Adquiridas: Integer;
    Subtotal: Real;
    Producto: CL_PRODUCTO;
  public
    constructor Create(clv: Integer; ua: Integer; prod: CL_PRODUCTO);
    procedure set_Clave(x: Integer);
    procedure set_Unidades_Adquiridas(x: Integer);
    procedure set_Subtotal(x: Real);
    procedure set_Producto(x: CL_PRODUCTO);
    function get_Clave: Integer;
    function get_Unidades_Adquiridas: Integer;
    function get_Subtotal: Real;
    function get_Producto: CL_PRODUCTO;
    procedure calcular_subtotal;
  end;

implementation

constructor CL_LINEA_DETALLE.Create(clv: Integer; ua: Integer; prod:
CL_PRODUCTO);
begin
  Self.Clave := clv;
  Self.Unidades_Adquiridas := ua;
  Self.Subtotal := 0;
  Self.Producto := prod;
end;

procedure CL_LINEA_DETALLE.set_Clave(x: Integer);
begin
```

```
  Clave := x;
end;

procedure CL_LINEA_DETALLE.set_Unidades_Adquiridas(x: Integer);
begin
  Unidades_Adquiridas := x;
end;

procedure CL_LINEA_DETALLE.set_Subtotal(x: Real);
begin
  Subtotal := x;
end;

procedure CL_LINEA_DETALLE.set_Producto(x: CL_PRODUCTO);
begin
  Producto := x;
end;

function CL_LINEA_DETALLE.get_Clave: Integer;
begin
  Result := Clave;
end;

function CL_LINEA_DETALLE.get_Unidades_Adquiridas: Integer;
begin
  Result := Unidades_Adquiridas;
end;

function CL_LINEA_DETALLE.get_Subtotal: Real;
begin
  Result := Subtotal;
end;

function CL_LINEA_DETALLE.get_Producto: CL_PRODUCTO;
begin
  Result := Producto;
end;

procedure CL_LINEA_DETALLE.calcular_subtotal;
begin
  Subtotal := Producto.get_Precio * Unidades_Adquiridas;
end;

end.
```

6.3. CL_VENTA.pas

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

interface

type
  CL_VENTA = class
```

```
  private
    Clave: Integer;
    Total: Real;
    Fecha: String;
  public
    constructor Create(clv: Integer; fech: String);
    procedure set_Clave(x: Integer);
    procedure set_Total(x: Real);
    procedure set_Fecha(x: String);
    function get_Clave: Integer;
    function get_Total: Real;
    function get_Fecha: String;
    procedure calcular_total(subtotal: Real);
  end;

implementation

constructor CL_VENTA.Create(clv: Integer; fech: String);
begin
  Self.Clave := clv;
  Self.Total := 0;
  Self.Fecha := fech;
end;

procedure CL_VENTA.set_Clave(x: Integer);
begin
  Clave := x;
end;

procedure CL_VENTA.set_Total(x: Real);
begin
  Total := x;
end;

procedure CL_VENTA.set_Fecha(x: String);
begin
  Fecha := x;
end;

function CL_VENTA.get_Clave: Integer;
begin
  Result := Clave;
end;

function CL_VENTA.get_Total: Real;
begin
  Result := Total;
end;

function CL_VENTA.get_Fecha: String;
begin
  Result := Fecha;
end;
```

```
procedure CL_VENTA.calcular_total(subtotal: Real);
begin
  Total := Total + subtotal;
end;

end.
```

6.4. CL_VISTA_MENSAJE.pas

```
unit U_CL_VISTA_MENSAJE;

{$mode objfpc}{$H+}

{
Autor: Rodrigo Díaz Salguero
Fecha: 16/10/2024
Actualización: ---
}

interface

uses
  Crt, SysUtils, Dialogs;

type
  CL_VISTA_MENSAJE = class
  private
    mensaje: String;
  public
    procedure set_mensaje(x: String);
    function get_mensaje: String;
    procedure mostrar_mensaje;
  end;

implementation

procedure CL_VISTA_MENSAJE.set_mensaje(x: String);
begin
  mensaje := x;
end;

function CL_VISTA_MENSAJE.get_mensaje: String;
begin
  Result := mensaje;
end;

procedure CL_VISTA_MENSAJE.mostrar_mensaje;
begin
  ShowMessage(mensaje);
end;

end.
```

6.5. CL_IUSR_01.pas

```
unit u_cl_ejecutar_tienda;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls,
  u_cl_iusr_02,u_cl_iusr_03, U_CL_VENTA;

type

  { TIUSR_01 }

  TIUSR_01 = class(TForm)
    btn_registrar: TButton;
    btn_venta: TButton;
    btn_salir: TButton;
    Label1: TLabel;
    procedure evt_mostrar_registrar(Sender: TObject);
    procedure evt_mostrar_venta(Sender: TObject);
    procedure evt_terminar_ejecucion(Sender: TObject);
  private

  public

  end;

var
  IUSR_01: TIUSR_01;

implementation

{$R *.lfm}

{ TIUSR_01 }

procedure TIUSR_01.evt_mostrar_registrar(Sender: TObject);
begin
   IUSR_02.ShowModal;
end;

procedure TIUSR_01.evt_mostrar_venta(Sender: TObject);
begin
    IUSR_03.ShowModal;
end;

procedure TIUSR_01.evt_terminar_ejecucion(Sender: TObject);
begin
  IUSR_01.Close;
end;
```

```
end.
```

6.6. CL_IUSR_02.pas

```
unit u_cl_iusr_02;

{$mode ObjFPC}{$H+}

interface

uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls,
U_CL_PRODUCTO, U_CL_VISTA_MENSAJE;

type

  { TIUSR_02 }

  TIUSR_02 = class(TForm)
    btn_cerrar: TButton;
    btn_registrar_prod: TButton;
    ct_nombre_prod: TEdit;
    ct_clave_prod: TEdit;
    ct_precio_prod: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    procedure evt_cerrar_iusr_02(Sender: TObject);
    procedure evt_registrar_prod(Sender: TObject);

  private

  public

  end;

var
  v_nombre_prod, v_clave_prod, v_precio_prod: String;
  v_cont_prod: Integer=0;
  PRODUCTOS: array[1..500] of CL_PRODUCTO;
  IUSR_02: TIUSR_02;
  mensaje: CL_VISTA_MENSAJE;

implementation

{$R *.lfm}

{ TIUSR_02 }
```

```
procedure TIUSR_02.evt_cerrar_iusr_02(Sender: TObject);
begin
   IUSR_02.Close;
end;

procedure TIUSR_02.evt_registrar_prod(Sender: TObject);
begin

     mensaje:=CL_VISTA_MENSAJE.Create;
     v_nombre_prod:= IUSR_02.ct_nombre_prod.Text;
     v_clave_prod:= IUSR_02.ct_clave_prod.Text;
     v_precio_prod:= IUSR_02.ct_precio_prod.Text;
     IF NOT(v_nombre_prod='') AND NOT(v_clave_prod='') AND
NOT(v_precio_prod='') then
     begin
          v_cont_prod:= v_cont_prod+1;
          PRODUCTOS[v_cont_prod]:=
CL_PRODUCTO.Create(StrToInt(v_clave_prod),v_nombre_prod,StrToFloat(v_p
recio_prod));
          mensaje.set_mensaje('PRODUCTO REGISTRADO');
          mensaje.mostrar_mensaje;
     end
     else
     begin
          mensaje.set_mensaje('VERIFIQUE QUE LAS CAJAS DE TEXTO
CONTENGAN DATOS');
          mensaje.mostrar_mensaje;
     end;

end;

end.
```

6.7. CL_IUSR_03.pas

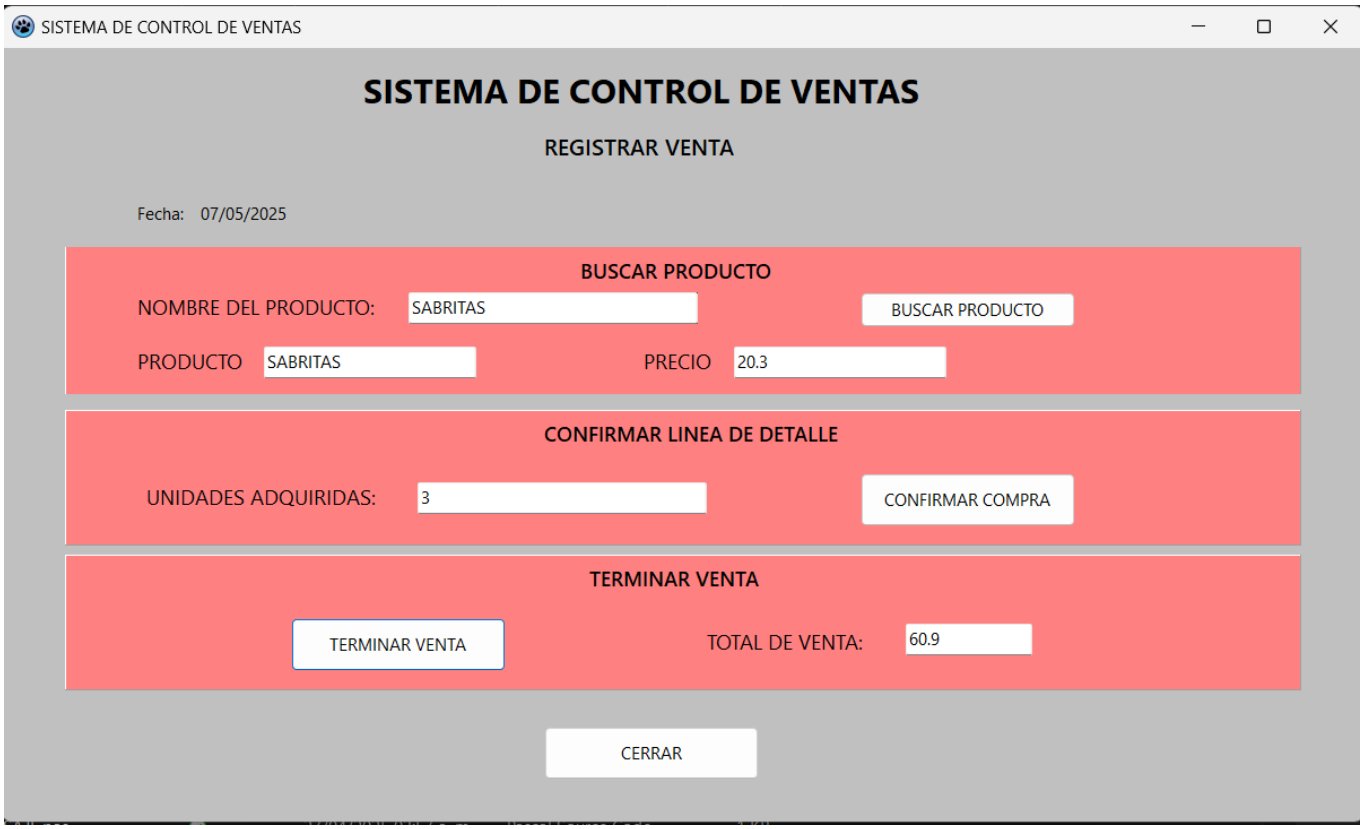
```
unit u_cl_iusr_03;

{$mode ObjFPC}{$H+}

interface

uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls,
ExtCtrls,
  u_cl_iusr_02, U_CL_LINEA_DETALLE, U_CL_VENTA;

type

  { TIUSR_03 }

  TIUSR_03 = class(TForm)
    btn_buscar: TButton;
    btn_conf_LD: TButton;
```

```
    btn_term_vta: TButton;
    btn_cerrar: TButton;
    ct_nombre_busc: TEdit;
    ct_prod_enc: TEdit;
    ct_preprod_enc: TEdit;
    ct_unidades: TEdit;
    ct_total: TEdit;
    Label1: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label2: TLabel;
    lbl_fecha: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    lbl_fecha1: TLabel;
    Panel1: TPanel;
    Panel2: TPanel;
    Panel3: TPanel;
    Panel4: TPanel;
    procedure evt_buscar_producto(Sender: TObject);
    procedure evt_cerrar_iusr_03(Sender: TObject);
    procedure evt_conf_LD(Sender: TObject);
    procedure evt_iniciar_venta(Sender: TObject);
    procedure evt_term_vta(Sender: TObject);
  private

  public

  end;

var
  v_nombre_busc, v_nombre_p,v_unidades:String;
  v_encontrado:Boolean;
  i, v_i_encontrado: Integer;
  v_cont_LD:Integer=0;
  v_subtotal, v_total:Real;
  LINEA_DETALLE: array[1..500] of CL_LINEA_DETALLE;
  v_cont_venta:Integer=0;
  v_fecha: TDateTime;
  VENTA: array[1..500] of CL_VENTA;
  IUSR_03: TIUSR_03;

implementation

{$R *.lfm}
{ TIUSR_03 }


procedure TIUSR_03.evt_cerrar_iusr_03(Sender: TObject);
begin
```

```
  If v_cont_LD>0 THEN
  BEGIN
       v_cont_LD:=0;
       IUSR_03.Close;
  end;
  If v_cont_LD=0 then
  begin
       VENTA[v_cont_venta].Destroy;
       v_cont_venta:=v_cont_venta-1;
       IUSR_03.Close;
  end;

end;

procedure TIUSR_03.evt_conf_LD(Sender: TObject);
begin
     v_unidades:= IUSR_03.ct_unidades.Text;
     If NOT(v_unidades='') then
     begin
          v_cont_LD:= v_cont_LD+1;
          LINEA_DETALLE[v_cont_LD]:=
CL_LINEA_DETALLE.Create(v_cont_LD,StrToInt(v_unidades),PRODUCTOS[v_i_e
ncontrado]);
          LINEA_DETALLE[v_cont_LD].calcular_subtotal;
          v_subtotal:= LINEA_DETALLE[v_cont_LD].get_Subtotal;
          VENTA[v_cont_venta].calcular_total(v_subtotal);
     end
     else
     begin
          mensaje.set_mensaje('Por favor llene el campo UNIDADES
ADQUIRIDAS para confirmar la Linea de Detalle');
          mensaje.mostrar_mensaje;
     end;
end;

procedure TIUSR_03.evt_iniciar_venta(Sender: TObject);
{EVENTO ONSHOW}
begin
    v_cont_venta:=v_cont_venta+1;
    v_fecha:=Date;

VENTA[v_cont_venta]:=CL_VENTA.Create(v_cont_venta,DateToStr(v_fecha));
    IUSR_03.lbl_fecha.Caption:=DateToStr(v_fecha);
end;

procedure TIUSR_03.evt_term_vta(Sender: TObject);
begin
     v_total:= VENTA[v_cont_venta].get_Total;
     IUSR_03.ct_total.Text:=FloatToStr(v_total);
end;

procedure TIUSR_03.evt_buscar_producto(Sender: TObject);
begin
  v_nombre_busc:=IUSR_03.ct_nombre_busc.Text;
```

```
  If v_nombre_busc='' then
  begin
     mensaje.set_mensaje('PARA BUSCAR UN PRODUCTO, INGRESE EL NOMBRE
EN EL CAMPO CORRESPONDIENTE');
     mensaje.mostrar_mensaje;
  end else
  begin
     v_encontrado := False;
     //MI METODO DE BUSQUEDA
     for i:=1 to v_cont_prod do
     begin
          v_nombre_p := PRODUCTOS[i].get_Nombre;
        if v_nombre_p = v_nombre_busc then
        begin
             v_i_encontrado := i;
             v_encontrado := True;
        end;
     end;
     If v_encontrado=True then
     begin
          v_nombre_prod:=PRODUCTOS[v_i_encontrado].get_Nombre;

v_precio_prod:=FloatToStr(PRODUCTOS[v_i_encontrado].get_Precio);
          IUSR_03.ct_prod_enc.Text:=v_nombre_prod;
          IUSR_03.ct_preprod_enc.Text:=v_precio_prod;
     end else
     begin
        mensaje.set_mensaje('PRODUCTO NO ENCONTRADO');
        mensaje.mostrar_mensaje;
     end;
   end;
end;

end.
```

6.8. EJECUCIÓN



7. IMPLEMENTACIÓN EN JAVA

7.1. CL_PRODUCTO.java

```
package com.mycompany.sistema_control_ventas;
/*
NOMBRE: RODRIGO DÍAZ SALGUERO
FECHA: 18/02/2025
ACTUALIZACIÓN: ---
*/
public class CL_PRODUCTO {
//---SECCIÓN DE ATRIBUTOS---//
    private int Clave;
    private String Nombre;
    private float Precio;
//---SECCIÓN DE MÉTODOS---//
    public void CL_PRODUCTO(int clave, String nom, float precio){
        Clave = clave;
        Nombre = nom;
        Precio = precio;
    }

    public void set_Clave(int x){
        Clave = x;
```

```
    }

    public void set_Nombre(String x){
        Nombre = x;
    }

    public void set_Precio(float x){
        Precio = x;
    }

    public int get_Clave(){
        return Clave;
    }

    public String get_Nombre(){
        return Nombre;
    }

    public float get_Precio(){
        return Precio;
    }
}
```

7.2. CL_LINEA_DETALLE.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package com.mycompany.sistema_control_ventas;

/**
 *
 * @author rodri
 */
public class CL_LINEA_DETALLE {
/*
NOMBRE: RODRIGO DÍAZ SALGUERO
FECHA: 18/02/2025
ACTUALIZACIÓN: ---
*/
    // SECCIÓN DE ATRIBUTOS
    private int Clave;
    private int Unidades_Adquiridas;
    private double Subtotal;
    private CL_PRODUCTO Producto;

    // SECCIÓN DE MÉTODOS

    public CL_LINEA_DETALLE(int clave, int ua, CL_PRODUCTO prod) {
        Clave = clave;
        Unidades_Adquiridas = ua;
```

```
        Subtotal = 0;
        Producto = prod;
    }

    public void set_Clave(int x) {
        Clave = x;
    }

    public void set_Unidades_Adquiridas(int x) {
        Unidades_Adquiridas = x;
    }

    public void set_Subtotal(double x) {
        Subtotal = x;
    }

    public void set_Producto(CL_PRODUCTO x) {
        Producto = x;
    }

    public int get_Clave() {
        return Clave;
    }

    public int get_Unidades_Adquiridas() {
        return Unidades_Adquiridas;
    }

    public double get_Subtotal() {
        return Subtotal;
    }

    public CL_PRODUCTO get_Producto() {
        return Producto;
    }

    public void calcular_subtotal() {
        Subtotal = Producto.get_Precio() * Unidades_Adquiridas;
    }
}
```

7.3. CL_VENTA.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package com.mycompany.sistema_control_ventas;

/**
 *
 * @author rodri
```

```
 */
public class CL_VENTA {
/*
NOMBRE: RODRIGO DÍAZ SALGUERO
FECHA: 18/02/2025
ACTUALIZACIÓN: ---
*/
    // SECCIÓN DE ATRIBUTOS
    private int Clave;
    private double Total;
    private String Fecha;

    // SECCIÓN DE MÉTODOS

    public CL_VENTA(int clave, String fech) {
        Clave = clave;
        Total = 0;
        Fecha = fech;
    }

    public void set_Clave(int x) {
        Clave = x;
    }

    public void set_Total(double x) {
        Total = x;
    }

    public void set_Fecha(String x) {
        Fecha = x;
    }

    public int get_Clave() {
        return Clave;
    }

    public double get_Total() {
        return Total;
    }

    public String get_Fecha() {
        return Fecha;
    }

    public void calcular_total(double subtotal) {
        Total = Total + subtotal;
    }
}
```

7.4. CL_VISTA_MENSAJE.java

```
package com.mycompany.sistema_control_ventas;
import javax.swing.JOptionPane;

public class CL_VISTA_MENSAJE {
/*
Autor: Rodrigo Díaz Salguero
Fecha: 16/10/2024
Actualización: 25/03/2025
COMENTARIO: Se actualizó para poder mostrar
mensajes en una interfaz GUI
*/

    // SECCIÓN DE ATRIBUTOS
    private String mensaje;

    // MÉTODO set_mensaje
    public void set_mensaje(String x) {
        mensaje = x;
    }

    // MÉTODO get_mensaje
    public String get_mensaje() {
        return mensaje;
    }

    // MÉTODO mostrar_mensaje
    public void mostrar_mensaje() {
        javax.swing.JOptionPane.showMessageDialog(null, mensaje);
    }
}
```