# UNIVERSIDAD DE IXTLAHUACA CUI

**FACULTAD DE INGENIERÍA**

**LICENCIATURA EN INGENIERÍA EN COMPUTACIÓN**

**CLAVE DE INCORPORACIÓN: 091-E**

# DOCUMENTACIÓN PARA EL DESARROLLO DE UN SISTEMA DE CONTROL DE VENTAS

**Autor: RODRIGO DÍAZ SALGUERO**

**Catedrático: Ing. ALBERTO CANO GARCÍA**

**CICLO ESCOLAR 2025A**

Fecha: 10/07/2025

# 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. Esta tienda tendrá 100 productos registrados con una taza de crecimiento del 0% a tres años, tiene aproximadamente (N cantidad) de ventas anualmente.

## 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. De igual manera el tendero tiene la necesidad de poder consultar la información de sus productos y ventas de una forma efectiva, así como de registrar productos, consultarlos, modificar si existiera algún error, o eliminar, así como de consultar, modificar o eliminar ventas.

## 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 permanentemente los productos y las ventas en memoria secundaria por medio de archivos estructurados de acceso directo y permitir las operaciones de un CRUD, el segundo módulo deberá permitir realizar el proceso de venta de una forma ágil y satisfacer las necesidades del tendero.

## 1.4 Análisis

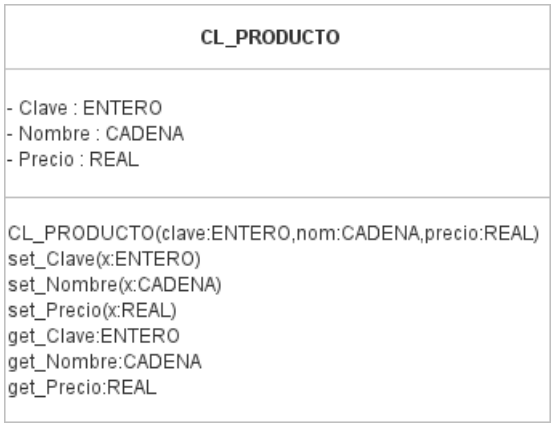
Primero podemos lograr identificar nuestras clases modelo y las relaciones entre estas.

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

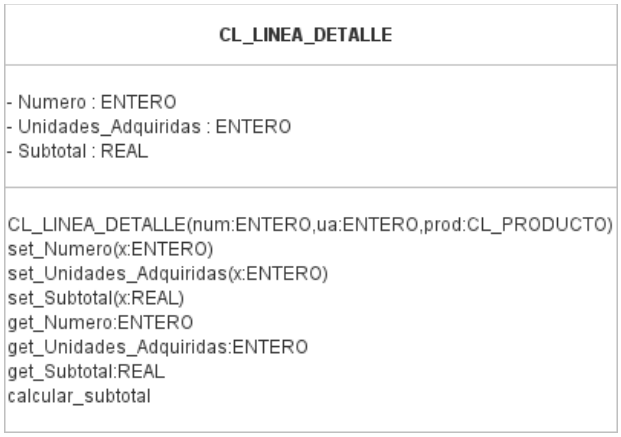
identificar otro elemento significativo, entonces podemos pensar en una clase CL_LINEA_DETALLE de la cuál se puede identificar las unidades adquiridas, el nombre del producto, el precio del producto, clave y subtotal.
De igual manera podemos identificar que una venta se compone por un número de venta, una fecha de venta, por las líneas de detalle y por un total. Entonces podemos identificar la clase CL_VENTA.

## 2. MODELADO DE CLASES MODELO

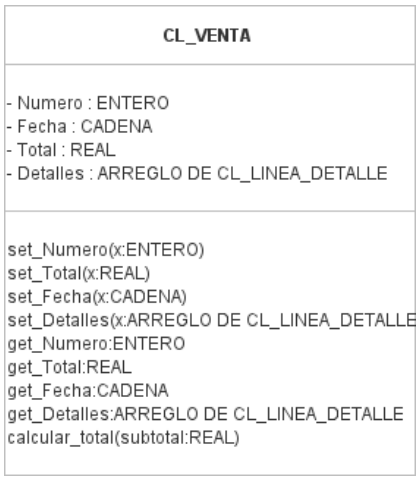
### 2.1. Modelado de CL_PRODUCTO




### 2.2. Modelado de CL_LINEA_DETALLE




### 2.3. Modelado de CL_VENTA




## 3. PSEUDOCÓDIGOS DE 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

      Numero : ENTERO, PRIVADO
      Unidades_Adquiridas : ENTERO, PRIVADO
      Subtotal : REAL, PRIVADO
      Producto : CL_PRODUCTO

      SECCION DE METODOS

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

      set_Numero(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_Numero: 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

      Numero : ENTERO, PRIVADO
      Total : REAL, PRIVADO
      Fecha : CADENA, PRIVADO
      Detalles : ARREGLO DE CL_LINEA_DETALLE, PRIVADO
```

```
      SECCIÓN DE MÉTODOS

      set_Numero(x:ENTERO)
      INICIO
            Numero <- x
      FIN MÉTODO set_Numero

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

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

      set_Detalles(x:ARREGLO DE CL_LINEA_DETALLE)
      INICIO
            Detalles <- x
      FIN MÉTODO set_Detalles

      get_Numero:ENTERO
      INICIO
            REGRESAR Numero
      FIN MÉTODO get_Numero

      get_Total:REAL
      INICIO
            REGRESAR Total
      FIN MÉTODO get_Total

      get_Fecha:CADENA
      INICIO
            REGRESAR Fecha
      FIN MÉTODO get_Fecha

      get_Detalles:ARREGLO DE CL_LINEA_DETALLE
      INICIO
            REGRESAR Detalles
      FIN MÉTODO get_Detalles

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

FIN CLASE CL_VENTA
```

## 4. DEFINICIÓN DE ESTRUCTURA DE ARCHIVOS

Este programa se trabajará con almacenamiento permanente en archivos estructurados con una organización de acceso directa y secuencial, por lo que es necesario definir el tamaño de nuestro archivo para los productos. Los antecedentes del problema nos indican que en la tienda se tendrán registrados 100 productos durante tres años con una taza de crecimiento del 0% anual, por lo tanto vamos a calcular el total de registros necesarios para los productos, al total de registros se le añadirá un 10% extra que se reservará como zona especial del archivo al final para almacenar colisiones cuando ocurra un duplicado de hash.

$$tam = 100 \times 1.1$$

$$tam = 110$$

Los cálculos realizados nos muestran que el tamaño de nuestro archivo es de 110 registros, 100 registros para hashing directo y a partir del registro número 101 y hasta el 110 tenemos 10 posiciones para la zona de colisiones.

Las claves únicas de productos serán de 4 dígitos y se utilizará la técnica de direccionamiento ***Hash Módulo 99*** para obtener el índice de registro según la clave de cada registro. Se decidió que la zona de colisiones iniciaría a partir del registro número 99 y hasta el 110 debido a que esta técnica de direccionamiento resulta números enteros entre 0 y 98.

Estructura del archivo 'PRODUCTOS.rod':

| CLAVE | NOMBRE | PRECIO |
|---|---|---|
| 0000 | SABRITAS | 18.7 |

Se tomó una decisión de diseño en la arquitectura de nuestro programa para separar las responsabilidades de nuestras clases modelo CL_PRODUCTO y CL_VENTA, creando clases dedicadas específicamente a la gestión de los archivos que usamos a lo largo de la ejecución de nuestro programa para hacer permanentes los datos, por lo que se implementarán las clases GESTOR_ARCH_PRODUCTO y GESTOR_ARCH_VENTA. Así nuestras clases modelo están dedicadas únicamente a modelar el producto, línea de detalle y venta.

El archivo de ventas se manejará con archivos estructurados de organización de acceso secuencial.

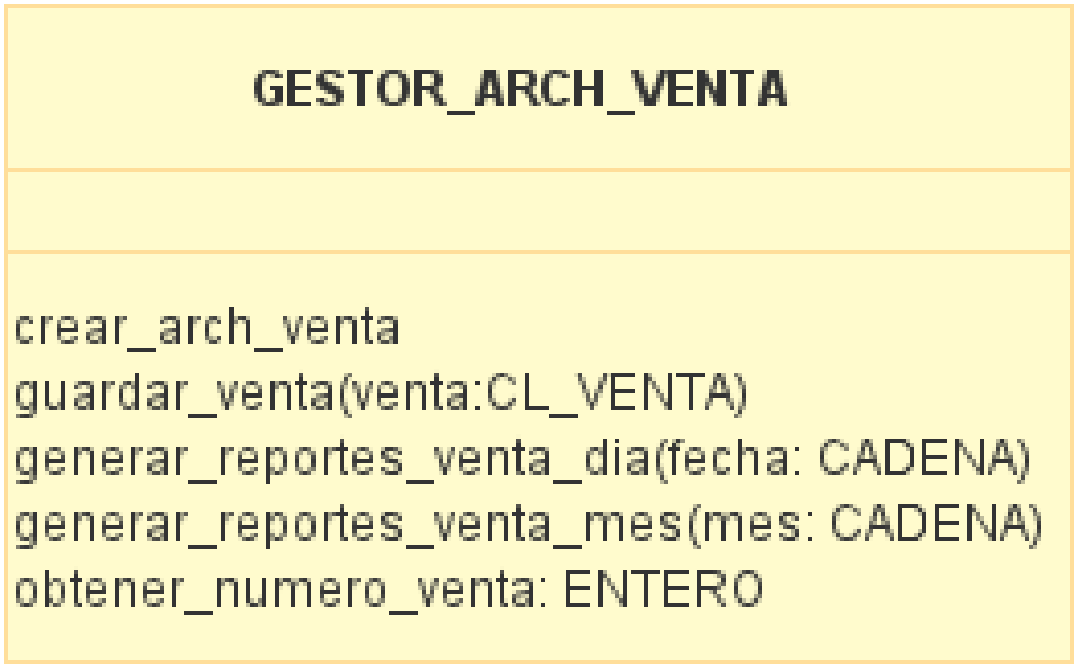
Estructura del archivo 'VENTAS.rod':

| No. Venta | Fecha | Línea de detalle [1] | | | | ... | Línea de detalle [20] | | | | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Producto | Precio | Unidades | Subtotal | | Producto | Precio | Unidades | Subtotal | |
| | | | | | | | | | | | |

## 5. MODELADO DE CLASES DE GESTIÓN DE ARCHIVOS

### 5.1. Modelado de GESTOR_ARCH_PRODUCTO


### 5.2. Modelado de GESTOR_ARCH_VENTA



## 6. PSEUDOCÓDIGO DE LAS CLASES GESTORAS DE ARCHIVOS

### 6.1. Pseudocódigo de clase GESTOR_ARCH_PRODUCTO

```
CLASE GESTOR_ARCH_PRODUCTO
/*
Autor: Rodrigo Díaz Salguero
Fecha: 07/06/2025
Actualización: ---
*/
INICIO
      SECCIÓN DE ATRIBUTOS
            porcentaje_uso_log : REAL
            porcentaje_uso_fis : REAL

      SECCIÓN DE MÉTODOS
            set_porcentaje_uso_log(x:REAL)
            INICIO
                  porcentaje_uso_log<--x
            FIN MÉTODO set_porcentaje_uso_log

            set_porcentaje_uso_fis(x:REAL)
            INICIO
                  porcentaje_uso_fis<--x
            FIN MÉTODO set_porcentaje_uso_fis

            get_porcentaje_uso_log:REAL
            INICIO
                  REGRESAR porcentaje_uso_log
            FIN MÉTODO get_porcentaje_uso_log

            get_porcentaje_uso_fis:REAL
            INICIO
                  REGRESAR porcentaje_uso_fis
            FIN MÉTODO get_porcentaje_uso_fis

            MÉTODO PÚBLICO crear_arch_producto(tam:ENTERO)
            SECCIÓN DE TIPOS
                  reg_producto=ESTRUCTURA
                        campo_clave: ENTERO
                        campo_nombre: CADENA
                        campo_precio: REAL
                        campo_disponible: ENTERO
                  FIN ESTRUCTURA
            SECCIÓN DE VARIABLES
                  f_producto: ARCHIVO DE reg_producto
                  v_reg_producto: reg_producto
                  i:ENTERO
            INICIO
                  CREAR("PRODUCTOS.rod",f_producto)
                  v_reg_producto.campo_clave<--0
                  v_reg_producto.campo_nombre<--""
                  v_reg_producto.campo_precio<--0
                  v_reg_producto.campo_disponible<-- -99
                  PARA i desde 1 hasta tam
```

```
                        ESCRIBIR(v_reg_producto,f_producto)
                  FIN PARA
                  CERRAR(f_producto)
            FIN MÉTODO crear_arch_producto

            MÉTODO PÚBLICO
guardar_producto(producto:CL_PRODUCTO):BOOLEANO
            SECCIÓN DE TIPOS
                  reg_producto=ESTRUCTURA
                        campo_clave: ENTERO
                        campo_nombre: CADENA
                        campo_precio: REAL
                        campo_disponible:ENTERO
                  FIN ESTRUCTURA
            SECCIÓN DE VARIABLES
                  v_reg_producto,v_leer_reg: reg_producto
                  f_producto:ARCHIVO DE v_reg_producto
                  v_pos, v_disponible:ENTERO
                  v_retorno: BOOLEANO
            INICIO
                  v_retorno<--FALSO
                  v_reg_producto.campo_clave <-- producto.get_Clave
                  v_reg_producto.campo_nombre <-- producto.get_Nombre
                  v_reg_producto.campo_precio <-- producto.get_Precio
                  v_reg_producto.campo_disponible<--1
                  v_pos <-- v_reg_producto.campo_clave MOD 99
                  ASIGNAR('PRODUCTOS.rod',f_producto)
                  ABRIR(f_producto)
                  POSICIONAR(f_producto,v_pos)
                  LEER(v_leer_reg,f_producto)
                  SI v_leer_reg.campo_disponible=-99
                        POSICIONAR(f_producto,v_pos)
                        ESCRIBIR(v_reg_producto,f_producto)
                        v_retorno<--VERDADERO
                  SI NO //COLISIÓN
                        SI v_leer_reg.campo_clave=/producto.get_Clave
                              v_pos<--99
                              POSICIONAR(f_producto,v_pos)//ZONA DE
COLISIONES
                              REPETIR //ACCESO SECUECNIAL EN ZONA DE
COLISIONES
                                    LEER(v_leer_reg,f_producto)
                                    v_disponible<--
v_leer_reg.campo_disponible
                                    v_pos<--v_pos+1
                              HASTA (v_disponible=-99 O
v_pos=110)//TAMAÑO PARA 132 REGISTROS
                              ESCRIBIR(v_reg_producto,f_producto)
                              v_retorno<--VERDADERO
                        FIN SI
                  FIN SI
                  CERRAR(f_producto)
                  RETORNAR v_retorno
            FIN MÉTODO guardar_producto
```

```
            MÉTODO PÚBLICO
consultar_producto(clave:ENTERO,producto:CL_PRODUCTO):BOOLEANO//produc
to por referencia
            SECCIÓN DE TIPOS
                  reg_producto=ESTRUCTURA
                        campo_clave: ENTERO
                        campo_nombre: CADENA
                        campo_precio: REAL
                        campo_disponible:ENTERO
                  FIN ESTRUCTURA
            SECCIÓN DE VARIABLES
                  f_producto: ARCHIVO DE reg_producto
                  v_reg_producto: reg_producto
                  v_retorno: BOOLEANO
                  v_pos:ENTERO
            INICIO
                  ASIGNAR('PRODUCTOS.rod',f_producto)
                  ABRIR(f_producto)
                  v_retorno<--FALSO
                  v_pos<--clave MOD 99
                  POSICIONAR(f_producto,v_pos)
                  LEER(v_reg_producto,f_producto)
                  SI v_reg_producto.campo_disponible=1 Y
v_reg_producto.campo_clave=clave
                        v_retorno<--VERDADERO
                        producto.set_Clave(v_reg_producto.campo_clave)

      producto.set_Nombre(v_reg_producto.campo_nombre)

      producto.set_Precio(v_reg_producto.campo_precio)
                  SI NO
                        v_pos <-- 99 //Zona de colisiones
                        POSICIONAR(f_producto,v_pos)
                        REPITE
                              LEER(v_reg_producto,f_producto)
                              v_pos <-- v_pos+1
                        HASTA FDA(f_producto)=VERDADERO O
(v_reg_producto.campo_disponible=1 Y v_reg_producto.campo_clave=clave)
                        SI v_reg_producto.campo_disponible=1 Y
v_reg_producto.campo_clave=clave
                              v_retorno <-- VERDADERO

      producto.set_Clave(v_reg_producto.campo_clave)

      producto.set_Nombre(v_reg_producto.campo_nombre)

      producto.set_Precio(v_reg_producto.campo_precio)
                        FIN SI
                  FIN SI
                  CERRAR(f_producto)
                  RETORNAR v_retorno
            FIN MÉTODO consultar_producto
```

```
            MÉTODO PÚBLICO eliminar_producto(clave:ENTERO)
            SECCIÓN DE TIPOS
                  reg_producto=ESTRUCTURA
                        campo_clave: ENTERO
                        campo_nombre: CADENA
                        campo_precio: REAL
                        campo_disponible: ENTERO
                  FIN ESTRUCTURA
            SECCIÓN DE VARIABLES
                  f_producto: ARCHIVO DE reg_producto
                  v_reg_producto: reg_producto
                  v_retorno: BOOLEANO
                  v_pos: ENTERO
            INICIO
                  v_retorno<--FALSO
                  v_pos<--clave MOD 99
                  ABRIR(f_producto)
                  POSICIONAR(f_producto,v_pos)
                  LEER(v_reg_producto,f_producto)
                  SI v_reg_producto.campo_disponible=1 Y
v_reg_producto.campo_clave=clave //Existe
                        POSICIONAR(f_producto,v_pos)
                        v_reg_producto.campo_disponible<-- -99
                        ESCRIBIR(v_reg_producto,f_producto)
                        v_retorno<--VERDADERO
                  SI NO
                        v_pos <-- 99 //Zona de colisiones
                        POSICIONAR(f_producto,v_pos)
                        REPITE
                              LEER(v_reg_producto,f_producto)
                              v_pos <-- v_pos+1
                        HASTA FDA(f_producto)=VERDADERO O
(v_reg_producto.campo_disponible=1 Y v_reg_producto.campo_clave=clave)
                        SI v_reg_producto.campo_disponible=1 Y
v_reg_producto.campo_clave=clave
                              v_retorno <-- VERDADERO
                              v_reg_producto.campo_disponible <-- -99
                              POSICIONAR(f_producto,v_pos-1)
                              ESCRIBIR(v_reg_producto,f_producto)
                        FIN SI
                  FIN SI
                  CERRAR(f_producto)
                  RETORNAR v_retorno
            FIN MÉTODO eliminar_producto

            MÉTODO PÚBLICO borrado_fisico
            SECCIÓN DE TIPOS
                  reg_producto=ESTRUCTURA
                        campo_clave: ENTERO
                        campo_nombre: CADENA
                        campo_precio: REAL
                        campo_disponible: ENTERO
                  FIN ESTRUCTURA
            SECCIÓN DE VARIABLES
```

```
            f_producto, f_temporal: ARCHIVO DE reg_producto
            v_reg_producto: reg_producto
            v_marca, v_contador:ENTERO
      INICIO
            v_contador<--0
            CREAR('TEMPORAL.rod',f_temporal) //QUEDA ABIERTO PARA
ESCRITURA
            ABRIR(f_producto)
            MIENTRAS FDA(f_producto)=FALSO
                  LEER(v_reg_producto,f_producto)
                  v_contador<--v_contador+1
                  v_marca<--v_reg_producto.campo_disponible
                  SI v_marca=/-99
                        POSICIONAR(f_temporal,v_contador)
                        ESCRIBIR(v_reg_producto,f_temporal)
                  FIN SI
            FIN MIENTRAS
            CERRAR(f_producto)
            ELIMINAR(f_producto)
            RENOMBRAR('PRODUCTOS.rod',f_temporal)
            CERRAR(f_temporal)
      FIN MÉTODO borrado_fisico

      MÉTODO PÚBLICO obtener_porcentaje_uso_log(tam:ENTERO)
      SECCIÓN DE TIPOS
            reg_producto=ESTRUCTURA
                  campo_clave: ENTERO
                  campo_nombre: CADENA
                  campo_precio: REAL
                  campo_disponible: ENTERO
            FIN ESTRUCTURA
      SECCIÓN DE VARIABLES
            f_producto: ARCHIVO DE reg_producto
            v_reg_producto: reg_producto
            v_contador,cant_reg_log:ENTERO
      INICIO
      v_contador<--1
      cant_reg_log<--0
      ASIGNAR('PRODUCTOS.rod',f_producto)
      ABRIR(f_producto)
      REPETIR
            POSICIONAR(f_producto,v_contador)
            LEER(v_reg_producto,f_producto)
            SI v_reg_producto.campo_disponible=1 ENTONCES
                  cant_reg_log<--cant_reg_log+1
            FIN SI
            v_contador<--v_contador+1
      HASTA v_contador>tam
      CERRAR(f_producto)
      //Porcentaje de uso lógico
      porcentaje_uso_log<--(cant_reg_log/tam)*100
      FIN MÉTODO obtener_porcentaje_uso_log

      MÉTODO PÚBLICO obtener_porcentaje_uso_fis(tam:ENTERO)
```

```
            SECCIÓN DE TIPOS
                  reg_producto=ESTRUCTURA
                        campo_clave: ENTERO
                        campo_nombre: CADENA
                        campo_precio: REAL
                        campo_disponible: ENTERO
                  FIN ESTRUCTURA
            SECCIÓN DE VARIABLES
                  f_producto: ARCHIVO DE reg_producto
                  v_reg_producto: reg_producto
                  v_contador,cant_reg_fis:ENTERO
            INICIO
            v_contador<--1
            cant_reg_fis<--0
            ASIGNAR('PRODUCTOS.rod',f_producto)
            ABRIR(f_producto)
            REPETIR
                  POSICIONAR(f_producto,v_contador)
                  LEER(v_reg_producto,f_producto)
                  SI v_reg_producto.campo_clave=/0 ENTONCES
                        cant_reg_fis<--cant_reg_fis+1
                  FIN SI
                  v_contador<--v_contador+1
            HASTA v_contador>tam
            CERRAR(f_producto)
            //Porcentaje de uso físico
            porcentaje_uso_fis<--(cant_reg_fis/tam)*100
            FIN MÉTODO obtener_porcentaje_uso_fis

            MÉTODO PÚBLICO obtener_diferencia_uso:REAL
            SECCIÓN DE VARIABLES
                  diferencia:REAL
            INICIO
                  diferencia<-- ABS(porcentaje_uso_log-
porcentaje_uso_fis)
                  REGRESAR diferencia
            FIN MÉTODO obtener_diferencia_uso

FIN CLASE GESTOR_ARCH_PRODUCTO
```

### 6.2. Pseudocódigo de clase GESTOR_ARCH_VENTA

```
CLASE GESTOR_ARCH_VENTA
/*
Autor: Rodrigo Díaz Salguero
Fecha: 07/06/2025
Actualización: ---
*/
INICIO

      SECCIÓN DE MÉTODOS

      MÉTODO crear_arch_venta
```

```
SECCIÓN DE TIPOS
      reg_venta=ESTRUCTURA
      campo_numero: ENTERO
      campo_fecha: CADENA
      campo_linea_det: ARREGLO DE CL_LINEA_DETALLE
      campo_total: REAL
      FIN ESTRUCTURA
SECCIÓN DE VARIABLES
      f_venta: ARCHIVO DE reg_venta
INICIO
      CREAR("VENTAS.rod",f_venta)
      CERRAR(f_venta)
FIN MÉTODO crear_arch_venta

MÉTODO guardar_venta(venta: CL_VENTA)
SECCIÓN DE TIPOS
      reg_venta=ESTRUCTURA
      campo_numero: ENTERO
      campo_fecha: CADENA
      campo_linea_det: ARREGLO DE CL_LINEA_DETALLE
      campo_total: REAL
      FIN ESTRUCTURA
SECCIÓN DE VARIABLES
      v_reg_venta: reg_venta
      f_venta:ARCHIVO DE reg_venta
INICIO
      ASIGNAR(f_venta,'VENTAS.rod')
      ABRIR(f_venta)
      MIENTRAS FDA(f_venta)=FALSO
            LEER(v_reg_venta,f_venta)
      FIN MIENTRAS
      v_reg_venta.campo_numero <-- venta.get_Numero
      v_reg_venta.campo_fecha <-- venta.get_Fecha
      v_reg_venta.campo_linea_det <-- venta.get_Detalles
      v_reg_venta.campo_total <-- venta.get_Total
      ESCRIBIR(v_reg_venta,f_venta)
      CERRAR(f_venta)
FIN MÉTODO guardar_venta

MÉTODO generar_reporte_ventas_dia(fecha: CADENA)
SECCIÓN DE TIPOS
      reg_venta=ESTRUCTURA
      campo_numero: ENTERO
      campo_fecha: CADENA
      campo_linea_det: ARREGLO DE CL_LINEA_DETALLE
      campo_total: REAL
      FIN ESTRUCTURA
SECCIÓN DE VARIABLES
      f_venta: ARCHIVO DE reg_venta
      f_reporte: ARCHIVO DE TEXTO
      v_reg_venta: reg_venta
      registro_encontrado:CADENA
      total_dia: REAL
INICIO
```

```
            total_dia <-- 0
            ASIGNAR(f_reporte,'REPORTE-DIA.rod')
            ABRIR(f_reporte)
            ESCRIBIR('REPORTE DE VENTAS DEL DÍA '+fecha,f_reporte)
            ESCRIBIR('NO. VENTA'+' | '+'  FECHA  '+' | '+'TOTAL
VENTA',f_reporte)
            ASIGNAR(f_venta,'VENTAS.rod')
            ABRIR(f_venta)
            MIENTRAS FDA(f_venta)=FALSO
                  LEER(v_reg_venta, f_venta)
                  SI v_reg_venta.campo_fecha=fecha
                        registro_encontrado <-- '
'+v_reg_venta.campo_numero+'   '+' | '+'v_reg_venta.campo_fecha+' |
'+'   '+v_reg_venta.campo_total
                        ESCRIBIR(registro_encontrado,f_reporte)
                        total_dia <-- total_dia+v_reg_venta.campo_total
                  FIN SI
            FIN MIENTRAS
            ESCRIBIR('TOTAL VENDIDO EN EL DIA: '+total_dia)
            CERRAR(f_reporte)
            CERRAR(f_venta)
      FIN MÉTODO generar_reporte_ventas_dia

      MÉTODO generar_reporte_ventas_mes(mes: ENTERO)
      SECCIÓN DE TIPOS
            reg_venta=ESTRUCTURA
            campo_numero: ENTERO
            campo_fecha: CADENA
            campo_linea_det: ARREGLO DE CL_LINEA_DETALLE
            campo_total: REAL
            FIN ESTRUCTURA
      SECCIÓN DE VARIABLES
            f_venta: ARCHIVO DE reg_venta
            f_reporte: ARCHIVO DE TEXTO
            v_reg_venta: reg_venta
            registro_encontrado,mes_registrado:CADENA
            total_mes: REAL
      INICIO
            total_dia <-- 0
            ASIGNAR(f_reporte,'REPORTE_MENSUAL--'+mes+'.rod')
            ABRIR(f_reporte)
            ESCRIBIR('REPORTE DE VENTAS DEL MES '+mes,f_reporte)
            ESCRIBIR('NO. VENTA'+' | '+'  FECHA  '+' | '+'TOTAL
VENTA',f_reporte)
            ASIGNAR(f_venta,'VENTAS.rod')
            ABRIR(f_venta)
            MIENTRAS FDA(f_venta)=FALSO
                  LEER(v_reg_venta, f_venta)
                  mes_registrado <-- v_reg_venta.campo_fecha
                  mes_registrado <-- SUBCADENA(mes_registrado,4,7)
                  SI mes_registrado=mes
                        registro_encontrado <-- '
'+v_reg_venta.campo_numero+'   '+' | '+'v_reg_venta.campo_fecha+' |
'+'   '+v_reg_venta.campo_total
```

```
                        ESCRIBIR(registro_encontrado,f_reporte)
                        total_mes <-- total_mes+v_reg_venta.campo_total
                  FIN SI
            FIN MIENTRAS
            ESCRIBIR('TOTAL VENDIDO EN EL MES: '+total_mes)
            CERRAR(f_reporte)
            CERRAR(f_venta)
      FIN MÉTODO generar_reporte_ventas_mes

      MÉTODO obtener_numero_venta: ENTERO
      SECCIÓN DE TIPOS
            reg_venta=ESTRUCTURA
            campo_numero: ENTERO
            campo_fecha: CADENA
            campo_linea_det: ARREGLO DE CL_LINEA_DETALLE
            campo_total: REAL
            FIN ESTRUCTURA
      SECCIÓN DE VARIABLES
            f_venta: ARCHIVO DE reg_venta
            v_reg_venta: reg_venta
            v_retorno: ENTERO
      INICIO
            ASIGNAR(f_venta,'VENTAS.rod')
            ABRIR(f_venta)
            v_retorno <-- 0
            MIENTRAS (FDA(f_venta)=FALSO) ENTONCES
                  LEER(v_reg_venta,f_venta)
                  v_retorno <-- v_reg_venta.campo_numero
            FIN MIENTRAS
            CERRAR(f_venta)
            REGRESAR v_retorno
      FIN MÉTODO obtener_numero_venta

FIN CLASE GESTOR_ARCH_VENTA
```

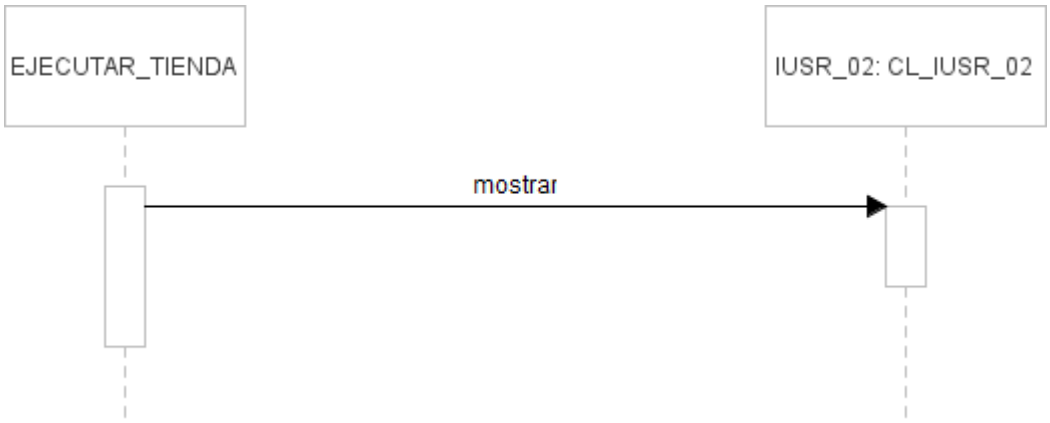
## 7. MODELADO DE LA CLASE VISTA

### 7.1. Diseño de la interfaz IUSR_01


### 7.2. Modelado de la clase CL_IUSR_01

### 7.3. Determinación de eventos de la interfaz IUSR_01

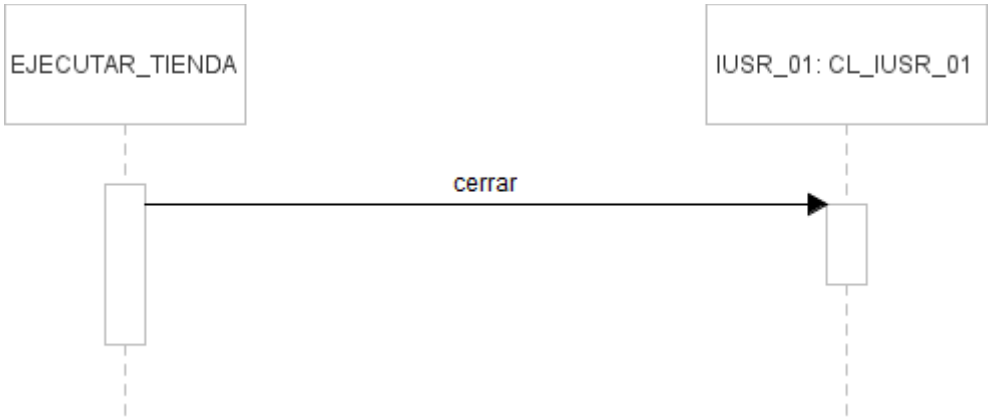
- evt_mostrar_gestionar: Evento CLICK en btn_gestionar, el programa deberá generar las acciones necesarias para que se muestre la interfaz IUSR_02




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

- evt_mostrar_gen_reporte: Evento Click en btn_gen_reporte debe generar las acciones necesarias para mostrar la interfaz IUSR_04.


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




### 7.4. Pseudocódigos de los eventos de IUSR_01

- **Pseudocódigo de evt_mostrar_gestionar**

```
INICIO
      IUSR_02.mostrar
FIN
```

- **Pseudocódigo de evt_mostrar_reg_venta**

```
INICIO
      IUSR_03.mostrar
FIN
```

- **Pseudocódigo de evt_mostrar_gen_reporte**

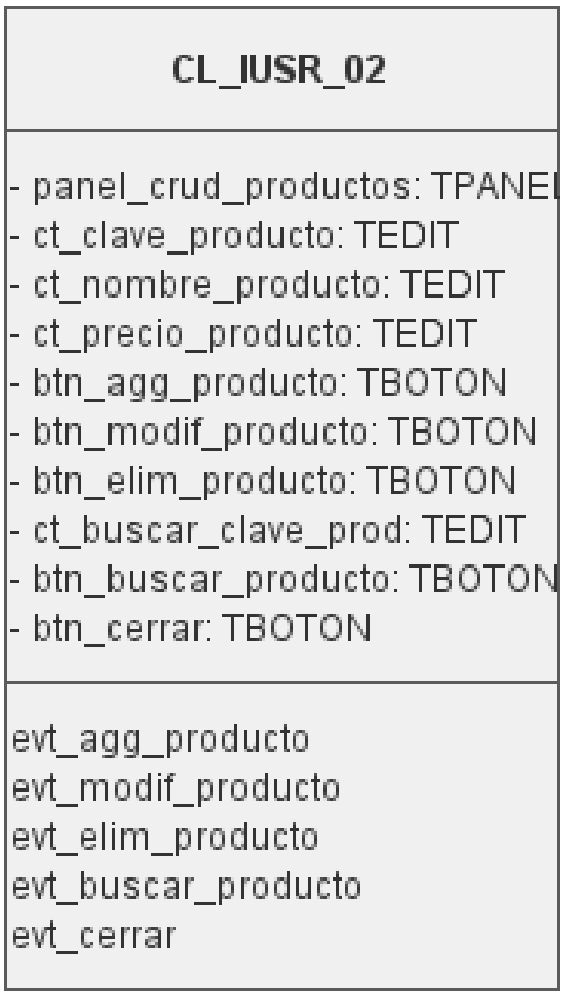
```
INICIO
      IUSR_04.mostrar
FIN
```

- **Pseudocódigo de evt_term_ejecucion**

```
INICIO
      IUSR_01.cerrar
FIN
```

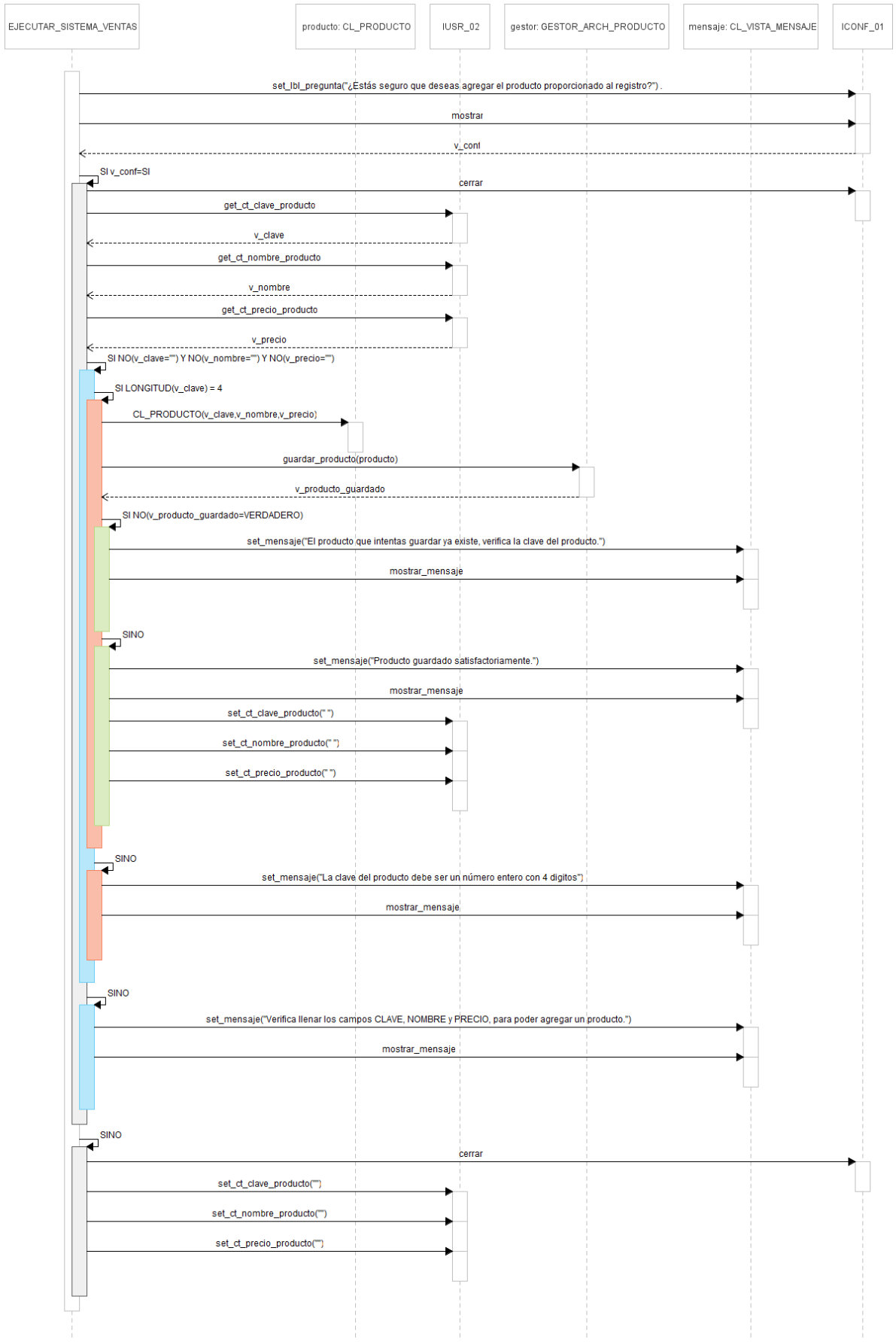
### 7.5. Diseño de interfaz IUSR_02

### 7.6. Modelado de la clase CL_IUSR_02

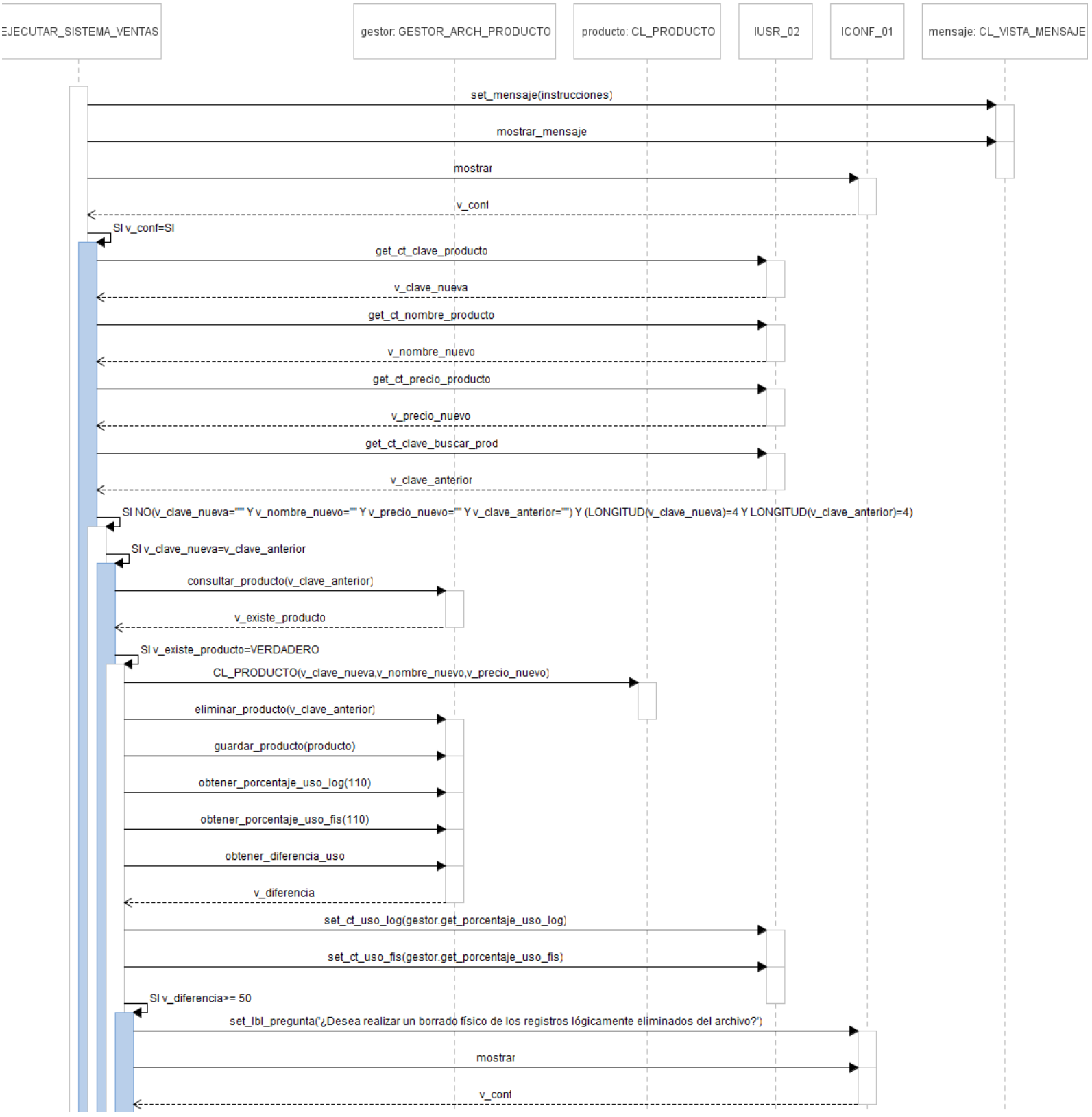


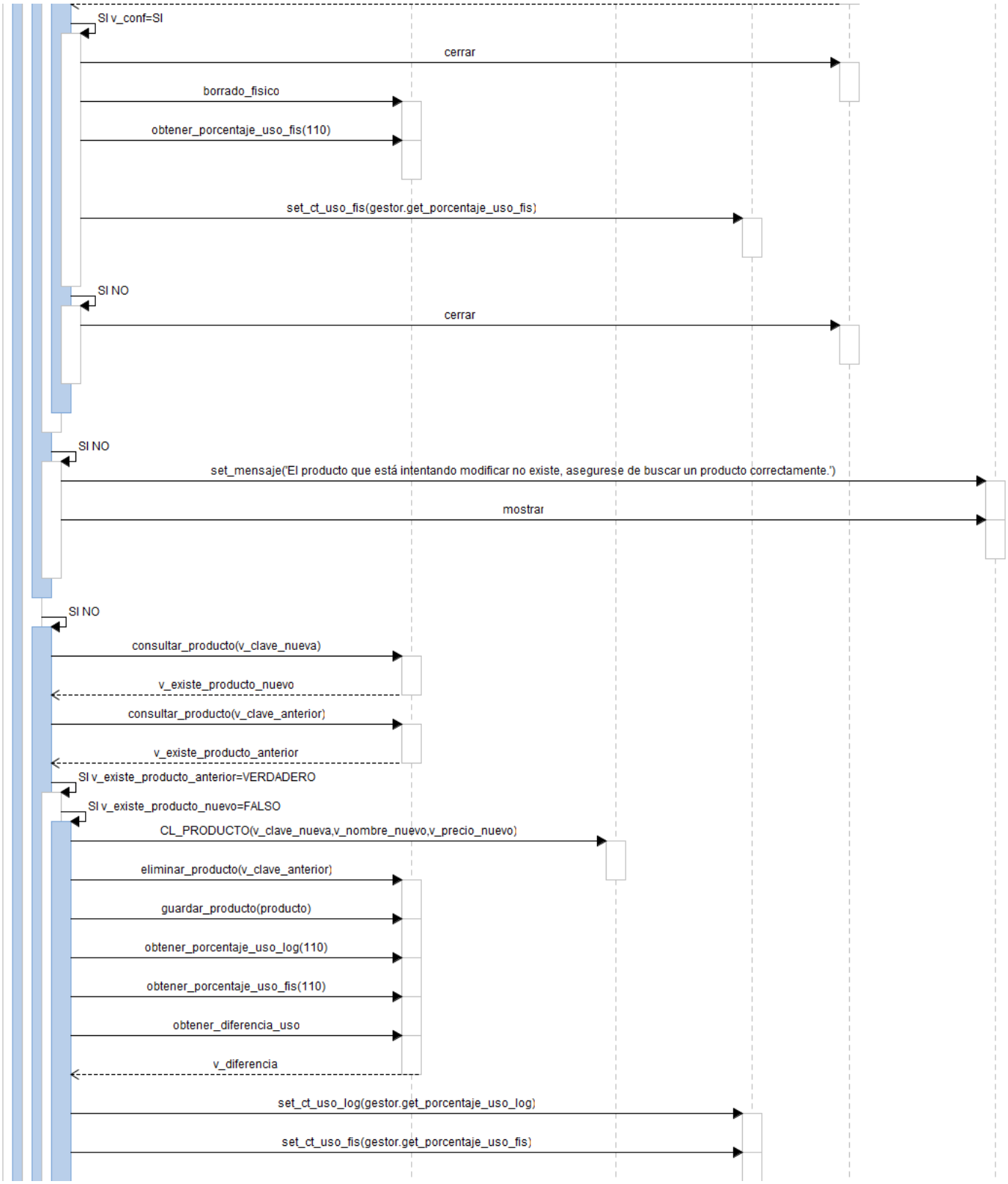

### 7.7. Determinación de eventos de la interfaz IUSR_02

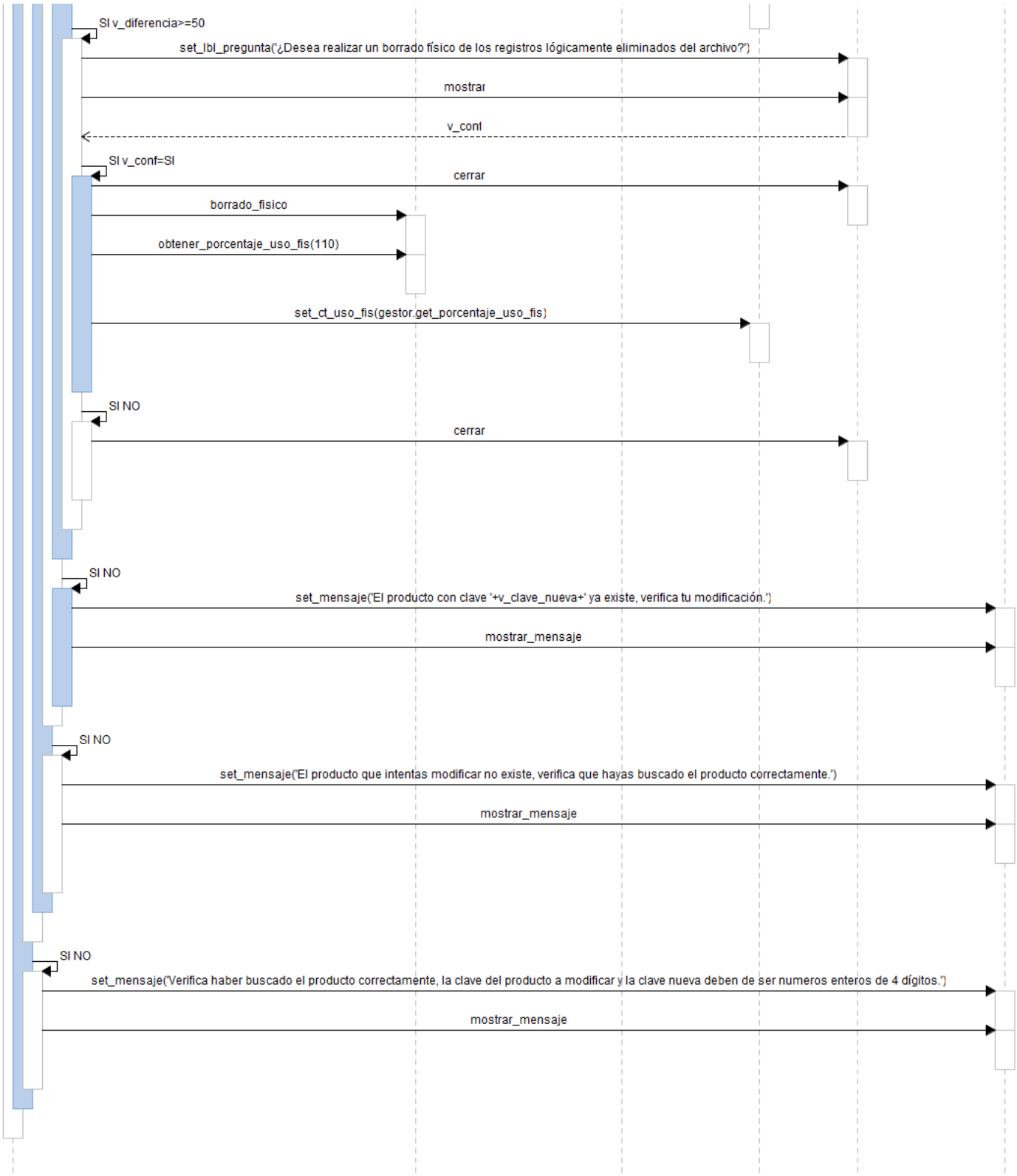
- **evt_agg_producto:** Debe generar las acciones necesarias para mostrar la ventana de confirmación ICONF_01, y si el resultado obtenido es Cancelar, limpiar las cajas de texto, de lo contrario, deberá inicialmente verificar que todas las cajas de texto no sean vacías, así mismo verificar que la clave del producto sea un número entero de 4 dígitos, de no ser así mandar mensajes de error, ya validado todo, construir un objeto de CL_PRODUCTO con el contenido de las cajas de texto y guardarlo en un archivo. Si al guardarlo el método de guardar_producto regresa un falso mostrar un mensaje de que ya existe un producto con esa clave.


EJECUTAR_SISTEMA_VENTAS
producto: CL_PRODUCTO
IUSR_02
gestor: GESTOR_ARCH_PRODUCTO
mensaje: CL_VISTA_MENSAJE
ICONF_01
set_lbl_pregunta("¿Estás seguro que deseas agregar el producto proporcionado al registro?") .
mostrar
v_conf
SI v_conf=SI
cerrar
get_ct_clave_producto
v_clave
get_ct_nombre_producto
v_nombre
get_ct_precio_producto
v_precio
SI NO(v_clave="") Y NO(v_nombre="") Y NO(v_precio="")
SI LONGITUD(v_clave) = 4
CL_PRODUCTO(v_clave,v_nombre,v_precio)
guardar_producto(producto)
v_producto_guardado
SI NO(v_producto_guardado=VERDADERO)
set_mensaje("El producto que intentas guardar ya existe, verifica la clave del producto.")
mostrar_mensaje
SINO
set_mensaje("Producto guardado satisfactoriamente.")
mostrar_mensaje
set_ct_clave_producto(" ")
set_ct_nombre_producto(" ")
set_ct_precio_producto(" ")
SINO
set_mensaje("La clave del producto debe ser un número entero con 4 digitos")
mostrar_mensaje
SINO
set_mensaje("Verifica llenar los campos CLAVE, NOMBRE y PRECIO, para poder agregar un producto.")
mostrar_mensaje
SINO
cerrar
set_ct_clave_producto("")
set_ct_nombre_producto("")
set_ct_precio_producto("")

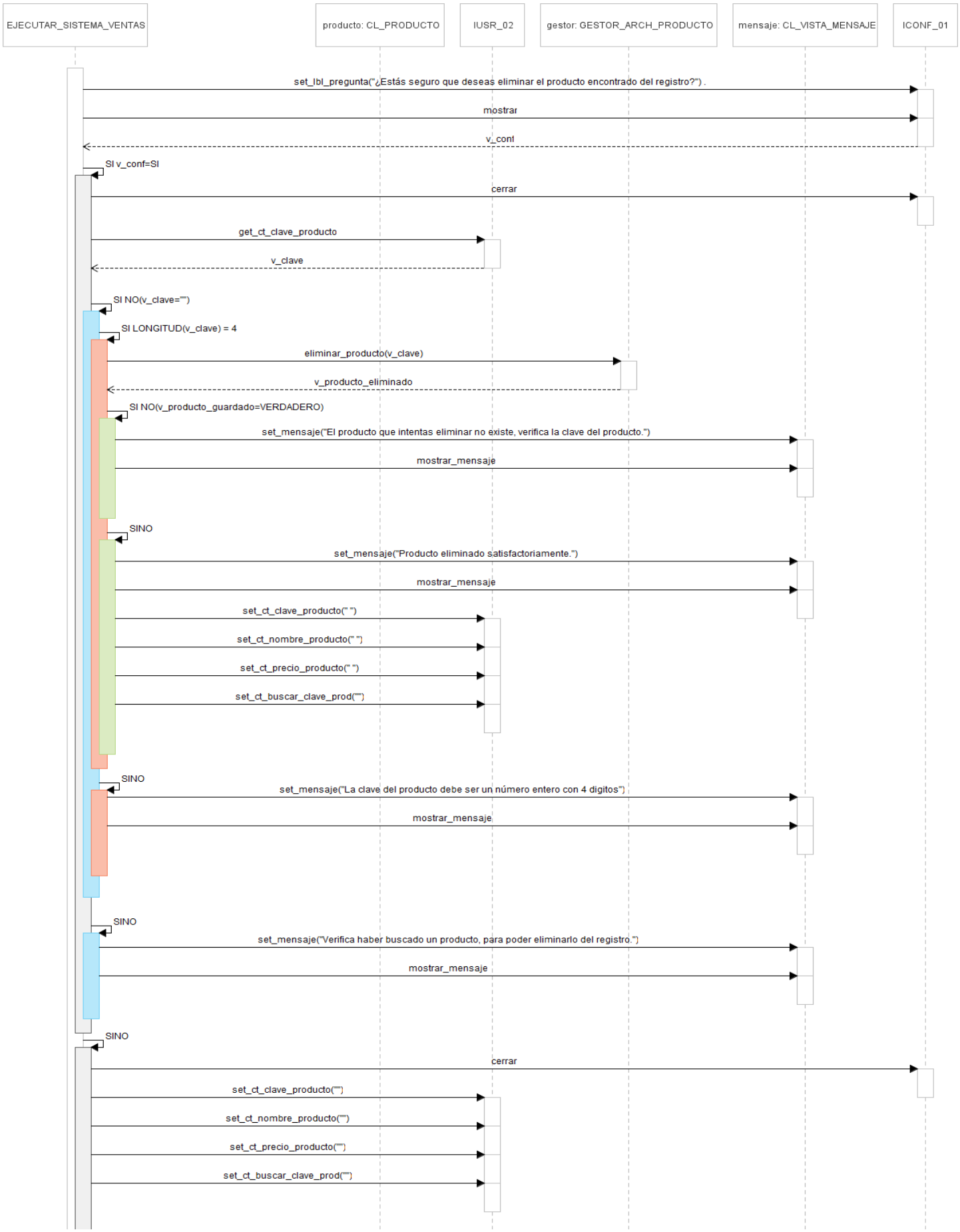
- **evt_modif_producto:** Debe generar las acciones necesarias para mostrar las instrucciones necesarias para modificar un registro, mostrar la ventana de confirmación ICONF_01, si el resultado es Aceptar, verificar el contenido de todas las cajas de texto, y debe permitir modificar el contenido de un registro de producto.


EJECUTAR_SISTEMA_VENTAS
gestor: GESTOR_ARCH_PRODUCTO
producto: CL_PRODUCTO
IUSR_02
ICONF_01
mensaje: CL_VISTA_MENSAJE
set_mensaje(instrucciones)
mostrar_mensaje
mostrar
v_conf
SI v_conf=SI
get_ct_clave_producto
v_clave_nueva
get_ct_nombre_producto
v_nombre_nuevo
get_ct_precio_producto
v_precio_nuevo
get_ct_clave_buscar_prod
v_clave_anterior
SI NO(v_clave_nueva="" Y v_nombre_nuevo="" Y v_precio_nuevo="" Y v_clave_anterior="") Y (LONGITUD(v_clave_nueva)=4 Y LONGITUD(v_clave_anterior)=4)
SI v_clave_nueva=v_clave_anterior
consultar_producto(v_clave_anterior)
v_existe_producto
SI v_existe_producto=VERDADERO
CL_PRODUCTO(v_clave_nueva,v_nombre_nuevo,v_precio_nuevo)
eliminar_producto(v_clave_anterior)
guardar_producto(producto)
obtener_porcentaje_uso_log(110)
obtener_porcentaje_uso_fis(110)
obtener_diferencia_uso
v_diferencia
set_ct_uso_log(gestor.get_porcentaje_uso_log)
set_ct_uso_fis(gestor.get_porcentaje_uso_fis)
SI v_diferencia>= 50
set_lbl_pregunta('¿Desea realizar un borrado físico de los registros lógicamente eliminados del archivo?')
mostrar
v_conf

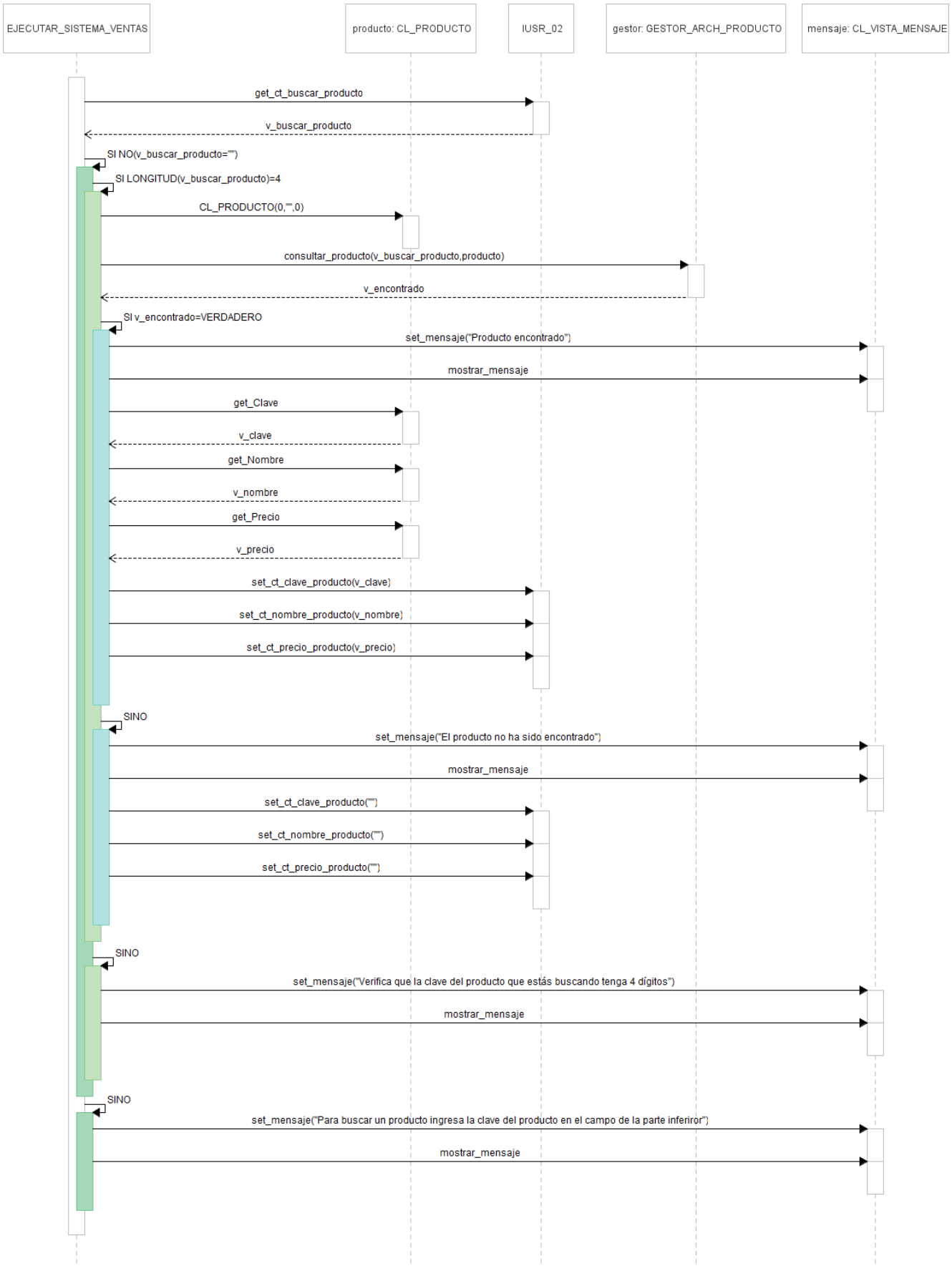

SI v_conf=SI
cerrar
borrado_fisico
obtener_porcentaje_uso_fis(110)
set_ct_uso_fis(gestor.get_porcentaje_uso_fis)
SI NO
cerrar
SI NO
set_mensaje('El producto que está intentando modificar no existe, asegurese de buscar un producto correctamente.')
mostrar
SI NO
consultar_producto(v_clave_nueva)
v_existe_producto_nuevo
consultar_producto(v_clave_anterior)
v_existe_producto_anterior
SI v_existe_producto_anterior=VERDADERO
SI v_existe_producto_nuevo=FALSO
CL_PRODUCTO(v_clave_nueva,v_nombre_nuevo,v_precio_nuevo)
eliminar_producto(v_clave_anterior)
guardar_producto(producto)
obtener_porcentaje_uso_log(110)
obtener_porcentaje_uso_fis(110)
obtener_diferencia_uso
v_diferencia
set_ct_uso_log(gestor.get_porcentaje_uso_log)
set_ct_uso_fis(gestor.get_porcentaje_uso_fis)


SI v_diferencia>=50
set_lbl_pregunta('¿Desea realizar un borrado físico de los registros lógicamente eliminados del archivo?')
mostrar
v_conf
SI v_conf=SI
cerrar
borrado_fisico
obtener_porcentaje_uso_fis(110)
set_ct_uso_fis(gestor.get_porcentaje_uso_fis)
SI NO
cerrar
SI NO
set_mensaje('El producto con clave '+v_clave_nueva+' ya existe, verifica tu modificación.')
mostrar_mensaje
SI NO
set_mensaje('El producto que intentas modificar no existe, verifica que hayas buscado el producto correctamente.')
mostrar_mensaje
SI NO
set_mensaje('Verifica haber buscado el producto correctamente, la clave del producto a modificar y la clave nueva deben de ser numeros enteros de 4 dígitos.')
mostrar_mensaje

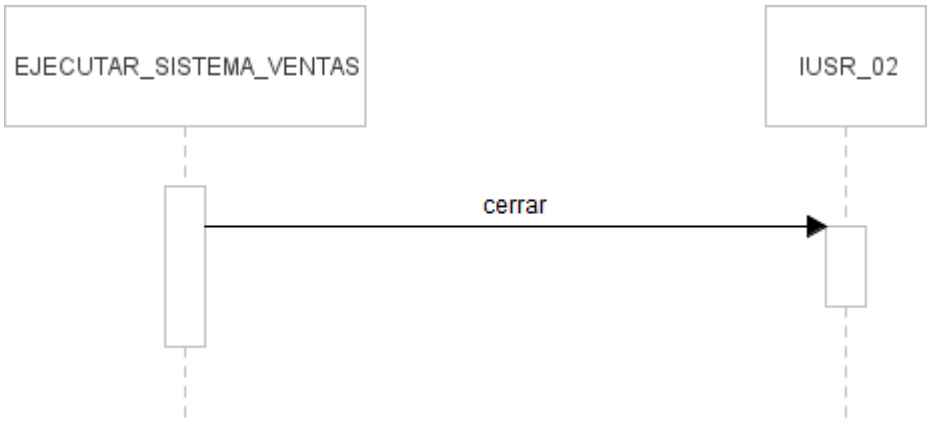
- **evt_elim_producto:** Debe generar las acciones necesarias para mostrar la ventana de confirmación ICONF_01, y si el resultado obtenido es Cancelar, limpiar las cajas de texto, de lo contrario, deberá inicialmente verificar que todas las cajas de texto no sean vacías, así mismo verificar que la clave del producto sea un número entero de 4 dígitos, de no ser así mandar mensajes de error, ya validado todo, llamar al método eliminar_producto de la clase GESTOR_ARCH_PRODUCTO. Si al eliminarlo el método de guardar_producto regresa un falso mostrar un mensaje de que no existe un producto con esa clave.


EJECUTAR_SISTEMA_VENTAS
producto: CL_PRODUCTO
IUSR_02
gestor: GESTOR_ARCH_PRODUCTO
mensaje: CL_VISTA_MENSAJE
ICONF_01
set_lbl_pregunta("¿Estás seguro que deseas eliminar el producto encontrado del registro?") .
mostrar
v_conf
SI v_conf=SI
cerrar
get_ct_clave_producto
v_clave
SI NO(v_clave="")
SI LONGITUD(v_clave) = 4
eliminar_producto(v_clave)
v_producto_eliminado
SI NO(v_producto_guardado=VERDADERO)
set_mensaje("El producto que intentas eliminar no existe, verifica la clave del producto.")
mostrar_mensaje
SINO
set_mensaje("Producto eliminado satisfactoriamente.")
mostrar_mensaje
set_ct_clave_producto(" ")
set_ct_nombre_producto(" ")
set_ct_precio_producto(" ")
set_ct_buscar_clave_prod("")
SINO
set_mensaje("La clave del producto debe ser un número entero con 4 digitos")
mostrar_mensaje
SINO
set_mensaje("Verifica haber buscado un producto, para poder eliminarlo del registro.")
mostrar_mensaje
SINO
cerrar
set_ct_clave_producto("")
set_ct_nombre_producto("")
set_ct_precio_producto("")
set_ct_buscar_clave_prod("")

- **evt_buscar_producto:** Debe generar las acciones necesarias para obtener el contenido de la caja de texto ct_clave_buscar_prod, y verificar que esta contenga un número entero de 4 dígitos, de no ser así mandar mensajes de error. Ya validado construir un objeto de tipo CL_PRODUCTO y ejecutar el método consultar_producto, si el método devuelve falso mandar mensaje de que el producto no existe y vaciar las cajas de texto, si no, obtener los valores del objeto producto pasado por referencia y setearlos en las cajas de texto.


EJECUTAR_SISTEMA_VENTAS
producto: CL_PRODUCTO
IUSR_02
gestor: GESTOR_ARCH_PRODUCTO
mensaje: CL_VISTA_MENSAJE
get_ct_buscar_producto
v_buscar_producto
SI NO(v_buscar_producto="")
SI LONGITUD(v_buscar_producto)=4
CL_PRODUCTO(0,"",0)
consultar_producto(v_buscar_producto,producto)
v_encontrado
SI v_encontrado=VERDADERO
set_mensaje("Producto encontrado")
mostrar_mensaje
get_Clave
v_clave
get_Nombre
v_nombre
get_Precio
v_precio
set_ct_clave_producto(v_clave)
set_ct_nombre_producto(v_nombre)
set_ct_precio_producto(v_precio)
SINO
set_mensaje("El producto no ha sido encontrado")
mostrar_mensaje
set_ct_clave_producto("")
set_ct_nombre_producto("")
set_ct_precio_producto("")
SINO
set_mensaje("Verifica que la clave del producto que estás buscando tenga 4 dígitos")
mostrar_mensaje
SINO
set_mensaje("Para buscar un producto ingresa la clave del producto en el campo de la parte inferior")
mostrar_mensaje

- **evt_cerrar:** Debe generar las acciones necesarias para cerrar la interfaz IUSR_02.




### 7.8. Pseudocódigos de los eventos de IUSR_02

- **Pseudocódigo evt_agg_producto**

```
SECCIÓN DE VARIABLES
      v_conf, v_clave: ENTERO
      v_nombre: CADENA
      v_precio: REAL
      mensaje: CL_VISTA_MENSAJE
      producto: CL_PRODUCTO
      gestor: GESTOR_ARCH_PRODUCTO
INICIO
      mensaje.CL_VISTA_MENSAJE
      gestor.GESTOR_ARCH_PRODUCTO
      ICONF_01.set_lbl_pregunta("¿Estás seguro que deseas agregar el
producto proporcionado al registro?")
      v_conf <-- ICONF_01.mostrar
      SI v_conf=SI ENTONCES
            ICONF_01.cerrar
            v_clave <--
CONVERTIR_A_ENTERO(IUSR_02.get_ct_clave_producto)
            v_nombre <-- IUSR_02.get_ct_nombre_producto
            v_precio <--
CONVERTIR_A_REAL(IUSR_02.get_ct_precio_producto)
            SI NO(v_clave="") Y NO(v_nombre="") Y NO(v_precio="")
ENTONCES
                  SI LONGITUD(CONVERTIR_A_CADENA(v_clave))=4 ENTONCES
                        producto.CL_PRODUCTO(v_clave,v_nombre,v_precio)
                        v_producto_guardado <--
gestor.guardar_producto(producto)
                        SI NO(v_producto_guardado=VERDADERO) ENTONCES
                              mensaje.set_mensaje("El producto que
intentas guardar ya existe, verifica la clave del producto.")
                              mensaje.mostrar_mensaje
                        SINO
                              mensaje.set_mensaje("Producto guardado
satisfactoriamente.")
                              mensaje.mostrar_mensaje
```

```
                              IUSR_02.set_ct_clave_producto("")
                              IUSR_02.set_ct_nombre_producto("")
                              IUSR_02.set_ct_precio_producto("")
                        FIN SI
                  SINO
                        mensaje.set_mensaje("La clave del producto debe
ser un número entero con 4 digitos")
                        mensaje.mostrar_mensaje
                  FIN SI
            SINO
                  mensaje.set_mensaje("Verifica llenar los campos
CLAVE, NOMBRE y PRECIO, para poder agregar un producto.")
                  mensaje.mostrar_mensaje
            FIN SI
      SINO
            ICONF_01.cerrar
            IUSR_02.set_ct_clave_producto(" ")
            IUSR_02.set_ct_nombre_producto(" ")
            IUSR_02.set_ct_precio_producto(" ")
      FIN SI
      mensaje.destructor
FIN
```

- **Pseudocódigo evt_modif_producto**

```
SECCIÓN DE VARIABLES
      gestor: GESTOR_ARCH_PRODUCTO
      producto: CL_PRODUCTO
      mensaje: CL_VISTA_MENSAJE
      v_conf: ENTERO
      v_clave_nueva,v_nombre_nuevo,v_precio_nuevo
      v_clave_anterior,instrucciones: CADENA
      v_existe_producto_nuevo,
      v_existe_producto_anterior: BOOLEANO
      v_diferencia: REAL
INICIO
      mensaje.CL_VISTA_MENSAJE
      gestor.GESTOR_ARCH_PRODUCTO
      instrucciones <-- 'Para poder modificar un registro;'+FinLinea+
      '1.- Busca el producto a modificar'+FinLinea+
      '2.- Realiza los cambios en los 3 campos de la
izquierda'+FinLinea+
      '3.- Da click en el botón "Modificar"'+FinLinea+
      '4.- Da click en "Aceptar"'+FinLinea+
      'Nota: El registro a modificar será el buscado en al campo de la
parte inferior.'
      mensaje.set_mensaje(instrucciones);
      mensaje.mostrar_mensaje;
      v_conf <-- ICONF_01.mostrar
      SI v_conf=SI ENTONCES
            v_clave_nueva <-- IUSR_02.get_ct_clave_producto
            v_nombre_nuevo <-- IUSR_02.get_ct_nombre_producto
            v_precio_nuevo <-- IUSR_02.get_ct_precio_producto
            v_clave_anterior <-- IUSR_02.get_ct_clave_buscar_prod
```

```
           SI NO(v_clave_nueva='' Y v_nombre_nuevo='' Y
v_precio_nuevo='' Y v_clave_anterior='') Y (LONGITUD(v_clave_nueva)=4
Y LONGITUD(v_clave_anterior)=4)
                  SI v_clave_nueva=v_clave_anterior
                        v_existe_producto_anterior <--
gestor.consultar_producto(v_clave_anterior)
                        SI v_existe_producto_anterior = VERDADERO

      producto.CL_PRODUCTO(v_clave_nueva,v_nombre_nuevo,v_precio_nuevo
)
                              gestor.eliminar_producto(v_clave_anterior)
                              gestor.guardar_producto(producto)
                              producto.destructor
                              gestor.obtener_porcentaje_uso_log(110)
                              gestor.obtener_porcentaje_uso_fis(110)
                              v_diferencia <--
gestor.obtener_diferencia_uso

      IUSR_02.set_ct_uso_log(gestor.get_porcentaje_uso_log)

      IUSR_02.set_ct_uso_fis(gestor.get_porcentaje_uso_fis)
                              SI v_diferencia>=50
                                    ICONF_01.set_lbl_pregunta('¿Desea
realizar un borrado físico de los registros lógicamente eliminados del
archivo?')
                                    v_conf <-- ICONF_01.mostrar
                                    SI v_conf=SI
                                          ICONF_01.Cerrar
                                          gestor.borrado_fisico

      gestor.obtener_porcentaje_uso_fis(110)

      IUSR_02.set_ct_uso_fis(gestor.get_porcentaje_uso_fis)
                                    SI NO
                                          ICONF_01.Cerrar
                                    FIN (SI)
                              FIN (SI)
                        SI NO
                              mensaje.set_mensaje('El producto que está
intentando modificar no existe, asegurese de buscar un producto
correctamente.')
                              mensaje.mostrar_mensaje
                        FIN (SI)
                  SI NO
                        v_existe_producto_nuevo <--
gestor.consultar_producto(v_clave_nueva)
                        v_existe_producto_anterior <--
gestor.consultar_producto(v_clave_anterior)
                        SI v_existe_producto_anterior=VERDADERO
                              SI v_existe_producto_nuevo=FALSO

      producto.CL_PRODUCTO(v_clave_nueva,v_nombre_nuevo,v_precio_nuevo
)
```

```
      gestor.eliminar_producto(v_clave_anterior)
                                    gestor.guardar_producto(producto)
                                    producto.destructor

      gestor.obtener_porcentaje_uso_log(110)

      gestor.obtener_porcentaje_uso_fis(110)
                                    v_diferencia <--
gestor.obtener_diferencia_uso

      IUSR_02.set_ct_uso_log(gestor.get_porcentaje_uso_log)

      IUSR_02.set_ct_uso_fis(gestor.get_porcentaje_uso_fis)
                                    SI v_diferencia>=50

      ICONF_01.set_lbl_pregunta('¿Desea realizar un borrado físico de
los registros lógicamente eliminados del archivo?')
                                          v_conf <-- ICONF_01.mostrar
                                          SI v_conf=SI
                                                ICONF_01.Cerrar
                                                gestor.borrado_fisico

      gestor.obtener_porcentaje_uso_fis(110)

      IUSR_02.set_ct_uso_fis(gestor.get_porcentaje_uso_fis)
                                          SI NO
                                                ICONF_01.Cerrar
                                          FIN (SI)
                                    FIN (SI)
                               SI NO
                                    mensaje.set_mensaje('El producto con
clave '+v_clave_nueva+' ya existe, verifica tu modificación.');
                                    mensaje.mostrar_mensaje;
                               FIN (SI)
                         SI NO
                               mensaje.set_mensaje('El producto que
intentas modificar no existe, verifica que hayas buscado el producto
correctamente.');
                               mensaje.mostrar_mensaje
                         FIN (SI)
                  FIN (SI)
            SI NO
                  mensaje.set_mensaje('Verifica haber buscado el
producto correctamente, la clave del producto a modificar y la clave
nueva deben de ser numeros enteros de 4 dígitos.')
                  mensaje.mostrar_mensaje
            FIN (SI)
      SI NO
            ICONF_01.Cerrar
      FIN (SI)
      mensaje.destructor
      gestor.destructor
FIN
```

- **Pseudocódigo evt_elim_producto**

```
SECCIÓN DE VARIABLES
      v_conf, v_clave: ENTERO
      mensaje: CL_VISTA_MENSAJE
      gestor: GESTOR_ARCH_PRODUCTO
      v_producto_eliminado: BOOLEANO
INICIO
      mensaje.CL_VISTA_MENSAJE
      gestor.GESTOR_ARCH_PRODUCTO
      ICONF_01.set_lbl_pregunta("¿Estás seguro que deseas eliminar el
producto encontrado del registro?")
      v_conf <-- ICONF_01.mostrar
      SI v_conf=SI ENTONCES
            ICONF_01.cerrar
            v_clave <--
CONVERTIR_A_ENTERO(IUSR_02.get_ct_clave_producto)
            SI NO(v_clave="") ENTONCES
                  SI LONGITUD(CONVERTIR_A_CADENA(v_clave))=4 ENTONCES
                        v_producto_eliminado <--
gestor.eliminar_producto(v_clave)
                        SI NO(v_producto_eliminado=VERDADERO) ENTONCES
                              mensaje.set_mensaje("El producto que
intentas eliminar no existe, verifica la clave del producto.")
                              mensaje.mostrar_mensaje
                        SINO
                              mensaje.set_mensaje("Producto eliminado
satisfactoriamente.")
                              mensaje.mostrar_mensaje
                              IUSR_02.set_ct_clave_producto("")
                              IUSR_02.set_ct_nombre_producto("")
                              IUSR_02.set_ct_precio_producto("")
                              IUSR_02.set_ct_clave_buscar_prod("")
                        FIN SI
                  SINO
                        mensaje.set_mensaje("La clave del producto debe
ser un número entero con 4 digitos")
                        mensaje.mostrar_mensaje
                  FIN SI
            SINO
                  mensaje.set_mensaje("Verifica haber buscado un
producto, para poder eliminarlo del registro.")
                  mensaje.mostrar_mensaje
            FIN SI
      SINO
            ICONF_01.cerrar
            IUSR_02.set_ct_clave_producto(" ")
            IUSR_02.set_ct_nombre_producto(" ")
            IUSR_02.set_ct_precio_producto(" ")
            IUSR_02.set_ct_clave_buscar_prod("")
      FIN SI
      mensaje.destructor
      gestor.destructor
```

```
FIN
```

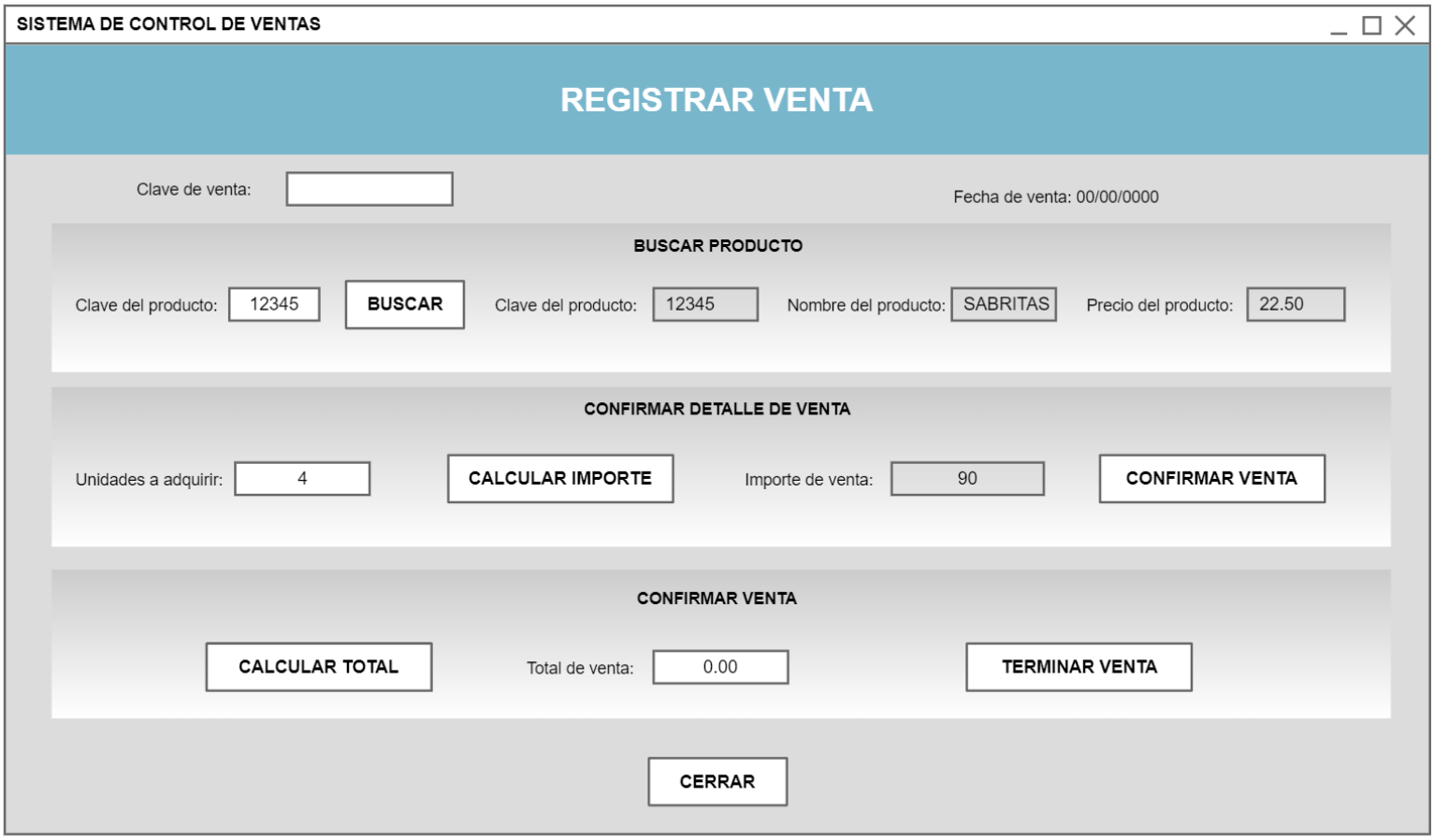
- **Pseudocódigo evt_buscar_producto**

```
SECCIÓN DE VARIABLES
      v_buscar_producto:CADENA
      mensaje: CL_VISTA_MENSAJE
      gestor: GESTOR_ARCH_PRODUCTO
      producto: CL_PRODUCTO
      v_encontrado: Booleano
INICIO
      mensaje.CL_VISTA_MENSAJE
      gestor.GESTOR_ARCH_PRODUCTO
      v_buscar_producto <-- IUSR_02.get_ct_buscar_producto
      SI NO(v_buscar_producto='') ENTONCES
            SI LONGITUD(v_buscar_producto)=4 ENTONCES
                  producto.CL_PRODUCTO(0,'',0)
                  v_encontrado <--
gestor.consultar_producto(v_buscar_producto,producto);
                  SI v_encontrado=VERDADERO ENTONCES
                        mensaje.set_mensaje('Producto encontrado.');
                        mensaje.mostrar_mensaje;
                        v_clave <--
CONVERTIR_A_CADENA(producto.get_Clave);
                        v_nombre <-- producto.get_Nombre;
                        v_precio <--
CONVERTIR_A_CADENA(producto.get_Precio);
                        IUSR_02.set_ct_clave_producto(v_clave)
                        IUSR_02.set_ct_nombre_producto(v_nombre)
                        IUSR_02.set_ct_precio_producto(v_precio)
                        producto.destructor
                  SINO
                        mensaje.set_mensaje('El producto no ha sido
encontrado');
                        mensaje.mostrar_mensaje;
                        IUSR_02.set_ct_clave_producto("")
                        IUSR_02.set_ct_nombre_producto("")
                        IUSR_02.set_ct_precio_producto("")
                  FIN SI
            SINO
                  mensaje.set_mensaje('Verifica que la clave del
producto que estás buscando tenga 4 dígitos');
                  mensaje.mostrar_mensaje;
            FIN SI
      SINO
            mensaje.set_mensaje('Para buscar un producto ingresa la
clave del producto en el campo de la parte inferiror');
            mensaje.mostrar_mensaje;
      FIN SI
      mensaje.destructor
      gestor.destructor
FIN
```

- **Pseudocódigo evt_cerrar**

```
INICIO
      IUSR_02.cerrar
FIN
```

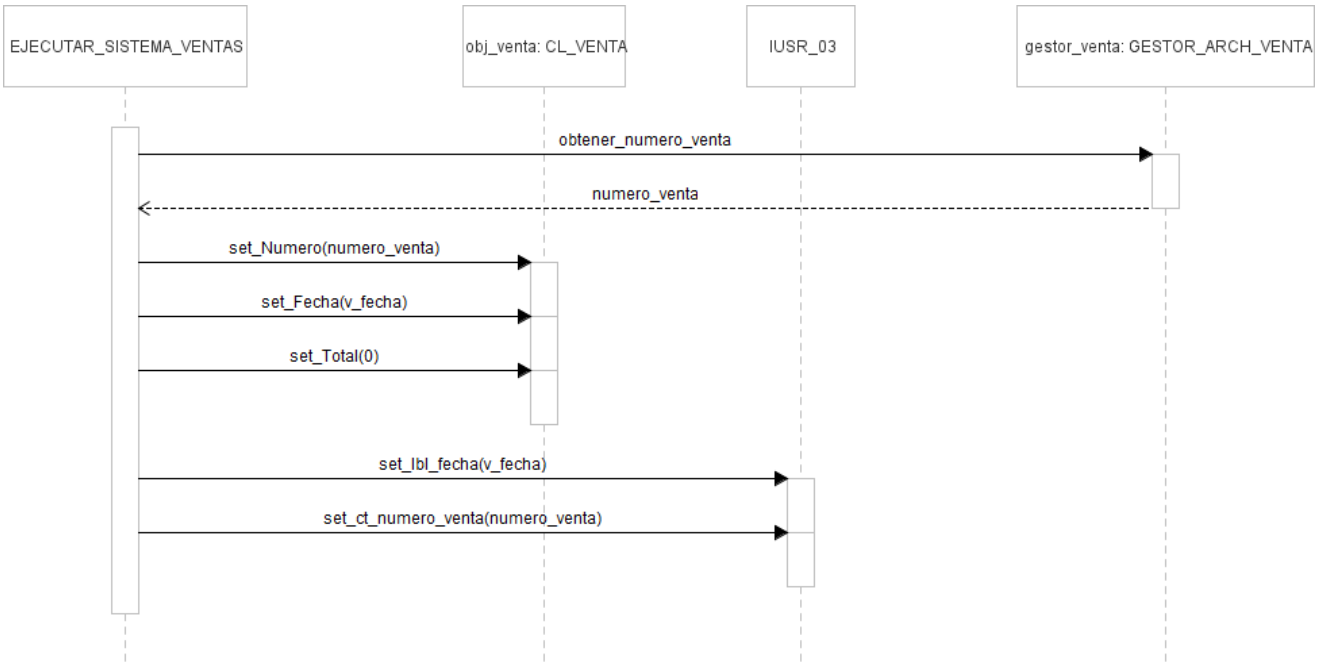
### 7.9. Diseño de la clase IUSR_03



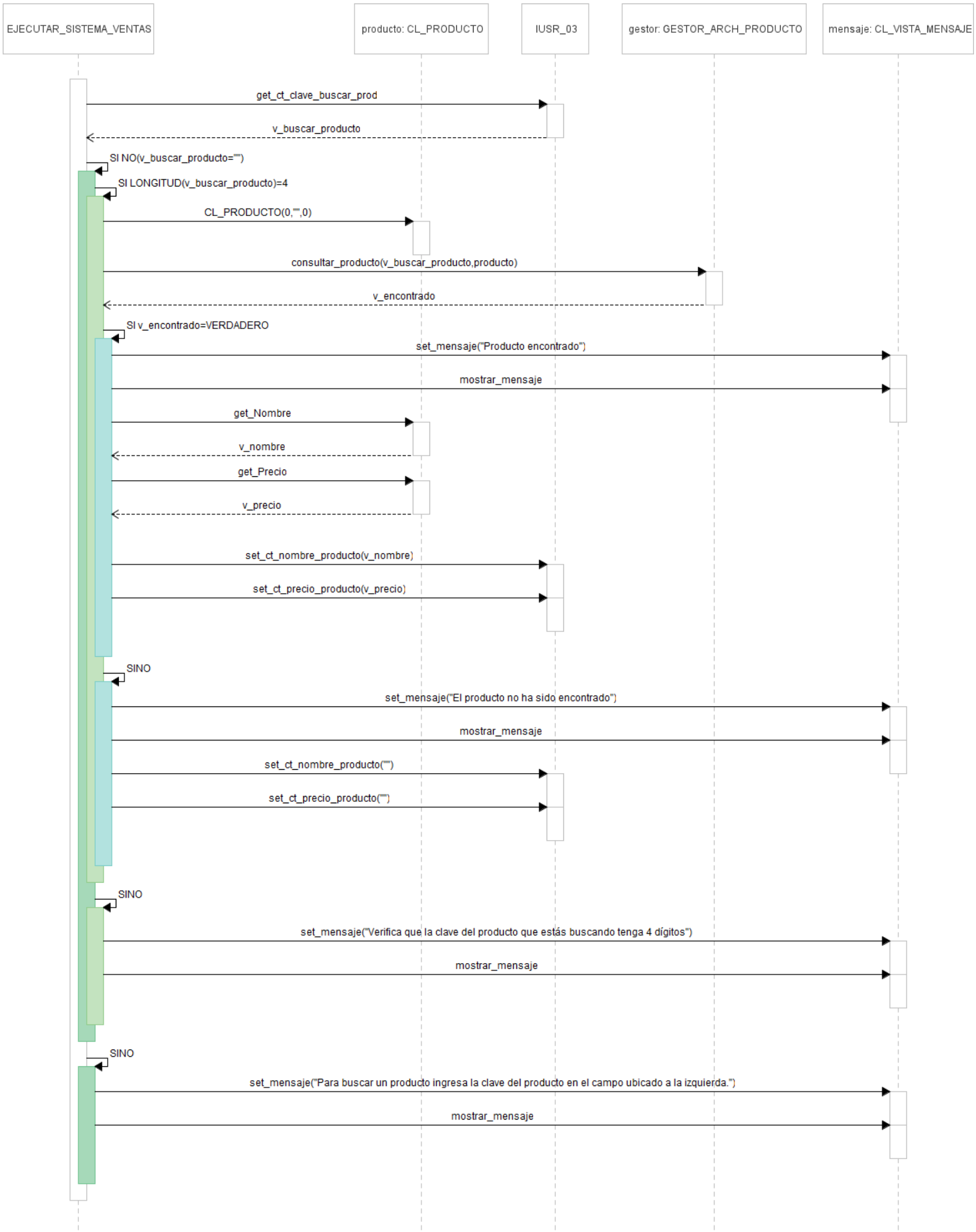
### 7.10. Modelado de la clase CL_IUSR_03


### 7.11. Determinación de eventos de la interfaz IUSR_03

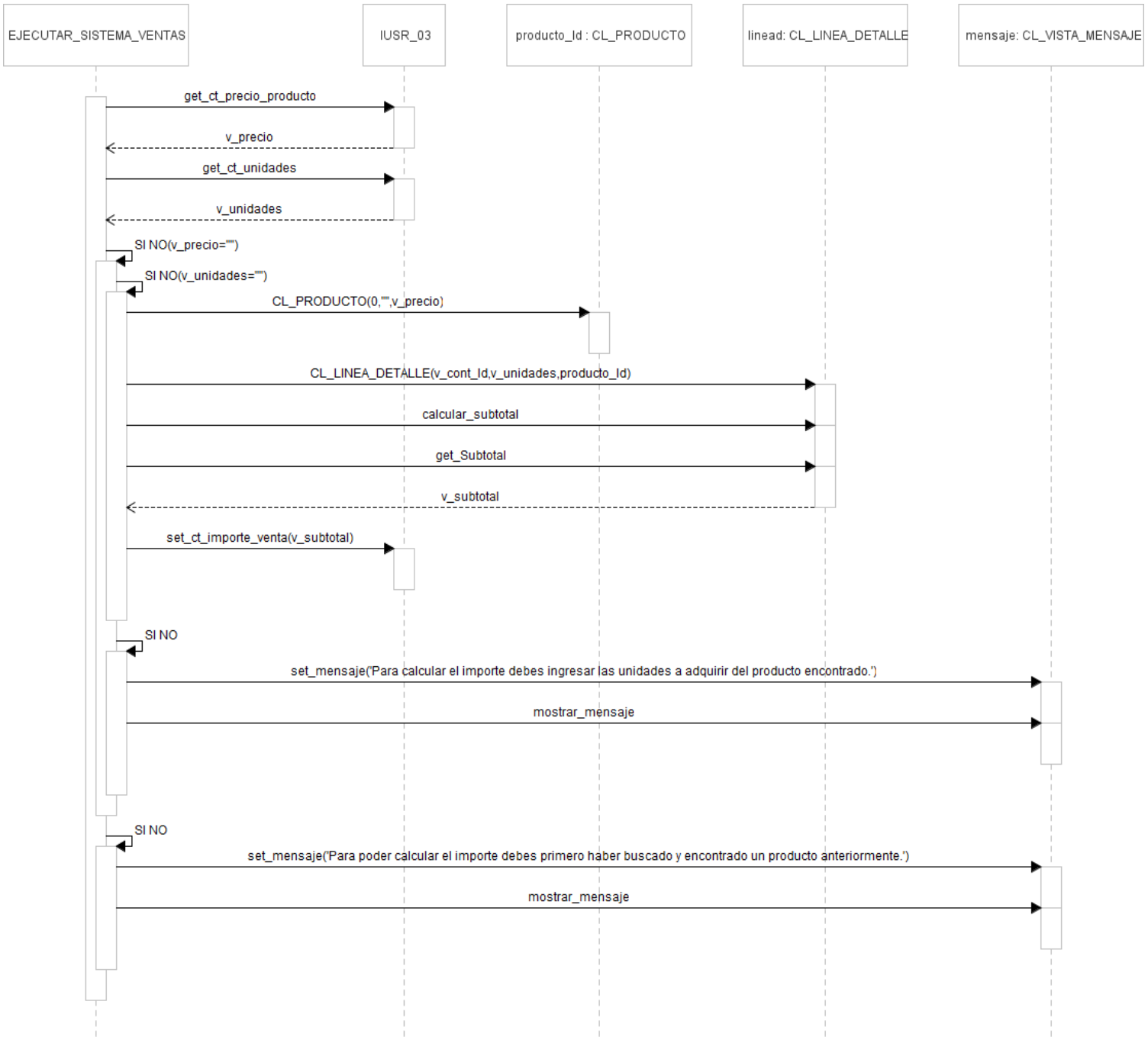
- **evt_iniciar_venta:** Debe generar las acciones necesarias para que al iniciar un proceso de venta se instancie un objeto de CL_VENTA inicie en 0 el contador de las lineas de detalle, se obtenga la fecha del sistema y se muestre en la interfaz IUSR_03.



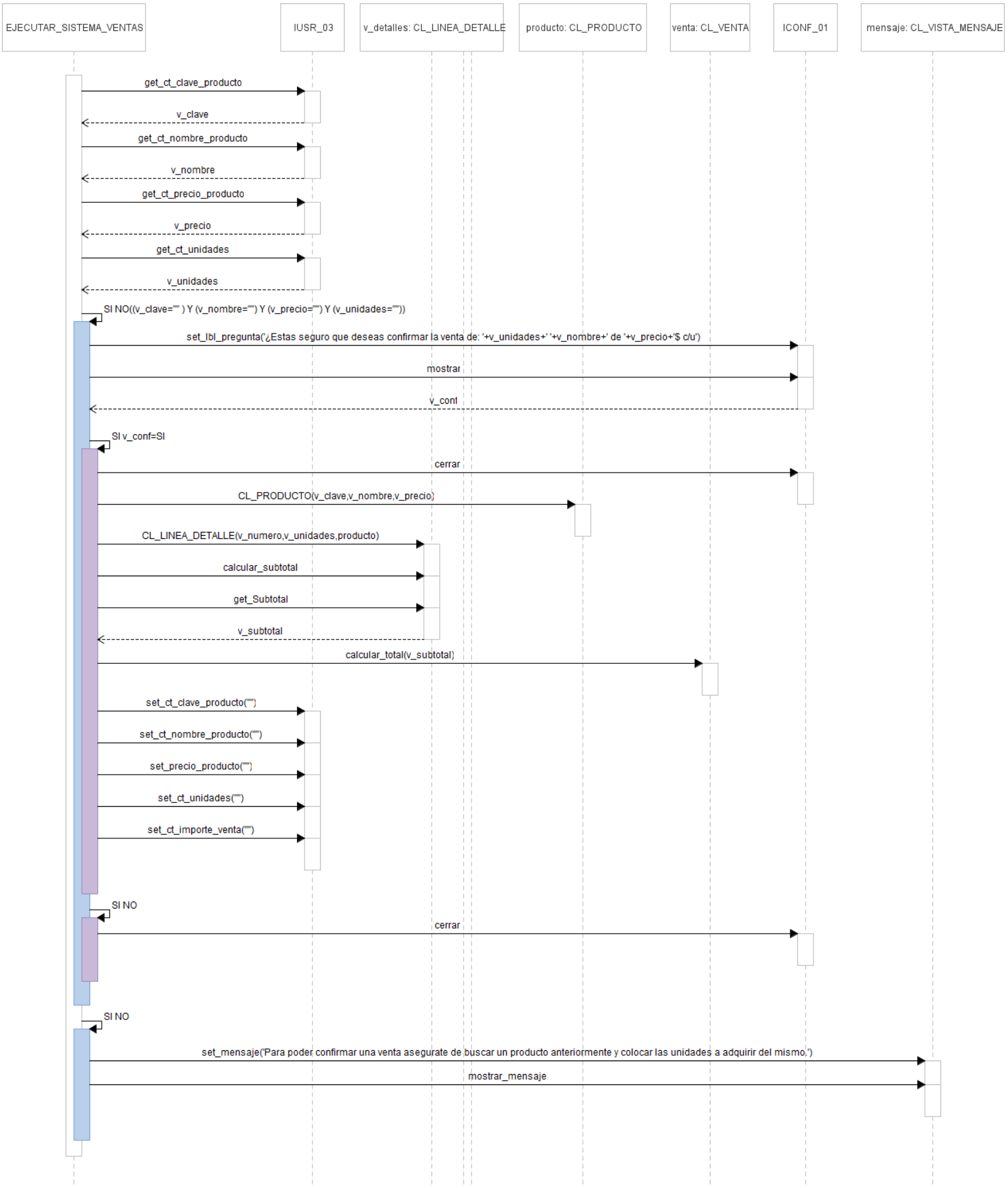

- **evt_buscar_producto:** Debe generar las acciones necesarias para obtener el contenido de la caja de texto ct_clave_buscar_prod, y verificar que esta contenga un número entero de 4 dígitos, de no ser así mandar mensajes de error. Ya validado construir un objeto de tipo CL_PRODUCTO y ejecutar el método consultar_producto, si el método devuelve falso mandar mensaje de que el producto no existe y vaciar las cajas de texto, si no, obtener los valores del objeto producto pasado por referencia y setearlos en las cajas de texto.


EJECUTAR_SISTEMA_VENTAS
producto: CL_PRODUCTO
IUSR_03
gestor: GESTOR_ARCH_PRODUCTO
mensaje: CL_VISTA_MENSAJE
get_ct_clave_buscar_prod
v_buscar_producto
SI NO(v_buscar_producto="")
SI LONGITUD(v_buscar_producto)=4
CL_PRODUCTO(0,"",0)
consultar_producto(v_buscar_producto,producto)
v_encontrado
SI v_encontrado=VERDADERO
set_mensaje("Producto encontrado")
mostrar_mensaje
get_Nombre
v_nombre
get_Precio
v_precio
set_ct_nombre_producto(v_nombre)
set_ct_precio_producto(v_precio)
SINO
set_mensaje("El producto no ha sido encontrado")
mostrar_mensaje
set_ct_nombre_producto("")
set_ct_precio_producto("")
SINO
set_mensaje("Verifica que la clave del producto que estás buscando tenga 4 dígitos")
mostrar_mensaje
SINO
set_mensaje("Para buscar un producto ingresa la clave del producto en el campo ubicado a la izquierda.")
mostrar_mensaje

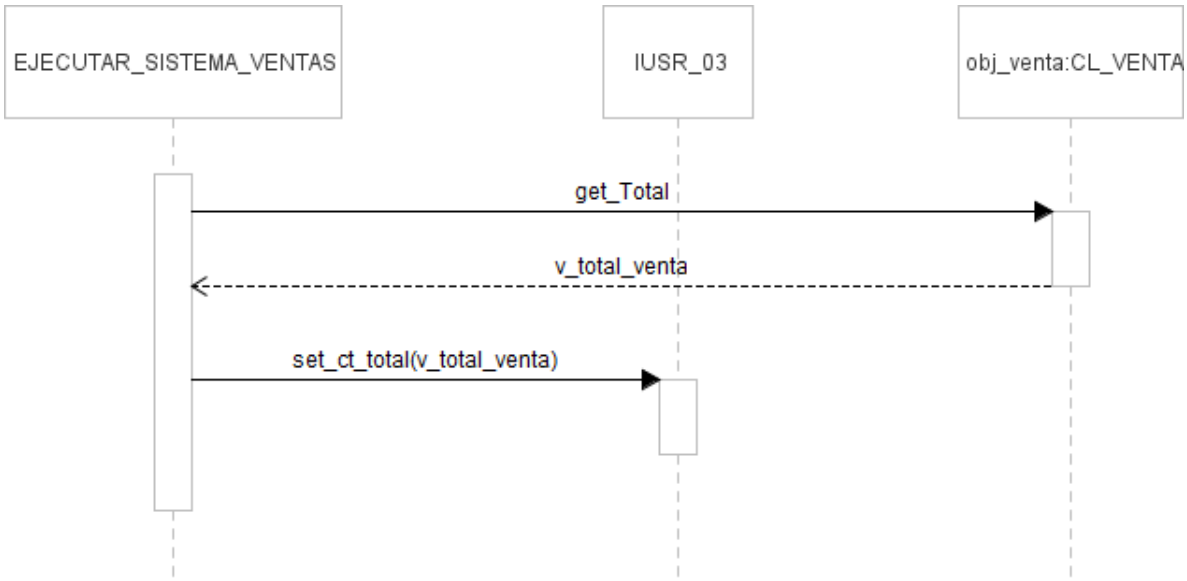
- **evt_calcular_importe:** Debe generar las acciones necesarias para que se calcule el subtotal de multiplicar las unidades adquiridas por el precio del producto seleccionado.



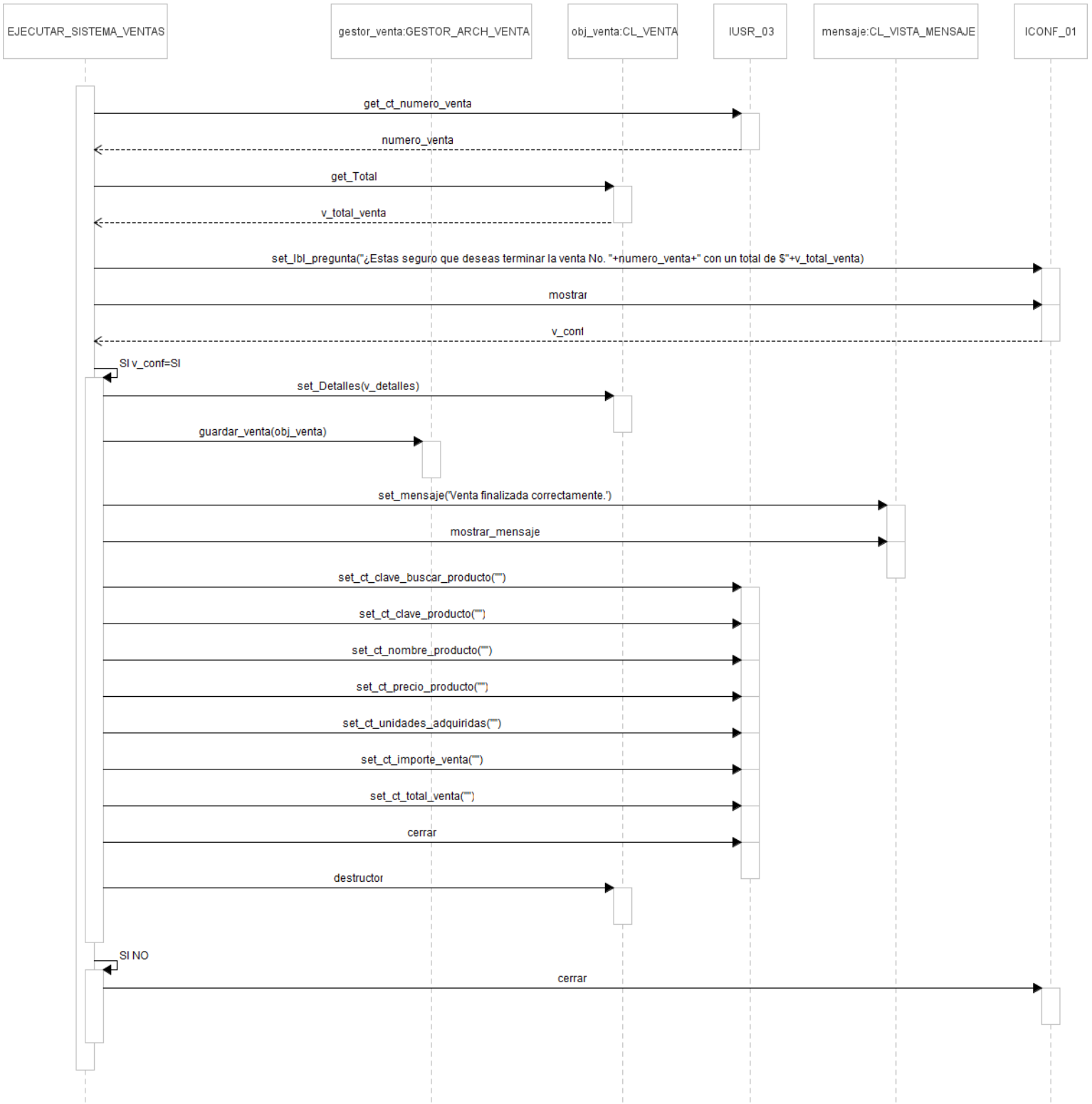
- **evt_confirmar_venta:** Debe generar las acciones necesarias para abrir las interfaz de confirmación ICONF_01 y preguntar si en verdad el usuario desea confirmar la venta, si la respuesta es SI, deberá verificar que contenga algo las cajas de texto del producto buscado, y que contenga algo la caja de texto ct_unidades, si es así, deberá instanciarse un objeto de CL_LINEA_DETALLE y este objeto debe ser ingresado al arreglo v_detalles, y se deberán limpiar las cajas de texto dentro del panel de la búsqueda de un producto y las cajas de texto ct_unidades y ct_importe_venta. Al mismo tiempo, se van a ir sumando los subtotales de cada línea de detalle en una variable v_total_venta.


EJECUTAR_SISTEMA_VENTAS
IUSR_03
v_detalles: CL_LINEA_DETALLE
producto: CL_PRODUCTO
venta: CL_VENTA
ICONF_01
mensaje: CL_VISTA_MENSAJE
get_ct_clave_producto
v_clave
get_ct_nombre_producto
v_nombre
get_ct_precio_producto
v_precio
get_ct_unidades
v_unidades
SI NO((v_clave="" ) Y (v_nombre="") Y (v_precio="") Y (v_unidades=""))
set_lbl_pregunta('¿Estas seguro que deseas confirmar la venta de: '+v_unidades+' '+v_nombre+' de '+v_precio+'$ c/u')
mostrar
v_conf
SI v_conf=SI
cerrar
CL_PRODUCTO(v_clave,v_nombre,v_precio)
CL_LINEA_DETALLE(v_numero,v_unidades,producto)
calcular_subtotal
get_Subtotal
v_subtotal
calcular_total(v_subtotal)
set_ct_clave_producto("")
set_ct_nombre_producto("")
set_precio_producto("")
set_ct_unidades("")
set_ct_importe_venta("")
SI NO
cerrar
SI NO
set_mensaje('Para poder confirmar una venta asegurate de buscar un producto anteriormente y colocar las unidades a adquirir del mismo.')
mostrar_mensaje

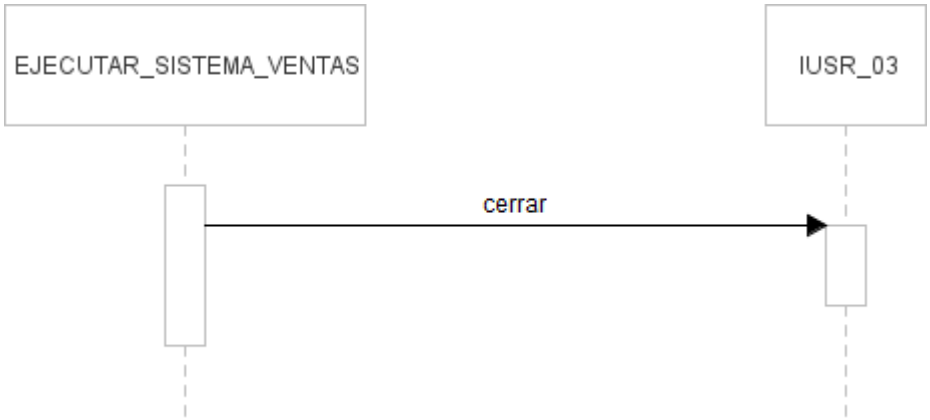
- **evt_calcular_total:** Debe generar las acciones necesarias para setear v_total_venta en ct_total_venta.




- **evt_terminar_venta:** Debe generar las acciones necesarias para mostrar la interfaz ICONF_01, y preguntarle al usuario si desea terminar la venta informando el total acumulado, si la respuesta es SI, se deberá setear el arreglo v_detalles al objeto obj_venta instanciada en el evento evt_iniciar_venta, y se deberá llamar el método guardar_venta de GESTOR_ARCH_VENTA.


EJECUTAR_SISTEMA_VENTAS
gestor_venta:GESTOR_ARCH_VENTA
obj_venta:CL_VENTA
IUSR_03
mensaje:CL_VISTA_MENSAJE
ICONF_01
get_ct_numero_venta
numero_venta
get_Total
v_total_venta
set_lbl_pregunta("¿Estas seguro que deseas terminar la venta No. "+numero_venta+" con un total de $"+v_total_venta)
mostrar
v_conf
SI v_conf=SI
set_Detalles(v_detalles)
guardar_venta(obj_venta)
set_mensaje('Venta finalizada correctamente.')
mostrar_mensaje
set_ct_clave_buscar_producto("")
set_ct_clave_producto("")
set_ct_nombre_producto("")
set_ct_precio_producto("")
set_ct_unidades_adquiridas("")
set_ct_importe_venta("")
set_ct_total_venta("")
cerrar
destructor
SI NO
cerrar

- **evt_cerrar:** Debe generar las acciones necesarias para que se muestre un mensaje de advertencia de que si no ha terminado la venta, al cerrar esta interfaz se perderán los datos de la venta y la misma no será guardada y si la respuesta es Sin cerrar la interfaz IUSR_03.




### 7.12. Pseudocódigos de los eventos de IUSR_03

- **Pseudocódigo de evt_iniciar_venta**

```
SECCIÓN DE VARIABLES
      gestor_venta: GESTOR_ARCH_VENTA
      v_fecha: FECHA
      numero_venta: ENTERO
INICIO
      gestor.GESTOR_ARCH_VENTA
      numero_venta <-- gestor_venta.obtener_numero_venta
      numero_venta <-- numero_venta+1
      v_fecha <-- FECHA_SISTEMA
      obj_venta.CL_VENTA
      obj_venta.set_Numero(numero_venta)
      obj_venta.set_Fecha(CONVERTIR_A_CADENA(v_fecha))
      obj_venta.set_Total(0)
      IUSR_03.set_lbl_fecha(CONVERTIR_A_CADENA(v_fecha))
      IUSR_03.set_ct_numero_venta(CONVERTIR_A_CADENA(numero_venta))
      numero_linea_detalle <-- 0
FIN
```

- **Pseudocódigo de evt_buscar_producto:**

```
SECCIÓN DE VARIABLES
      v_buscar_producto,v_clave,v_nombre,v_precio:CADENA
      mensaje: CL_VISTA_MENSAJE
      gestor: GESTOR_ARCH_PRODUCTO
      producto: CL_PRODUCTO
      v_encontrado: Booleano
INICIO
      mensaje.CL_VISTA_MENSAJE
```

```
      gestor.GESTOR_ARCH_PRODUCTO
      v_buscar_producto <-- IUSR_03.get_ct_clave_buscar_prod
      SI NO(v_buscar_producto='') ENTONCES
            SI LONGITUD(v_buscar_producto)=4 ENTONCES
                  producto.CL_PRODUCTO(0,'',0)
                  v_encontrado <--
gestor.consultar_producto(v_buscar_producto,producto);
                  SI v_encontrado=VERDADERO ENTONCES
                        mensaje.set_mensaje('Producto encontrado.');
                        mensaje.mostrar_mensaje;
                        v_nombre <-- producto.get_Nombre;
                        v_precio <--
CONVERTIR_A_CADENA(producto.get_Precio);
                        IUSR_03.set_ct_nombre_producto(v_nombre)
                        IUSR_03.set_ct_precio_producto(v_precio)
                        producto.destructor
                  SINO
                        mensaje.set_mensaje('El producto no ha sido
encontrado');
                        mensaje.mostrar_mensaje;
                        IUSR_02.set_ct_nombre_producto("")
                        IUSR_02.set_ct_precio_producto("")
                  FIN SI
            SINO
                  mensaje.set_mensaje('Verifica que la clave del
producto que estás buscando tenga 4 dígitos');
                  mensaje.mostrar_mensaje;
            FIN SI
      SINO
            mensaje.set_mensaje('Para buscar un producto ingresa la
clave del producto en el campo ubicado a la izquierda');
            mensaje.mostrar_mensaje;
      FIN SI
      mensaje.destructor
      gestor.destructor
FIN
```

- **Pseudocódigo de evt_calcular_importe:**

```
SECCIÓN DE VARIABLES
      producto_ld: CL_PRODUCTO;
      linead: CL_LINEA_DETALLE;
      mensaje: CL_VISTA_MENSAJE;
      v_precio,v_unidades: CADENA;
      v_subtotal: Real;
INICIO
      mensaje.CL_VISTA_MENSAJE
      v_precio <-- IUSR_03.get_ct_precio_producto
      v_unidades <-- IUSR_03.get_ct_unidades
      SI NO(v_precio='')
            SI NO(v_unidades='')
                  producto_ld.CL_PRODUCTO(0,'',v_precio)
                  linead.CL_LINEA_DETALLE(0,v_unidades,producto_ld)
                  linead.calcular_subtotal
```

```
                v_subtotal <-- linead.get_Subtotal
                IUSR_03.set_ct_importe_venta(v_subtotal)
                producto_ld.destructor
                linead.destructor
          SI NO
                mensaje.set_mensaje('Para calcular el importe debes
ingresar las unidades a adquirir del producto encontrado.');
                mensaje.mostrar_mensaje;
          FIN (SI)
     SI NO
          mensaje.set_mensaje('Para poder calcular el importe debes
primero haber buscado y encontrado un producto anteriormente.');
          mensaje.mostrar_mensaje;
     FIN (SI)
     mensaje.destructor
FIN
```

- **Pseudocódigo de evt_confirmar_venta:**

```
SECCIÓN DE VARIABLES
     v_detalle: CL_LINEA_DETALLE
     producto: CL_PRODUCTO
     mensaje: CL_VISTA_MENSAJE
     v_clave, v_nombre, v_precio, v_unidades: CADENA
     v_conf: ENTERO
     v_subtotal: REAL
INICIO
     mensaje.CL_VISTA_MENSAJE
     v_clave <-- IUSR_03.get_ct_clave_producto
     v_nombre <-- IUSR_03.get_ct_nombre_producto
     v_precio <-- IUSR_03.get_ct_precio_producto
     v_unidades <-- IUSR_03.get_ct_unidades
     SI
(NO((v_clave="")Y(v_nombre="")Y(v_precio="")Y(v_unidades="")))
ENTONCES
          ICONF_01.set_lbl_pregunta("Estas seguro que deseas
confirmar la venta de:"
          + v_unidades+" "+v_nombre+" de "+v_precio+"$ c/u")
          v_conf <-- ICONF_01.mostrar
          SI v_conf=SI
                numero_linea_detalle <-- numero_linea_detalle+1
                ICONF_01.cerrar
                producto.CL_PRODUCTO(v_clave,v_nombre,v_precio)
                v_detalle <--
CL_LINEA_DETALLE(numero_linea_detalle,v_unidades,producto)
                v_detalle.calcular_subtotal
                v_Detalles[numero_linea_detalle] <-- v_detalle
                v_subtotal <-- v_detalle.get_Subtotal
                obj_venta.calcular_total(v_subtotal)
                IUSR_03.set_ct_clave_producto("")
                IUSR_03.set_ct_nombre_producto("")
                IUSR_03.set_ct_precio_producto("")
                IUSR_03.set_ct_unidades("")
          SI NO
```

```
                ICONF_01.cerrar
            FIN (SI)
      SI NO
            mensaje.set_mensaje("Para poder confirmar una venta
asegurate de buscar un producto
            anteriormente y colocar las unidades a adquirir del
mismo.")
            mensaje.mostrar_mensaje
      FIN (SI)
      mensaje.destructor
FIN
```

- **Pseudocódigo de evt_calcular_total:**

```
SECCIÓN DE VARIABLES
      v_total_venta:Real
INICIO
      v_total_venta <-- obj_venta.get_Total
      IUSR_03.set_ct_total(v_total_venta)
FIN
```

- **Pseudocódigo de evt_terminar_venta:**

```
SECCIÓN DE VARIABLES
      gestor_venta: GESTOR_ARCH_VENTA
      mensaje: CL_VISTA_MENSAJE
      numero_venta, v_conf: ENTERO
      v_total_venta: REAL
INICIO
      gestor_venta.GESTOR_ARCH_VENTA
      mensaje.CL_VISTA_MENSAJE
      numero_venta <-- IUSR_03.get_ct_numero_venta
      v_total_venta <-- obj_venta.get_Total
      ICONF_01.set_lbl_pregunta('Estas seguro que deseas terminar la
venta No. '+numero_venta+' con total de $'+v_total_venta)
      v_conf <-- ICONF_01.mostrar
      SI v_conf=SI ENTONCES
            ICONF_01.cerrar
            obj_venta.set_Detalles(v_detalles)
            gestor_venta.guardar_venta(obj_venta)
            mensaje.set_mensaje("Venta finalizada correctamente")
            mensaje.mostrar_mensaje
            IUSR_03.set_ct_clave_buscar_producto("")
            IUSR_03.set_ct_clave_producto("")
            IUSR_03.set_ct_nombre_producto("")
            IUSR_03.set_ct_precio_producto("")
            IUSR_03.set_ct_unidades_adquiridas("")
            IUSR_03.set_ct_importe_venta("")
            IUSR_03.set_ct_total_venta("")
            IUSR_03.cerrar
            gestor_venta.destructor
            obj_venta.destructor
            PARA i DESDE 1 HASTA 10 HACER
```

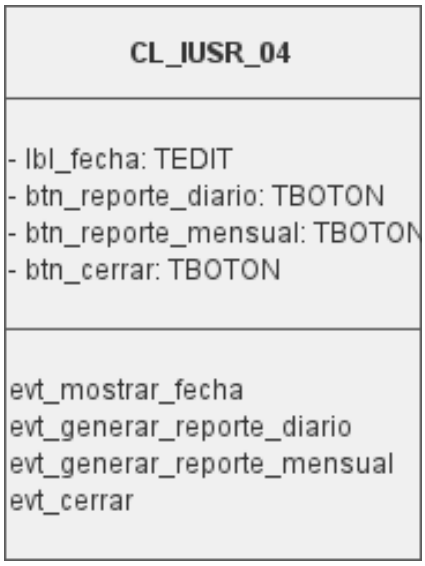
```
                v_detalles[i].Free
            FIN PARA
      SI NO
            ICONF_01.cerrar
      FIN (SI)
FIN
```

- **Pseudocódigo de evt_cerrar:**

```
INICIO
      IUSR_03.cerrar
FIN
```

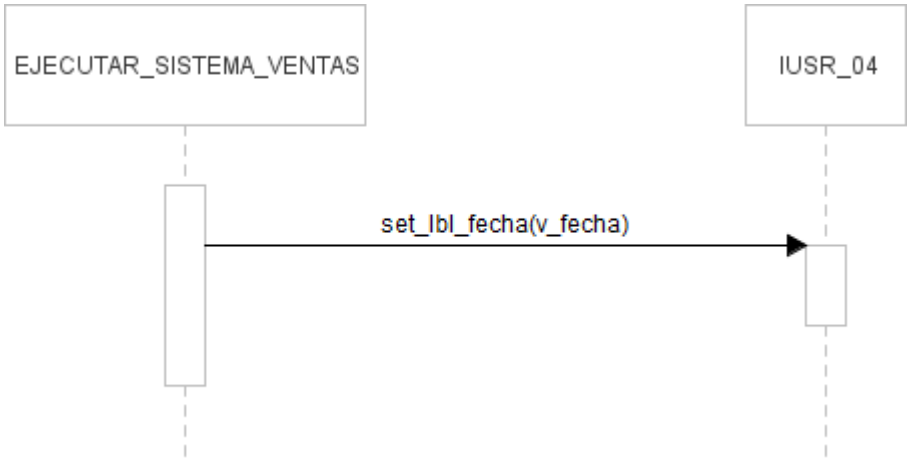
### 7.13. Diseño de la interfaz IUSR_04

### 7.14. Modelo de la clase CL_IUSR_04

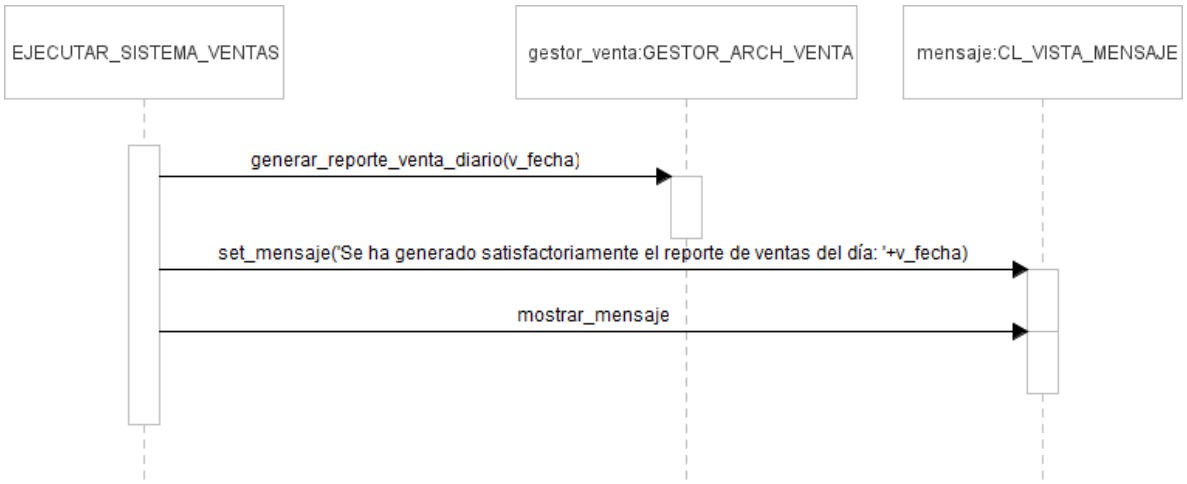



### 7.15. Determinación de eventos de la interfaz IUSR_04

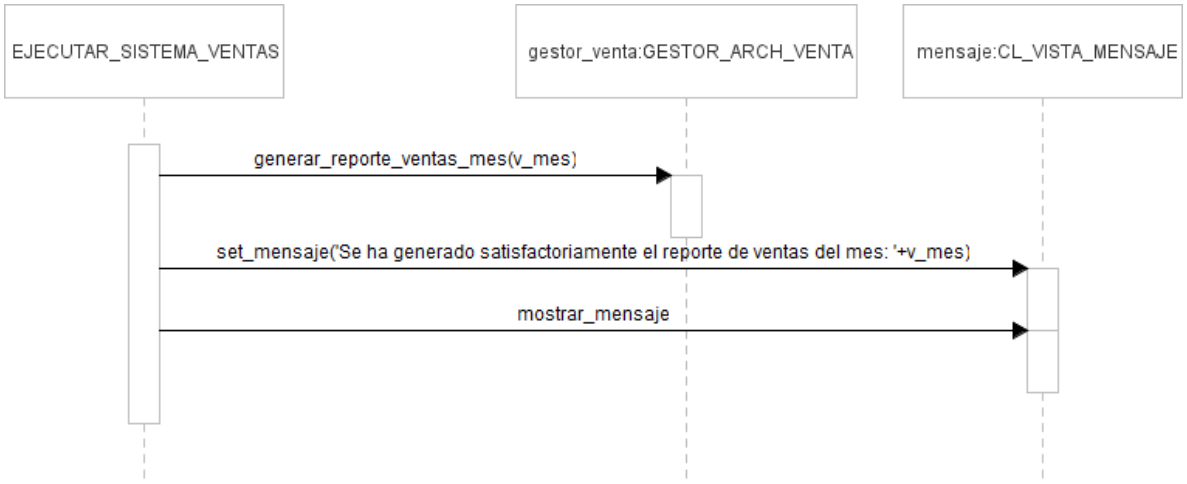
- **evt_mostrar_fecha:** Debe generar las acciones necesarias para que al mostrarse la interfaz IUSR_04 la fecha del sistema se coloque en el componente lbl_fecha.



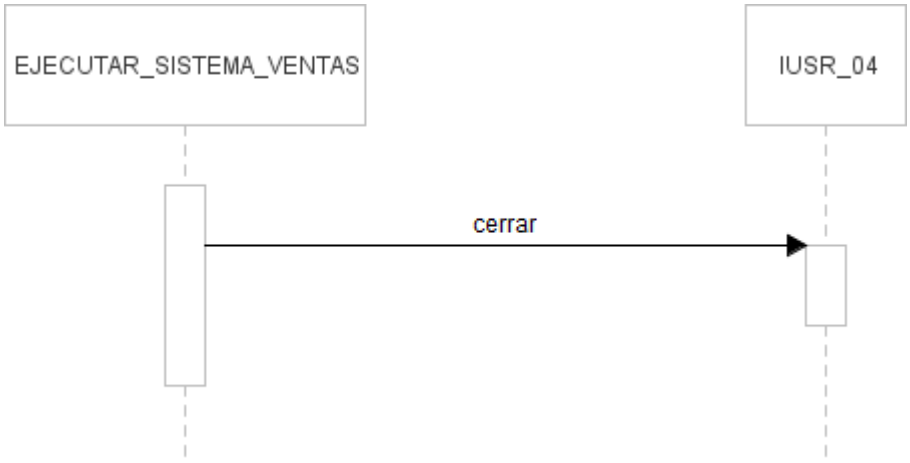

- **evt_generar_reporte_diario:** Debe generar las acciones necesarias para que se genere un reporte de venta del día de la fecha del sistema y que se guarde en la carpeta REPORTES dentro del proyecto.




- **evt_generar_reporte_mensual:** Debe generar las acciones necesarias para que se genere un reporte de venta del mes actual y que se guarde en la carpeta REPORTES dentro del proyecto.




- **evt_cerrar:** Debe generar las acciones necesarias para cerrar la interfaz IUSR_04.



### 7.16. Pseudocódigo de los eventos de la interfaz IUSR_04

- **Pseudocódigo de evt_mostrar_fecha**

```
SECCIÓN DE VARIABLES
      v_fecha: FECHA
INICIO
      v_fecha <-- FECHA_SISTEMA
      IUSR_04.set_lbl_fecha(CONVERTIR_A_CADENA(v_fecha))
FIN
```

- **Pseudocódigo de evt_generar_reporte_diario**

```
SECCIÓN DE VARIABLES
      v_fecha: FECHA
      gestor_venta: GESTOR_ARCH_VENTA
      mensaje: CL_VISTA_MENSAJE
INICIO
      v_fecha <-- FECHA_SISTEMA
      gestor_venta.GESTOR_ARCH_VENTA
      gestor_venta.generar_reporte_ventas_dia(CONVERTIR_A_CADENA(v_fec
ha))
      mensaje.CL_VISTA_MENSAJE
      mensaje.set_mensaje('Se ha generado satisfactoriamente el
reporte de ventas del día: '+v_fecha)
      mensaje.mostrar_mensaje
FIN
```

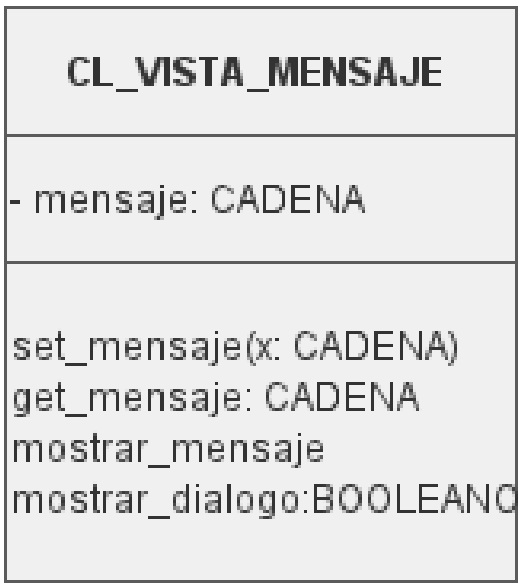
- **Pseudocódigo de evt_generar_reporte_mensual**

```
SECCIÓN DE VARIABLES
      v_fecha: FECHA
      v_mes: CADENA
      gestor_venta: GESTOR_ARCH_VENTA
      mensaje: CL_VISTA_MENSAJE
INICIO
      v_fecha <-- FECHA_SISTEMA
      v_mes <-- SUBCADENA(CONVERTIR_A_CADENA(v_fecha),4,7)
      gestor_venta.GESTOR_ARCH_VENTA
      gestor_venta.generar_reporte_ventas_mes(v_mes)
      mensaje.CL_VISTA_MENSAJE
      mensaje.set_mensaje('Se ha generado satisfactoriamente el
reporte de ventas del mes: '+v_mes)
      mensaje.mostrar_mensaje
FIN
```

- **Pseudocódigo de evt_cerrar**

```
INICIO
      IUSR_04.cerrar
FIN
```

### 7.17. Modelado de clase CL_VISTA_MENSAJE

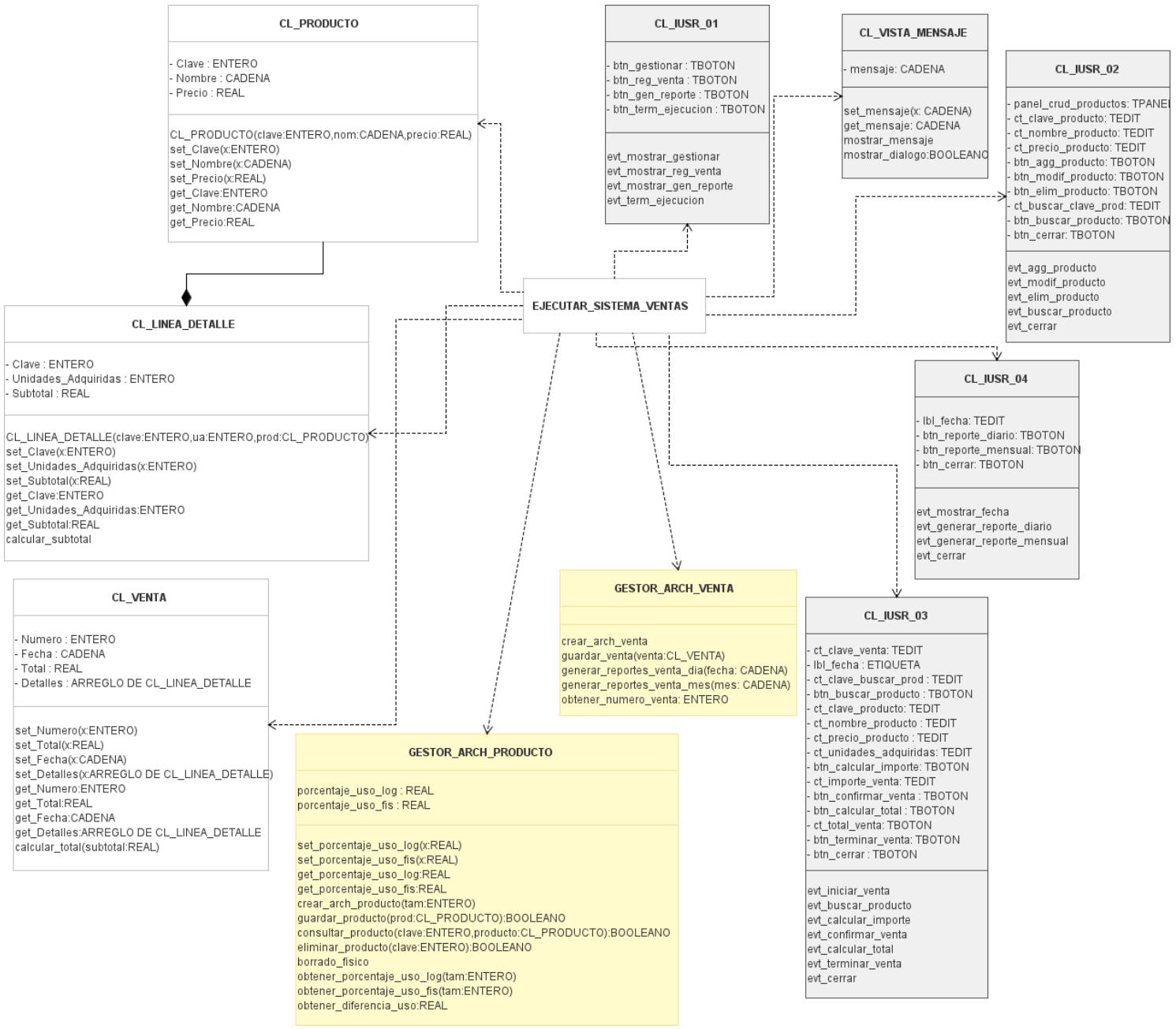



### 7.18. Pseudocódigo de clase CL_VISTA_MENSAJE

```
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
```

```
        MÉTODO mostrar_dialogo:BOOLEANO
        SECCIÓN DE VARIABLES
              resultado: ENTERO
              v_retorno:BOOLEANO
        INICIO
              //Recibe como parametro el mensaje y los botones
              resultado <-- DIALOGO(mensaje, [OK, CANCELAR])
              SI resultado=OK
                    v_retorno <-- VERDADERO
              SI NO
                    v_retorno <-- FALSO
              FIN SI
              REGRESAR v_retorno
        FIN MÉTODO mostrar_dialogo

FIN CLASE CL_VISTA_MENSAJE
```

## 8. DIAGRAMA DE CLASES



## 9. IMPLEMENTACIÓN CON PASCAL EN LAZARUS

### 9.1. 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.
```

### 9.2. 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.
```

### 9.3. 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.
```

### 9.4. 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.
```

### 9.5. GESTOR_ARCH_PRODUCTO.pas

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

interface

uses
    U_CL_PRODUCTO;

type
    GESTOR_ARCH_PRODUCTO = class
    private
        porcentaje_uso_log: Real;
        porcentaje_uso_fis: Real;
    public
        procedure set_porcentaje_uso_log(x: Real);
        procedure set_porcentaje_uso_fis(x: Real);
        function get_porcentaje_uso_log: Real;
        function get_porcentaje_uso_fis: Real;
        procedure crear_arch_producto(tam: Integer);
        function guardar_producto(producto: CL_PRODUCTO): Boolean;
        function consultar_producto(clave: Integer;var producto:
CL_PRODUCTO): Boolean;
        function modificar_producto(clave: Integer; producto:
CL_PRODUCTO): Boolean;
        function eliminar_producto(clave: Integer): Boolean;
        procedure borrado_fisico;
        procedure obtener_porcentaje_uso_log(tam:Integer);
        procedure obtener_porcentaje_uso_fis(tam:Integer);
        function obtener_diferencia_uso:Real;
    end;

implementation

uses
    SysUtils;

{ GESTOR_ARCH_PRODUCTO }

procedure GESTOR_ARCH_PRODUCTO.set_porcentaje_uso_log(x: Real);
begin
    porcentaje_uso_log := x;
end;

procedure GESTOR_ARCH_PRODUCTO.set_porcentaje_uso_fis(x: Real);
begin
    porcentaje_uso_fis := x;
end;

function GESTOR_ARCH_PRODUCTO.get_porcentaje_uso_log: Real;
begin
    Result := porcentaje_uso_log;
end;
```

```
function GESTOR_ARCH_PRODUCTO.get_porcentaje_uso_fis: Real;
begin
    Result := porcentaje_uso_fis;
end;

procedure GESTOR_ARCH_PRODUCTO.crear_arch_producto(tam: Integer);
type
    reg_producto = record
        campo_clave: Integer;
        campo_nombre: string[25];
        campo_precio: Real;
        campo_disponible: Integer;
    end;
var
    f_producto: file of reg_producto;
    v_reg_producto: reg_producto;
    i: Integer;
begin
    AssignFile(f_producto, 'Archivos\PRODUCTOS.rod');
    Rewrite(f_producto);
    v_reg_producto.campo_clave := 0;
    v_reg_producto.campo_nombre := '';
    v_reg_producto.campo_precio := 0;
    v_reg_producto.campo_disponible := -99;
    for i := 1 to tam do
        Write(f_producto, v_reg_producto);
    CloseFile(f_producto);
end;

function GESTOR_ARCH_PRODUCTO.guardar_producto(producto: CL_PRODUCTO):
Boolean;
type
    reg_producto = record
        campo_clave: Integer;
        campo_nombre: string[25];
        campo_precio: Real;
        campo_disponible: Integer;
    end;
var
    v_reg_producto, v_leer_reg: reg_producto;
    f_producto: file of reg_producto;
    v_pos, v_disponible: Integer;
    v_retorno: Boolean;
begin
    v_retorno := False;
    v_reg_producto.campo_clave := producto.get_Clave;
    v_reg_producto.campo_nombre := producto.get_Nombre;
    v_reg_producto.campo_precio := producto.get_Precio;
    v_reg_producto.campo_disponible := 1;

    { HASH MODULO 99 }
    v_pos := v_reg_producto.campo_clave mod 99;
```

```
    { REGISTRAR PRODUCTO }
    AssignFile(f_producto, 'Archivos\PRODUCTOS.rod');
    Reset(f_producto);
    Seek(f_producto, v_pos);
    Read(f_producto, v_leer_reg);
    if v_leer_reg.campo_disponible = -99 then
    begin
        Seek(f_producto, v_pos);
        Write(f_producto, v_reg_producto);
        v_retorno := True;
    end
    else { COLISION }
    begin
        if v_leer_reg.campo_clave <> producto.get_Clave then
        begin
            v_pos := 99;
            Seek(f_producto, v_pos);
            repeat
                Read(f_producto, v_leer_reg);
                v_disponible := v_leer_reg.campo_disponible;
                v_pos := v_pos + 1;
            until (v_disponible = -99) or (v_pos = 110);
            Write(f_producto, v_reg_producto);
            v_retorno := True;
        end;
    end;
    CloseFile(f_producto);
    Result := v_retorno;
end;

function GESTOR_ARCH_PRODUCTO.consultar_producto(clave: Integer;var
producto: CL_PRODUCTO): Boolean;
type
    reg_producto = record
        campo_clave: Integer;
        campo_nombre: string[25];
        campo_precio: Real;
        campo_disponible: Integer;
    end;
var
    f_producto: file of reg_producto;
    v_reg_producto: reg_producto;
    v_retorno: Boolean;
    v_pos: Integer;
begin
    AssignFile(f_producto, 'Archivos\PRODUCTOS.rod');
    Reset(f_producto);
    v_retorno := False;
    v_pos := clave mod 99;
    Seek(f_producto, v_pos);
    Read(f_producto, v_reg_producto);
    if v_reg_producto.campo_disponible <> -99 then
    begin
        v_retorno := True;
```

```
        producto.set_Clave(v_reg_producto.campo_clave);
        producto.set_Nombre(v_reg_producto.campo_nombre);
        producto.set_Precio(v_reg_producto.campo_precio);
    end;
    CloseFile(f_producto);
    Result := v_retorno;
end;

function GESTOR_ARCH_PRODUCTO.modificar_producto(clave: Integer;
producto: CL_PRODUCTO): Boolean;
type
    reg_producto = record
        campo_clave: Integer;
        campo_nombre: string[25];
        campo_precio: Real;
        campo_disponible: Integer;
    end;
var
    f_producto: file of reg_producto;
    v_reg_producto, v_reg_leer: reg_producto;
    v_retorno: Boolean;
    v_pos, v_contador: Integer;
begin
    AssignFile(f_producto, 'Archivos\PRODUCTOS.rod');
    Reset(f_producto);
    v_retorno := False;
    v_pos := clave mod 99;
    v_reg_producto.campo_clave := producto.get_Clave;
    v_reg_producto.campo_nombre := producto.get_Nombre;
    v_reg_producto.campo_precio := producto.get_Precio;
    v_reg_producto.campo_disponible := 1;

    Seek(f_producto, v_pos);
    Read(f_producto, v_reg_leer);

    if (v_reg_leer.campo_disponible = 1) and (v_reg_leer.campo_clave =
clave) then
    begin
        v_retorno := True;
        Seek(f_producto, v_pos);
        Write(f_producto, v_reg_producto); // Sobrescribe el registro
    end
    else
    begin
        Seek(f_producto, 99); // Zona de colisiones
        v_contador := 99;
        while not Eof(f_producto) do
        begin
            Read(f_producto, v_reg_leer);
            v_contador := v_contador + 1;

            if (v_reg_leer.campo_clave = clave) and
(v_reg_leer.campo_disponible = 1) then
            begin
```

```
                v_retorno := True;
                Seek(f_producto, v_contador - 1);
                Write(f_producto, v_reg_producto);
            end;
        end;
    end;

    CloseFile(f_producto);
    Result := v_retorno;
end;

function GESTOR_ARCH_PRODUCTO.eliminar_producto(clave: Integer):
Boolean;
type
    reg_producto = record
        campo_clave: Integer;
        campo_nombre: string[25];
        campo_precio: Real;
        campo_disponible: Integer;
    end;
var
    f_producto: file of reg_producto;
    v_reg_producto: reg_producto;
    v_retorno: Boolean;
    v_pos, v_contador: Integer;
begin
    v_retorno := False;
    v_pos := clave mod 99;
    AssignFile(f_producto, 'Archivos\PRODUCTOS.rod');
    Reset(f_producto);
    Seek(f_producto, v_pos);
    Read(f_producto, v_reg_producto);
    if (v_reg_producto.campo_disponible = 1) and
(v_reg_producto.campo_clave = clave) then
    begin
        Seek(f_producto, v_pos);
        v_reg_producto.campo_disponible := -99;
        Write(f_producto, v_reg_producto);
        v_retorno := True;
    end
    else
    begin
        Seek(f_producto, 99);
        v_contador := 99;
        while not Eof(f_producto) do
        begin
            Read(f_producto, v_reg_producto);
            v_contador := v_contador + 1;
            if (v_reg_producto.campo_clave = clave) and
(v_reg_producto.campo_disponible = 1) then
            begin
                v_reg_producto.campo_disponible := -99;
                Seek(f_producto, v_contador - 1);
                Write(f_producto, v_reg_producto);
```

```
                v_retorno := True;
            end;
        end;
    end;
    CloseFile(f_producto);
    Result := v_retorno;
end;

procedure GESTOR_ARCH_PRODUCTO.borrado_fisico;
type
    reg_producto = record
        campo_clave: Integer;
        campo_nombre: string[25];
        campo_precio: Real;
        campo_disponible: Integer;
    end;
var
    f_producto, f_temporal: file of reg_producto;
    v_reg_producto: reg_producto;
    v_marca, v_contador: Integer;
begin
    v_contador := 0;
    AssignFile(f_temporal, 'Archivos\TEMPORAL.rod');
    Rewrite(f_temporal);
    AssignFile(f_producto, 'Archivos\PRODUCTOS.rod');
    Reset(f_producto);
    while not Eof(f_producto) do
    begin
        Read(f_producto, v_reg_producto);
        v_contador := v_contador + 1;
        v_marca := v_reg_producto.campo_disponible;
        if v_marca <> -99 then
        begin
            Seek(f_temporal, v_contador);
            Write(f_temporal, v_reg_producto);
        end;
    end;
    CloseFile(f_producto);
    DeleteFile('Archivos\PRODUCTOS.rod');
    CloseFile(f_temporal);
    RenameFile('Archivos\TEMPORAL.rod', 'Archivos\PRODUCTOS.rod');
end;

procedure
GESTOR_ARCH_PRODUCTO.obtener_porcentaje_uso_log(tam:Integer);
type
    reg_producto = record
        campo_clave: Integer;
        campo_nombre: string[25];
        campo_precio: Real;
        campo_disponible: Integer;
    end;
var
    f_producto: file of reg_producto;
```

```
    v_reg_producto: reg_producto;
    v_contador, cant_reg_log: Integer;
begin
    v_contador:=1;
    cant_reg_log:=0;
    AssignFile(f_producto,'Archivos\PRODUCTOS.rod');
    Reset(f_producto);
    repeat
        Seek(f_producto,v_contador);
        Read(f_producto,v_reg_producto);
        If v_reg_producto.campo_disponible=1 then
        begin
            cant_reg_log:=cant_reg_log+1;
        end;
        v_contador:=v_contador+1;
    until v_contador=tam;
    CloseFile(f_producto);
    //Porcentaje de uo lógico
    porcentaje_uso_log:=(cant_reg_log/tam)*100;
end;

procedure
GESTOR_ARCH_PRODUCTO.obtener_porcentaje_uso_fis(tam:Integer);
type
    reg_producto = record
        campo_clave: Integer;
        campo_nombre: string[25];
        campo_precio: Real;
        campo_disponible: Integer;
    end;
var
    f_producto: file of reg_producto;
    v_reg_producto: reg_producto;
    v_contador, cant_reg_fis: Integer;
begin
    v_contador:=1;
    cant_reg_fis:=0;
    AssignFile(f_producto,'Archivos\PRODUCTOS.rod');
    Reset(f_producto);
    repeat
        Seek(f_producto,v_contador);
        Read(f_producto,v_reg_producto);
        If v_reg_producto.campo_clave<>0 then
        begin
            cant_reg_fis:=cant_reg_fis+1;
        end;
        v_contador:=v_contador+1;
    until v_contador=tam;
    CloseFile(f_producto);
    //Porcentaje de uo físico
    porcentaje_uso_fis:=(cant_reg_fis/tam)*100;
end;

function GESTOR_ARCH_PRODUCTO.obtener_diferencia_uso:Real;
```

```
var
   diferencia: Real;
begin
   diferencia:= Abs(porcentaje_uso_log-porcentaje_uso_fis);
   Result:=diferencia;
end;

end.
```

### 9.6. GESTOR_ARCH_VENTA.pas

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

interface

uses
    U_CL_LINEA_DETALLE, U_CL_VENTA,SysUtils;

type
    GESTOR_ARCH_VENTA = class
    private

    public
        procedure crear_arch_venta;
        procedure guardar_venta(venta: CL_VENTA);
        procedure generar_reporte_ventas_dia(fecha:String);
        procedure generar_reporte_ventas_mes(mes: String);
        function obtener_numero_venta:Integer;
    end;

implementation

procedure GESTOR_ARCH_VENTA.crear_arch_venta;
type
    reg_venta = record
        campo_numero: Integer;
        campo_fecha: string[10];
        campo_linea_det: array [1..10] of CL_LINEA_DETALLE;
      campo_total: Real;
    end;
var
    f_venta: file of reg_venta;
begin
    AssignFile(f_venta, 'Archivos\VENTAS.rod');
    Rewrite(f_venta);
    CloseFile(f_venta);
end;

procedure GESTOR_ARCH_VENTA.guardar_venta(venta: CL_VENTA);
type
    reg_venta = record
        campo_numero: Integer;
        campo_fecha: string[10];
```

```
        campo_linea_det: array [1..10] of CL_LINEA_DETALLE;
      campo_total: Real;
    end;
var
    v_reg_venta: reg_venta;
    f_venta: file of reg_venta;
begin
    AssignFile(f_venta, 'Archivos\VENTAS.rod');
    Reset(f_venta);
    While (EOF(f_venta)=False) do
    begin
      Read(f_venta, v_reg_venta);
    end;
      v_reg_venta.campo_numero := venta.get_Numero;
    v_reg_venta.campo_fecha := venta.get_Fecha;
    v_reg_venta.campo_linea_det := venta.get_Detalles;
    v_reg_venta.campo_total := venta.get_Total;
    Write(f_venta, v_reg_venta);
    CloseFile(f_venta);
end;

procedure GESTOR_ARCH_VENTA.generar_reporte_ventas_dia(fecha:String);
type
    reg_venta = record
        campo_numero: Integer;
        campo_fecha: string[10];
        campo_linea_det: array [1..10] of CL_LINEA_DETALLE;
      campo_total: Real;
    end;
var
    v_reg_venta: reg_venta;
    f_venta: file of reg_venta;
    f_reporte: Text;
    registro_encontrado:String;
    total_dia:Real;
begin
    total_dia:=0;
    Assign(f_reporte,'REPORTES\REPORTE-DIA.txt');
    REWRITE(f_reporte);
    WriteLn(f_reporte,'REPORTE DE VENTAS DEL DÍA '+ fecha);
    WriteLn(' ');
    WriteLn(f_reporte,'NO. VENTA'+' | '+'  FECHA  '+' | '+'TOTAL
VENTA');
    Assign(f_venta,'Archivos\VENTAS.rod');
    Reset(f_venta);
    While EOF(f_venta)=False do
    begin
        Read(f_venta,v_reg_venta);
        If v_reg_venta.campo_fecha=fecha then
        begin
          registro_encontrado:= '
'+IntToStr(v_reg_venta.campo_numero)+'   '+' |
'+v_reg_venta.campo_fecha+' | '+'
'+FloatToStr(v_reg_venta.campo_total);
```

```
          WriteLn(f_reporte,registro_encontrado);
          total_dia:=total_dia+v_reg_venta.campo_total;
        end;
    end;
    WriteLn(f_reporte,'TOTAL VENDIDO EN EL DÍA:
'+FloatToStr(total_dia));
    Close(f_reporte);
    Close(f_venta);
end;

procedure GESTOR_ARCH_VENTA.generar_reporte_ventas_mes(mes: String);
type
    reg_venta = record
        campo_numero: Integer;
        campo_fecha: string[10];
        campo_linea_det: array [1..10] of CL_LINEA_DETALLE;
      campo_total: Real;
    end;
var
    v_reg_venta: reg_venta;
    f_venta: file of reg_venta;
    f_reporte: Text;
    registro_encontrado, mes_registrado:String;
    total_mes:Real;
begin
    total_mes:=0;
    Assign(f_reporte,'REPORTES\REPORTE-MES.txt');
    REWRITE(f_reporte);
    WriteLn(f_reporte,'REPORTE DE VENTAS DEL MES '+ mes);
    WriteLn(' ');
    WriteLn(f_reporte,'NO. VENTA'+' | '+'  FECHA  '+' | '+'TOTAL
VENTA');
    Assign(f_venta,'Archivos\VENTAS.rod');
    Reset(f_venta);
    While EOF(f_venta)=False do
    begin
        Read(f_venta,v_reg_venta);
        mes_registrado:=v_reg_venta.campo_fecha;
        mes_registrado:=Copy(mes_registrado,4,7);
        If mes_registrado=mes then
        begin
          registro_encontrado:= '
'+IntToStr(v_reg_venta.campo_numero)+'   '+' |
'+v_reg_venta.campo_fecha+' | '+'
'+FloatToStr(v_reg_venta.campo_total);
          WriteLn(f_reporte,registro_encontrado);
          total_mes:=total_mes+v_reg_venta.campo_total;
        end;
    end;
    WriteLn(f_reporte,'TOTAL VENDIDO EN EL MES:
'+FloatToStr(total_mes));
    Close(f_reporte);
    Close(f_venta);
end;
```

```
function GESTOR_ARCH_VENTA.obtener_numero_venta:Integer;
type
    reg_venta = record
        campo_numero: Integer;
        campo_fecha: string[10];
        campo_linea_det: array [1..10] of CL_LINEA_DETALLE;
      campo_total: Real;
    end;
var
    v_reg_venta: reg_venta;
    f_venta: file of reg_venta;
    v_retorno: Integer;
begin
    AssignFile(f_venta, 'Archivos\VENTAS.rod');
    Reset(f_venta);
    v_retorno:=0;
    While (EOF(f_venta)=False) do
    begin
      Read(f_venta, v_reg_venta);
         v_retorno:=v_reg_venta.campo_numero;
    end;
    CloseFile(f_venta);
    Result:=v_retorno;
end;

end.
```

### 9.7. CL_IUSR_01.pas

```
unit CL_IUSR_01;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls,
StdCtrls,
  cl_iusr_02, cl_iusr_03, cl_iusr_04;

type

  { TIUSR_01 }

  TIUSR_01 = class(TForm)
    btn_gestionar: TButton;
    btn_reg_venta: TButton;
    btn_gen_reporte: TButton;
    btn_term_ejecucion: TButton;
    Label1: TLabel;
    Panel1: TPanel;
    procedure evt_mostrar_gen_reporte(Sender: TObject);
    procedure evt_term_ejecucion(Sender: TObject);
```

```
    procedure evt_mostrar_gestionar(Sender: TObject);
    procedure evt_mostrar_reg_venta(Sender: TObject);
  private

  public

  end;

var
  IUSR_01: TIUSR_01;

implementation

{$R *.lfm}

{ TIUSR_01 }

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

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

procedure TIUSR_01.evt_mostrar_gen_reporte(Sender: TObject);
begin
   IUSR_04.ShowModal;
end;

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

end.
```

### 9.8. CL_IUSR_02.pas

```
unit CL_IUSR_02;

{$mode ObjFPC}{$H+}

interface

uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls,
StdCtrls,
  Buttons, cl_iconf_01, U_CL_PRODUCTO, U_GESTOR_ARCH_PRODUCTO,
U_CL_VISTA_MENSAJE;
```

```
type

  { TIUSR_02 }

  TIUSR_02 = class(TForm)
    btn_cerrar: TButton;
    btn_buscar_producto: TButton;
    ct_clave_producto: TEdit;
    ct_porcentaje_fisico_p: TEdit;
    ct_nombre_producto: TEdit;
    ct_porcentaje_logico_p: TEdit;
    ct_precio_producto: TEdit;
    ct_buscar_clave_prod: TEdit;
    Label1: TLabel;
    Label10: TLabel;
    Label2: TLabel;
    Label22: TLabel;
    Label23: TLabel;
    Label24: TLabel;
    Label25: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Panel1: TPanel;
    panel_crud_productos: TPanel;
    btn_agg_producto: TSpeedButton;
    btn_modif_producto: TSpeedButton;
    btn_elim_producto: TSpeedButton;
    procedure evt_agg_producto(Sender: TObject);
    procedure evt_buscar_producto(Sender: TObject);
    procedure evt_cerrar(Sender: TObject);
    procedure evt_elim_producto(Sender: TObject);
    procedure evt_modif_producto(Sender: TObject);
  private

  public

  end;

var
  IUSR_02: TIUSR_02;

implementation

{$R *.lfm}

{ CONTROL}

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

```
end;

procedure TIUSR_02.evt_elim_producto(Sender: TObject);
var
  v_conf: Integer;
  v_clave: String;
  mensaje: CL_VISTA_MENSAJE;
  gestor: GESTOR_ARCH_PRODUCTO;
  v_producto_eliminado:Boolean;
  v_diferencia: Real;
begin
  mensaje:=CL_VISTA_MENSAJE.Create;
  gestor:=GESTOR_ARCH_PRODUCTO.Create;
  ICONF_01.lbl_pregunta.Caption:='¿Estas seguro que deseas eliminar el
producto'+LineEnding+'encontrado del registro?';
  v_conf:=ICONF_01.ShowModal;
  If v_conf=mrYes then
  begin
    ICONF_01.Close;
    v_clave:=(IUSR_02.ct_clave_producto.Text);
    If NOT(v_clave='') then
    begin
      If(Length(v_clave)=4) then
      begin

v_producto_eliminado:=gestor.eliminar_producto(StrToInt(v_clave));
       If NOT(v_producto_eliminado=True) then
       begin
         mensaje.set_mensaje('El producto que intentas eliminar no
existe, verifica la clave del producto.');
       mensaje.mostrar_mensaje;
       end else
       begin
         mensaje.set_mensaje('Producto eliminado
satisfactoriamente.');
       mensaje.mostrar_mensaje;
         IUSR_02.ct_clave_producto.Text:='';
         IUSR_02.ct_nombre_producto.Text:='';
         IUSR_02.ct_precio_producto.Text:='';
         IUSR_02.ct_buscar_clave_prod.Text:='';
         gestor.obtener_porcentaje_uso_log(110);
         gestor.obtener_porcentaje_uso_fis(110);

IUSR_02.ct_porcentaje_logico_p.Text:=FloatToStr(gestor.get_porcentaje_
uso_log);

IUSR_02.ct_porcentaje_fisico_p.Text:=FloatToStr(gestor.get_porcentaje_
uso_fis);
         v_diferencia:=gestor.obtener_diferencia_uso;
         If v_diferencia>=50 then
         begin
             ICONF_01.lbl_pregunta.Caption:='¿Deseas realizar un
borrado físico de'+LineEnding+
```

```
                                                    'los registros
lógicamente eliminados?';
              v_conf:=ICONF_01.ShowModal;
              If v_conf=mrYes then
              begin
                ICONF_01.Close;
                gestor.borrado_fisico;
                gestor.obtener_porcentaje_uso_fis(110);

IUSR_02.ct_porcentaje_fisico_p.Text:=FloatToStr(gestor.get_porcentaje_
uso_fis);
              end else
              begin
                ICONF_01.Close;
              end;
          end;
        end;
      end else
      begin
        mensaje.set_mensaje('La clave del producto debe ser un número
entero con 4 digitos');
      mensaje.mostrar_mensaje;
      end;
    end else
    begin
      mensaje.set_mensaje('Verifica haber buscado un producto, para
poder eliminarlo del registro.');
      mensaje.mostrar_mensaje;
    end;
  end else
  begin
    ICONF_01.Close;
    IUSR_02.ct_clave_producto.Text:='';
    IUSR_02.ct_nombre_producto.Text:='';
    IUSR_02.ct_precio_producto.Text:='';
    IUSR_02.ct_buscar_clave_prod.Text:='';
  end;
  mensaje.Free;
  gestor.Free;
end;

procedure TIUSR_02.evt_agg_producto(Sender: TObject);
var
  v_conf: Integer;
  v_nombre, v_clave, v_precio: String;
  mensaje: CL_VISTA_MENSAJE;
  producto: CL_PRODUCTO;
  gestor: GESTOR_ARCH_PRODUCTO;
  v_producto_guardado:Boolean;
  v_diferencia: Real;
begin
  mensaje:=CL_VISTA_MENSAJE.Create;
  gestor:=GESTOR_ARCH_PRODUCTO.Create;
```

```
  ICONF_01.lbl_pregunta.Caption:='¿Estas seguro que deseas agregar el
producto'+LineEnding+'proporcionado al registro?';
  v_conf:=ICONF_01.ShowModal;
  If v_conf=mrYes then
  begin
    ICONF_01.Close;
    v_clave:=(IUSR_02.ct_clave_producto.Text);
    v_nombre:=IUSR_02.ct_nombre_producto.Text;
    v_precio:=(IUSR_02.ct_precio_producto.Text);
    If NOT(v_clave='') AND NOT(v_nombre='') AND NOT(v_precio='') then
    begin
      If(Length(v_clave)=4) then
      begin

producto:=CL_PRODUCTO.Create(StrToInt(v_clave),v_nombre,StrToFloat(v_p
recio));
        v_producto_guardado:=gestor.guardar_producto(producto);
        producto.Free;
        If NOT(v_producto_guardado=True) then
        begin
          mensaje.set_mensaje('El producto que intentas guardar ya
existe, verifica la clave del producto.');
       mensaje.mostrar_mensaje;
        end else
        begin
          mensaje.set_mensaje('Producto guardado
satisfactoriamente.');
       mensaje.mostrar_mensaje;
          IUSR_02.ct_clave_producto.Text:='';
          IUSR_02.ct_nombre_producto.Text:='';
          IUSR_02.ct_precio_producto.Text:='';
          gestor.obtener_porcentaje_uso_log(110);
          gestor.obtener_porcentaje_uso_fis(110);

IUSR_02.ct_porcentaje_logico_p.Text:=FloatToStr(gestor.get_porcentaje_
uso_log);

IUSR_02.ct_porcentaje_fisico_p.Text:=FloatToStr(gestor.get_porcentaje_
uso_fis);
          v_diferencia:=gestor.obtener_diferencia_uso;
          If v_diferencia>=50 then
          begin
              ICONF_01.lbl_pregunta.Caption:='¿Deseas realizar un
borrado físico de'+LineEnding+
                                             'los registros
lógicamente eliminados?';
              v_conf:=ICONF_01.ShowModal;
              If v_conf=mrYes then
              begin
                ICONF_01.Close;
                gestor.borrado_fisico;
                gestor.obtener_porcentaje_uso_fis(110);
```

```
IUSR_02.ct_porcentaje_fisico_p.Text:=FloatToStr(gestor.get_porcentaje_
uso_fis);
              end else
              begin
                ICONF_01.Close;
              end;
          end;
        end;
      end else
      begin
        mensaje.set_mensaje('La clave del producto debe ser un número
entero con 4 digitos');
      mensaje.mostrar_mensaje;
      end;
    end else
    begin
      mensaje.set_mensaje('Verifica llenar los campos CLAVE, NOMBRE y
PRECIO, para poder agregar un producto.');
      mensaje.mostrar_mensaje;
    end;
  end else
  begin
    ICONF_01.Close;
    IUSR_02.ct_clave_producto.Text:='';
    IUSR_02.ct_nombre_producto.Text:='';
    IUSR_02.ct_precio_producto.Text:='';
  end;
  mensaje.Free;
  gestor.Free;
end;

procedure TIUSR_02.evt_buscar_producto(Sender: TObject);
var
  v_buscar_producto,v_clave,v_nombre,v_precio:String;
  mensaje: CL_VISTA_MENSAJE;
  gestor: GESTOR_ARCH_PRODUCTO;
  producto: CL_PRODUCTO;
  v_encontrado: Boolean;
begin
  mensaje:=CL_VISTA_MENSAJE.Create;
  gestor:=GESTOR_ARCH_PRODUCTO.Create;
  v_buscar_producto:=IUSR_02.ct_buscar_clave_prod.Text;
  If NOT(v_buscar_producto='') then
  begin
     If(Length(v_buscar_producto)=4) then
     begin
       producto:=CL_PRODUCTO.Create(0,'',0);

v_encontrado:=gestor.consultar_producto(StrToInt(v_buscar_producto),pr
oducto);
       If v_encontrado=True then
       begin
         mensaje.set_mensaje('Producto encontrado.');
```

```
       mensaje.mostrar_mensaje;
       v_clave:= IntToStr(producto.get_Clave);
       v_nombre:= producto.get_Nombre;
       v_precio:= FloatToStr(producto.get_Precio);
          IUSR_02.ct_clave_producto.Text:=v_clave;
          IUSR_02.ct_nombre_producto.Text:=v_nombre;
          IUSR_02.ct_precio_producto.Text:=v_precio;
          producto.Free;
       end else
       begin
         mensaje.set_mensaje('El producto no ha sido encontrado');
       mensaje.mostrar_mensaje;
          IUSR_02.ct_clave_producto.Text:='';
          IUSR_02.ct_nombre_producto.Text:='';
          IUSR_02.ct_precio_producto.Text:='';
       end;
     end else
     begin
       mensaje.set_mensaje('Verifica que la clave del producto que
estás buscando tenga 4 dígitos');
       mensaje.mostrar_mensaje;
     end;
  end else
  begin
    mensaje.set_mensaje('Para buscar un producto ingresa la clave del
producto en el campo de la parte inferiror');
    mensaje.mostrar_mensaje;
  end;
  mensaje.Free;
  gestor.Free;
end;

procedure TIUSR_02.evt_modif_producto(Sender: TObject);
var
  gestor: GESTOR_ARCH_PRODUCTO;
  producto: CL_PRODUCTO;
  mensaje: CL_VISTA_MENSAJE;
  v_conf: Integer;
  v_clave_nueva,v_nombre_nuevo,v_precio_nuevo,
  v_clave_anterior,instrucciones: String;
  v_existe_producto_nuevo, v_existe_producto_anterior,r: Boolean;
  v_diferencia: Real;
begin
  mensaje:=CL_VISTA_MENSAJE.Create;
  gestor:=GESTOR_ARCH_PRODUCTO.Create;
  instrucciones := 'Para poder modificar un registro;'+LineEnding+
  '1.- Busca el producto a modificar'+LineEnding+
  '2.- Realiza los cambios en los 3 campos de la
izquierda'+LineEnding+
  '3.- Da click en el botón "Modificar"'+LineEnding+
  '4.- Da click en "Aceptar"'+LineEnding+
  'Nota: El registro a modificar será el buscado en al campo de la
parte inferior.';
  mensaje.set_mensaje(instrucciones);
```

```
  mensaje.mostrar_mensaje;
  v_conf := ICONF_01.ShowModal;
  If v_conf=mrYes then
  begin
     v_clave_nueva := IUSR_02.ct_clave_producto.Text;
     v_nombre_nuevo := IUSR_02.ct_nombre_producto.Text;
     v_precio_nuevo := IUSR_02.ct_precio_producto.Text;
     v_clave_anterior := IUSR_02.ct_buscar_clave_prod.Text;
     If NOT((v_clave_nueva='') AND (v_nombre_nuevo='') AND
(v_precio_nuevo='') AND (v_clave_anterior='')) AND
((Length(v_clave_nueva)=4) AND (Length(v_clave_anterior)=4)) then
     begin
        If v_clave_nueva=v_clave_anterior then
        begin
           producto:=CL_PRODUCTO.Create(0,'',0);
         v_existe_producto_anterior :=
gestor.consultar_producto(StrToInt(v_clave_anterior),producto);
           producto.Free;
         If v_existe_producto_anterior = True then
           begin

producto:=CL_PRODUCTO.Create(StrToInt(v_clave_nueva),v_nombre_nuevo,St
rToFloat(v_precio_nuevo));
            r:=gestor.eliminar_producto(StrToInt(v_clave_anterior));
            r:=gestor.guardar_producto(producto);
            producto.Free;
              IUSR_02.ct_clave_producto.Text:='';
              IUSR_02.ct_nombre_producto.Text:='';
              IUSR_02.ct_precio_producto.Text:='';
              IUSR_02.ct_buscar_clave_prod.Text:='';
              mensaje.set_mensaje('Producto modificado
satisfactoriamente.');
              mensaje.mostrar_mensaje;
            gestor.obtener_porcentaje_uso_log(110);
            gestor.obtener_porcentaje_uso_fis(110);
            v_diferencia := gestor.obtener_diferencia_uso;

IUSR_02.ct_porcentaje_logico_p.Text:=FloatToStr(gestor.get_porcentaje_
uso_log);

IUSR_02.ct_porcentaje_fisico_p.Text:=FloatToStr(gestor.get_porcentaje_
uso_fis);
            If v_diferencia>=50 then
              begin
               ICONF_01.lbl_pregunta.Caption:='¿Desea realizar un
borrado físico de los registros lógicamente eliminados del archivo?';
             v_conf := ICONF_01.ShowModal;
             If v_conf=mrYes then
                  begin
               ICONF_01.Close;
               gestor.borrado_fisico;
               gestor.obtener_porcentaje_uso_fis(110);
```

```
IUSR_02.ct_porcentaje_fisico_p.Text:=FloatToStr(gestor.get_porcentaje_
uso_fis);
             end else
                  begin
                ICONF_01.Close;
             end;
            end;
         end else
           begin
            mensaje.set_mensaje('El producto que está intentando
modificar no existe, asegurese de buscar un producto correctamente.');
            mensaje.mostrar_mensaje;
         end;
      end else
        begin
           producto:=CL_PRODUCTO.Create(0,'',0);
         v_existe_producto_nuevo :=
gestor.consultar_producto(StrToInt(v_clave_nueva),producto);
         v_existe_producto_anterior :=
gestor.consultar_producto(StrToInt(v_clave_anterior),producto);
           producto.Free;
         If v_existe_producto_anterior=True then
           begin
            If v_existe_producto_nuevo=False then
              begin

producto:=CL_PRODUCTO.Create(StrToInt(v_clave_nueva),v_nombre_nuevo,St
rToFloat(v_precio_nuevo));
             r:=gestor.eliminar_producto(StrToInt(v_clave_anterior));
             r:=gestor.guardar_producto(producto);
             producto.Free;
                  IUSR_02.ct_clave_producto.Text:='';
                  IUSR_02.ct_nombre_producto.Text:='';
                  IUSR_02.ct_precio_producto.Text:='';
                  IUSR_02.ct_buscar_clave_prod.Text:='';
                  mensaje.set_mensaje('Producto modificado
satisfactoriamente.');
                  mensaje.mostrar_mensaje;
             gestor.obtener_porcentaje_uso_log(110);
             gestor.obtener_porcentaje_uso_fis(110);
             v_diferencia := gestor.obtener_diferencia_uso;

IUSR_02.ct_porcentaje_logico_p.Text:=FloatToStr(gestor.get_porcentaje_
uso_log);

IUSR_02.ct_porcentaje_fisico_p.Text:=FloatToStr(gestor.get_porcentaje_
uso_fis);
             If v_diferencia>=50 then
                  begin
                     ICONF_01.lbl_pregunta.Caption:='¿Desea realizar un
borrado físico de los registros lógicamente eliminados del archivo?';
                 v_conf := ICONF_01.ShowModal;
                 If v_conf=mrYes then
```

```
                    begin
                     ICONF_01.Close;
                     gestor.borrado_fisico;
                     gestor.obtener_porcentaje_uso_fis(110);

IUSR_02.ct_porcentaje_fisico_p.Text:=FloatToStr(gestor.get_porcentaje_
uso_fis);
                end else
                     begin
                     ICONF_01.Close;
                end;
              end;
            end else
               begin
                mensaje.set_mensaje('El producto con clave
'+v_clave_nueva+' ya existe, verifica tu modificación.');
             mensaje.mostrar_mensaje;
            end;
         end else
            begin
             mensaje.set_mensaje('El producto que intentas modificar no
existe, verifica que hayas buscado el producto correctamente.');
             mensaje.mostrar_mensaje;
         end;
      end;
     end else
     begin
        mensaje.set_mensaje('Verifica haber buscado el producto
correctamente, la clave del producto a modificar y la clave nueva
deben de ser numeros enteros de 4 dígitos.');
      mensaje.mostrar_mensaje;
     end;
  end else
  begin
     ICONF_01.Close;
  end;
  mensaje.Free;
  gestor.Free;
end;

end.
```

### 9.9. CL_IUSR_03.pas

```
unit CL_IUSR_03;

{$mode ObjFPC}{$H+}

interface

uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls,
StdCtrls,
```

```
U_CL_PRODUCTO,U_CL_VISTA_MENSAJE,U_GESTOR_ARCH_PRODUCTO,U_CL_LINEA_DET
ALLE,
  U_CL_VENTA,U_GESTOR_ARCH_VENTA,CL_ICONF_01;

type

  { TIUSR_03 }

  TIUSR_03 = class(TForm)
    btn_cerrar: TButton;
    btn_buscar_producto: TButton;
    btn_calcular_importe: TButton;
    btn_confirmar_venta: TButton;
    btn_calcular_total: TButton;
    btn_terminar_venta: TButton;
    ct_clave_producto: TEdit;
    ct_clave_venta: TEdit;
    ct_precio_producto: TEdit;
    ct_nombre_producto: TEdit;
    ct_clave_buscar_prod: TEdit;
    ct_unidades_adquiridas: TEdit;
    ct_importe_venta: TEdit;
    ct_total_venta: TEdit;
    Label1: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label2: TLabel;
    lbl_fecha: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Panel1: TPanel;
    Panel2: TPanel;
    Panel3: TPanel;
    Panel4: TPanel;
    procedure evt_buscar_producto(Sender: TObject);
    procedure evt_calcular_importe(Sender: TObject);
    procedure evt_calcular_total(Sender: TObject);
    procedure evt_cerrar(Sender: TObject);
    procedure evt_confirmar_venta(Sender: TObject);
    procedure evt_terminar_venta(Sender: TObject);
    procedure iniciar_venta(Sender: TObject);
  private

  public

  end;
```

```
var
  IUSR_03: TIUSR_03;

implementation

var
  obj_venta: CL_VENTA;
  numero_linea_detalle: Integer;
  v_Detalles: array [1..10] of CL_LINEA_DETALLE;

{$R *.lfm}

{ TIUSR_03 }

procedure TIUSR_03.evt_buscar_producto(Sender: TObject);
var
  v_buscar_producto,v_nombre,v_precio, v_clave:String;
  mensaje: CL_VISTA_MENSAJE;
  gestor: GESTOR_ARCH_PRODUCTO;
  producto: CL_PRODUCTO;
  v_encontrado: Boolean;
begin
  mensaje:=CL_VISTA_MENSAJE.Create;
  gestor:=GESTOR_ARCH_PRODUCTO.Create;
  v_buscar_producto:=IUSR_03.ct_clave_buscar_prod.Text;
  If NOT(v_buscar_producto='') then
  begin
     If(Length(v_buscar_producto)=4) then
     begin
       producto:=CL_PRODUCTO.Create(0,'',0);

v_encontrado:=gestor.consultar_producto(StrToInt(v_buscar_producto),pr
oducto);
       If v_encontrado=True then
       begin
         mensaje.set_mensaje('Producto encontrado.');
       mensaje.mostrar_mensaje;
         v_clave:= IntToStr(producto.get_Clave);
       v_nombre:= producto.get_Nombre;
       v_precio:= FloatToStr(producto.get_Precio);
         IUSR_03.ct_clave_producto.Text:=v_clave;
         IUSR_03.ct_nombre_producto.Text:=v_nombre;
         IUSR_03.ct_precio_producto.Text:=v_precio;
         producto.Free;
       end else
       begin
         mensaje.set_mensaje('El producto no ha sido encontrado');
       mensaje.mostrar_mensaje;
         IUSR_03.ct_clave_producto.Text:='';
         IUSR_03.ct_nombre_producto.Text:='';
         IUSR_03.ct_precio_producto.Text:='';
       end;
     end else
```

```
     begin
       mensaje.set_mensaje('Verifica que la clave del producto que
estás buscando tenga 4 dígitos.');
       mensaje.mostrar_mensaje;
     end;
  end else
  begin
    mensaje.set_mensaje('Para buscar un producto ingresa la clave del
producto en el campo ubicado en la parte izquierda.');
    mensaje.mostrar_mensaje;
  end;
  mensaje.Free;
  gestor.Free;
end;

procedure TIUSR_03.evt_calcular_importe(Sender: TObject);
var
  producto_ld: CL_PRODUCTO;
  linead: CL_LINEA_DETALLE;
  mensaje: CL_VISTA_MENSAJE;
  v_precio,v_unidades: String;
  v_subtotal: Real;
begin
  mensaje:=CL_VISTA_MENSAJE.Create;
  v_precio := IUSR_03.ct_precio_producto.Text;
  v_unidades := IUSR_03.ct_unidades_adquiridas.Text;
  If NOT(v_precio='') then
  begin
     If NOT(v_unidades='') then
     begin
        producto_ld:=CL_PRODUCTO.Create(0,'',StrToFloat(v_precio));
      linead:=CL_LINEA_DETALLE.Create(0,StrToInt(v_unidades),producto_
ld);
      linead.calcular_subtotal;
      v_subtotal := linead.get_Subtotal;
      IUSR_03.ct_importe_venta.Text:=FloatToStr(v_subtotal);
      producto_ld.Free;
      linead.Free;
     end else
     begin
        mensaje.set_mensaje('Para calcular el importe debes ingresar
las unidades a adquirir del producto encontrado.');
      mensaje.mostrar_mensaje;
     end;
  end else
  begin
     mensaje.set_mensaje('Para poder calcular el importe debes primero
haber buscado y encontrado un producto anteriormente.');
     mensaje.mostrar_mensaje;
  end;
  mensaje.Free;
end;

procedure TIUSR_03.evt_calcular_total(Sender: TObject);
```

```
var
  v_total_venta:Real;
begin
  v_total_venta:=obj_venta.get_Total;
  IUSR_03.ct_total_venta.Text:=FloatToStr(v_total_venta);
end;

procedure TIUSR_03.evt_cerrar(Sender: TObject);
begin
  IUSR_03.Close;
end;

procedure TIUSR_03.evt_confirmar_venta(Sender: TObject);
var
  v_detalle: CL_LINEA_DETALLE;
  producto: CL_PRODUCTO;
  mensaje: CL_VISTA_MENSAJE;
  v_clave,v_nombre,v_precio,v_unidades:String;
  v_conf:Integer;
  v_subtotal:Real;
begin
  mensaje:=CL_VISTA_MENSAJE.Create;
  v_clave:=IUSR_03.ct_clave_producto.Text;
  v_nombre:=IUSR_03.ct_nombre_producto.Text;
  v_precio:=IUSR_03.ct_precio_producto.Text;
  v_unidades:=IUSR_03.ct_unidades_adquiridas.Text;
  If
Not((v_clave='')and(v_nombre='')and(v_precio='')and(v_unidades=''))
then
  begin
     ICONF_01.lbl_pregunta.Caption:='Estas seguro que deseas confirmar
la venta de:'+LineEnding+
     v_unidades+' unidades de '+v_nombre+' de '+v_precio+'$ c/u';
     v_conf:=ICONF_01.ShowModal;
     If v_conf=mrYes then
     begin
        ICONF_01.Close;
        numero_linea_detalle:=numero_linea_detalle+1;

producto:=CL_PRODUCTO.Create(StrToInt(v_clave),v_nombre,StrToFloat(v_p
recio));

v_detalle:=CL_LINEA_DETALLE.Create(numero_linea_detalle,StrToInt(v_uni
dades),producto);
        v_detalle.calcular_subtotal;
        v_Detalles[numero_linea_detalle]:=v_detalle;
        v_subtotal:=v_detalle.get_Subtotal;
        obj_venta.calcular_total(v_subtotal);
        IUSR_03.ct_clave_producto.Text:='';
        IUSR_03.ct_nombre_producto.Text:='';
        IUSR_03.ct_precio_producto.Text:='';
        IUSR_03.ct_unidades_adquiridas.Text:='';
        producto.Free;
        v_detalle.Free;
```

```
     end else
     begin
        ICONF_01.Close;
     end;
  end else
  begin
     mensaje.set_mensaje('Para poder confirmar una venta asegurate de
buscar un producto anteriormente y colocar las unidades a adquirir del
mismo.');
     mensaje.mostrar_mensaje;
  end;
  mensaje.Free;
end;

procedure TIUSR_03.evt_terminar_venta(Sender: TObject);
var
  gestor_venta: GESTOR_ARCH_VENTA;
  mensaje: CL_VISTA_MENSAJE;
  numero_venta: String;
  v_conf,i: Integer;
  v_total_venta: Real;
begin
  gestor_venta:=GESTOR_ARCH_VENTA.Create;
  mensaje:=CL_VISTA_MENSAJE.Create;
  numero_venta:=IUSR_03.ct_clave_venta.Text;
  v_total_venta:=obj_venta.get_Total;
  ICONF_01.lbl_pregunta.Caption:='¿Estas seguro que deseas terminar la
venta No. '+numero_venta+LineEnding+
  'con total de $'+FloatToStr(v_total_venta)+' mxn';
  v_conf:=ICONF_01.ShowModal;
  If v_conf=mrYes then
  begin
     ICONF_01.Close;
     obj_venta.set_Detalles(v_Detalles);
     gestor_venta.guardar_venta(obj_venta);
     mensaje.set_mensaje('Venta finalizada correctamente');
     mensaje.mostrar_mensaje;
     IUSR_03.ct_clave_buscar_prod.Text:='';
     IUSR_03.ct_clave_producto.Text:='';
     IUSR_03.ct_nombre_producto.Text:='';
     IUSR_03.ct_precio_producto.Text:='';
     IUSR_03.ct_unidades_adquiridas.Text:='';
     IUSR_03.ct_importe_venta.Text:='';
     IUSR_03.ct_total_venta.Text:='';
     IUSR_03.Close;
     gestor_venta.Free;
     obj_venta.Free;
     For i:=1 to 10 do
     begin
        v_Detalles[i]:=CL_LINEA_DETALLE.Create(0,0,NIL);
        v_Detalles[i].Free;
     end;
  end else
  begin
```

```
     ICONF_01.Close;
  end;
end;

procedure TIUSR_03.iniciar_venta(Sender: TObject);
var
  gestor_venta: GESTOR_ARCH_VENTA;
  v_fecha: TDate;
  numero_venta: Integer;
begin
  gestor_venta:=GESTOR_ARCH_VENTA.Create;
  numero_venta:=gestor_venta.obtener_numero_venta;
  numero_venta:=numero_venta+1;
  v_fecha:=Date;
  obj_venta:=CL_VENTA.Create;
  obj_venta.set_Numero(numero_venta);
  obj_venta.set_Fecha(DateToStr(v_fecha));
  obj_venta.set_Total(0);
  IUSR_03.lbl_fecha.Caption:=DateToStr(v_fecha);
  IUSR_03.ct_clave_venta.Text:=IntToStr(numero_venta);
  numero_linea_detalle:=0;
end;

end.
```

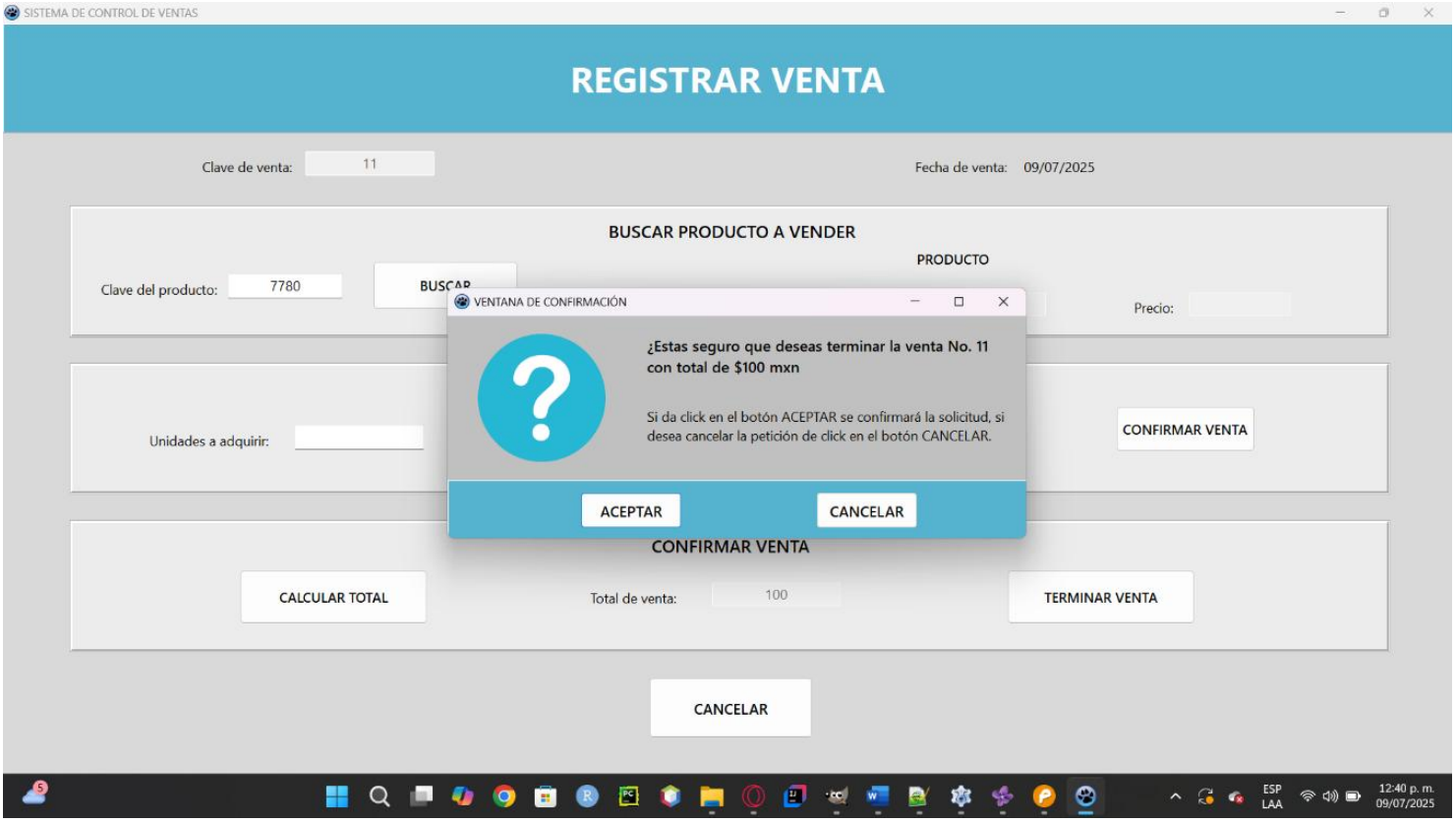
### 9.10. CL_IUSR_04.pas

```
unit CL_IUSR_04;

{$mode ObjFPC}{$H+}

interface

uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls,
StdCtrls,
  Buttons, U_CL_VISTA_MENSAJE,U_GESTOR_ARCH_VENTA;

type

  { TIUSR_04 }

  TIUSR_04 = class(TForm)
    btn_reporte_diario: TButton;
    btn_reporte_mensual: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    lbl_fecha: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Panel1: TPanel;
```

```
    btn_cerrar: TSpeedButton;
    procedure evt_cerrar(Sender: TObject);
    procedure evt_generar_reporte_diario(Sender: TObject);
    procedure evt_generar_reporte_mensual(Sender: TObject);
    procedure evt_mostrar_fecha(Sender: TObject);
  private

  public

  end;

var
  IUSR_04: TIUSR_04;

implementation

{$R *.lfm}

{ TIUSR_04 }

procedure TIUSR_04.evt_cerrar(Sender: TObject);
begin
  IUSR_04.Close;
end;

procedure TIUSR_04.evt_generar_reporte_diario(Sender: TObject);
var
  v_fecha:TDate;
  gestor_venta: GESTOR_ARCH_VENTA;
  mensaje: CL_VISTA_MENSAJE;
begin
  v_fecha:=Date;
  gestor_venta:=GESTOR_ARCH_VENTA.Create;
  gestor_venta.generar_reporte_ventas_dia(DateToStr(v_fecha));
  mensaje:=CL_VISTA_MENSAJE.Create;
  mensaje.set_mensaje('Se ha generado satisfactoriamente el reporte de
ventas del día: '+DateToStr(v_fecha));
  mensaje.mostrar_mensaje;
end;

procedure TIUSR_04.evt_generar_reporte_mensual(Sender: TObject);
var
  v_fecha:TDate;
  v_mes: String;
  gestor_venta: GESTOR_ARCH_VENTA;
  mensaje: CL_VISTA_MENSAJE;
begin
  v_fecha:=Date;
  v_mes:= Copy(DateToStr(v_fecha),4,7);
  gestor_venta:=GESTOR_ARCH_VENTA.Create;
  gestor_venta.generar_reporte_ventas_mes(v_mes);
  mensaje:=CL_VISTA_MENSAJE.Create;
  mensaje.set_mensaje('Se ha generado satisfactoriamente el reporte de
ventas del mes: '+v_mes);
```

```
  mensaje.mostrar_mensaje;
end;

procedure TIUSR_04.evt_mostrar_fecha(Sender: TObject);
var
  v_fecha: TDate;
begin
  v_fecha:=Date;
  IUSR_04.lbl_fecha.Caption:=DateToStr(v_fecha);
end;

end.
```

## 9.11. EJECUCIÓN PASCAL



SISTEMA DE CONTROL DE VENTAS
REGISTRAR VENTA
Clave de venta: 11
Fecha de venta: 09/07/2025
BUSCAR PRODUCTO A VENDER
PRODUCTO
Clave del producto: 7780
BUSCAR
Precio:
VENTANA DE CONFIRMACIÓN
¿Estas seguro que deseas terminar la venta No. 11 con total de $100 mxn
Si da click en el botón ACEPTAR se confirmará la solicitud, si desea cancelar la petición de click en el botón CANCELAR.
ACEPTAR
CANCELAR
Unidades a adquirir:
CONFIRMAR VENTA
CONFIRMAR VENTA
CALCULAR TOTAL
Total de venta: 100
TERMINAR VENTA
CANCELAR
ESP LAA
12:40 p. m. 09/07/2025


SISTEMA DE CONTROL DE VENTAS
REPORTES DE VENTAS
MÓDULO DE GENERACIÓN DE REPORTES DE VENTA DIARIOS Y MENSUALES
INDICACIONES
Fecha: 09/07/2025
1.- Elige el tipo de reporte que deseas generar (Diario/Mensual).
2.- Al dar click sobre el botón correspontiente a tu eleccións generará un archivo de texto el cuál contiene la información de las ventas diarias o mensuales según sea tu elección.
3.- El archivo de texto generado se guardará en la carpeta llamada 'REPORTES' dentro de la misma ruta de acceso a este programa.
REPORTE DIARIO
REPORTE MENSUAL
ESP LAA
12:40 p. m. 09/07/2025


**Reporte de venta del mes de Julio de 2025**

**10. ¿CÓMO FUNCIONA?**

Este producto de software tiene un menú principal dónde podemos observar 4 botones, al hacer click en el botón gestión de productos se dispara un evento el cual genera las acciones necesarias para mostrar la interfaz IUSR_02, en la cuál encontramos el módulo de la aplicación dónde podemos realizar la gestón de los productos de la tienda, podemos agregar productos, consultar algún producto por medio de la clave, modificar un producto ya existente, o eliminarlo. De igual manera en la parte inferior derecha podemos encontrar información sobre el archivo dónde se almacenan permanentemente los productos, en la cuál nos muestra el porcentaje utilizado lógicamente en el archivo y de igual manera el uso físico del archivo, al existir una diferencia de 50% o más nos debe de sugerir realizar un borrado físico dentro del archivo. AL hacer click en el botón de registrar venta nos muestra la interfaz IUSR_03 en la cuál se nos permite realizar ventas al menudeo, la interfza nos muestra la fecha actual del sistema, nos muestra la clave de venta la cuál es un contador que incrementa en 1 con cada venta registrada, para registrar una venta tienes que buscar un producto, ingresar las unidades a adquirir, confirmar la venta y podrías repetir este proceso dependiendo de la cantidad de productos a vender y al finalizar tendrías que terminar la venta para que esta sea

registrada. La interfaz de la generación de reportes nos muestra las indicaciones y la fecha actual del sistema.  Este nos permite generar un reporte de venta del día actual o un reporte de venta del mes actual. Estos reportes se generan en una carpeta llamada ‘REPORTES’ dentro de los archivos fuente del proyecto, se genera un archivo de texto con las ventas del día o del mes según sea la elección del usuario. El botón ‘Terminar Ejecución’ simplemente termina con la ejecución del programa.