Documento de diseño - Proyecto 2
ISIS-1226

David Valderrama Herrera
Martín Daniel Rincón Molina

1. Contexto del problema.

La figura 1 ilustra los requerimientos funcionales del proyecto. Los sistemas *POS* e *Inventario* no se usan nunca de manera paralela y ambos tienen que ser capaces de guardar la información en la memoria mediante algún mecanismo de persistencia.

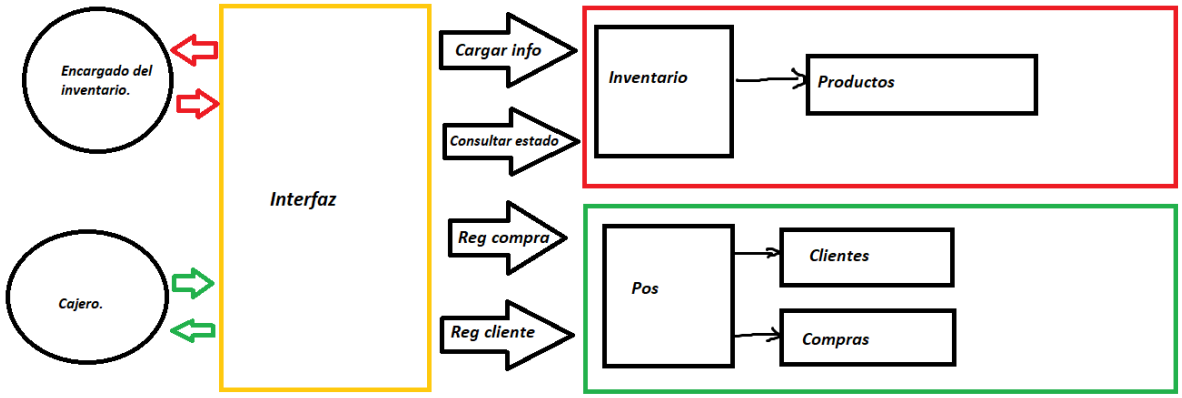

Figura 1: Contexto general del problema.

Los productos del sistema de inventarios se cargan por lotes, los cuales determinan ciertas características de los productos, como su fecha de caducidad o su precio de compra. Los productos se cargan por lotes mediante archivos *csv*.

Los usuarios interactúan con los 2 sistemas mediante una misma interfaz. Por medio de la interfaz se permite registrar nuevos clientes, nuevas compras, cargar un archivo con productos, consultar el estado del inventario para un producto dado su código, y eliminar todos los lotes del producto.

2. Sistema Inventario

2.1. Lógica del sistema

2.1.1. Nivel 1.

Roles y componentes

- Coordinador del inventario
- Cargador de archivos
- Controlador de productos

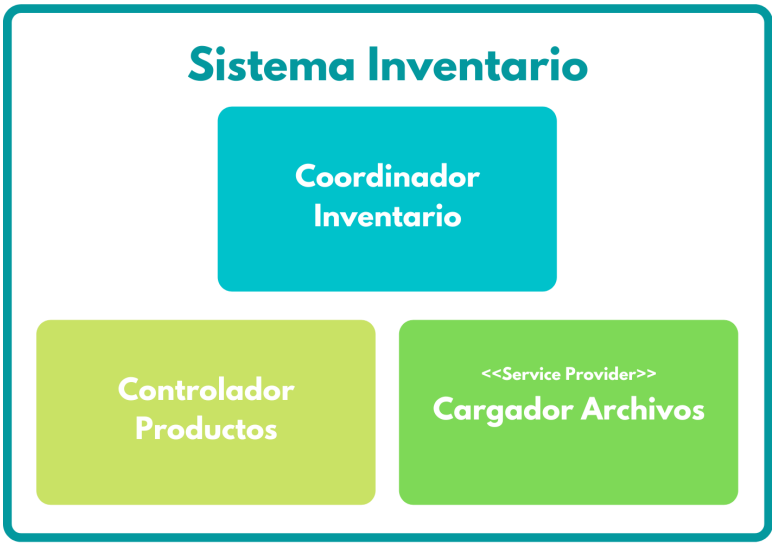

Figura 2: Componentes candidatos y sus respectivos estereotipos

Responsabilidades. En la tabla 1 se muestran las responsabilidades de cada rol para la iteración 1.

#	Responsabilidad	Componente
1	Facilitar el intercambio de datos entre la interfaz y el controlador	Coordinador inventario
2	Inicializar el proceso de carga de datos de los productos existentes.	
3	Leer el archivo csv que contiene los lotes de productos	Cargador de archivos
4	Se encarga de la persistencia de los productos del inventario	
5	Inicializar el proceso de escritura en los datos de los producto existentes	Controlador de productos
6	Añadir nuevos productos.	
7	Responder consultas sobre los productos del inventario	
8	Eliminar lotes vencidos	

Tabla 1: Responsabilidades de los roles del inventario para la iteracion 1.

Colaboraciones.

- **Cargar archivo CSV - Se entiende que el Coordinador POS recibe una petición desde la interfaz y:**

  a) Le indica al Controlador de Productos que le pida al Administrador de Archivos que abra un archivo con determinado nombre.

  b) El Administrador de Archivos abre el archivo y se lo retorna al Controlador de Productos como un objeto procesable.

  c) El Controlador de Productos procesa la información del objeto y le indica al Administrador de Archivos que persista los cambios.

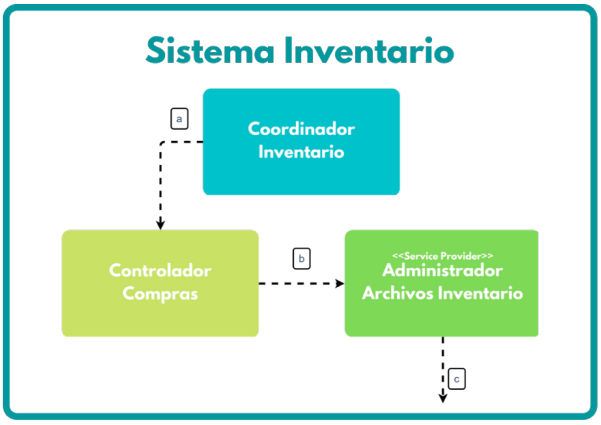

Figura 3: Colaboración entre componentes a la hora de leer el archivo csv

- **Borrar los productos vencidos - Se entiende que el Coordinador POS recibe una petición desde la interfaz y:**

  a) Le indica al Controlador de productos que quite los lotes vencidos y actualice las ganancias.

  b) El Controlador de Productos le pide al Administrador de Archivos que persista los cambios.

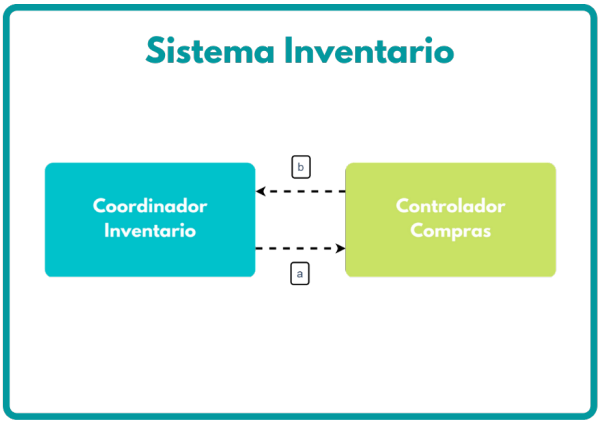

Figura 4: Colaboración entre componentes a la hora a de quitar los productos vencidos

Figura 5: Asociaciones entre componentes de acuerdo a las colaboraciones

2.1.2. Nivel 2

Coordinador de inventario. Componentes

Con base en las responsabilidades delegadas a este componente en la tabla **??** es importante resaltar que estas permiten, por su concordancia en cuanto a granularidad y sentido (peticiones que vienen desde la interfaz a otras clases que llevan la lógica de negocio), que este componente se materialice en un única clase con estereotipo coordinador. Esta clase tiene como único atributo un objeto que pertenece a la clase AdminProductos, así como métodos que responderán a cada una de las solicitudes que podrían venir desde la interfaz.

Responsabilidades y Colaboraciones

Dado que este componente consta de una única clase, no da lugar a colaboraciones internas, sin embargo, este componente es indispensable para todas las colaboraciones entre componentes descritas durante el nivel 1.

Administrador de archivos. Componentes

Las dos responsabilidades definidas durante el nivel 1 para este componente tienen en común que requieren de la interacción con elementos ajenos al entorno de Java (archivos alojados en memoria secundaria) para guardar y recuperar información , de manera que, se optó por implementar este componente en una única clase sin atributos que cumple con el esteretipo de proveedor de servicios para el resto del sistema por medio de sus métodos.

Responsabilidades y Colaboraciones

Al haber una sola clase, no se da lugar a colaboraciones dentro del componente. En primer lugar, como la información sobre los productos ha de ser persistente, se hace uso de serializadores para que el sistema logre mover la información de los productos entre memoria principal y secundaria. En segundo lugar, como se ha de leer la información sobre los lotes, este elemento encargado de seguir la ruta indicada por el usuario, abrir y leerlo con un BufferedReader para su posterior uso por el controlador de productos.

Administrador de productos Componentes

Este componente pasa en este nivel a ser cuatro componentes independientes. Es importante anotar que tres de estos corresponden con el estereotipo de Information Holders, pues se encargan de mantener información, en este caso de los productos y los lotes. Así pues, se separó este componente en elementos como sigue: la clase Producto que se encarga de contener la información de los productos, de manera que, se comporta como un contenedor de los lotes; Lote que se ocupa de contener la información de la cantidad de productos que comparten costos, precio de venta y fecha de vencimiento, lo cual es vital para calcular las ganancias; y Fecha que sirve, en el caso específico de este sistema, para facilitar la remoción de productos que han expirado. Luego, hay un último elemento que cumple el rol de controlador, este hace las modificaciones solicitadas por el usuario sobre información de Producto, Lote y Fecha, y al mismo tiempo, implementa métodos que construyen las salidas que necesita el usuario con base en los atributos de estos Information Holders.

Responsabilidades y Colaboraciones

Con respecto a las responsabilidades de los elementos que conforman este componente se debe entender que las clases que cumplen con el estereotipo de Information Holder tienen como responsabilidad representar cosas, por lo tanto, su responsabilidad es trabajar con sus atributos para cambiarlos con base en las entradas del controlador y/o producir las salidas que este requiera: añadir un producto, añadir un lote a un producto (la cual exige una colaboración con la clase Producto para modificar la estructura de datos que contiene los lotes), modificar la disponibilidad y borrar los productos vencidos (colaboración con los tres Information Holders de este componente), verificar la disponibilidad de un producto (colaboración con Producto y Lote).

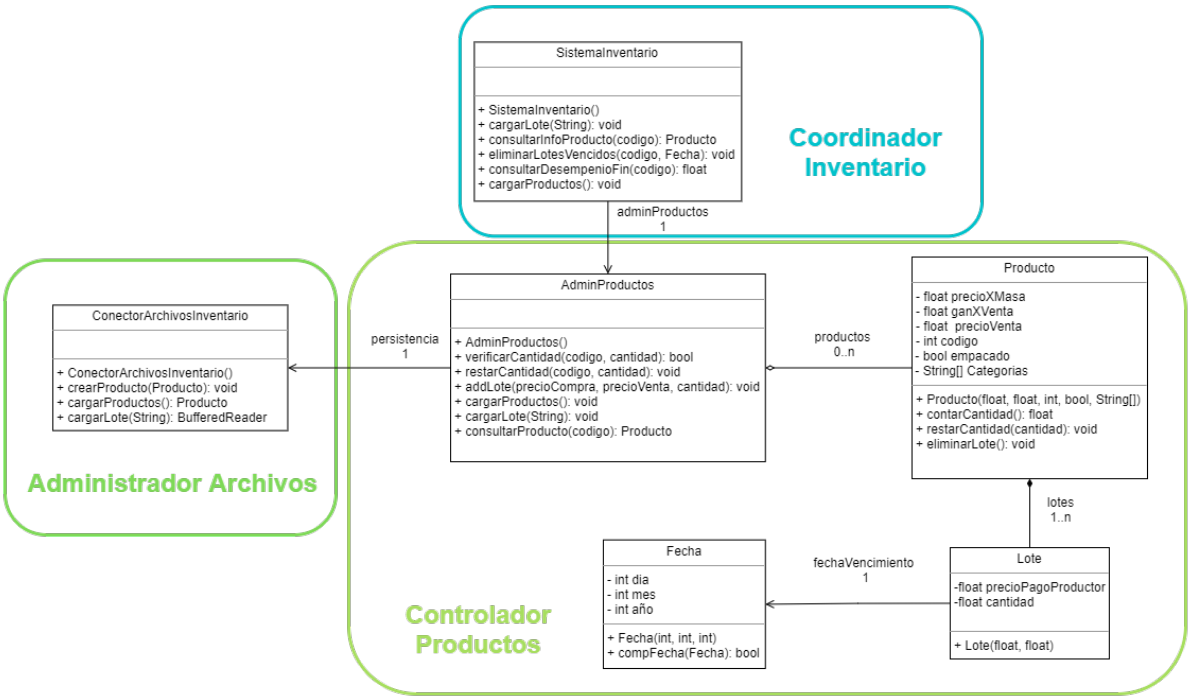

Figura 6: Diagrama de clases UML para la lógica del Inventario

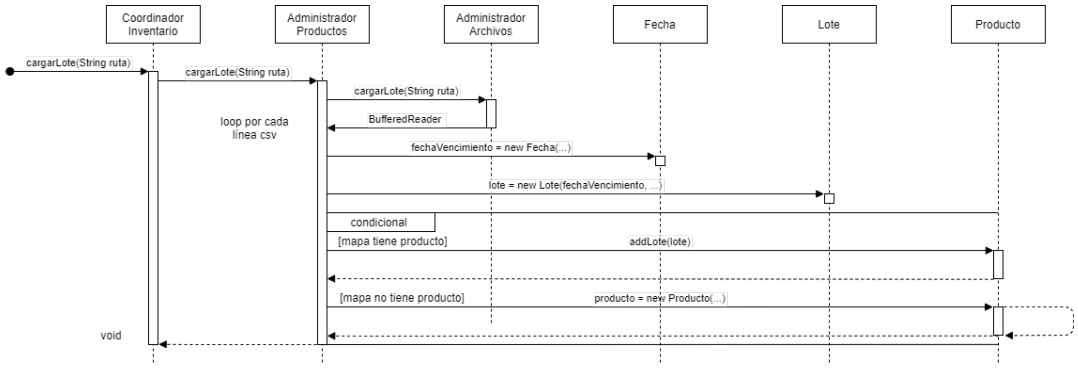

Figura 7: Diagrama de secuencia para cargar la información de nuevos lotes en el inventario

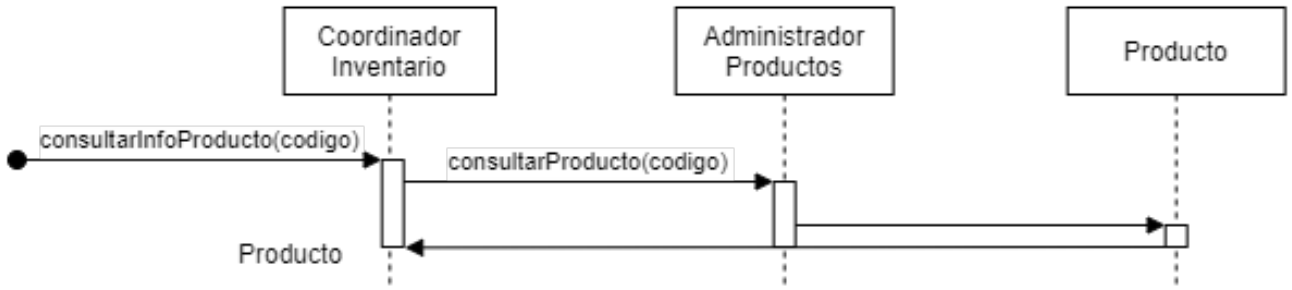

Figura 8: Diagrama de secuencia para consultar la información de un producto

Diagramas y diseño final:

2.2. GUI del sistema

2.2.1. Nivel 1

Responsabilidad	Componente
Es el puente entre la lógica y el resto de la interfaz gráfica	Contenedor
Contener los demás elementos que permiten el funcionamiento del sistema inventario	
Hacer las peticiones para empezar la carga del CSV por solicitud del encargado del inventario	Carga CSV
Presentar la información del producto elegido por el encargado del inventario	Visor Producto
Hacer las peticiones para cambiar la imagen asociada a un producto	

Figura 9: Requerimientos funcionales interfaz gráfica

2.2.2. Nivel 2

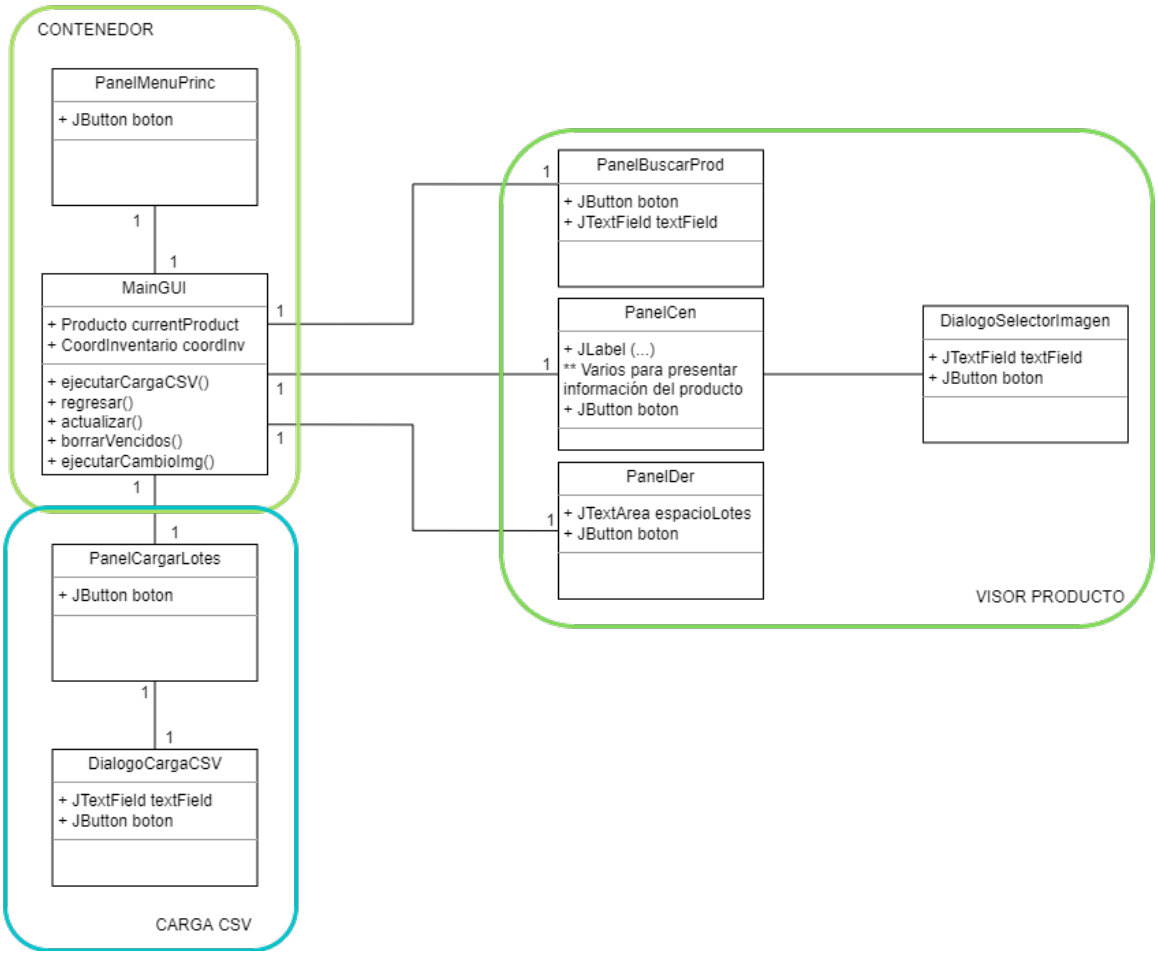

Figura 10: Diagrama de clases de todo el sistema Inventario

Responsabilidad	Componente
Es el puente entre la lógica y el resto de la interfaz gráfica	MainGUI
Contener los demás elementos que permiten el funcionamiento del sistema inventario	
Pedir la ruta del archivo csv con la información de los lotes que han de ser cargados	Diálogo Carga CSV
Hacer las peticiones para empezar la carga del CSV	
Se encarga de hacer visible el 'Diálogo Carga CSV'	Panel Carga CSV
Pedir el código del producto que se desea buscar en el sistema	Panel Buscar Producto
Hacer las peticiones correspondientes para traer la información del producto que desea el encargado del inventario	
Mostrar la información de los lotes	Panel Derecho
Hacer las peticiones correspondientes para eliminar los lotes vencidos	
Pedir el código del archivo de imagen que se desea vincular al producto elegido por el encargado del inventario	Diálogo Selector Imagen
Hacer las peticiones para cambiar la imagen asociada a un producto	
Mostrar los datos del producto que el encargado solicitó en 'Panel Buscar Producto'	Panel Centro
Se encarga de hacer visible el 'Diálogo Selector de Imagen'	
Permite volver al menú de entrada a los dos sistemas	Panel Menú Principal

Figura 11: Requerimientos funcionales interfaz gráfica

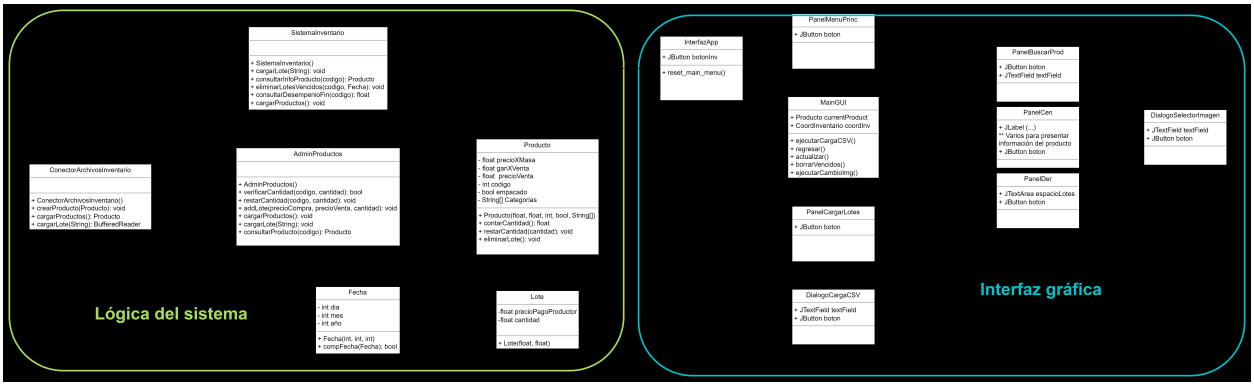

Figura 12: Diagrama Lógica + Interfaz gráfica

3. Sistema POS

3.1. Lógica del sistema

3.1.1. Nivel 1

Roles y componentes Se hace una división preliminar de roles para el sistema pos. La división se hace buscando trabajar con elementos de responsabilidades similares y que permitan el cumplimiento de los requerimientos del sistema. En la figura 13 se muestran los 4 roles sugeridos.

- **Coordinador Pos:** Es necesario un elemento que se encarga de la comunicación entre las otras partes del sistema y la interfaz de usuario. El coordinador recibe y entrega información al usuario, también delega las tareas a otras partes con más lógica en el sistema. Dado que la principal tarea de este componente es delegar tareas entre componentes, se le asigna el estereotipo de *coordinador.*

- **Controlador clientes:** Dado que los requerimientos exigen que los clientes puedan hacer compras y que se puedan agregar clientes, consideramos pertinentes tener un elemento encargado de administrar las operaciones sobre los clientes. Este componente es un *controlador.*

- **Controlador de compras:** Puesto que se requiere crear, modificar, y cerrar las compras de los clientes, consideramos pertinente tener un elemento que se encargue de la lógica que implican estas operaciones. Se espera también que este elemento sea capaz de comunicarse con el inventario. Este componente es un *controlador.*

  Vale resaltar que, a diferencia de los clientes, los requerimientos de la aplicación no necesitan que se guarden las compras realizadas para usarlas luego; solo se tiene que imprimir en consola el resultado de la compra. Por lo anterior, el controlador de compras no tiene ninguna responsabilidad relacionada a la persistencia de las compras finalizadas.

- **Administrador archivos:** Uno de los requerimientos es que la información de los clientes sea persistente. Para evitar asignar esta tarea también al controlador, contamos con un elemento especial que se encarga de escribir y recuperar la información de los clientes de la memoria secundaria del sistema. Este componente es un *proveedor de servicios* ya que su función es permitir escribir y rescatar información persistente de la memoria del sistema.

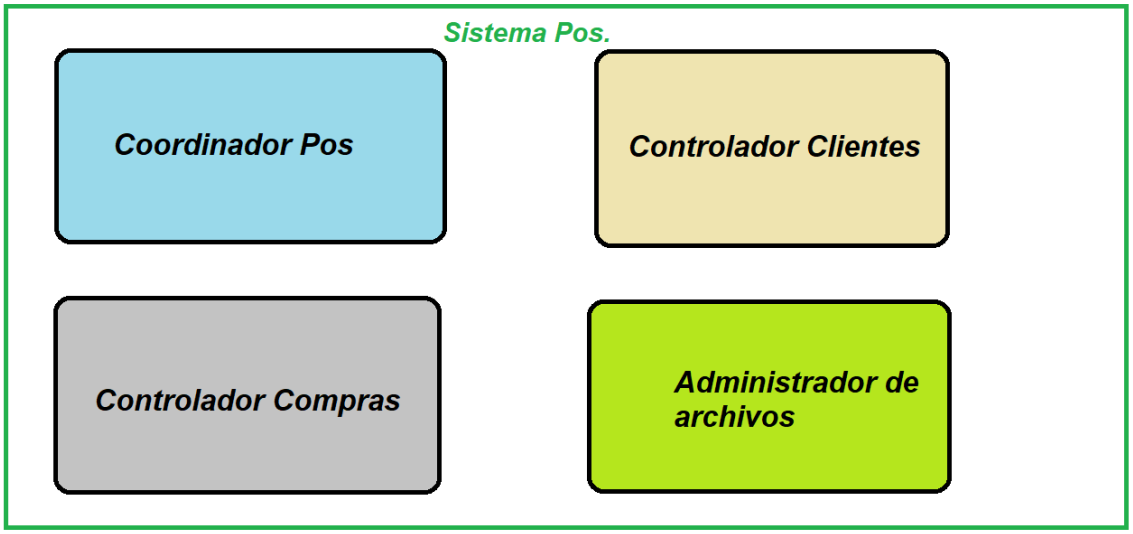

Figura 13: Roles para la iteración intermedia del sistema Pos.

Responsabilidades En la tabla 2 se muestran las diferentes responsabilidades y el componente al que se asigna cada una.

Tabla 2: Responsabilidades

#	Responsabilidad	Componente
1	Facilitar la comunicacion entre la interfaz y los controladores	Coordinador pos
2	Manejar el intercambio de informacion entre controladores	
3	Inicializar el proceso de carga de clientes.	
4	Añadir clientes	Controlador clientes
5	Buscar un cliente	
6	Gestionar los puntos de un cliente	
7	Dar la informacion de un cliente especifico	
8	Manejar la persistencia del sistema	
9	Abrir una nueva compra	Administrador de archivos
10	Añadir productos a una compra con los datos del inventario	Controlador de compras
11	Cerrar una compra	
12	Aplicar promociones a los articulos comprados	
13	Agregar un combo a la compra	
14	Cargar promociones y combos	

El sistema se inicia al abrir la consola de usuario, pero no queremos que el cajero tenga que solicitar que se carguen los clientes, por esto asignamos la responsabilidad al coordinador de que, una vez sea creado, inmediatamente genere las instrucciones que concluyan en la carga de los datos de los clientes guardados.

Colaboraciones Algunas de las colaboraciones más importantes de la aplicación se muestran a continuación. Su correcta ejecución depende del cumplimiento de las responsabilidades de cada uno de los componentes señalados previamente.

En las figuras 14, 15, 16 y 17 se muestran diagramas de los pasos que se ejecutan en las diferentes colaboraciones.

- **Crear una nueva compra:** Implica la solicitud por parte de la interfaz de iniciar una nueva compra.

1. El coordinador le solicita al controlador de clientes un cliente especifico; para quien se hará la orden de compra.
2. El coordinador le solicita al controlador de compras que cree una nueva compra para el cliente retornado por el controlador de clientes.
3. El controlador de compras le indicará al coordinador si se pudo crear la compra, o si actualmente existe una compra que debe ser cerrada primero.

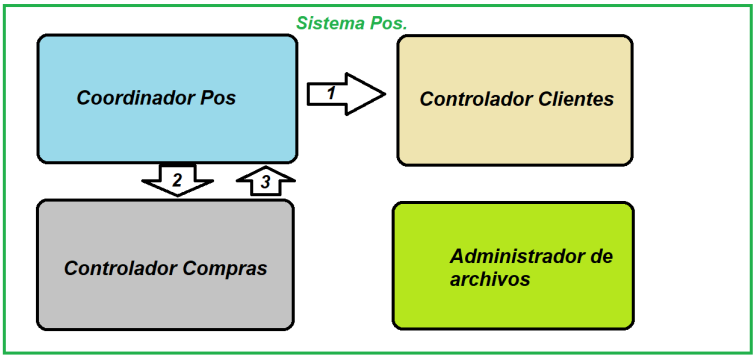

Figura 14: Secuencia de ejecución de la colaboración

- **Registrar un nuevo cliente:** Requiere que el cajero solicite un nuevo cliente e ingrese todos los datos del cliente solicitados.

1. El coordinador le indica al controlador que cree un nuevo cliente. Para esto le pasa los parámetros necesarios.
2. El controlador de clientes le envía el nuevo cliente al administrador de archivos para que lo guarde como un *serializable*.

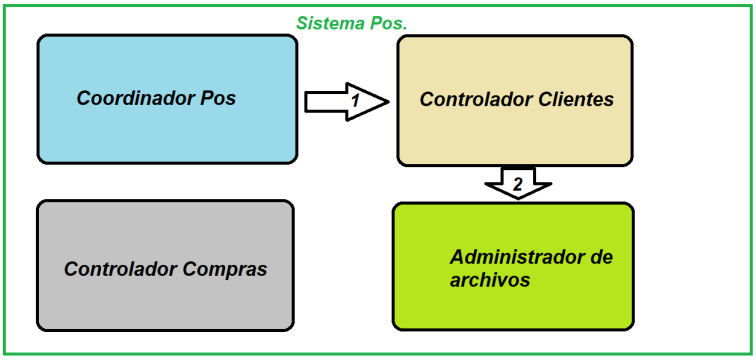

Figura 15: Secuencia de ejecución de la colaboración

- **Cerrar una compra y actualizar puntos de cliente:** Se inicia al recibir la instrucción por parte de la interfaz.

    1. El coordinador le indica al controlador de compras que cierre la compra.
    2. El controlador retorna la compra al coordinador.
    3. El coordinador envía la compra al controlador de clientes para que este se encargue de calcular los puntos a sumar, y los asigne al cliente correspondiente.

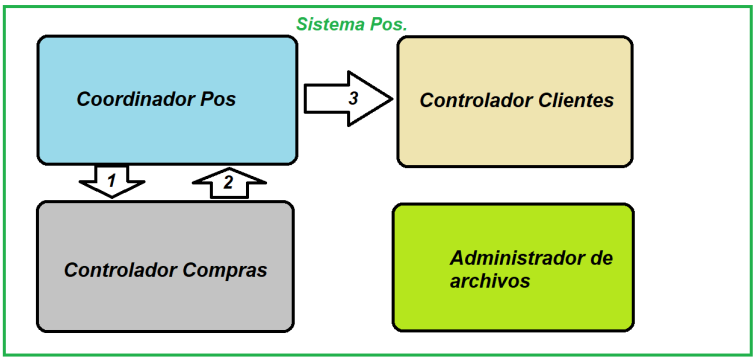

Figura 16: Secuencia de ejecución de la colaboración

- **Cerrar una compra y generar factura:** Esta colaboración se inicia igual que la anterior; cuando el cajero indica mediante la interfaz el cierre de la compra. Se separa de la colaboración anterior porque sirve a un requerimiento relacionado pero diferente.

    1. El coordinador le indica al controlador de compras que cierre la compra.
    2. El controlador retorna la compra al coordinador.
    3. El coordinador envía la información de la compra a la interfaz para que esta se encargue de mostrar los valores de la compra.

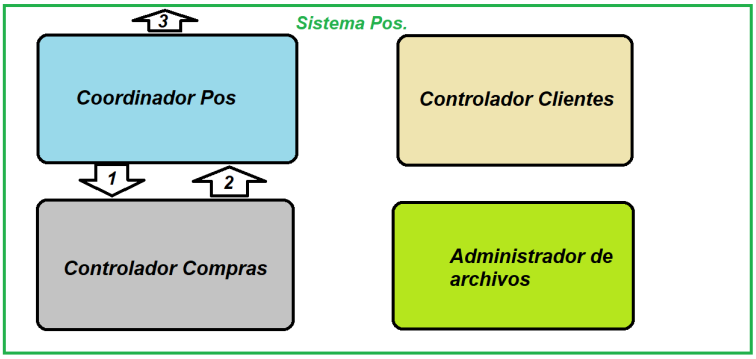

Figura 17: Secuencia de ejecución de la colaboración

De acuerdo a estas colaboraciones las asociaciones entre los componentes se muestran en la figura 18.

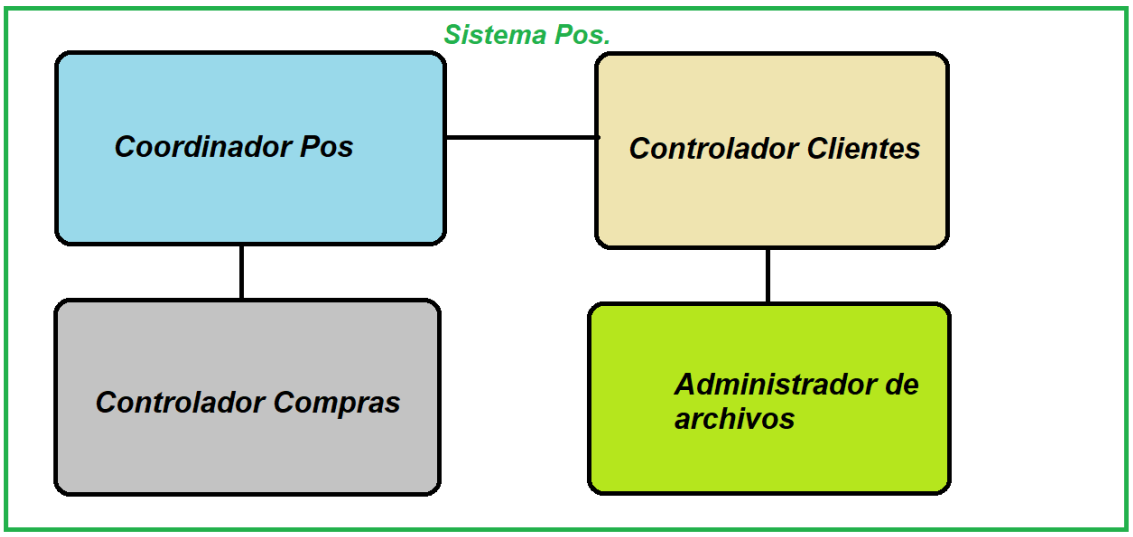

Figura 18: Asociaciones entre componentes de acuerdo a las colaboraciones para la iteración intermedia del sistema Pos.

3.1.2. Nivel 2

Se procede a descomponer los elementos propuestos en el nivel anterior.

Controlador clientes. Componentes.

Recordando las responsabilidades asignadas en la tabla 2, A partir de estas responsabilidades decidimos que lo mejor es dividir el componente en 2 componentes independientes. Uno, el administrador de clientes, que se encargara de añadir, buscar, y ejecutar acciones sobre los clientes cargados (controlador), mientras el otro se ocupa de mantener, retornar y modificar la información del cliente especifico (una clase cliente).

Responsabilidades.

El controlador se encarga de mantener y buscar por la cédula entre los clientes cargados y los nuevos clientes creados. El controlador se encarga de la creación de nuevos clientes. El controlador se ocupa de actualizar los puntos del cliente luego de haber cerrado una nueva compra; recibe la compra del cliente, de la cual obtiene los puntos redimidos en la compra, los puntos obtenidos por la compra, y los puntos extra dados por promociones, suma estos valores y el resultado lo suma a los puntos actuales del cliente.

La clase cliente simplemente tiene la responsabilidad de representar a un cliente, retornar sus atributos, y ser capaz de sumar los puntos indicados por el controlador.

Colaboraciones.

La colaboración de *cerrar una compra y actualizar puntos del cliente* no cambia fundamentalmente, solo que ahora se especifica que el coordinador envía la compra específicamente al objeto administrador de clientes para que este determine a que cliente debe sumar puntos y cuantos puntos debe sumar.

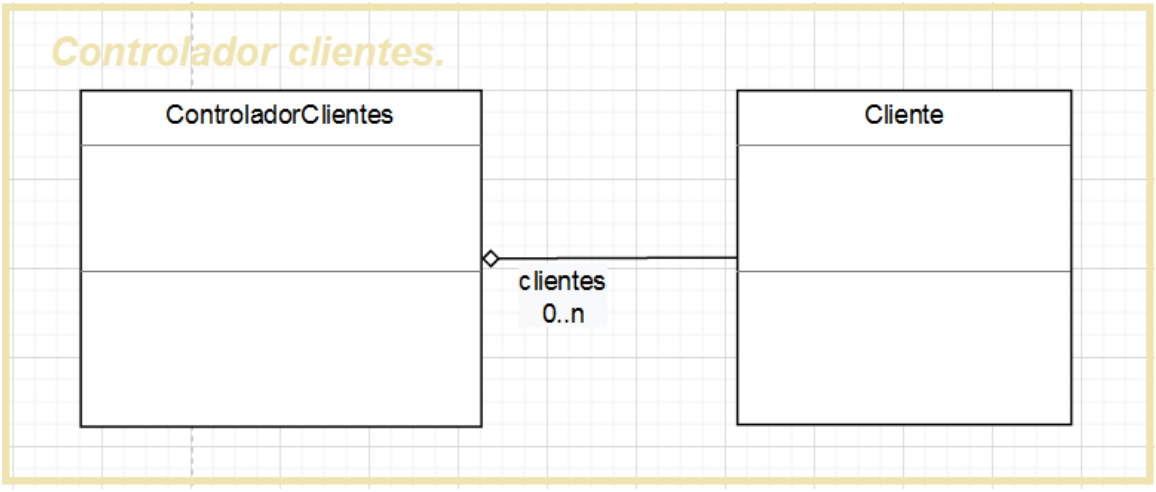

Figura 19: División del controlador de clientes planteado en el nivel 1.

Controlador de compras Componentes.

Similar al caso del controlador de productos, nos parece conveniente hacer una clase *Com-*

pra que se encargue de modelar la compra, y una clase *Admin compra*, que se encargue de recibir y hacer las solicitudes necesarias para cumplir con las responsabilidades de la tabla 2.

Dado que una compra esta compuesta por diferentes entradas de cierta cantidad de productos, también decidimos tener una clase *entrada*, la cual modela cada producto agregado a la compra y la cantidad de producto agregado. La clase *entrada* se incluye para facilitar la generación de la factura y el calculo del valor final de la compra, ya que toda la información de la compra se obtendrá iterando sobre las entradas y pidiendo los valores de sus atributos.

En cuanto a las promociones, se implemento una clase abstracta *Promoción*, de la cual heredan las clases *Descuento*, *Multiplicador*, y *Regalos*. Cada una de las clases cuenta con un mensaje para presentar en la factura en caso de ser aplicadas, el código del producto al que aplica, y la fecha en la que vencen. Además, cada clase implementa los métodos *VerificarAplicacion()* y *AplicarCompra()*; ambos reciben la compra que se ha cerrado por parámetro para modificarla con la promoción, si está aplica. Al cerrar una compra, está se entra como parámetro a todas las promociones cargadas. La implementación se hizo así para poder usar la herencia y ahorrar código haciendo 3 clases diferentes desde cero. La aplicación de las promociones se hace al cerrar la compra ya que si se hiciese al ingresar cada entrada, el programa tardaría más tiempo.

El caso de los cambos es particular, porque esté trata con más de un producto. Además hay que revisar que haya disponibilidad de cada uno de los productos que componen el combo. Para poder reutilizar el código que añade entradas individuales, se creo la clase *Combo*, la cual tiene toda la información que caracteriza a un combo. En la interfaz el cajero puede presionar el botón de agregar combo e ingresar el código del combo, luego de esto el sistema añadirá una entrada por cada producto del combo, modificando el precio de esta entrada en base al descuento del combo. Antes de añadir las entradas se verifica la disponibilidad de los productos, en caso de que no hay suficiente cantidad para el combo, se lanza una excepción *NoCantidadComboException* y no se agrega el combo.

Para cargar las promociones se Creo la clase *CargadorPromociones*. Esta clase implementa el patrón Singleton ya que solo se precisa de una clase del cargador. Además, al ser la instancia accesible desde el resto del sistema, no es necesario crear métodos intermedios entre la instancia y el controlador compras.

Al finalizar una compra esta se retorna a la interfaz, la cual se encarga de imprimir los valores importantes en una factura.

Responsabilidades y colaboraciones.

La responsabilidad de *Añadir un producto* se hace ahora como una colaboración entre el administrador de compras y la compra. Luego de verificar existencias mediante la colabora-

ción con un sistema de inventarios externo, el administrador de compras envía la información a la compra para que esta cree una nueva instancia que represente la compra del nuevo producto. Las responsabilidades 11, 12, y 13 se vuelven colaboraciones entre el controlador de compras, la Clase Compra, la clase Producto, y la clase Combo.

La responsabilidad 14 se asigna a la instancia de la clase *CargadorPromociones*.

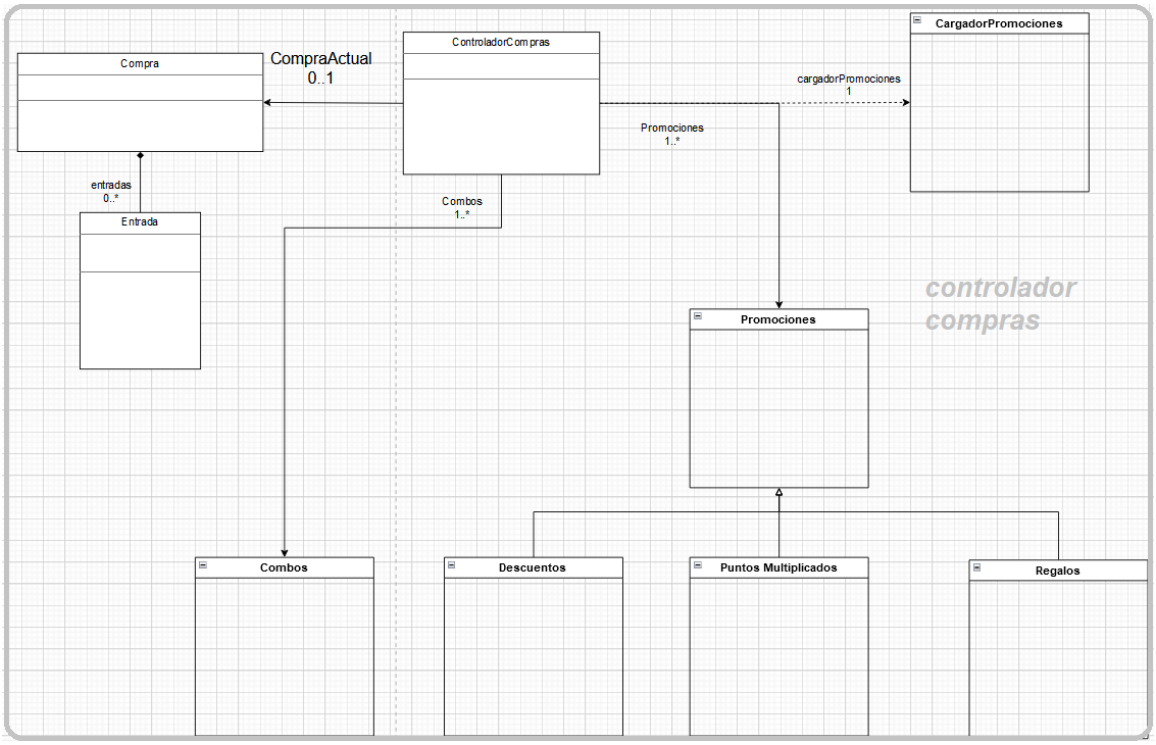

Figura 20: División del controlador de compras planteado en el nivel 1.

Coordinador Pos. Consideramos que las responsabilidades asignadas a este componente están lo suficientemente relacionadas entre sí como para que se implemente en una sola clase. La clase se llamara *"Sistema Pos"* y tendrá como atributos a instancias de las clases *AdminCompras* y *AdminClientes*, justificadas anteriormente.Todas las responsabilidades mostradas en la tabla 2 se delegan a la clase *CoordinadorPos*, de modo que no existen colaboraciones entre clases para este caso.

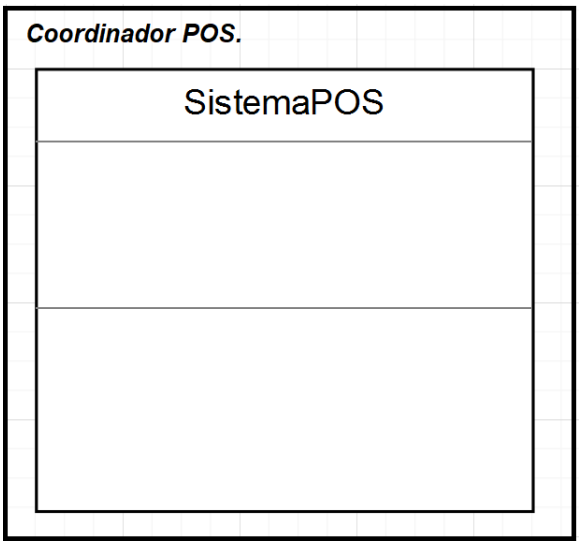

Figura 21: División del coordinador Pos planteado en el nivel 1.

Administrador de archivos. Las responsabilidades del administrador de archivos solo serán el poder serializar un cliente y cargar todos los clientes guardados en el disco. Estas responsabilidades se asignan a una única clase que se comunica directamente con la clase *AdminClientes* y con los archivos escritos.

Dada la diferencia en la naturaleza de las responsabilidades, se prefirió hacer esta clase aparte en vez de delegar sus responsabilidades al *AdminClientes*.

Figura 22: División del administrador de archivos planteado en el nivel 1.

Métodos y atributos. De acuerdo a las clases obtenidas anteriormente, se presenta un diagrama de alto nivel que pretende presentar las relaciones ya descritas entre clases.

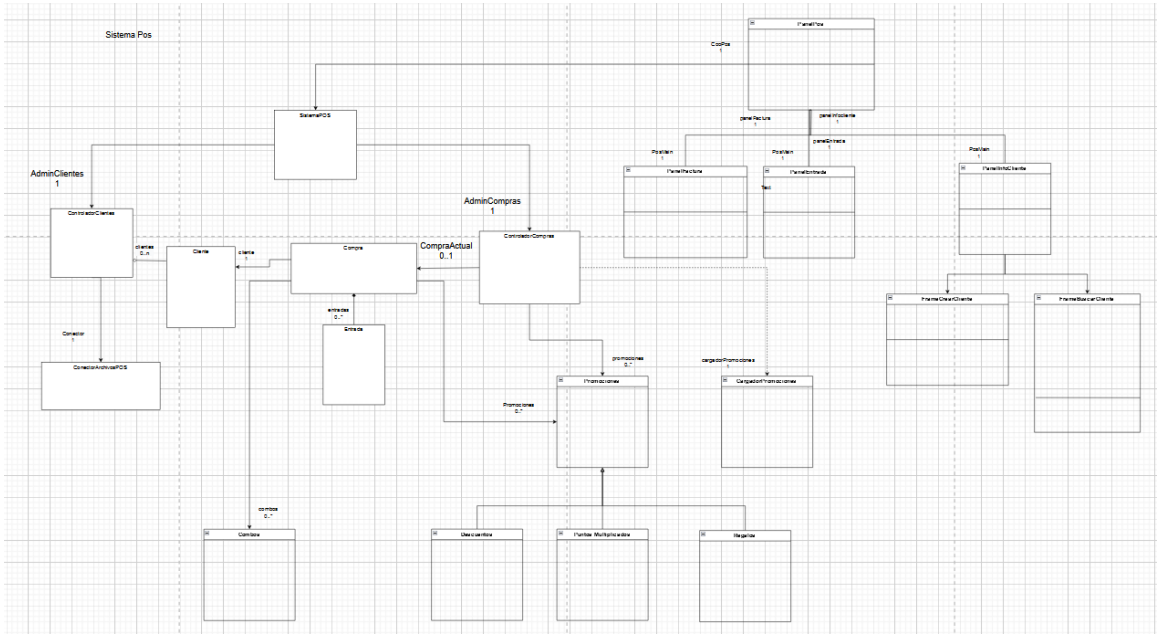

Figura 23: Diagrama UML de alto nivel de la lógica del sistema POS.

A continuación se presentan los métodos y atributos que tendrá cada una de las clases, con sus respectivas justificaciones.

SistemaPOS: Esta clase solo tendrá como atributos instancias de cada uno de los controladores, tal y como se indica en la figura 23. No se añaden más atributos ya que estos 2 son suficientes para cumplir con las responsabilidades asignadas en la tabla 2; cabe recordar que el coordinador no se dividió durante el nivel 2, así que las responsabilidades de la tabla se conservan.

En cuanto a los métodos, se necesitan los que inicien los procesos encargados de satisfacer requerimientos funcionales:

- addEntrada()
- registrarCliente()
- cerrarCompra()

- nuevaCompra()
- AddCombo()

También se necesitan métodos que ejecuten requerimientos no funcionales como:

- getCliente()
- cargarClientes()

Los nombres de todos estos métodos son bastante auto explicativos así que no se ahondará más en ellos.

Controlador Clientes: Los únicos 2 atributos de esta clase son una instancia del cargador de archivos y un HashMap con clientes como valores. En el nivel 2 se concluyó que el controlador se encargará de crear clientes; la persistencia solicitada hace que se requiera entonces que cuente con acceso a la clase que interactúa directamente escribiendo y leyendo en la memoria.

Respecto a la búsqueda de clientes, otra responsabilidad del controlador, proponemos un HashMap con todos los clientes registrados. La anterior propuesta se da ya que asumiendo cédulas únicas que identifican a cada cliente, esta estructura nos permite acceder a los clientes de forma rápida y natural.

Respecto a los métodos, tenemos a los que siguen la secuencia de los métodos de *sistemaPos* y hacen parte de una misma colaboración; estos métodos tendrán el mismo nombre que los de *sistemaPos*, para facilitar el entendimiento del código.

Un método especial del controlador es el método *SumarPuntos()*, el cual recibe una compra y un cliente. De la compra obtiene los puntos que se han de adicionar y los puntos que canjea el cliente, de acuerdo a estos estable los puntos que el cliente tiene luego de la compra.

Cliente: La responsabilidad de esta clase es modelar a los clientes. Los atributos serán los especificados directamente en el enunciado. Como métodos se tienen *getters* y *setters*.

ConectorArchivosPOS: Esta clase no tiene atributos; anteriormente dijimos que la separamos del coordinador por lo especializado de sus responsabilidades. Como métodos se *CrearCliente()*, que recibe un cliente por parámetro y lo escribe en la memoria permanente, y CargarCliente(), el cual lee los archivos guardados en memoria y retorna al controlador un HashMap con todos los clientes guardados de la ultima vez que se ejecutó la aplicación.

Controlador compras: Uno de sus atributos es una instancia de Compra que representa la compra que se esta realizando en el momento. Además, tiene una lista de promociones, sobre la cual se itera para validar la aplicación de las promociones sobre las entradas de la compra. También tiene un hash map de Combos, donde las llaves son los códigos de los combos; esto porque los combos se buscan usando el código ingresado por el cajero.

Esta clase también tiene el código principal algunos de los métodos llamados desde el sistemaPOS:

- addcombo()
- addEntrada()
- AplicarPromociones()
- CerrarCompra()
- NuevaCompra()

También tiene métodos invocados por el constructor para cargar todas las promociones:

- CargarCombos()
- CargarDescuentos()
- CargarMultiplicadores()
- CargarRegalos()

Compra: Tiene un cliente que representa al cliente que realiza la compra, una lista de entradas registradas por el cajero, una lista de promociones con los mensajes de las promociones que aplican a la compra, un atributo *puntos* con los puntos que el cliente desea redimir en la compra, y un atributo *puntosExtra* con los puntos adicionales que da la compra por promociones.

Algunos de sus métodos más importantes son:

- DescontarPrecioUltimaEntrada(): Se usa en la colaboración que lleva a añadir un combo a la compra. La entrada del producto se hace con los métodos usuales, pero al final se quiere hacer el descuento del combo; una vez añadida la entrada a la lista de entradas, se invoca este método para generar el descuento.

- DarPuntosAgregados(): Retorna los puntos que se suman por la compra, considerando los puntos normales y los puntos extra.
- AgregarPuntos(): Recibe los puntos que el cliente desea redimir en la compra y verifica que ese numero sea mayor a la cantidad de puntos con la que cuenta el cliente.
- ValidarPuntos(): Verifica que los puntos canjeados no sean demasiados como para llevar el precio final de la factura a un valor negativo.

Combo: Sus atributos son su código, su nombre, un array con los códigos de los productos que incluye, un array con las cantidades de cada producto, el descuento del combo, y una fecha de vencimiento.

Sus métodos son los getters de sus elementos y un método *VerificarFecha()*, que retorna true si el combo ha vencido respecto a la fecha de la maquina en la que se ejecuta el programa.

Entrada: Para modelar una entrada se incluyen el nombre del producto, su código, la cantidad de producto comprado, y el costo de lo comprado. Al igual que en *Cliente*, los métodos de esta clase son solo *getters* y *setters*.

Diagramas y diseño final: A continuación se presentan algunos diagramas importantes del diseño final propuesto.

En la figura 24 se muestra el diagrama de clases final, con sus atributos, métodos, y relaciones.

Por otra parte, en las figuras 25 y 26 se muestran los diagramas de secuencia de las colaboraciones para el registro de clientes y de compras. Dado que en nuestro diseño las compras funcionan como contenedores de entradas, el diagrama de la figura 26 muestra la secuencia que toma crear una compra e ingresar una entrada a la compra.

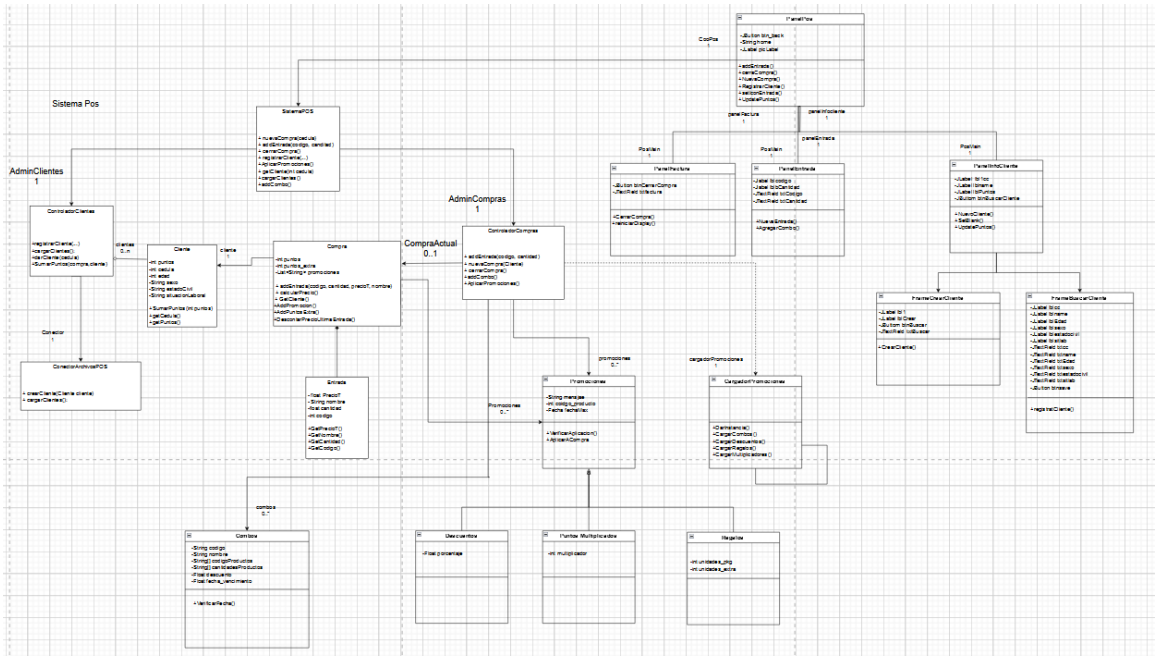

Figura 24: Diagrama de clases UML completo del diseño final.

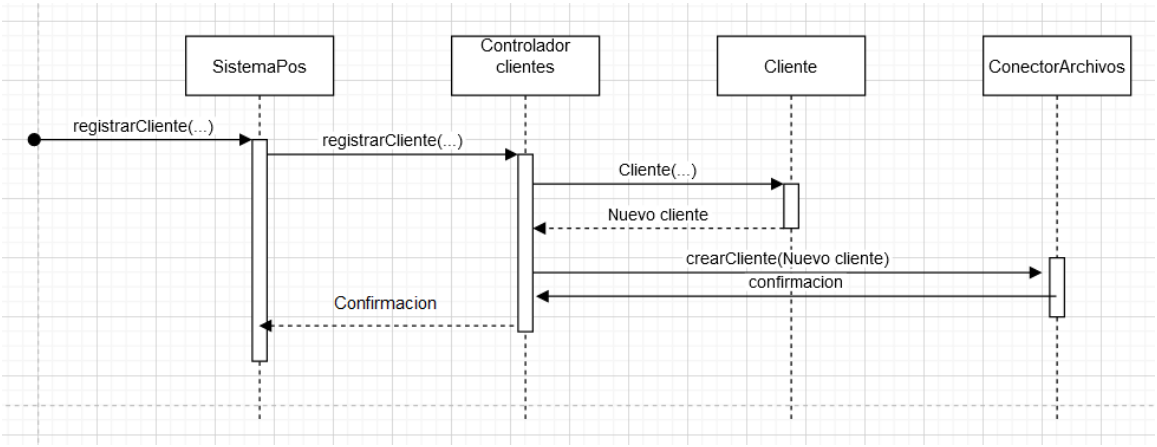

Figura 25: Diagrama de secuencia de la colaboración que lleva al registro de un cliente en el sistema Pos.

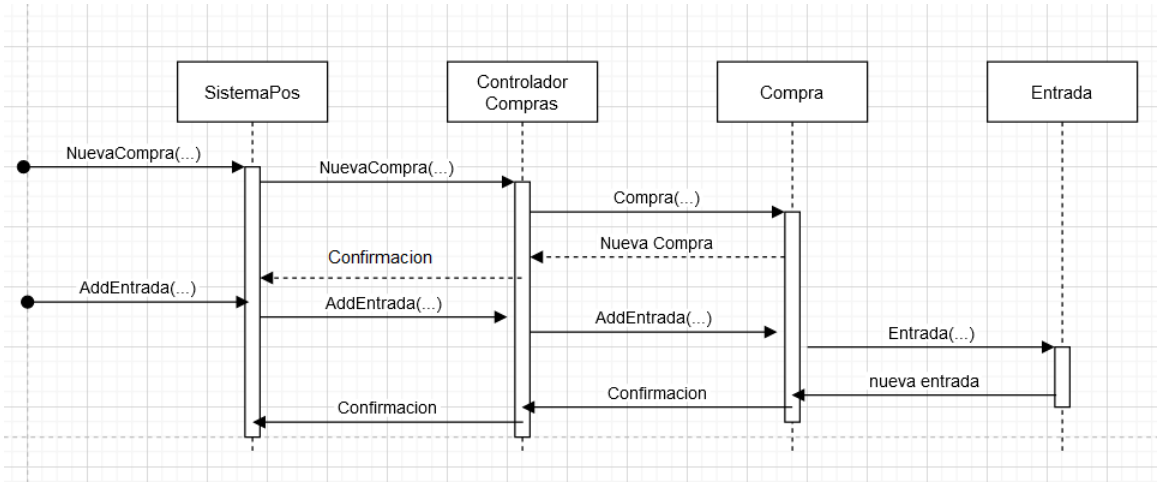

Figura 26: Diagrama de secuencia de la colaboración que lleva a la creación de una compra y el registro de una entrada.

En la figura 27 se muestra el diagrama de secuencia para la actualización de los puntos de un cliente luego de que este haya cerrado una compra.

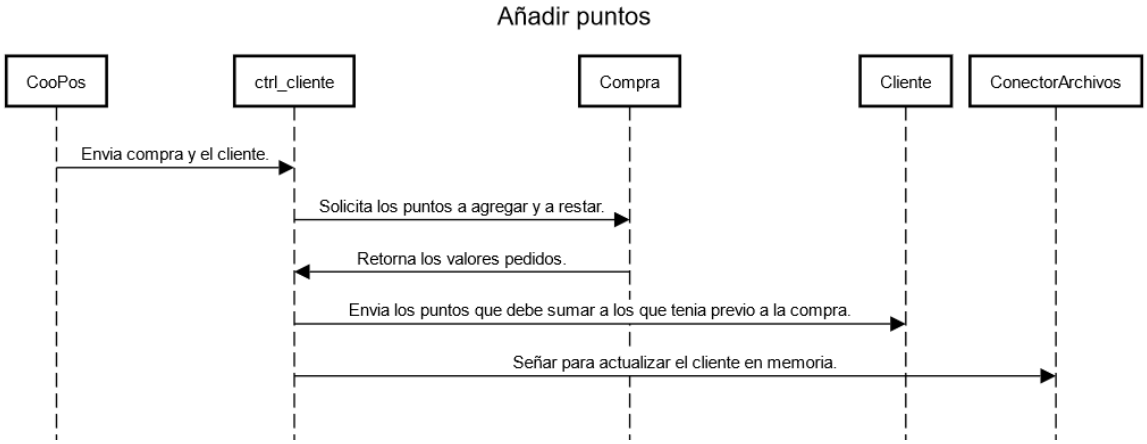

Figura 27: Diagrama de secuencia de la colaboración que lleva actualizar los puntos de un cliente tras una compra.

En la figura 28 se muestra el diagrama de secuencia para aplicar las promociones. Es necesario mencionar que el diagrama asume que la promoción a aplicar no esta vencida. El paso de modificación de entradas puede hacerse hasta tantas veces como numero de entradas tenga la compra. El proceso que se da desde $ctrl_compra$ se ejecuta tantas veces como promociones se hayan cargado.

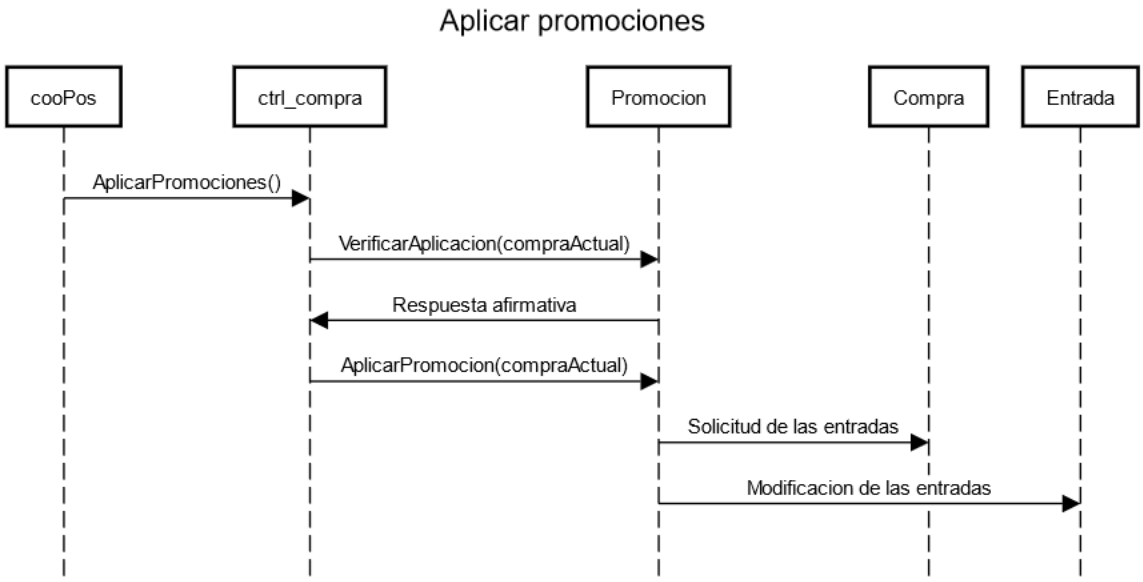

Figura 28: Diagrama de secuencia de la aplicación de las promociones.

3.2. GUI del sistema

3.2.1. Nivel 1

Roles y componentes. De manera preliminar, y similar al diseño de la lógica, se van a plantear 3 componentes principales en la interfaz. Uno con la tarea de organizar y contener otros elementos; los otros 2 componentes intercambian información directamente con el usuario. Estos 2 componentes se centran en lo referente a compras y clientes, los objetos del dominio en los que se centra POS.

- **Panel Pos:** Es necesario un elemento que se encargue de organizar y contener a los demás elementos de la interfaz. Funciona como un puente de comunicación entre la interfaz y la lógica del sistema ya que es el único elemento de la interfaz que intercambia información directamente con el *Coordinador pos*. Por lo anterior, este rol es un *interfacer*.
- **Interfaz clientes:** Su función es permitir las interacciones con el cajero que se requieran para cumplir los requerimientos de crear nuevos clientes y de buscar clientes.
- **Interfaz Compras:** Se encarga de interactuar con el cajero. Recibe señales del cajero referentes a la compra actual y es capaz de mostrar información relacionada.

Responsabilidades. Las responsabilidades asiganadas a cada rol se pueden ver en la tabla 3.

#	Responsabilidad	Componente
1	Organizar y contener otros componentes	Panel Pos
2	Puente de comunicacion entre interfaz y logica	
3	Recibir la informacion necesaria para registrar un nuevo cliente	Interfaz clientes
4	Recibir la informacion necesaria para buscar un cliente entre los existentes	
5	Mostar la informaciondel cliente sobre el cual se hace la compra actual	
6	Recibir la informacion necesaria para añadir una entrada a una compra	Interfaz compras
7	Recibir la señal de cerrar una compra	
8	Imprimir la factura final de la compra	
9	Mostrar una imagen del ultimo producto añadido a la compra	

Tabla 3: Responsabilidades iteración intermedia de la interfaz gráfica POS.

Colaboraciones. Las colaboraciones más importantes de la aplicación se muestran a continuación. Solo se explica la interacción entre elementos de la interfaz. La información proveniente de la lógica entra por el *Panel Pos*.

- **Crear una nueva compra:** Una nueva compra esta directamente relacionada a la búsqueda de un cliente, el cual está únicamente identificado por su cédula.

    1. La *Interfaz clientes* recibe una señal del usuario de iniciar la búsqueda de un cliente.
    2. Recibe la cédula del cliente y la envía al Panel Pos para que esté envié la señal a la lógica.
    3. Recibe la información del cliente desde el Panel Pos y muestra algunos datos del cliente, lo cual indica que se ha iniciado una nueva compra para ese cliente.

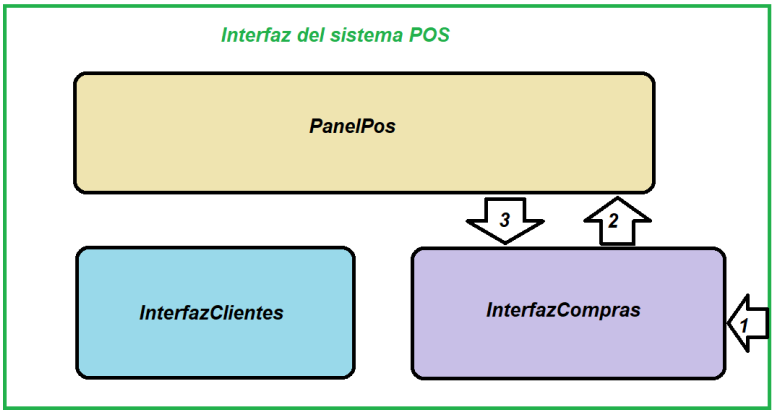

Figura 29: Colaboraciones para la creación de una nueva compra.

- **Registrar un nuevo cliente:**

  1. La *Interfaz clientes* recibe una señal del usuario de crear un nuevo cliente.
  2. La *Interfaz clientes* recibe los datos del nuevo cliente por parte del usuario.
  3. La *Interfaz clientes* recibe la señal de crear un nuevo cliente con los datos suministrados. Se envían los datos al Panel Pos.

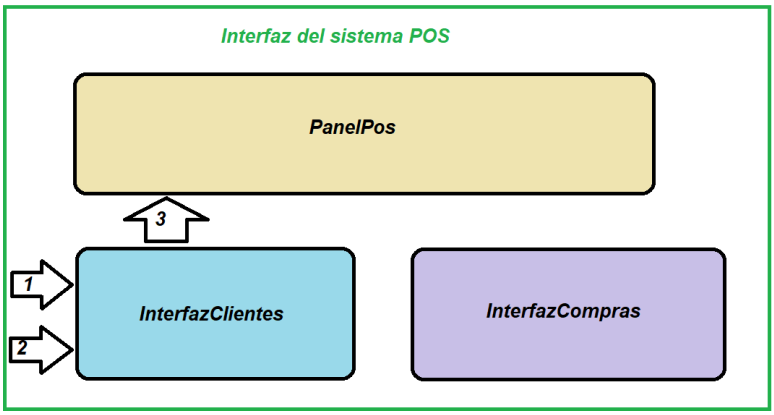

Figura 30: Colaboraciones para la creación de un nuevo cliente.

- **Añadir una entrada a la compra actual:**

  1. La *Interfaz Compras* recibe el código y la cantidad de producto que se quiere añadir a la compra

2. La *Interfaz compra* envía la información al Panel Pos, y este se encarga de iniciar la ejecución en la lógica.
3. El panel Pos retorna una respuesta de la lógica a la *Interfaz Compras*. Si la respuesta es afirmativa, se muestra un cuadro de dialogo que comunica al usuario el éxito de la operación; de lo contrario, el cuadro de dialogo indica que no hay suficiente cantidad del producto solicitado. Si la respuesta es afirmativa, la *Interfaz Compras* muestra una imagen relacionada al ultimo producto añadido.

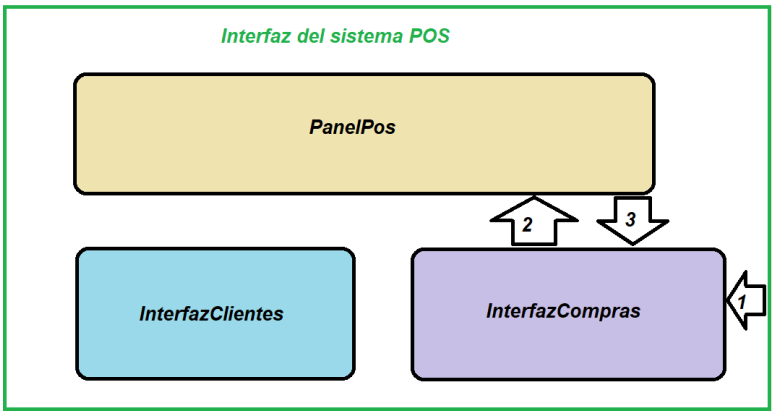

Figura 31: Colaboraciones para añadir una entrada a la compra actual.

- **Cerrar una compra y generar factura:**

1. La *Interfaz Compras* recibe la señal de terminar la compra actual.
2. La *Interfaz Compras* le transmite la señal al Panel Pos.
3. El Panel Pos retorna a la *Interfaz Compras* un objeto de tipo Compra desde la interfaz.
4. La *Interfaz Compras* extrae información del archivo e imprime una factura para que el cliente pueda verla. Luego de este, *Interfaz Compras* manda, a través de PanelPos, el costo total de la compra para que se actualicen los puntos del cliente en la lógica.

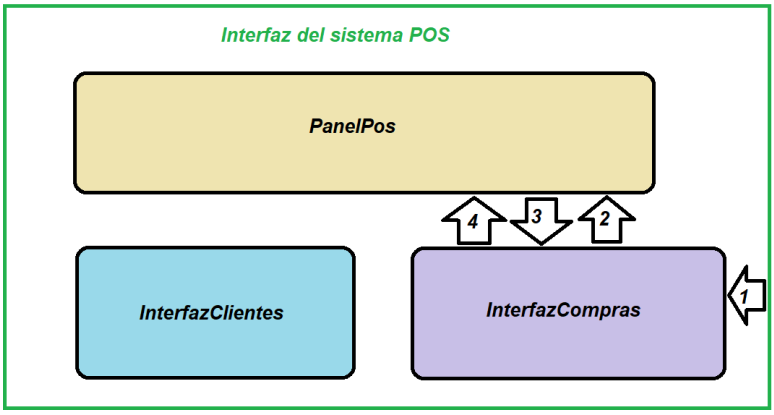

Figura 32: Colaboraciones para cerrar la compra actual.

Por ultimo, en la figura 33, se muestran las asociaciones entre los componentes.

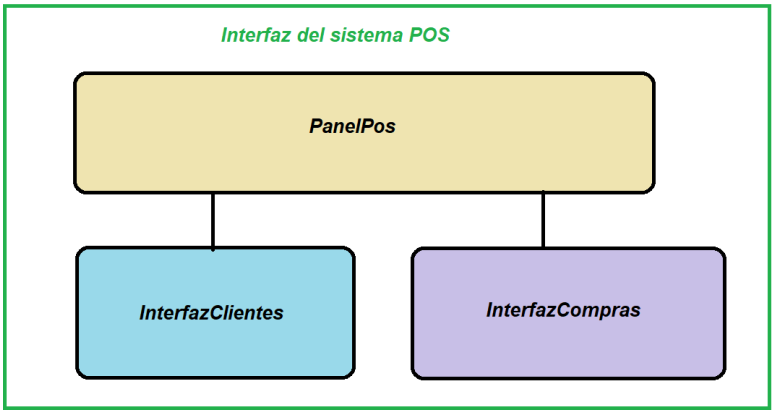

Figura 33: Asociaciones entre componentes de la iteración intermedia del diseño de la interfaz del sistema POS.

3.2.2. Nivel 2

Panel Pos El Panel Pos se implementa como un único objeto de una clase que hereda de JPanel. El objetivo de esté componente solo es organizar y almacenar los otros elementos gráficos, así como hacer de puente entre estos elementos y la lógica, de modo que no hay necesidad de dividirlo.

Las responsabilidades del componente se mantienen igual que las mostradas en la tabla 3. En la figura 34 se muestra la descomposición final del componente.

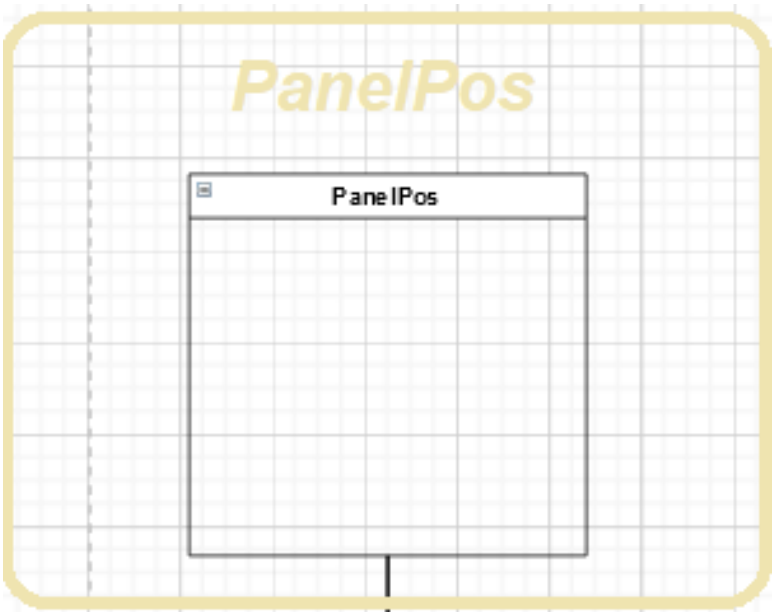

Figura 34: Descomposición final del panelPos.

Interfaz Clientes Componentes.

Se opta por repartir las responsabilidades asignadas a la interfaz en 3 componentes, de manera que cada uno se especialice una sola responsabilidad:

a) *PanelInfoCliente:* Se trata de un objeto de una clase que implementa JPanel. Contiene 3 elementos JLabel donde mostrar la cédula, el nombre, y los puntos del cliente. Este panel es visible al usuario desde que entra al sistema POS y tiene un botón que indica "buscar"; al presionar se inicia el proceso de buscar un cliente en especifico para iniciar una compra.

b) *FrameBuscarCliente:* Para evitar tener siempre a la vista el campo donde se ingresa la cédula del cliente a buscar, se decidió incluir esté componente en un JFrame, el cual emerge al pulsar el botón 'buscar', en *PanelInfoCliente*.

   FrameBuscarCliente también presenta un hipervínculo con el mensaje 'Nuevo cliente', el cual al activarse hace emerger un segundo JFrame.Se dio esta ubicación al vinculo para evitar sobrecargar el PanelPos.

c) *FrameCrearCliente:* Se despliega al presionar 'Nuevo cliente' en *FrameBuscarCliente*. Contiene 6 cuadros de texto en los que ingresar los datos más relevantes del cliente.

Una vez llenos los campos, el usuario puede presionar el botón guardar; luego de esto, se cierra el cuadro.

De nuevo, estos 2 Frames se implementan porque su contenido no es de uso continuo durante el proceso de compra, de modo que se puede ahorrar espacio si se muestran los cuadros solo cuando son requeridos. En la figura 35 se muestra la descomposición final del componente.

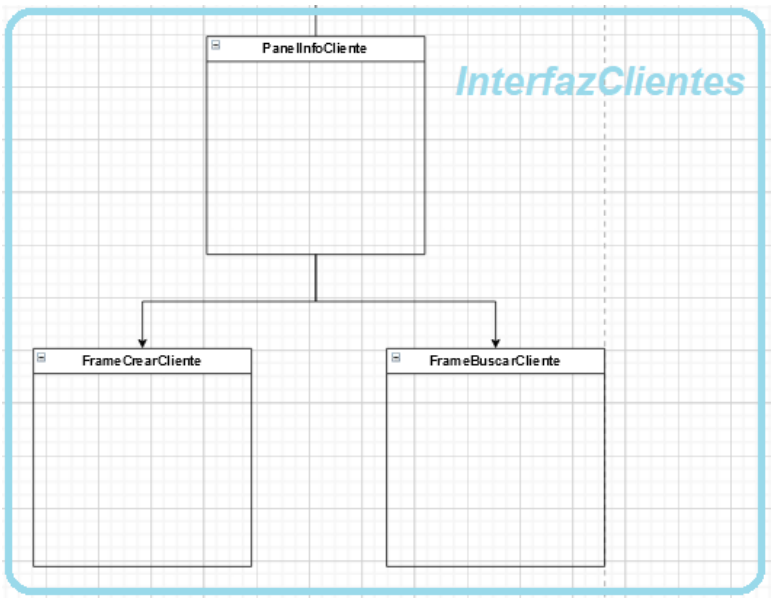

Figura 35: Descomposición final de InterfazClientes.

Responsabilidades y colaboraciones.

Como se ha descrito antes, y tomando las responsabilidades mostradas en la tabla 3, a PanelInfoCliente se le asigna la responsabilidad 5, a FrameBuscarCliente se le asigna la responsabilidad 4, y a FrameCrearCliente se le asigna la responsabilidad 3.

Las colaboraciones necesarias para crear y buscar un cliente consisten en que FrameCrearCliente y FrameBuscarCliente envien los datos ingresados por el usuario directamente a Panel Pos. Las nuevas colaboraciones se muestran más detalladamente en los diagramas de las figuras 37 y 36.

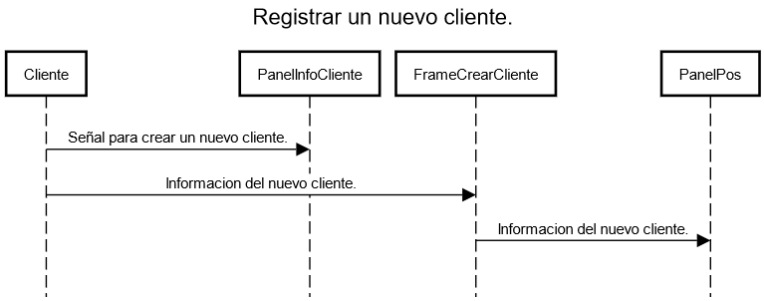

Figura 36: Colaboraciones para la creación de un nuevo cliente.

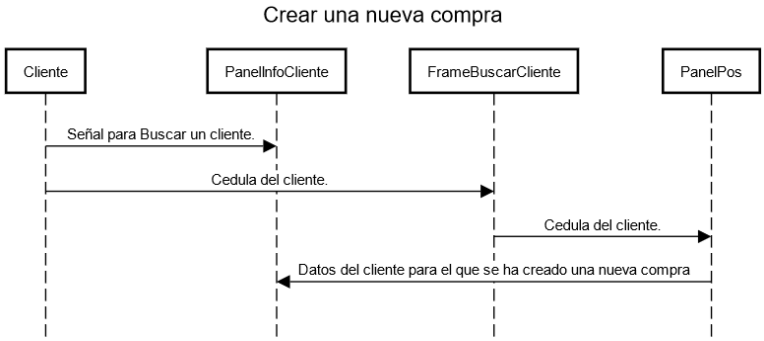

Figura 37: Colaboraciones para la creación de una nueva compra.

Interfaz Compras Componentes.

Por su relación, las responsabilidades 7 y 8 de la tabla 3 se asignan a un solo componente llamado *PanelFactura*. Las responsabilidades 6 y 9 se asignan a otro componente, llamado *PanelEntrada*.

a) *PanelFactura:* Tiene un cuadro de texto no editable, donde se imprime la factura al cerrar la compra. También cuenta con un botón que manda a que se cierre la compra.

   Al pulsar el boton de cierre de la compra, el panel imprime la factura con la información que recibe de la lógica. También manda, a través de panelPos, que se actualicen los puntos del cliente.

b) *PanelEntrada:* Tiene 2 campos de texto; uno para el código y el otro para la cantidad de producto que se quiere añadir a la compra que se esta procesando.

   Al presionar el botón no solo se agrega la entrada a la compra; también se manda al Panel Pos que cambie la imagen que se esté mostrando, por la imagen del producto agregado.

Ambos paneles permanecen siempre visibles para el usuario dado que su uso es continuo durante el proceso de registro de una compra.En la figura 41 se muestra la descomposición final del componente.

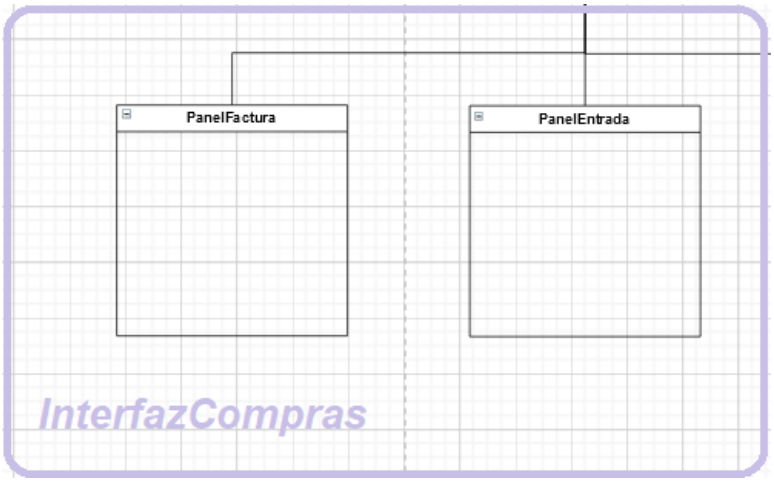

Figura 38: Descomposición final de InterfazCompras.

Por ultimo, en la figura 39 se muestra el diagrama de clases de alto nivel de la interfaz del sistema POS.

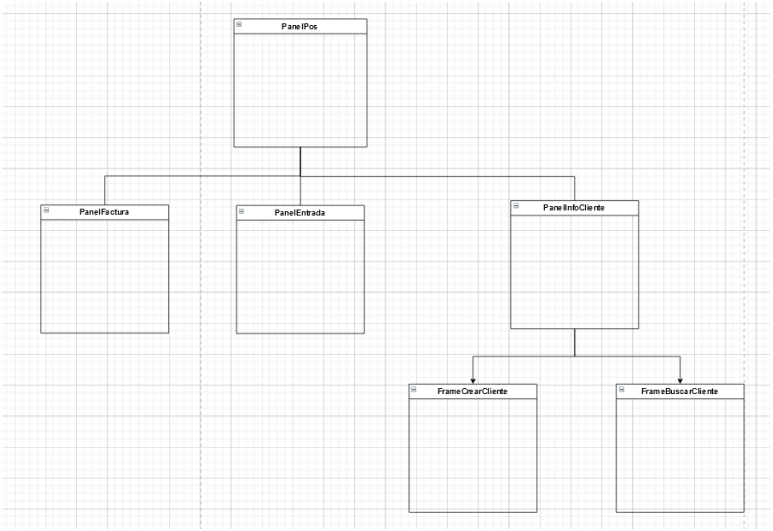

Figura 39: Diagrama de alto nivel de la interfaz POS.

Responsabilidades y colaboraciones.

Las colaboraciones necesarias para añadir una entrada a la compra son similares a las descritas en la iteración intermedia. La diferencia es que ahora las acciones *Interfaz Compras* en colaboración para añadir una entrada a la compra actual son realizadas por *PanelEntrada*.

De igual manera, para las colaboraciones que llevan a cerrar una compra, las acciones ejecutadas por *Interfaz Compras* en la iteración intermedia pasan a ser tomadas por*PanelFactura*. Las nuevas colaboraciones se muestran más detalladamente en las figuras 40 y 41

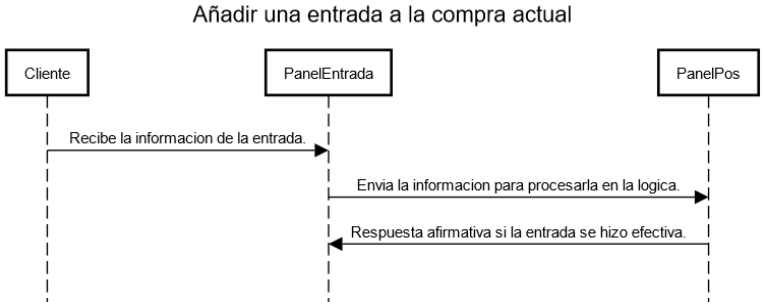

Figura 40: Colaboraciones para añadir una entrada a la compra actual.

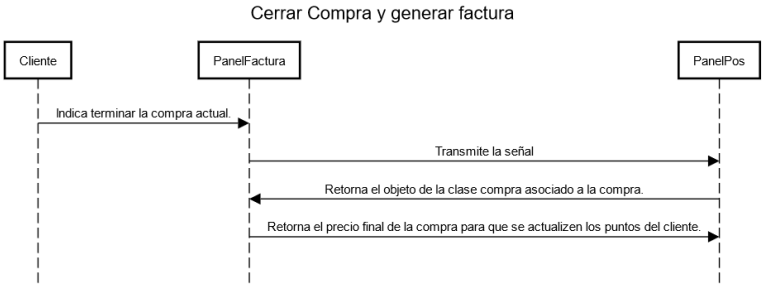

Figura 41: Colaboraciones para cerrar la compra actual.

En cuanto a las responsabilidades especificas de los componentes en esta, la iteración final, ya se han discutido. El resumen de las responsabilidades asignadas se muestra como la tabla 4.

#	Responsabilidad	Componente
1	Organizar y contener otros componentes	Panel Pos
2	Puente de comunicacion entre interfaz y logica	
3	Recibir informacion necesaria para registrar un nuevo cliente	FrameCrearCliente
4	Recibir informacion necesaria para buscar un cliente entre los existentes	FrameBuscarCliente
5	Mostrar la informacion del cliente sobre el que se hace la compra actual	PanelInfoCliente
7	Recibir la señal de cerrar una compra	PanelFactura
8	Imprimir la factura final de la compra	
6	Recibir informacion necesaria para añadir una entrada a una compra	PanelEntrada
9	Mostrar una imagen del ultimo producto añadido a la compra	

Tabla 4: Responsabilidades interfaz POS para la iteración final.

Diagramas detallados del diseño final. En la figura 42 se muestra el diagrama de clases detallado de la interfaz de la aplicación. Este diagrama cuenta con los métodos y atributos más importantes de cada clase.

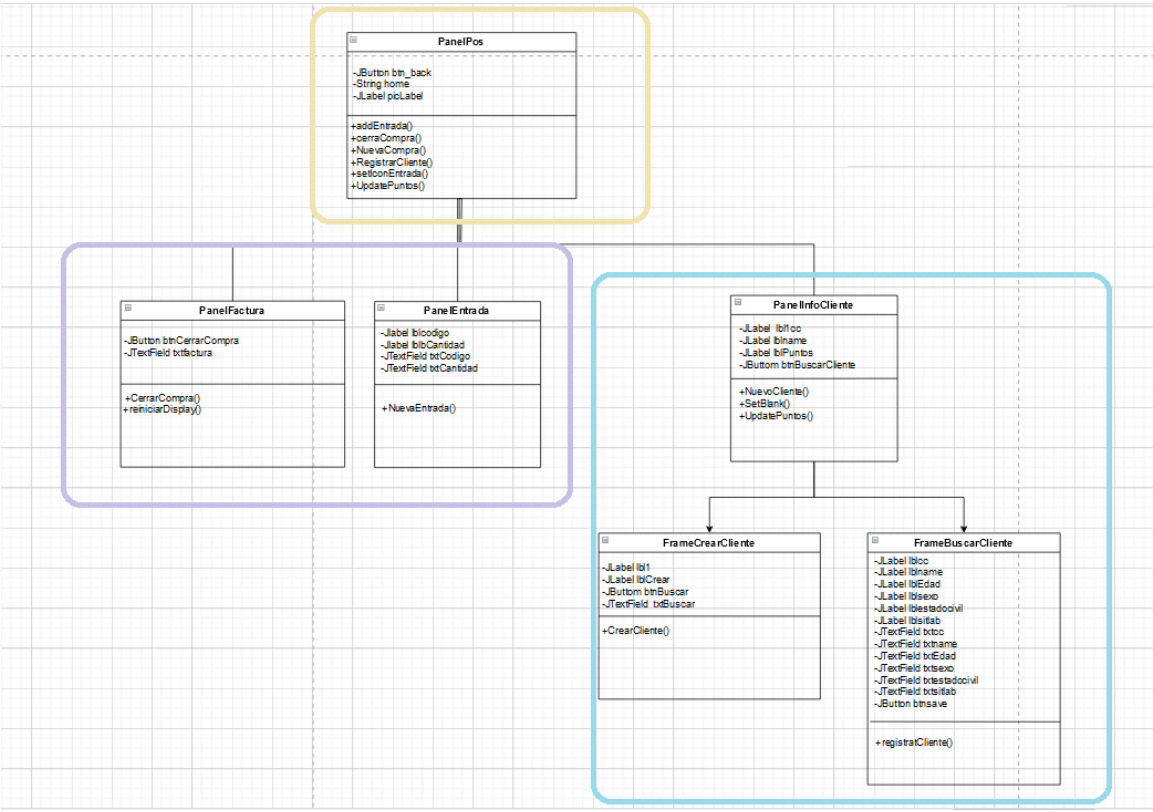

Figura 42: Diagrama detallado de la interfaz POS.

En la figura 43 se muestra el diagrama de clases detallado del sistema POS completo; interfaz y lógica. Como se puede ver, ambas partes se conectan mediante el panelPos y el coordinador pos.

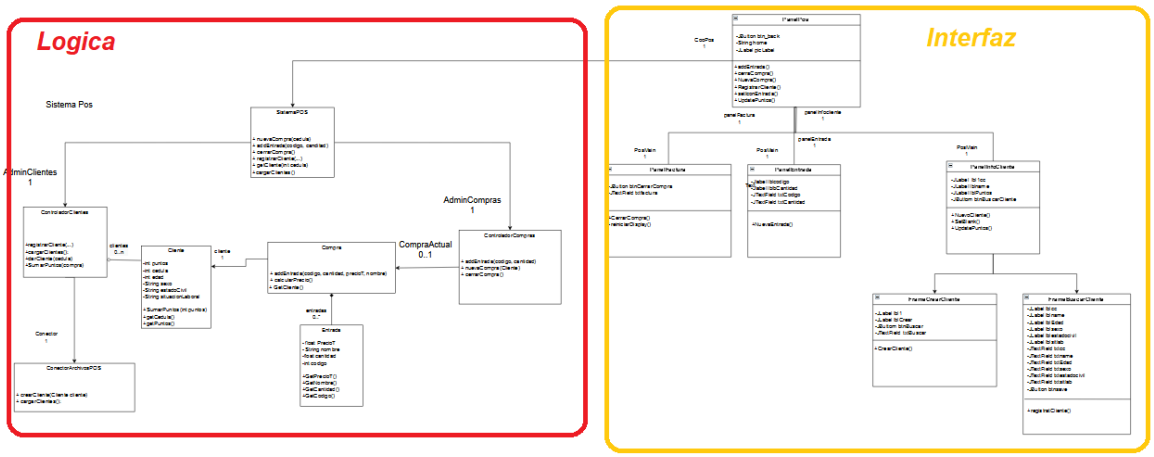

Figura 43: Diagrama de clases de todo el sistema POS.