

# Universidad de los Andes

FACULTAD DE INGENIERÍA  
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS



## ENTREGA 2: PROYECTO 2

Jesús Alberto Jiménez Garizao  
Juan Diego Yepes Parra

Noviembre 16 de 2021  
Bogotá D.C.

# Contenido

|   |           |
|---|-----------|
| <b>1 Caso de estudio</b>  | <b>2</b>  |
| <b>2 Modelo</b>   | <b>2</b>  |
| 2.1 Roles del sistema Inventario . . . . .                        | 2         |
| 2.1.1 Dominio del sistema . . . . .                               | 2         |
| 2.1.2 Lo que hace el sistema . . . . .                            | 2         |
| 2.1.3 Elementos que fluyen . . . . .                              | 2         |
| 2.1.4 Agrupadores . . . . .                                       | 2         |
| 2.2 Roles del sistema POS . . . . .                               | 2         |
| 2.2.1 Dominio del sistema . . . . .                               | 2         |
| 2.2.2 Elementos que fluyen . . . . .                              | 3         |
| 2.2.3 Agrupadores . . . . .                                       | 3         |
| 2.3 Responsabilidades . . . . .                                   | 3         |
| 2.3.1 Inventario . . . . .  | 3         |
| 2.3.2 POS . . . . .   | 4         |
| 2.4 Colaboraciones . . . . .                                      | 6         |
| 2.5 Reflexión . . . . .   | 12        |
| 2.5.1 Sobre el diseño obtenido . . . . .                          | 12        |
| 2.5.2 Sobre los cambios con respecto al diseño original . . . . . | 13        |
| <b>3 Interfaz</b>   | <b>13</b> |
| 3.1 Producto Final: INVENTARIO . . . . .                          | 13        |
| 3.2 Producto Final: POS . . . . .                                 | 22        |
| 3.3 Diagramas de diseño . . . . .                                 | 31        |
| 3.3.1 Explicación de las interfaces . . . . .                     | 44        |
| 3.4 Reflexión . . . . .   | 45        |

# 1 Caso de estudio

Dentro del proyecto del curso vamos a construir un conjunto de aplicaciones para manejar el inventario y las ventas de un supermercado. Es decir que la entrega del proyecto tendrá que incluir necesariamente dos aplicaciones: el sistema **POS** (point-of-sale) que usará un cajero para registrar las compras de un cliente y el sistema **INVENTARIO** que usarán los encargados de manejar el inventario del supermercado.

Para este proyecto, nuestro objetivo es hacer una interfaz gráfica del proyecto 1.

## 2 Modelo

### 2.1 Roles del sistema Inventario

Los objetos que se mencionan a continuación están divididos de acuerdo a estas secciones. Cada objeto tiene su respectivo estereotipo.

#### 2.1.1 Dominio del sistema

- Producto <<Information Holder>>

#### 2.1.2 Lo que hace el sistema

- Inventario <<structurer>>

#### 2.1.3 Elementos que fluyen

- Lote <<Information Holder>>

#### 2.1.4 Agrupadores

- Aplicación <<coordinator>>
- FileManager <<interfacer>>

### 2.2 Roles del sistema POS

Los objetos que se mencionan a continuación están divididos de acuerdo a estas secciones. Cada objeto tiene su respectivo estereotipo.

#### 2.2.1 Dominio del sistema

- Compra <<Information Holder>>
- Cliente <<Information Holder>>
- Producto <<Information Holder>>

### **2.2.2 Elementos que fluyen**

- Inventario <<Service provider>>

### **2.2.3 Agrupadores**

- POS <<structurer>>
- Aplicación <<coordinator>>
- FileManager <<interfacer>>

## **2.3 Responsabilidades**

A continuación se enumeran responsabilidades y a qué objeto le corresponden

### **2.3.1 Inventario**

#### **Interfaz gráfica**

- Interacción con el usuario

#### **Módulo Inventario**

- Estructurar el programa, es decir llamar otras clases para ejecutar colaboraciones
- Cargar los archivos
- Modificar los archivos .csv

#### **Lote**

Este componente es solo útil para consultar información del mismo, luego su responsabilidad debe ser:

- Guardar y modificar información del lote

```
1 private String fechaVencimiento;
2 private String fechaRecibido;
3 private int precioCompra;
4 private int precioVenta;
5 private int unidades;
6 private int unidadesRestantes;
7 private Producto productoAsociado; // Objeto producto asociado al lote
8 private boolean loteEliminado; // Determina si el lote ha sido eliminado
```

#### **Producto**

Este componente es solo útil para consultar información del mismo, luego su responsabilidad debe ser:

- Guardar y modificar la información del producto

```

1 private int codigoBarras;
2 private String nombre;
3 private int numUnidadesMedida;
4 private String unidadMedida; // Determinante de la unidad de medida (e.g. Gramos,
   Litros)
5 private Categoria categoria; // Objeto categoria a la que pertenece
6 private String almacenamiento; // Gondola a la que pertenece
7 private boolean empacado; // Determina si esta empacado o no

```

### Categoría

Este componente es solo útil para consultar información del mismo, luego su responsabilidad debe ser:

- Guardar y modificar información de la categoría

```

1 private String nombre;
2 private Categoria superCategoria = null;

```

Este componente funciona de manera recursiva. Para hallar la supercategoría de una categoría, solo se revisa su atributo, sin embargo, si se quisiera hallar un nivel superior de supercategoría, se haría el llamado hacia la misma categoría, para hallar su supercategoría; las n veces necesarias.

### 2.3.2 POS

Para efectos del funcionamiento de la aplicación POS, se asignan las siguientes responsabilidades:

#### Interfaz Gráfica

- Interacción con el usuario

#### Módulo POS

- Estructurar el programa, es decir llamar otras clases para ejecutar colaboraciones
- Cargar los archivos
- Modificar los archivos .csv

#### Compra

Este componente se inicializará para cada nueva compra, y generará su propia factura para cada vez que sea ejecutado, luego sus responsabilidades deben ser:

- Hacer una compra
- Guardar factura de compra
- Guardar puntos que acumula la compra

```

1 Inventario inventario; // Llamado al componente inventario
2 ArrayList<Producto> productos = new ArrayList<Producto>(); // Lista de productos
   disponibles
3 int total = 0; // Total de compra
4 int puntos = 0; // Puntos de compra
5 int id_compra; // Identificador para generar factura

```

### Cliente

Este componente es solo útil para consultar información del mismo, luego su responsabilidad debe ser:

- Guardar y modificar información del cliente

```

1 private int cedula;
2 private String nombres;
3 private String apellidos;
4 private int edad;
5 private String idenGenero; // Decidimos denominar identidad de genero y no sexo porque
   no nos interesan los genitales del cliente sino como se identifica como persona
6 private String situacionLaboral;
7 public int puntos;

```

### Producto

Este componente es solo útil para consultar información del mismo, luego su responsabilidad debe ser:

- Guardar y modificar la información del producto

```

1 private int codigoBarras;
2 private String nombre;
3 private int numUnidadesMedida;
4 private String unidadMedida; // Determinante de la unidad de medida (e.g. Gramos,
   Litros)
5 private Categoria categoria; // Objeto categoria a la que pertenece
6 private String almacenamiento; // Gondola a la que pertenece
7 private boolean empacado; // Determina si esta empacado o no

```

### Inventario

Este componente es compartido con el sistema Inventario, es su módulo, luego a continuación solo se mencionan las responsabilidades estrictamente necesarias para el funcionamiento del sistema POS.

- Cargar toda la información del supermercado, los productos, los lotes y las categorías
- Modificar toda la información del supermercado, los productos, los lotes y las categorías

### FileManager

Este componente funciona sobretodo gracias a las colaboraciones que se describen después, sin embargo su responsabilidad principal es:

- Modificar y crear en archivo .csv

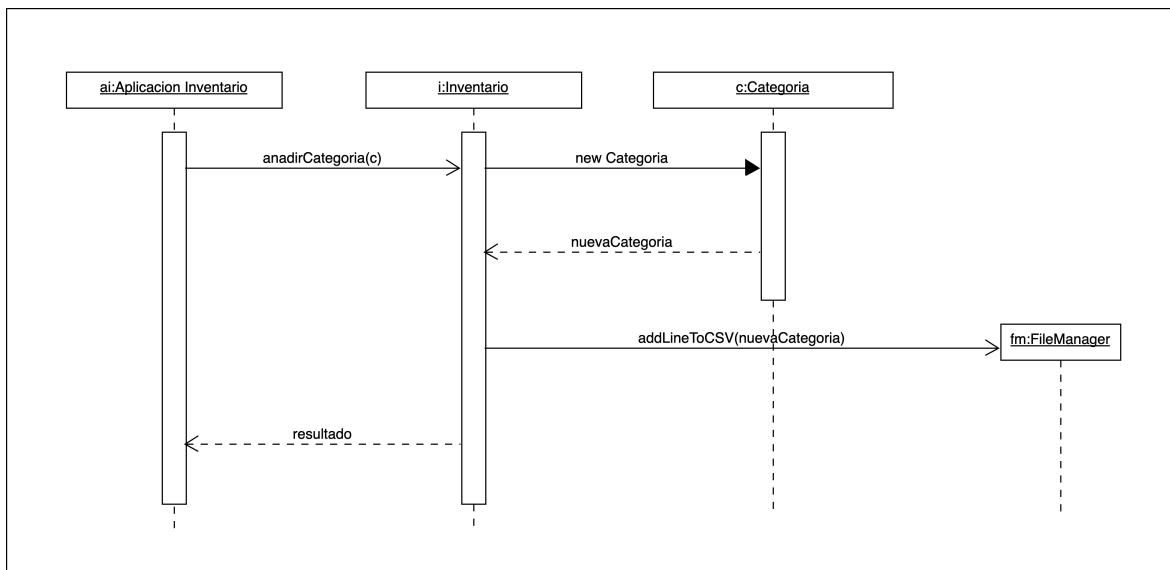
## 2.4 Colaboraciones

Para cumplir con todas las responsabilidades enmarcadas en la anterior sección se tuvo que tener en cuenta las siguientes colaboraciones para algunas cumplir estas funciones.

### Añadir categoría

El inventario luego de recibir la instrucción de añadir una categoría al supermercado por parte de la aplicación,

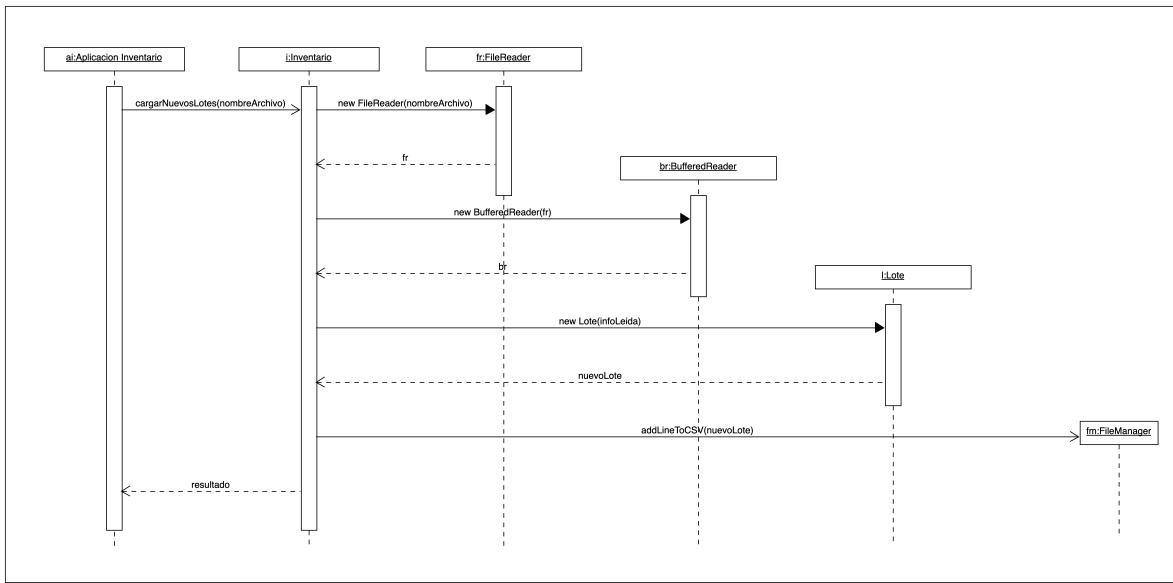
- Crea una nueva categoría a partir del nombre y la super categoría de la misma.
- Guarda esta categoría en la lista de categorías del inventario.
- Añade la categoria al archivo categorias.csv por medio del File Manager.



### Carga nuevo lote de productos al inventario

El inventario luego de recibir la instrucción de carga de un nuevo lote en el inventario,

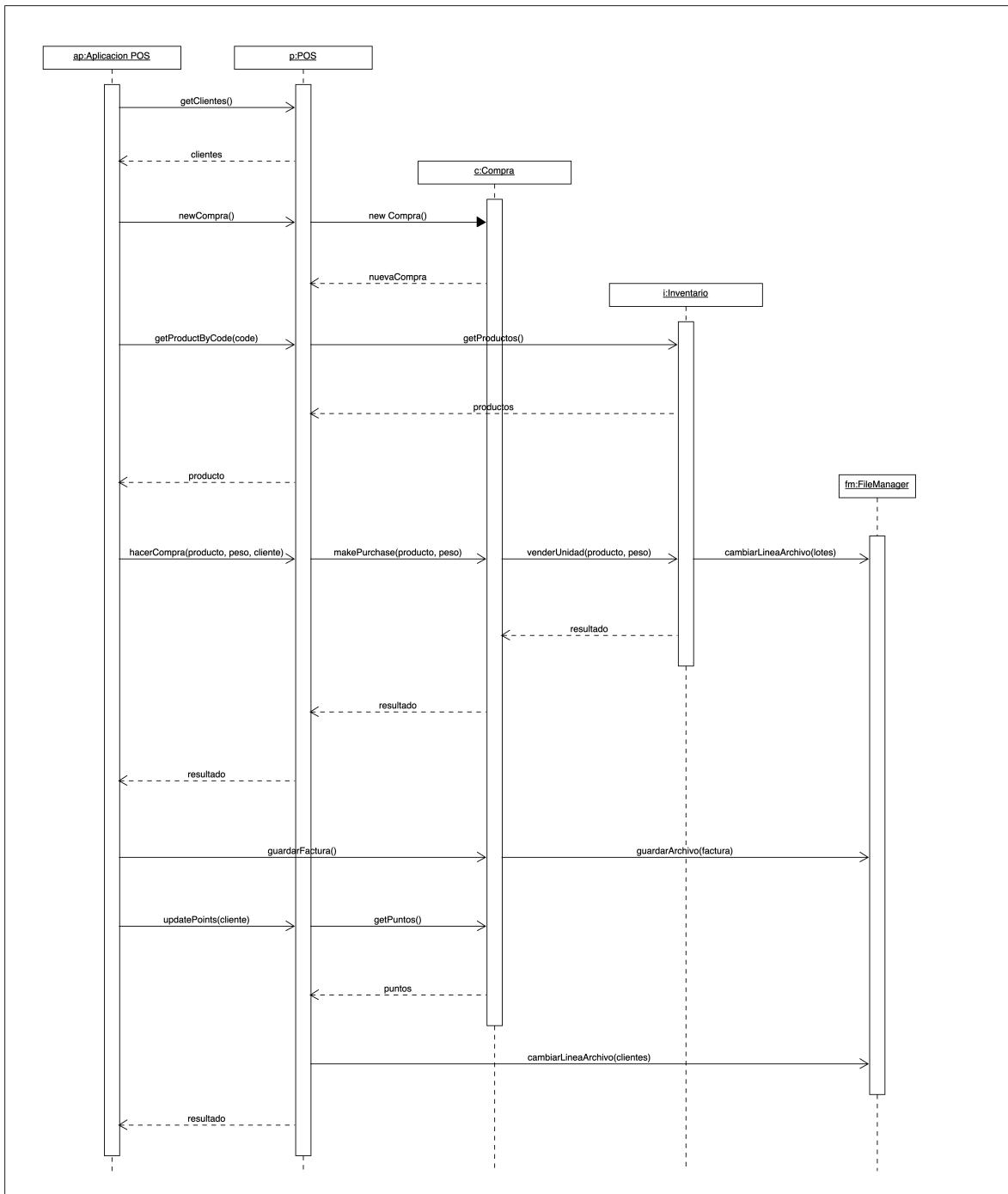
- Tiene que crear un File Reader pasándole el nombre del archivo al constructor del mismo.
- Crea un BufferedReader por medio del File Reader creado anteriormente.
- Lee el archivo y crea un nuevo lote con la información leída.
- Añade el lote al archivo lotes.csv por medio del File Manager.



### Realizar una compra en POS

La aplicación al ejecutar la opción de realizar una compra:

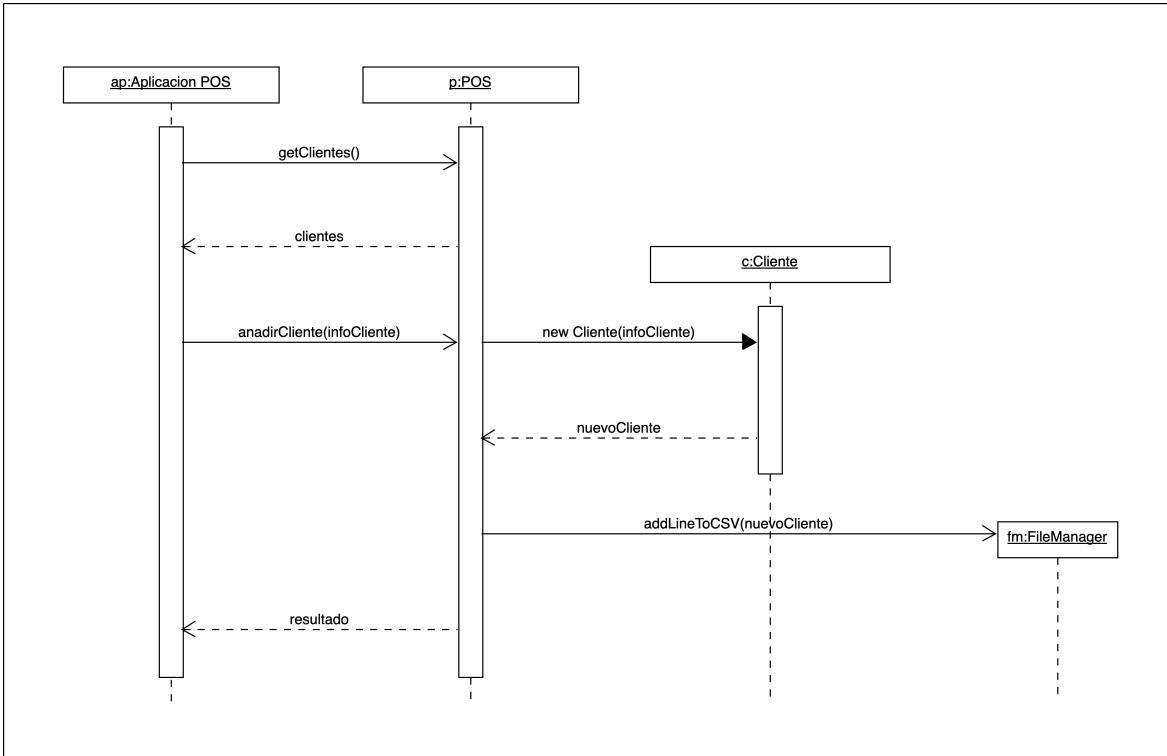
- Tiene que obtener los clientes para luego verificar si la persona que comprará obtendrá puntos o no.
- Iniciará una nueva compra.
- Escanea el código de un producto y obtiene los datos del mismo por medio del inventario.
- Realiza la compra verificando si hay stock del producto y dependiendo de eso el inventario editará el archivo de lotes por medio del File Manager para cambiar el número de unidades que ahora tenemos del producto.
- Le pide a la compra que guarde la factura por medio del File Manager.
- Le pide a la compra los puntos generados por la misma y los añade al cliente.
- Guarda los nuevos puntos del cliente al archivo clientes.csv.
- La compra termina.



## Registrar un nuevo cliente en el sistema

La aplicación al ejecutar la opción de registrar un cliente:

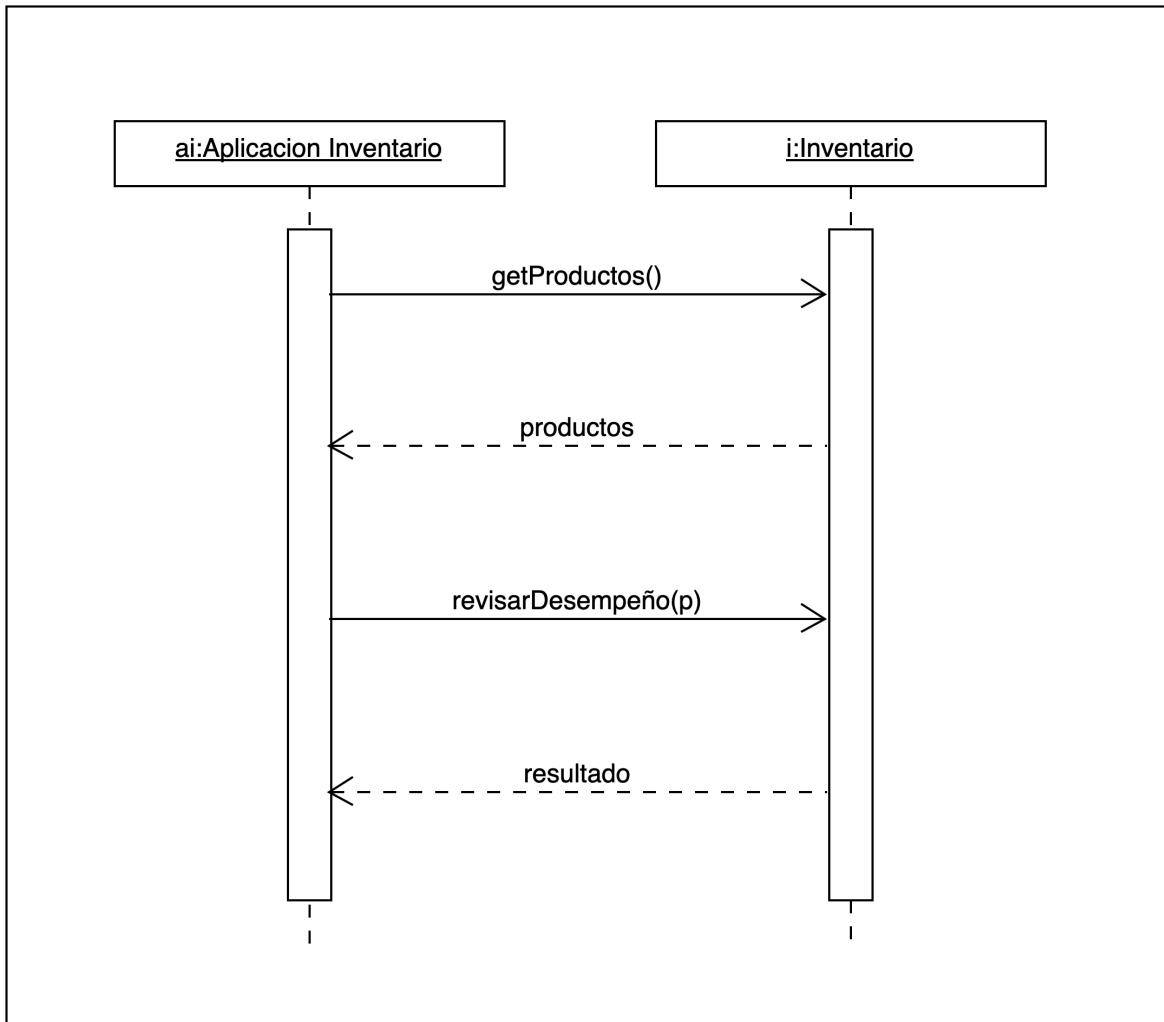
- Obtiene los clientes para verificar que la cédula no se encuentre ya registrada.
- Le pasa la información del cliente al POS y este crea un nuevo cliente con la misma.
- Añade el cliente al archivo clientes.csv por medio del File Manager.



## Revisar desempeño de un producto

La aplicación del inventario al recibir la instrucción de ver el desempeño de un producto:

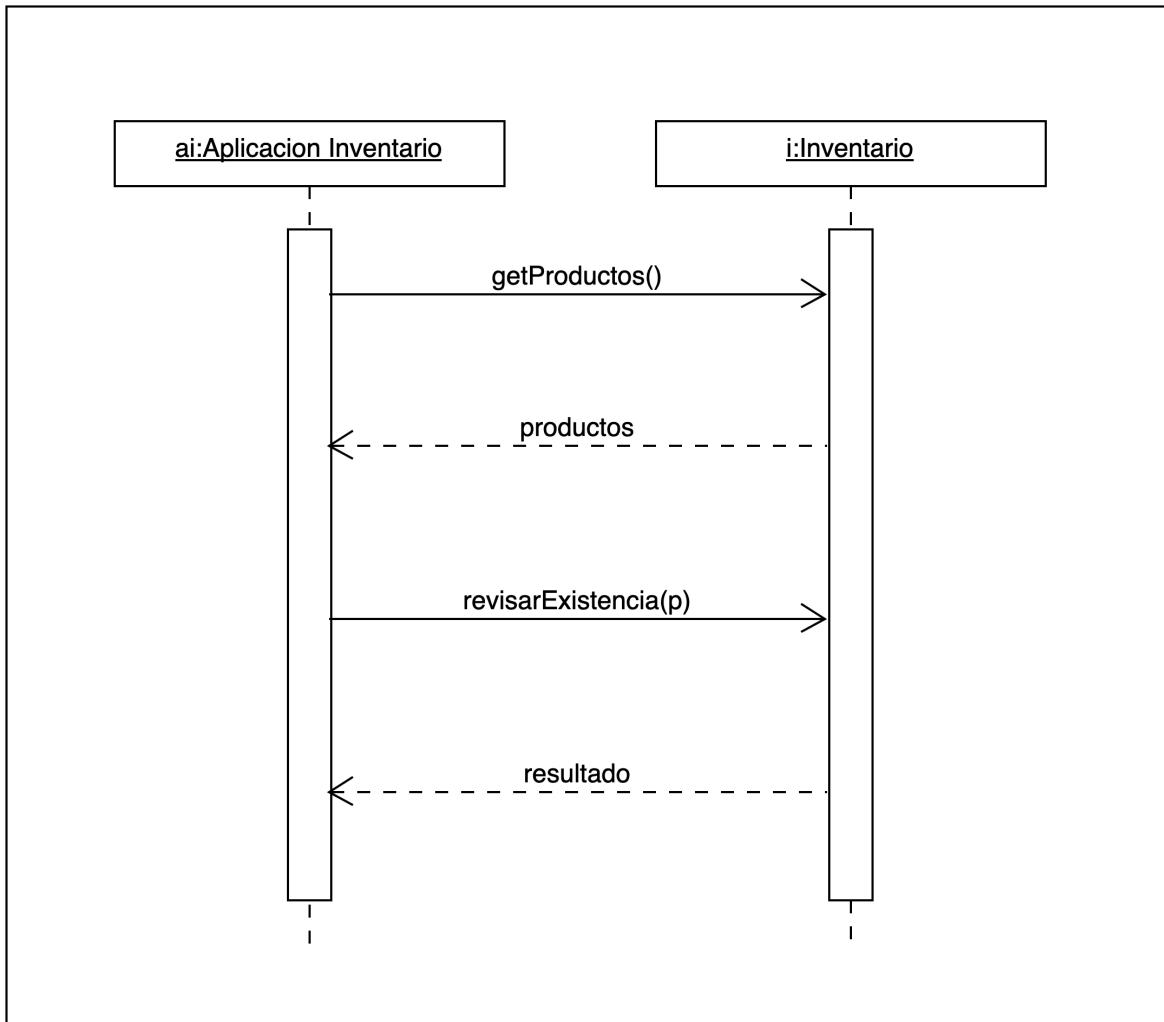
- Obtiene todos los productos del inventario.
- Enseña los productos al usuario para que elija uno.
- Le pide al inventario que chequee el desempeño del producto elegido.
- El inventario calcula las ganancias por la diferencia de las unidades restantes y las unidades originales de un lote específico y las perdidas a través de las unidades restantes de un lote que ha sido eliminado.
- Le muestra al usuario las ganancias y perdidas generadas por el producto escogido.



### Revisar existencias de un producto

La aplicación del inventario al recibir la instrucción de revisar las existencias de un producto:

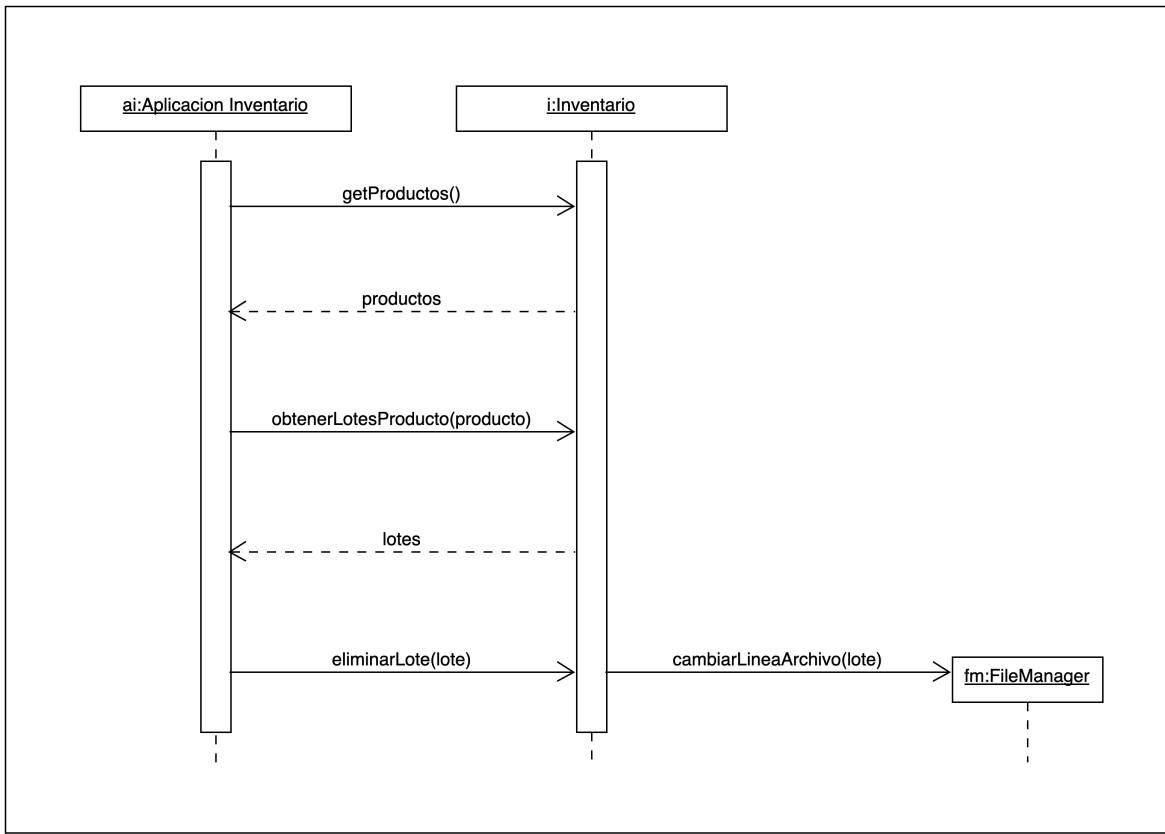
- Obtiene todos los productos del inventario.
- Enseña los productos al usuario para que elija uno.
- Le pide al inventario que chequee las existencias del producto elegido.
- El inventario calcula las existencias de los productos por medio de los lotes que no han sido eliminados y de las unidades restantes y lo retorna a la aplicación.
- Le muestra al usuario las cuantas unidades quedan por el producto escogido.



#### **Eliminar lote de un producto**

La aplicación del inventario al recibir la instrucción de eliminar un lote de un producto en todo el sistema,

- Obtiene todos los productos del inventario.
- Enseña los productos al usuario para que elija uno.
- Le pide al inventario que chequee los lotes del producto elegido.
- El inventario retorna los lotes que existen para el producto escogido.
- La aplicación el pide al inventario que elimine el lote escogido por el usuario.
- El inventario cambia el atributo LoteEliminado del lote almacenado en la lista lotes y modifica el lote del archivo lotes.csv por medio del File Manager.



## 2.5 Reflexión

### 2.5.1 Sobre el diseño obtenido

Teniendo en cuenta que hay funcionalidades del aplicativo que no se ven todavía bien reflejadas en la vida real, y que no son muy detalladas en el enunciado, como el separar los productos por góndolas y por categorías, o el sistema de puntos de los clientes, nuestro sistema maneja estas opciones de manera muy superficial, porque no pudimos encontrar más cosas para agregarle.

Por otro lado, nuestro diseño tiene algunas limitaciones, como que todo lo que guardamos está almacenado en archivos .csv lo que significa que se pueden modificar por fuera del sistema; o que nuestro sistema no fue diseñado con el propósito de poder expandirse en un futuro, por ende nuestro diseño no es muy flexible.

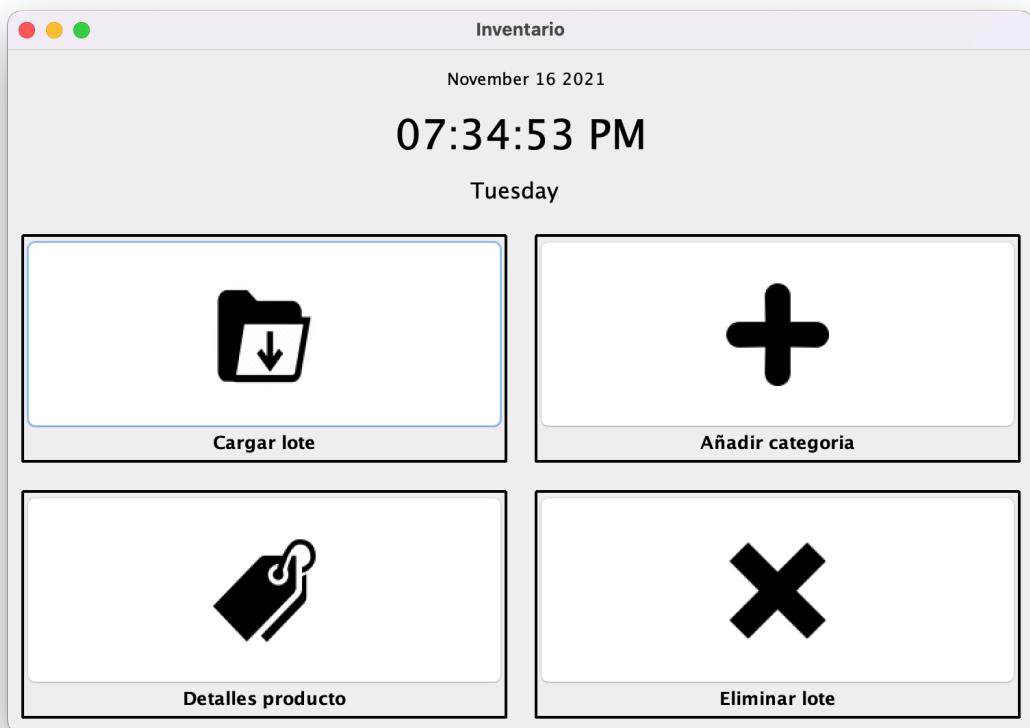
No obstante, hay varios pros en nuestro diseño, y es que si se quieren hacer modificaciones que no afecten la estructura de los sistemas, como por ejemplo agregar opciones a las clases preexistentes, esto es muy fácil de hacer, luego nuestro sistema tiene una adaptabilidad buena.

### 2.5.2 Sobre los cambios con respecto al diseño original

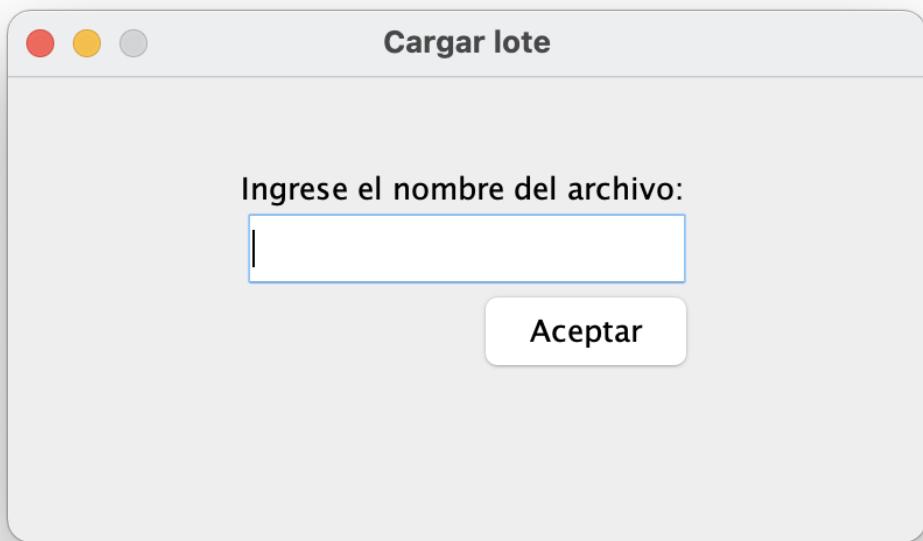
Teniendo en cuenta las nuevas restricciones del diseño de la interfaz gráfica, tuvimos que cambiar las clases: FileManager, y la razón de esto es que tuvimos que empezar a guardar información que antes no guardábamos, como la fecha en que un cliente hizo compras, etc.

## 3 Interfaz

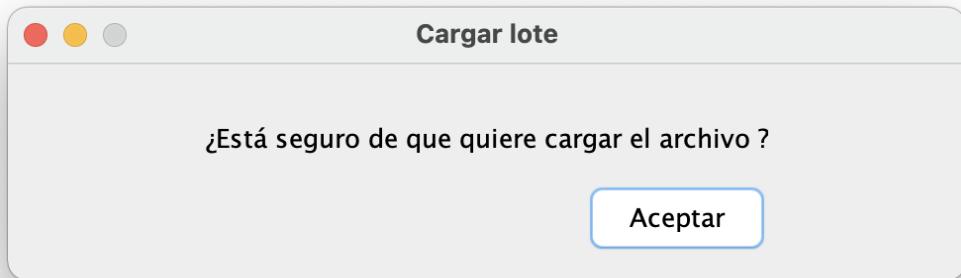
### 3.1 Producto Final: INVENTARIO



Si el usuario presiona **Cargar Lote** se abre la siguiente ventana

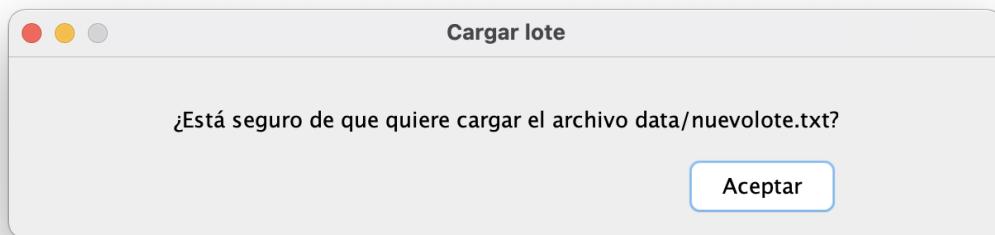


Si se digita un archivo vacío o que no existe



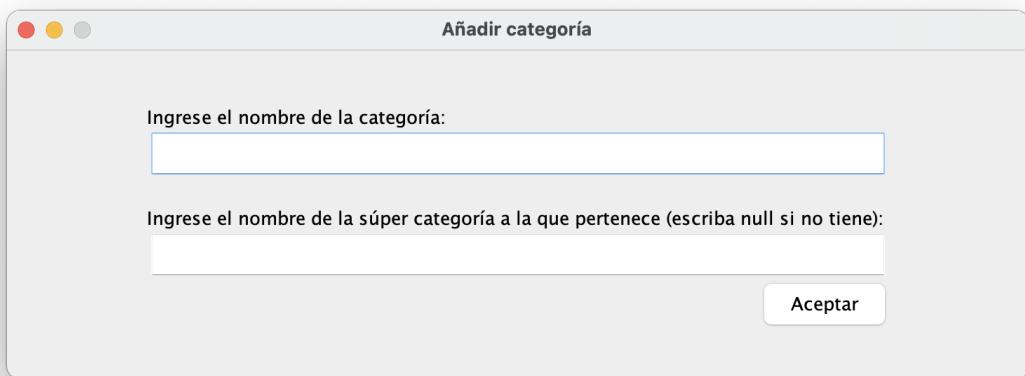


Si se digita un archivo correcto

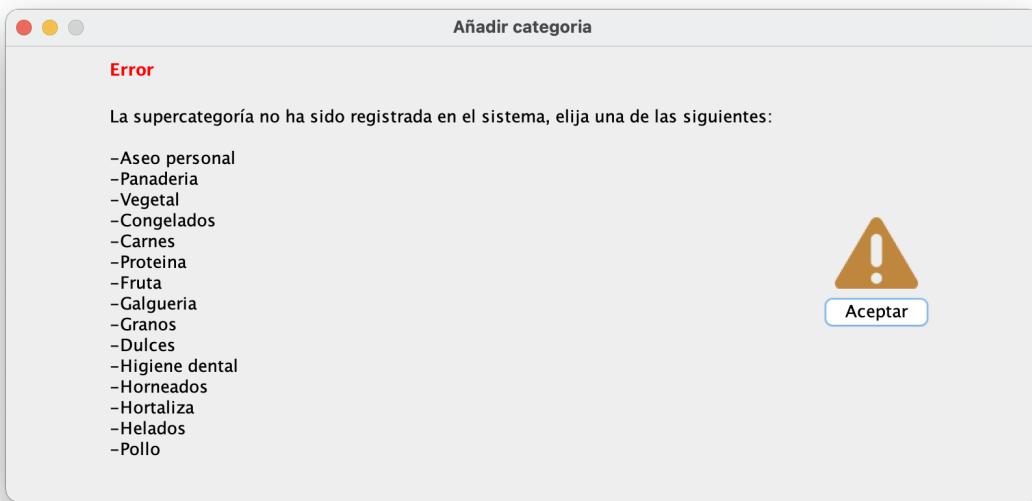




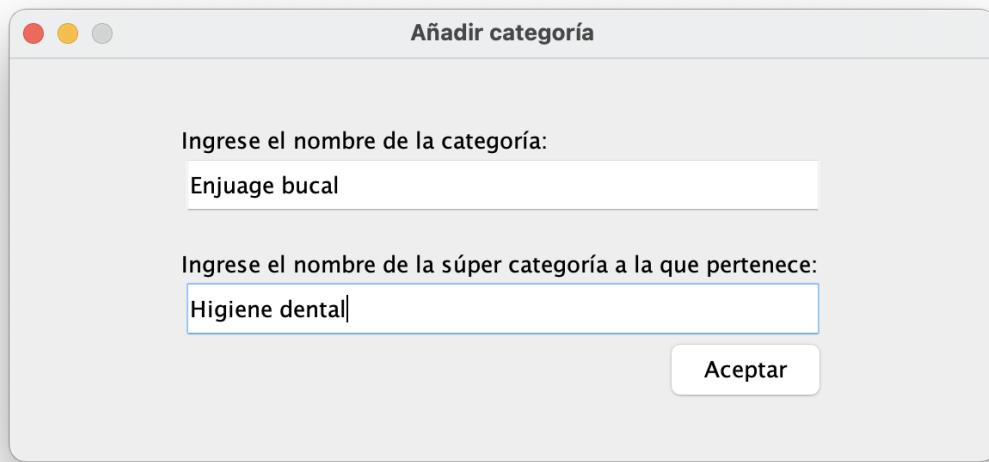
Si el usuario presiona **Añadir Categoría** se abre la siguiente ventana

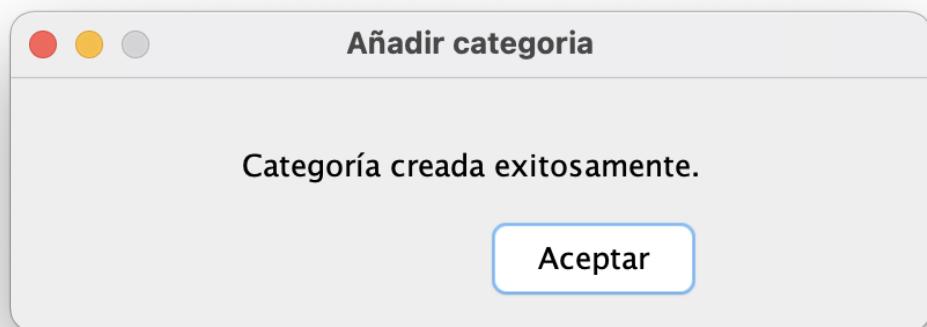


Si se intenta agregar una categoría que no tiene una supercategoría preexistente



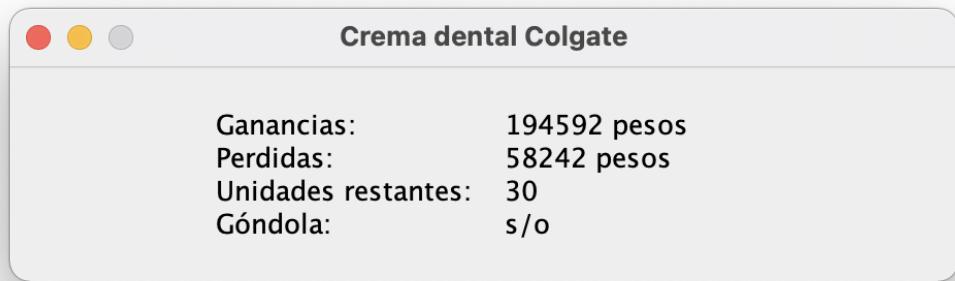
Si se intenta agregar una categoría que sí tiene una supercategoría preexistente



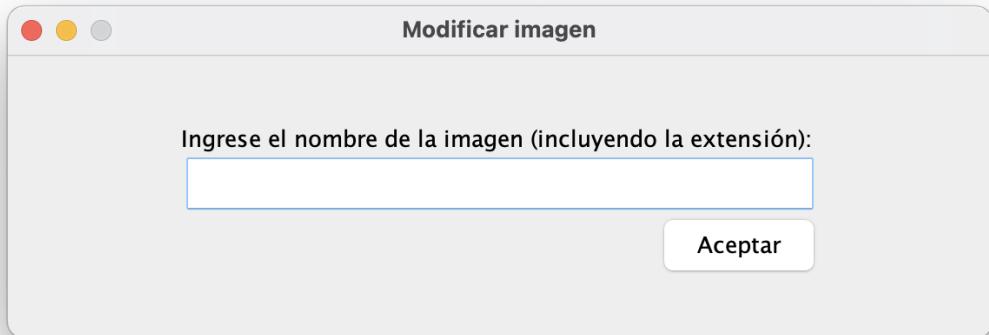


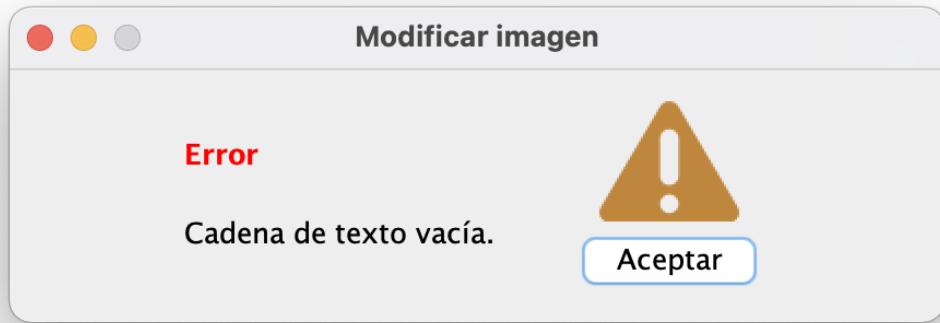
Si el usuario presiona **Detalles Producto** se abre la siguiente ventana



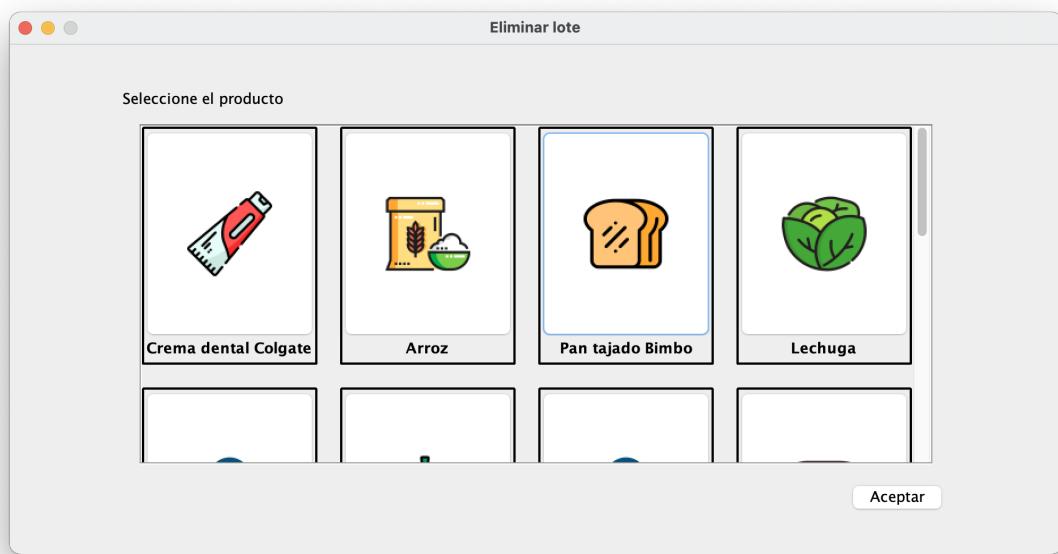


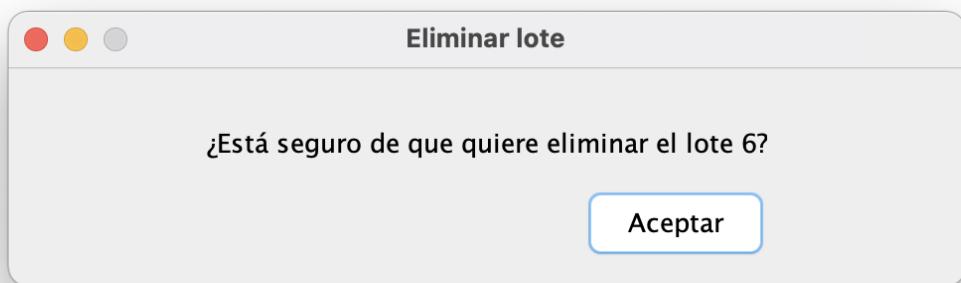
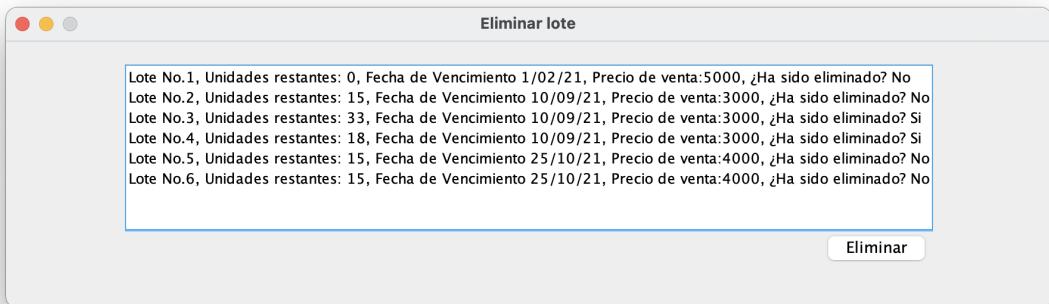
Si se intenta agregar una imagen y no se le pasa texto

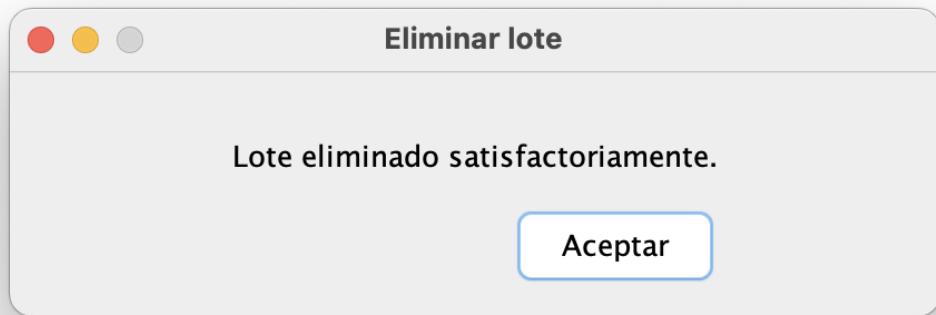




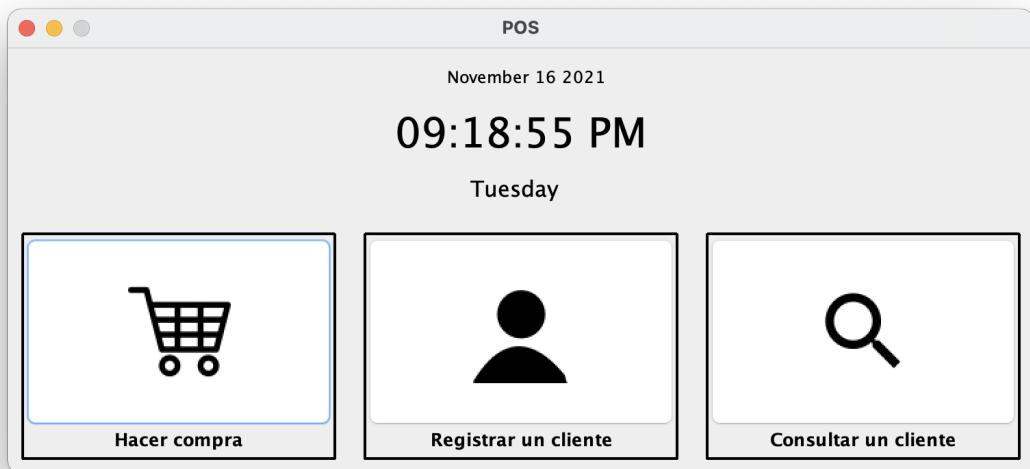
Si el usuario presiona **Eliminar lote** se abre la siguiente ventana



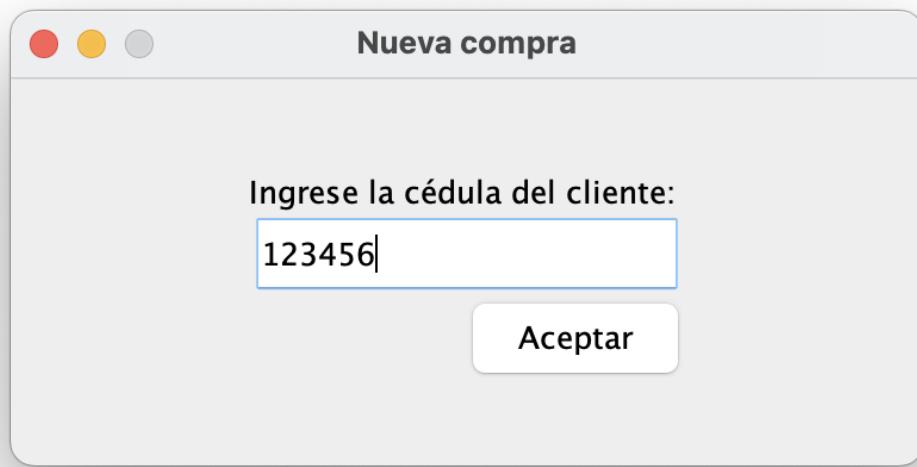




### 3.2 Producto Final: POS



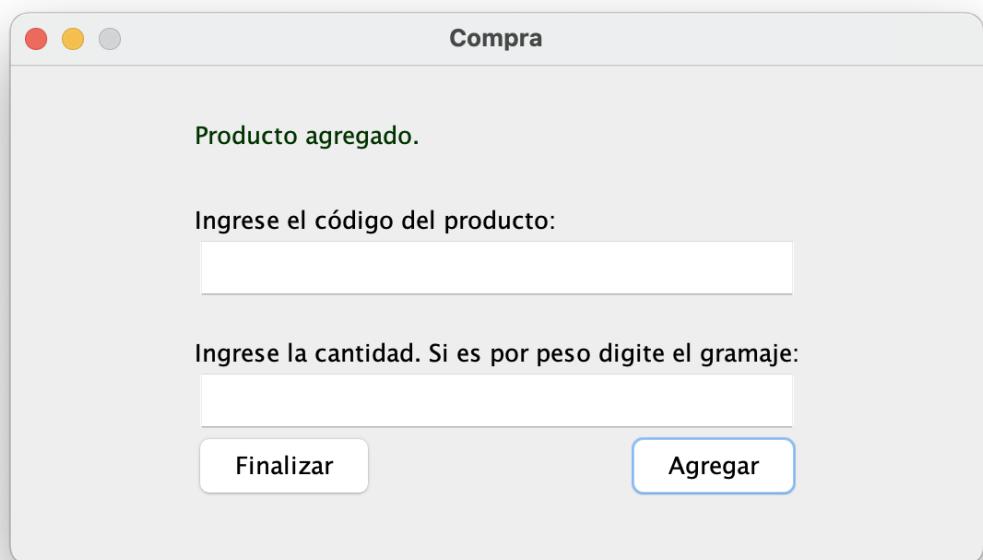
Si el usuario presiona **Hacer Compra** se abre la siguiente ventana

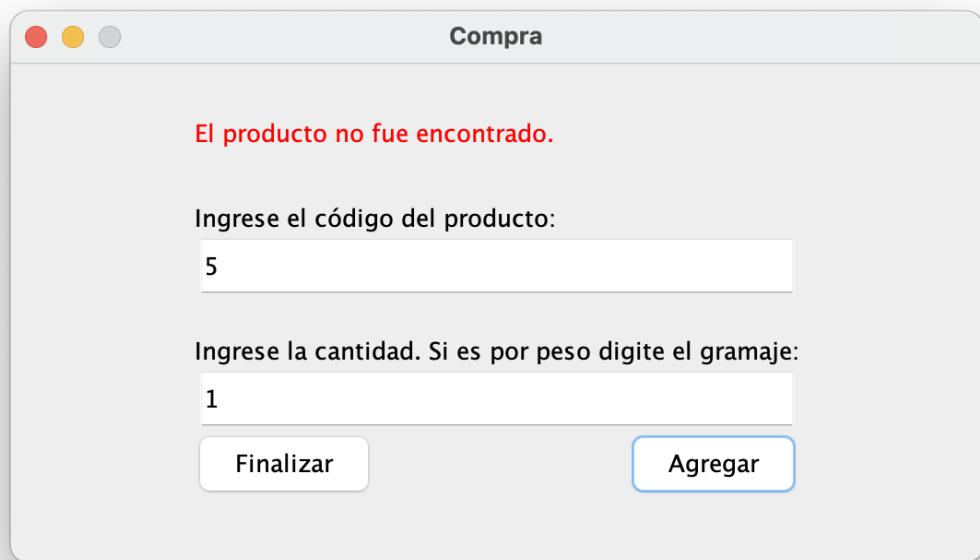


Compra

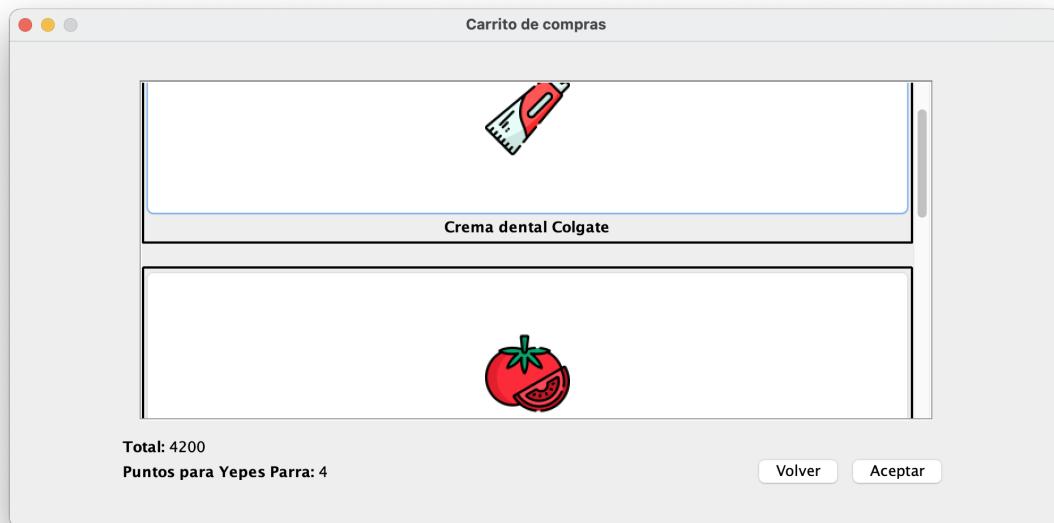
Ingrese el código del producto:

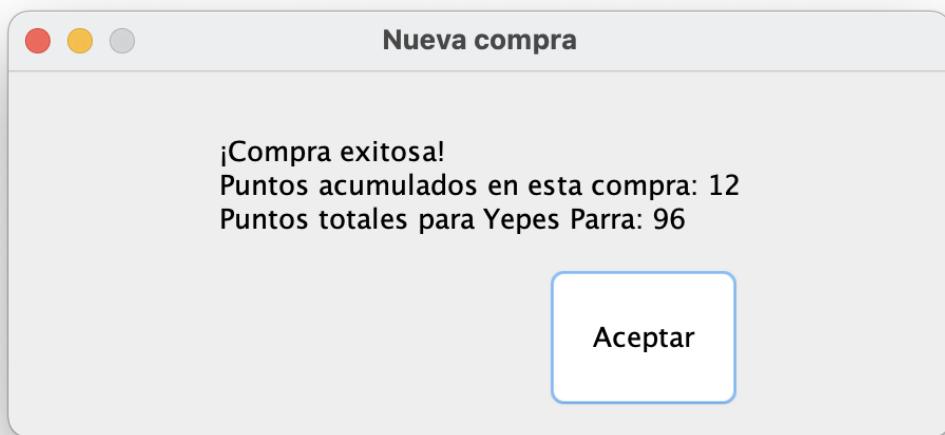
Ingrese la cantidad. Si es por peso digite el gramaje:





Se agregan los productos secuencialmente, si se oprime finalizar se puede ver toda la lista de los productos agregados



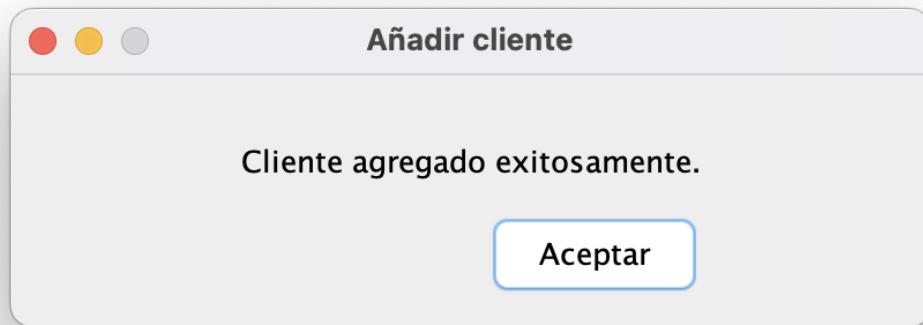


**Registro de cliente**

|                    |            |
|--------------------|------------|
| Cédula:            | 2131038    |
| Nombres:           | Nemo       |
| Apellidos:         | Buscando   |
| Edad:              | 19         |
| Pronombres:        | Él         |
| Situación Laboral: | Estudiante |

**Aceptar**

Si el usuario presiona **Registrar Cliente** se abre la siguiente ventana



**Registro de cliente**

Cédula:

Nombres:

Apellidos:

Edad:

Pronombres:

Situación Laboral:

**Aceptar**

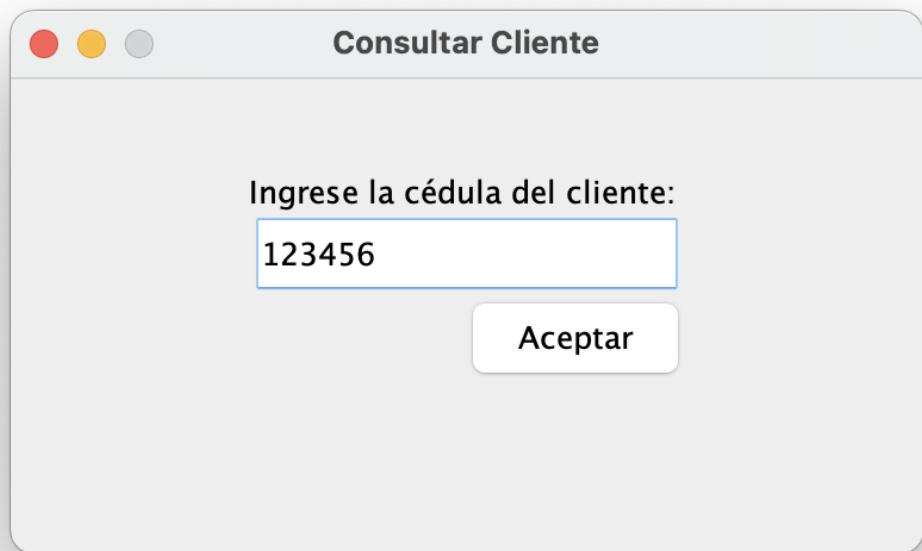
**Añadir Cliente**

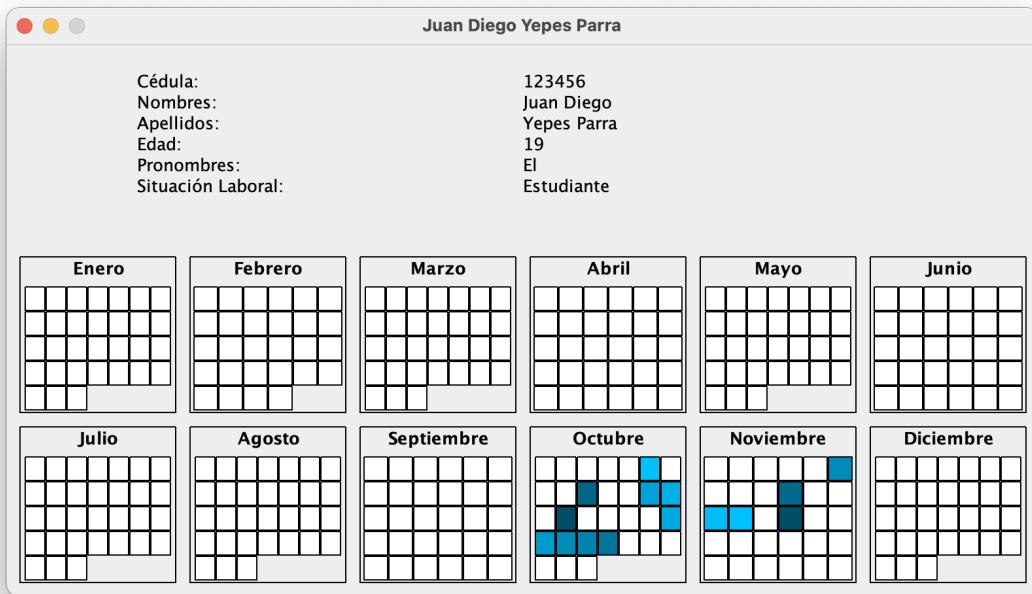
**Error**

Alguno de los campos es incorrecto

 Aceptar

Si el usuario presiona **Consultar cliente** se abre la siguiente ventana





### 3.3 Diagramas de diseño

Finalmente a través de los análisis hechos anteriormente se obtiene el siguiente diagrama de diseño.

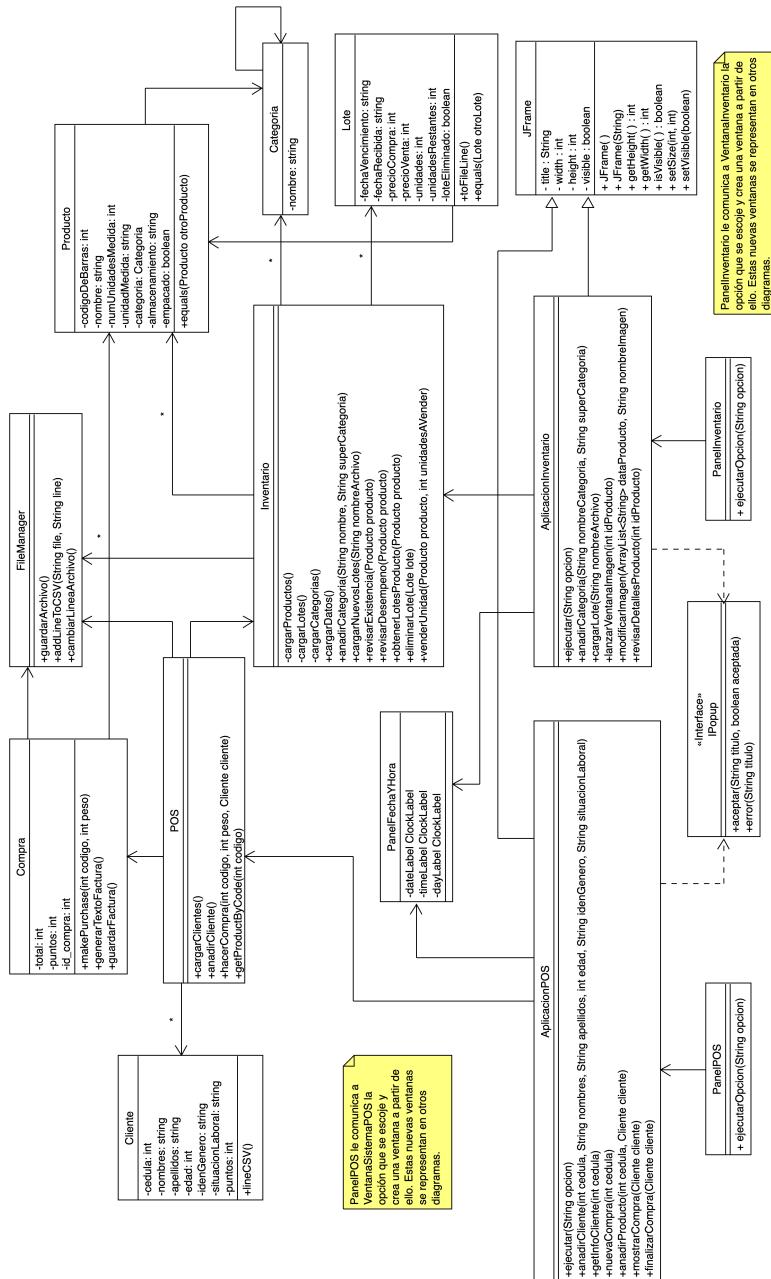


Diagrama de diseño general

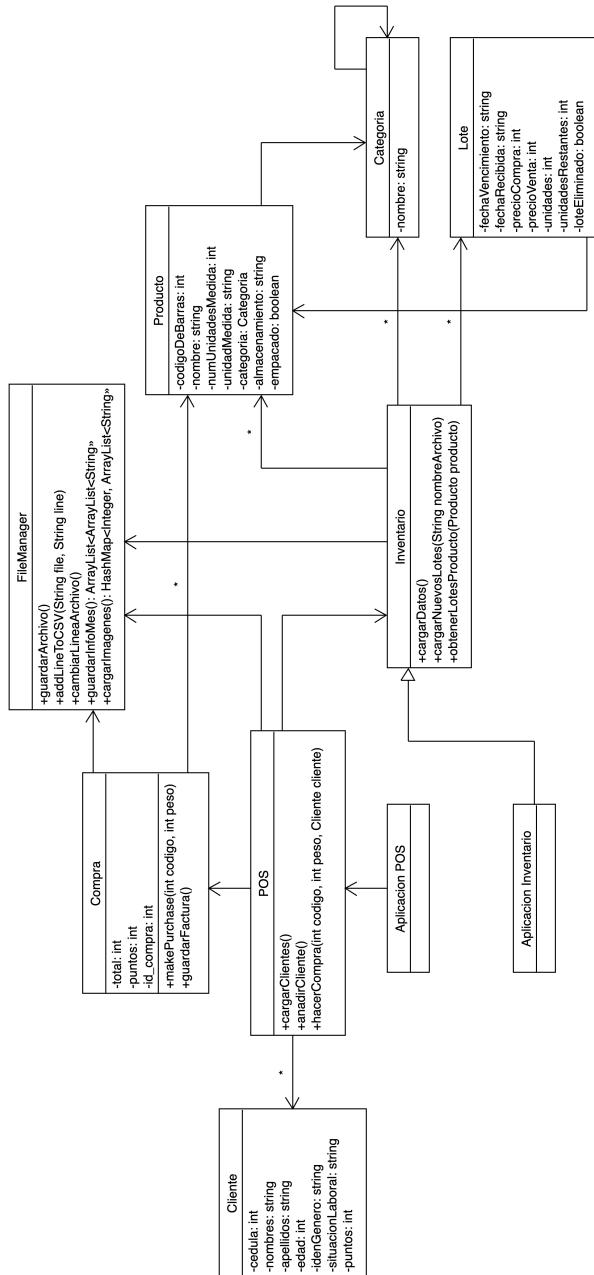


Diagrama de alto nivel para el modelo

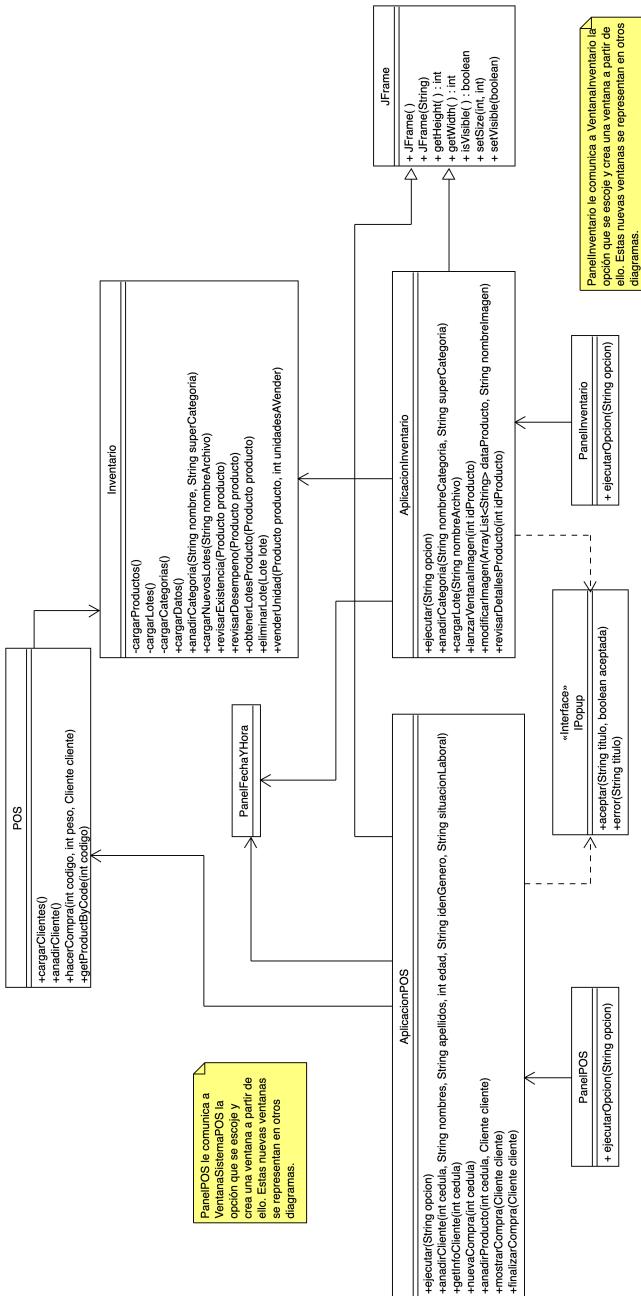
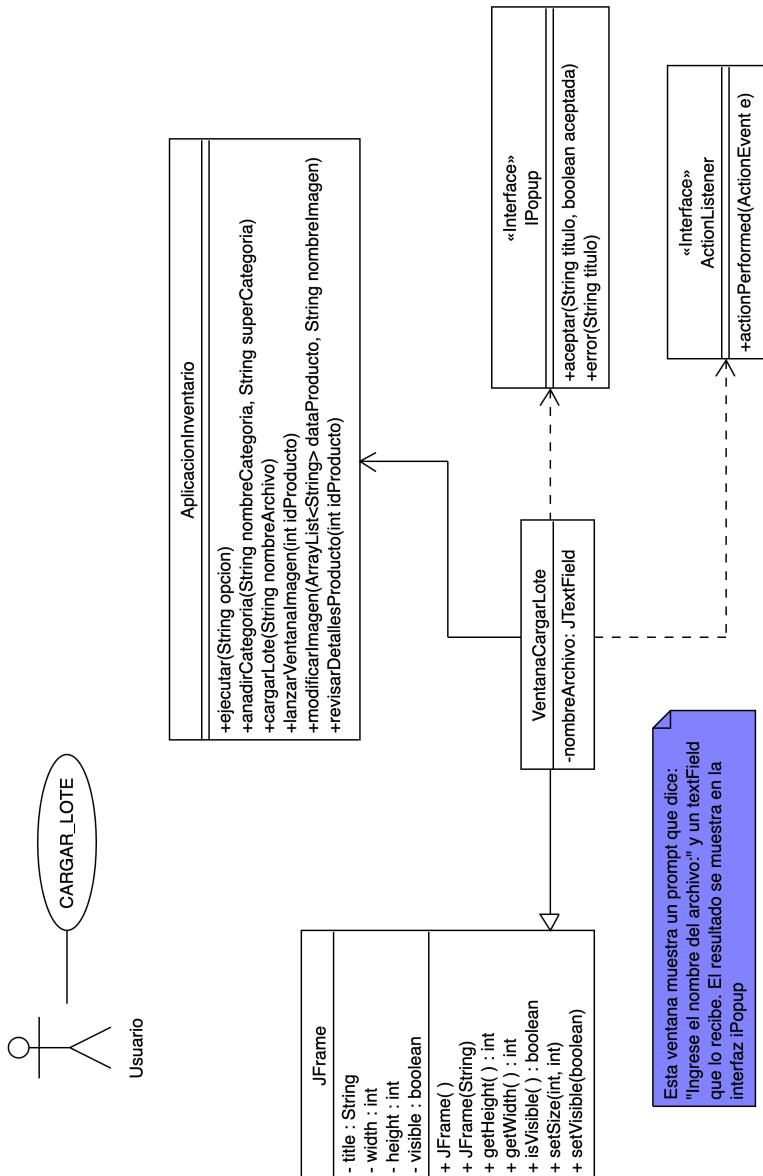
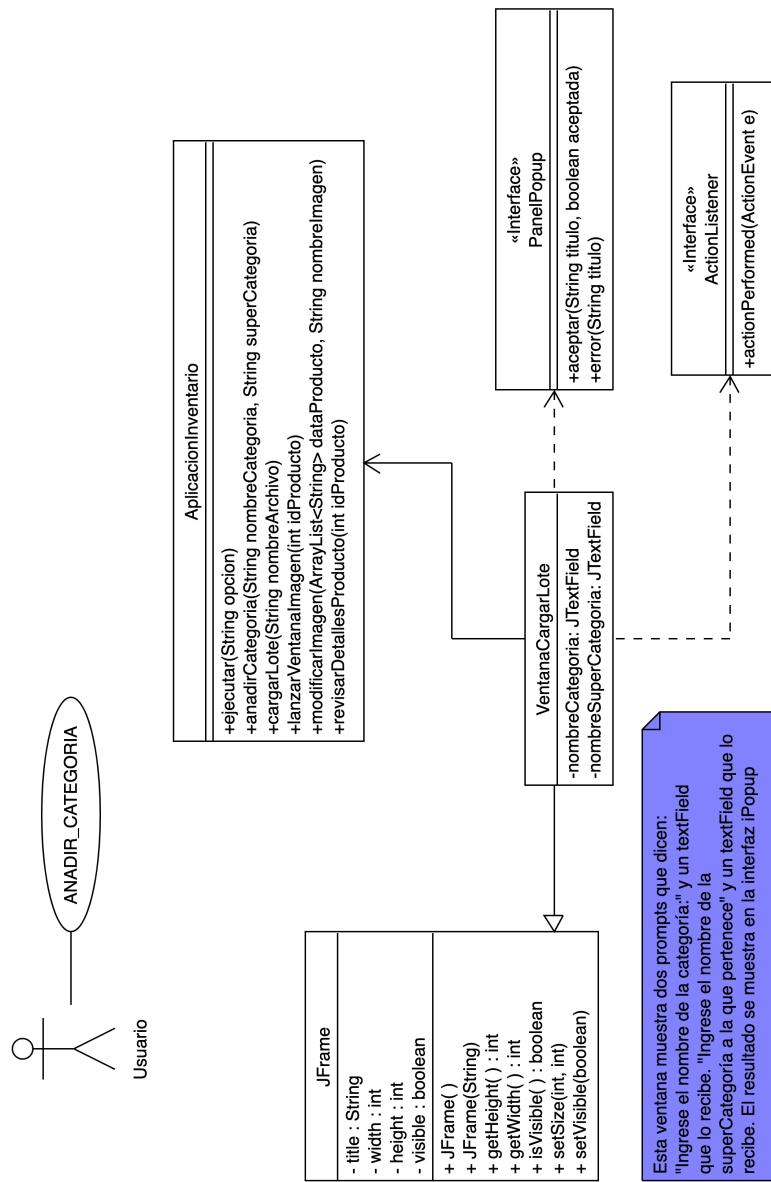


Diagrama de alto nivel para la interfaz

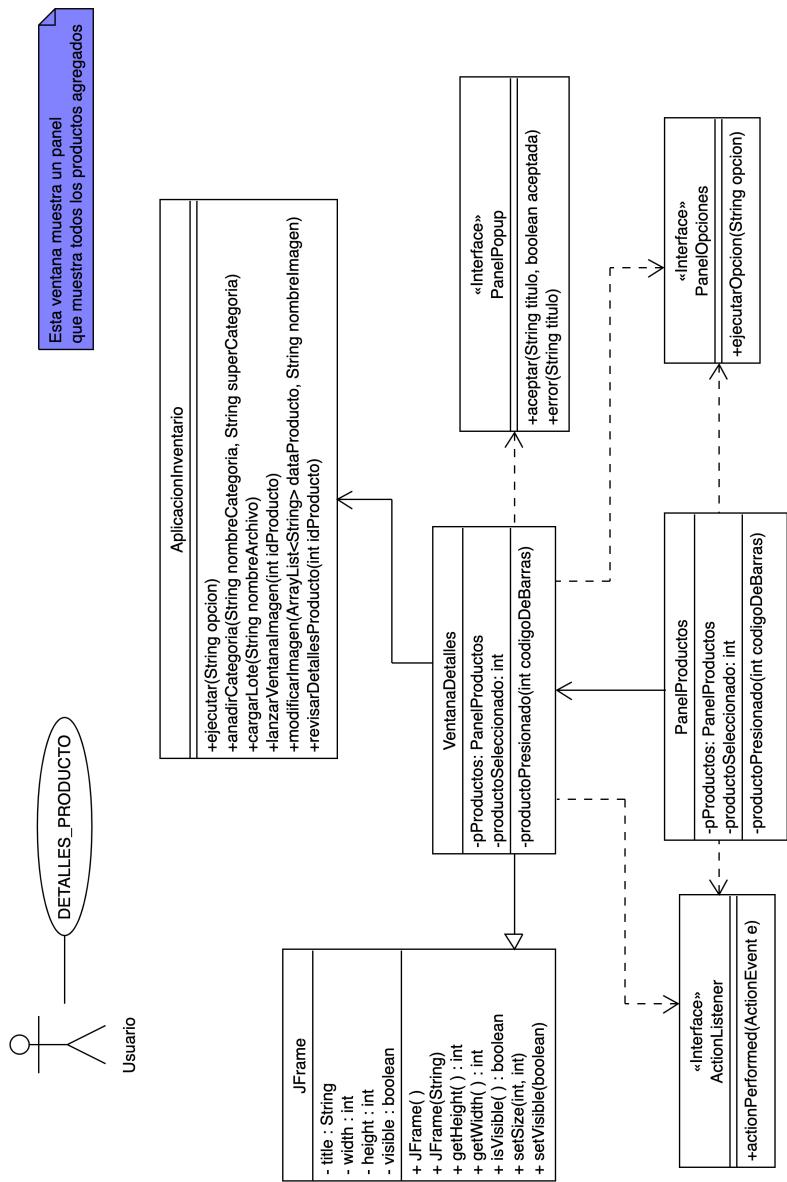
## INTERFAZ GRÁFICA SISTEMA INVENTARIO



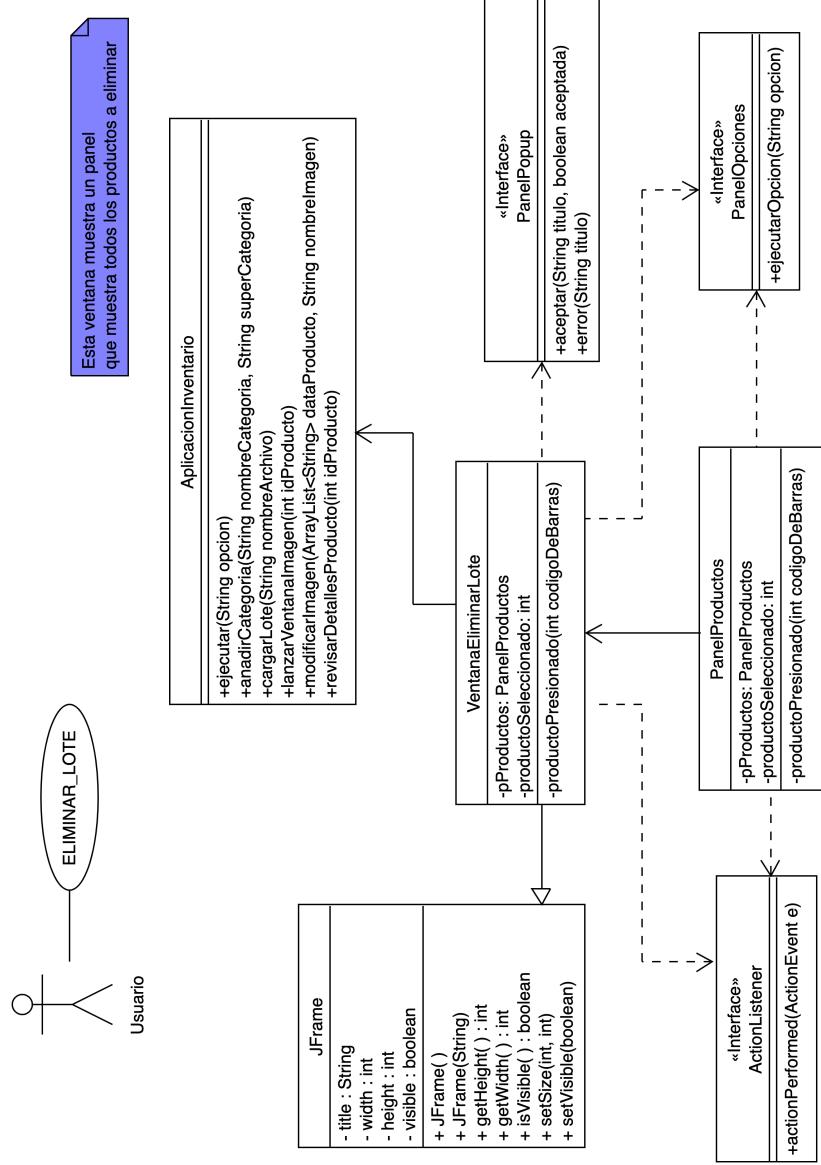
Caso 1: El *ActionListener* del PanelInventario recibió la instrucción CARGAR\_LOTE y se la comunicó a la AplicacionInventario



**Caso 2:** El *ActionListener* del PanelInventario recibió la instrucción ANADIR\_CATEGORIA y se la comunicó a la AplicacionInventario

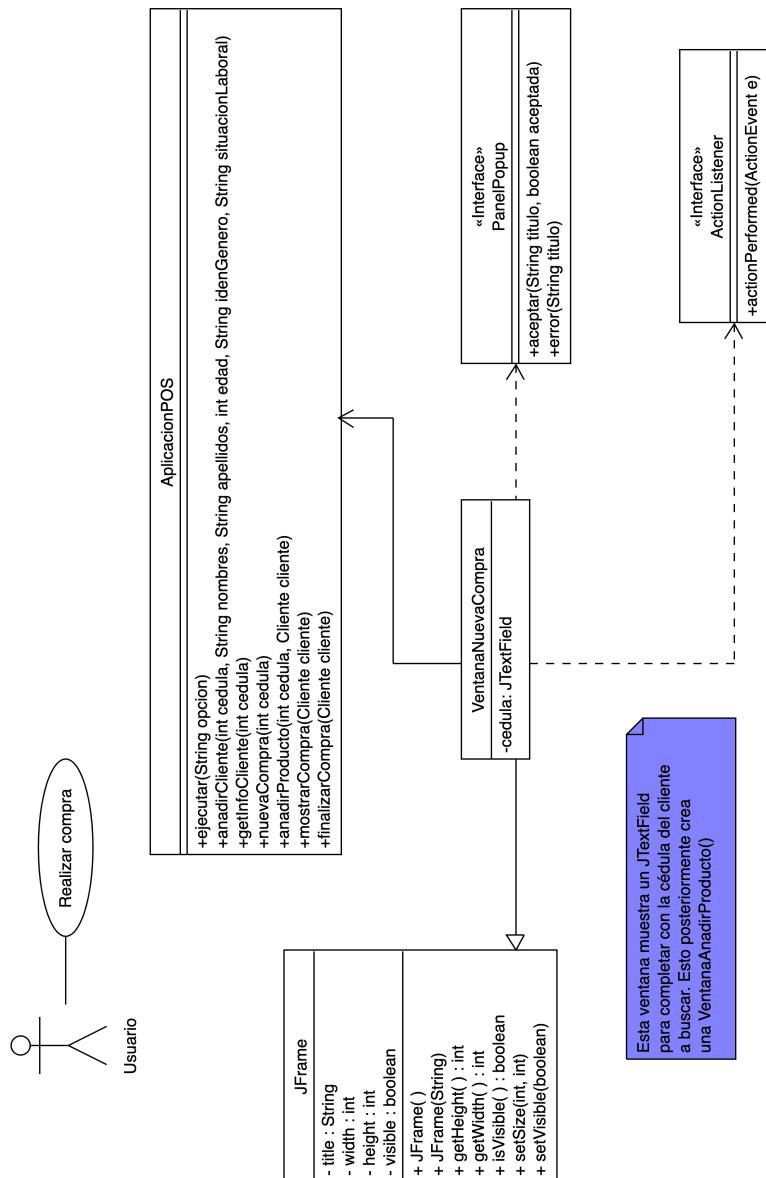


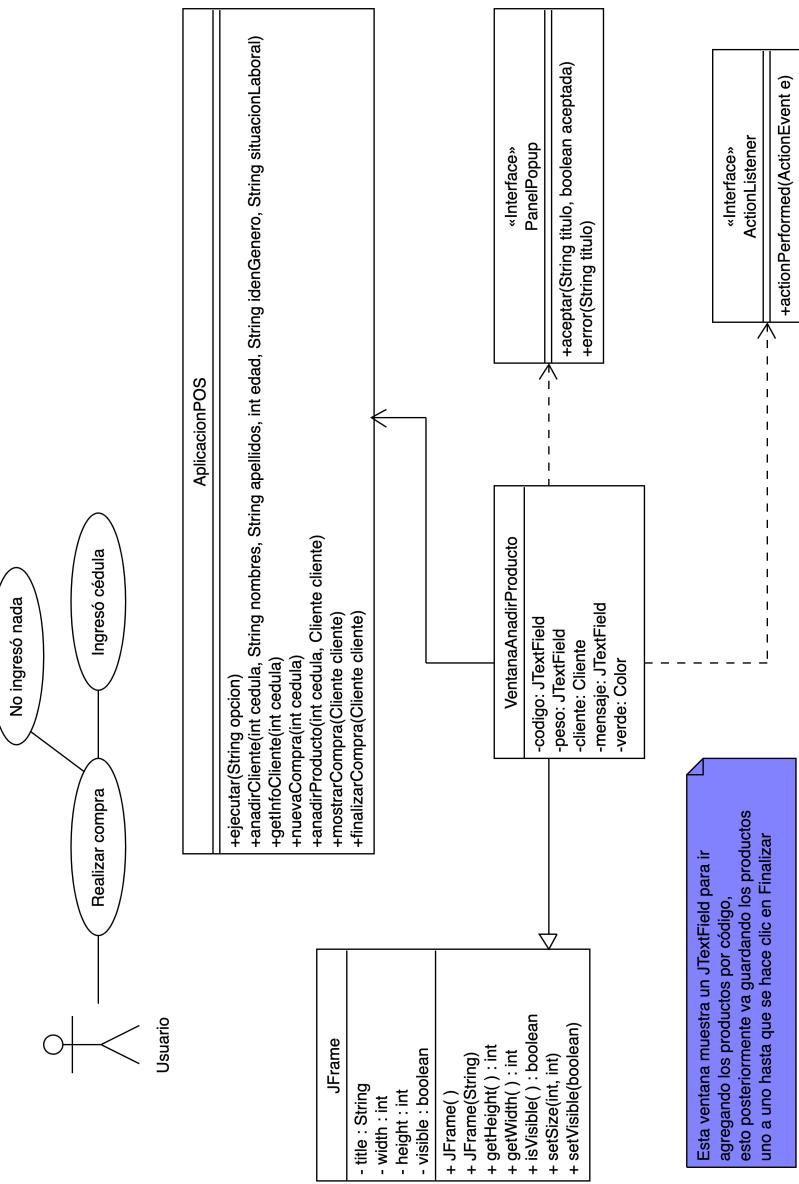
**Caso 3:** El *ActionListener* del PanelInventory recibió la instrucción DETALLES\_PRODUCTO y se la comunicó a la AplicacionInventory

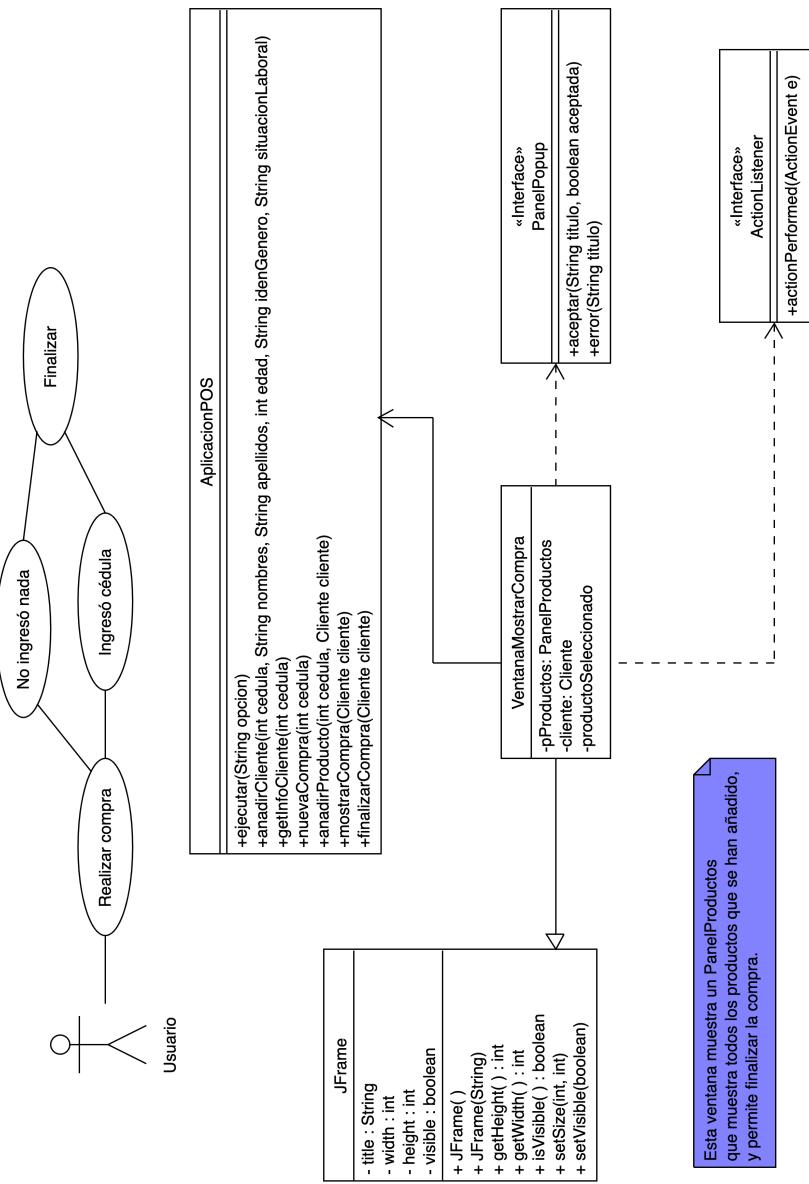


Caso 4: El *ActionListener* del *PanelInventoryario* recibió la instrucción **ELIMINAR\_LOTE** y se la comunicó a la *AplicacionInventory*

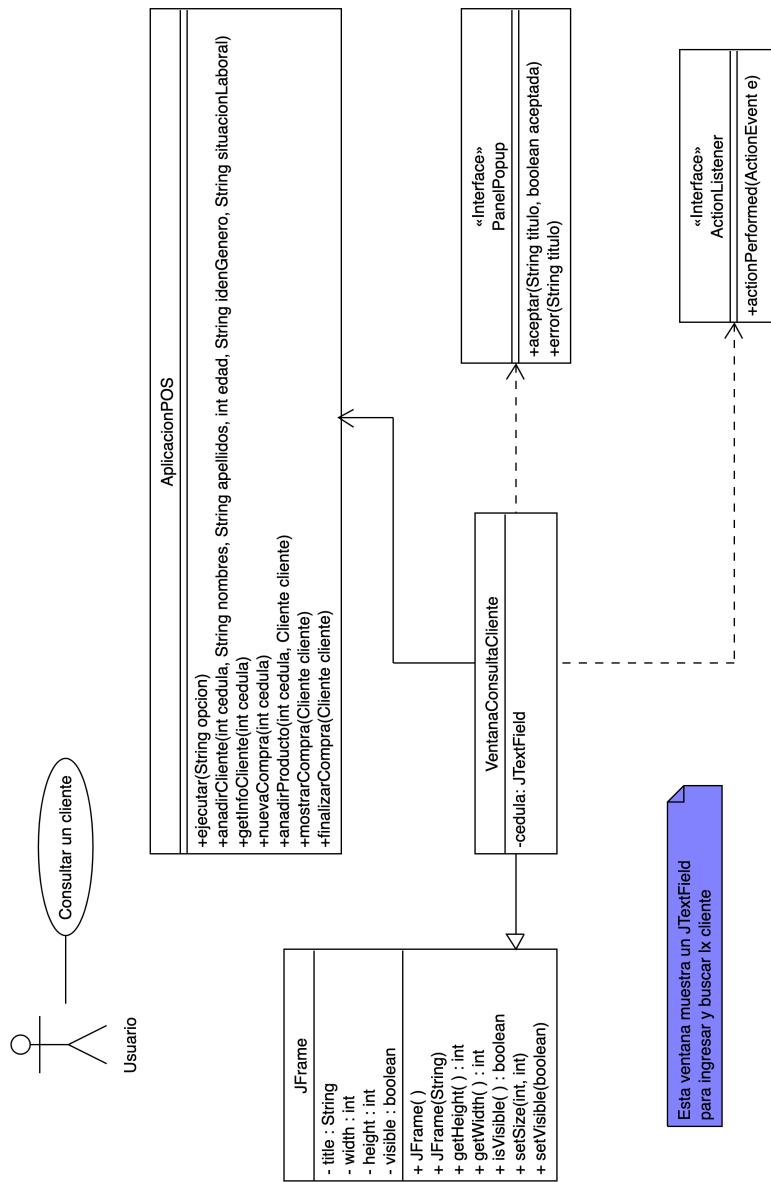
## INTERFAZ GRÁFICA SISTEMA POS

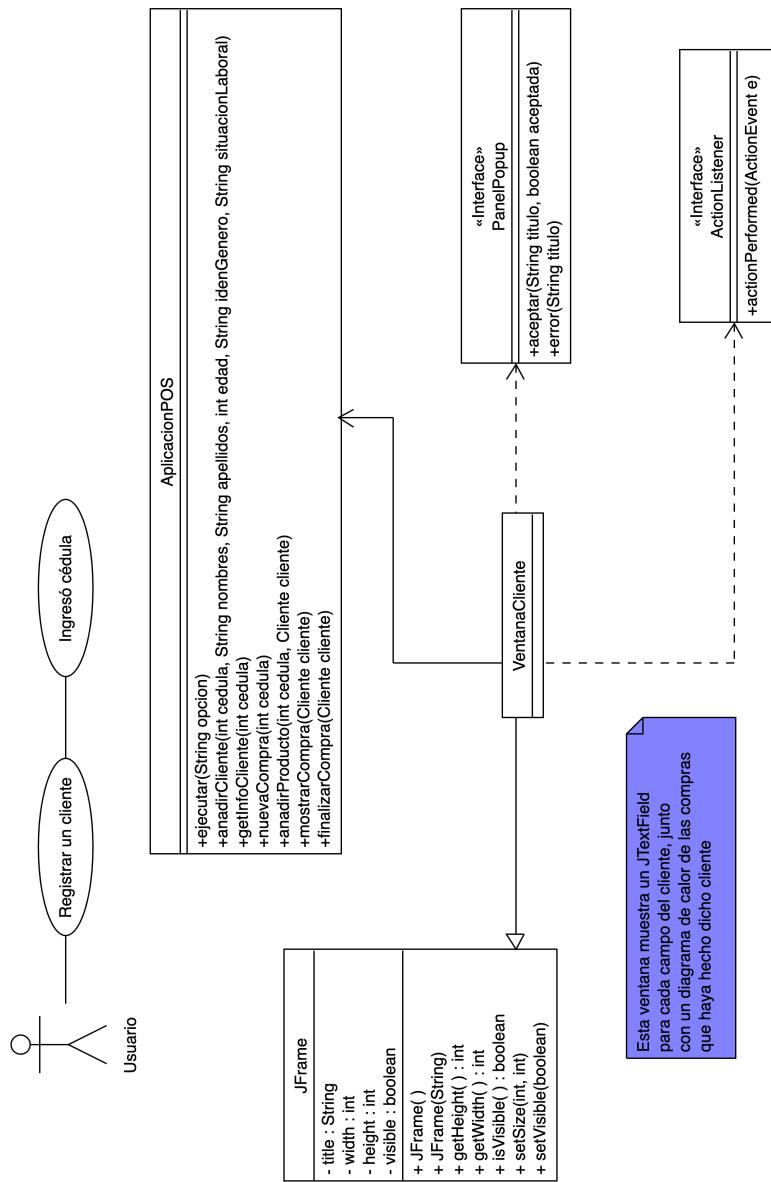




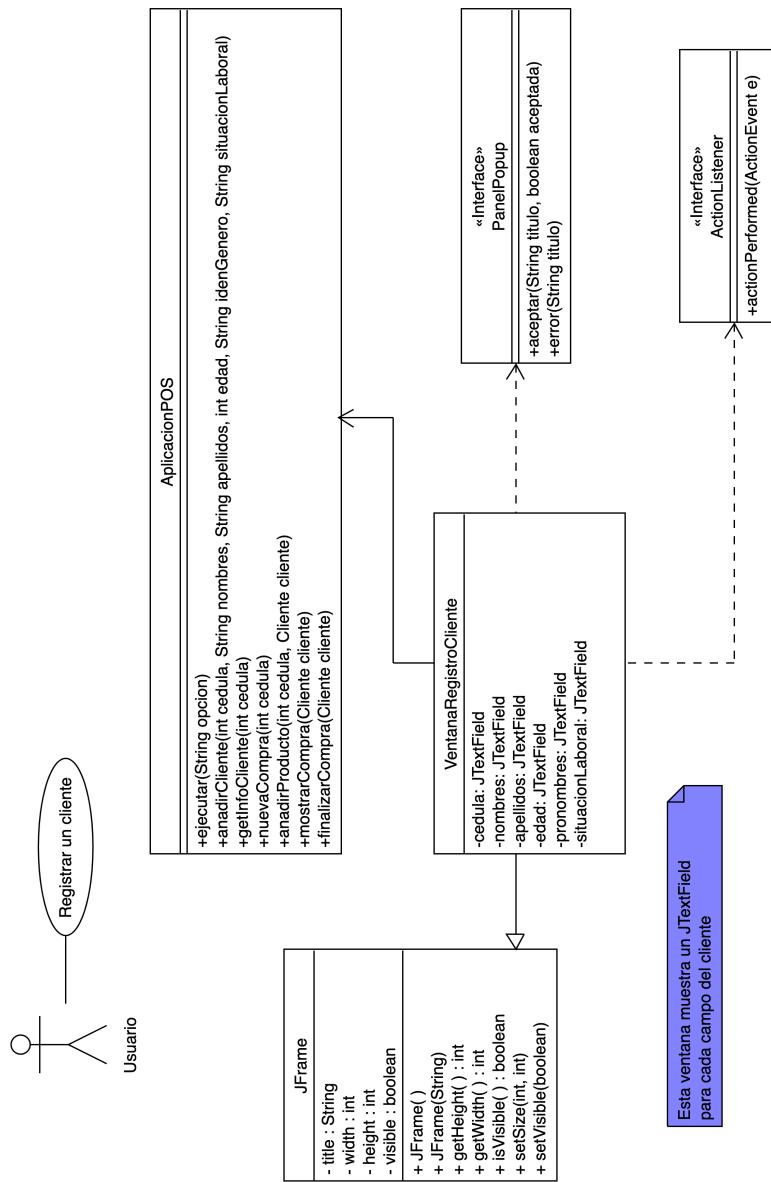


**Caso 1:** El *ActionListener* del PanelPOS recibió la instrucción "Hacer compra" y se la comunicó a la AplicacionInventario





**Caso 2:** El *ActionListener* del *PanelPOS* recibió la instrucción "Consultar Cliente" y se la comunicó a la *AplicacionInventario*



**Caso 3:** El *ActionListener* del PanelPOS recibió la instrucción "Registrar Cliente" y se la comunicó a la AplicacionInventario

### 3.3.1 Explicación de las interfaces

La interfaz **PanelPopup** es la que permite mandar los mensajes de error o de éxito. La implementan la mayoría de las ventanas.

```
1 public interface PanelPopup {
```

```
2 //titulo es el nombre de la ventana creada, aceptada es true si se presion aceptar
3 // sino se presion la x
4 public void aceptar(String titulo, boolean aceptada);
5 public void error(String titulo);
```

La interfaz **PanelOpciones** es la que permite mandar cuál opción se ejecuta a partir de los paneles de opciones de pos e inventario

```
1 public interface PanelOpciones {
2     public void ejecutarOpcion(String opcion);
3 }
```

### 3.4 Reflexión

Nuestra organización del código se justifica bajo la premisa de reutilizar el código, ya que tenemos varios paneles que utilizamos varias veces, como se va a mostrar en los diagramas de diseño posteriores. Igualmente, consideramos que de esta forma estamos aprovechando las restricciones de Java Swing, ya que es más fácil agregar paneles que agregar cada cosa individualmente afectando el layout.

Nuestra aproximación de tener primero un *mockup* del diseño y luego aplicarlo fue provechosa para poder tener un rumbo de diseño, e intentamos acercarnos lo más posible al mismo.

Finalmente, nuestro diseño tiene varias restricciones de acoplamiento, ya que esta interfaz gráfica es completamente dependiente del modelo, y no podría ser reutilizada en diseños diferentes de modelos. Sin embargo, esto no es necesariamente algo malo, y es una restricción predecible desde que teníamos el enunciado del proyecto.