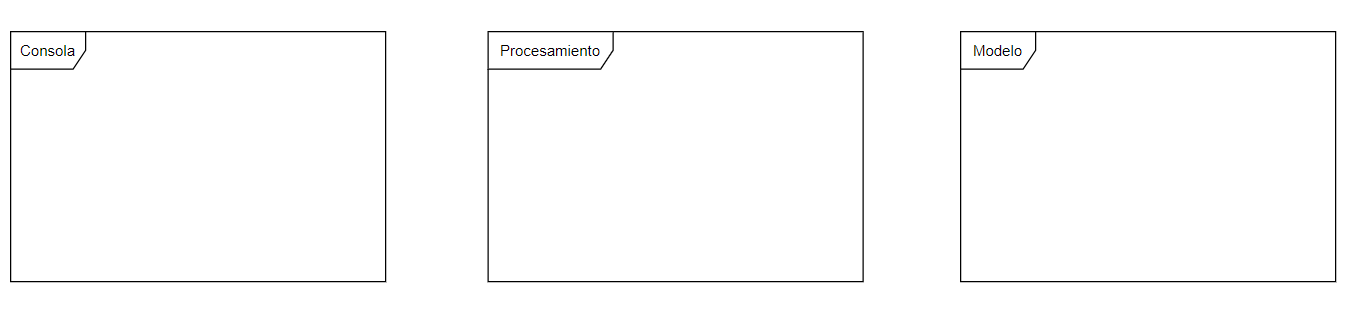
**Proyecto 1 – Diseño**

* Sergio Arango Arango - 201921814
* Juan José Sierra Alarcón - 202013642

Iniciamos separando el diseño del programa en tres partes principales: Consola, Procesamiento y Modelo.



Separamos las entidades del programa dentro de estas tres partes:

**Consola:**

* Sistema POS
* Sistema INVENTARIO

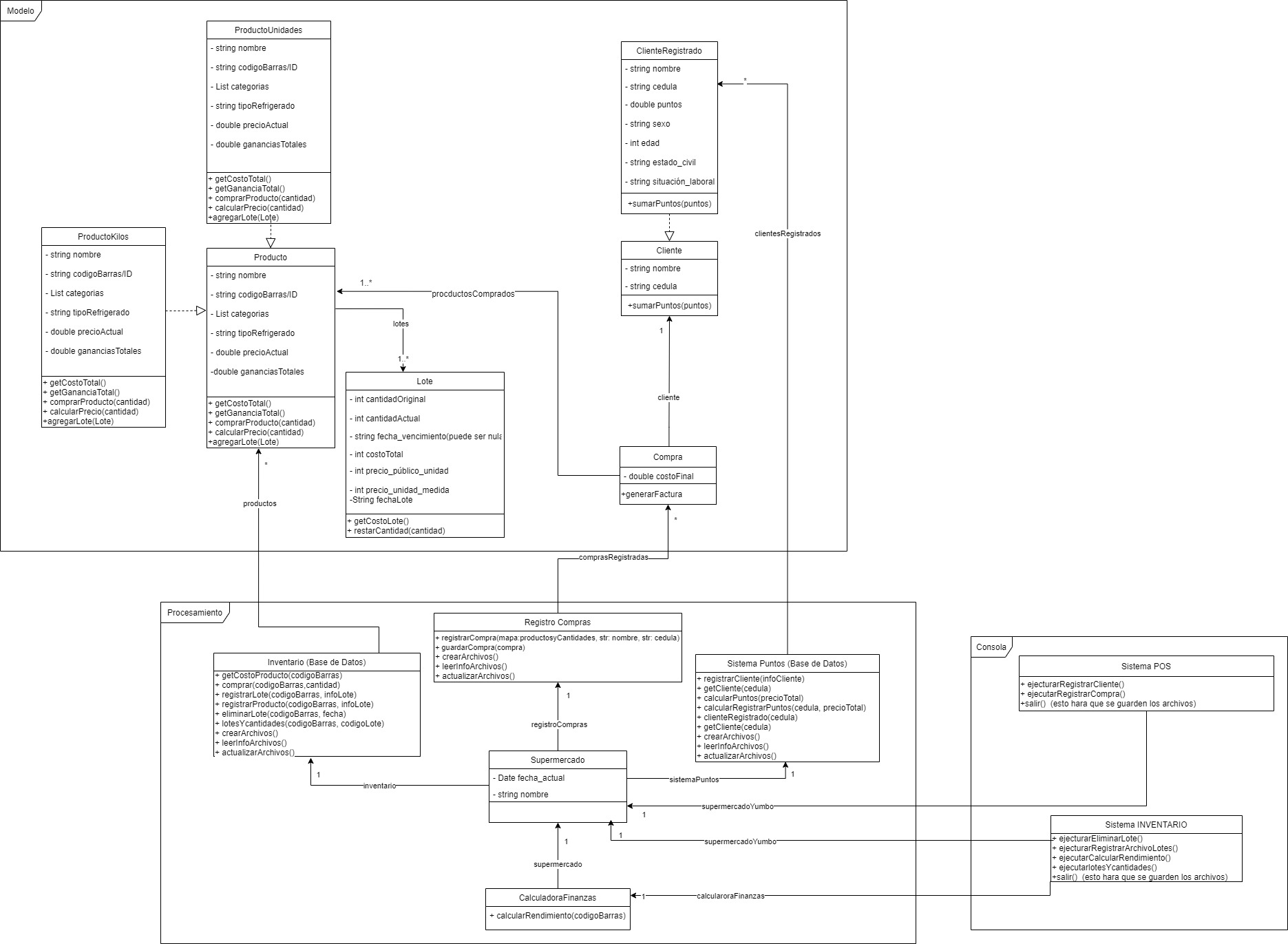
**Procesamiento:**

* Inventario (base de datos)
* Sistema de puntos (base de datos)
* Registro de compras
* Calculadora Finanzas
* Supermercado

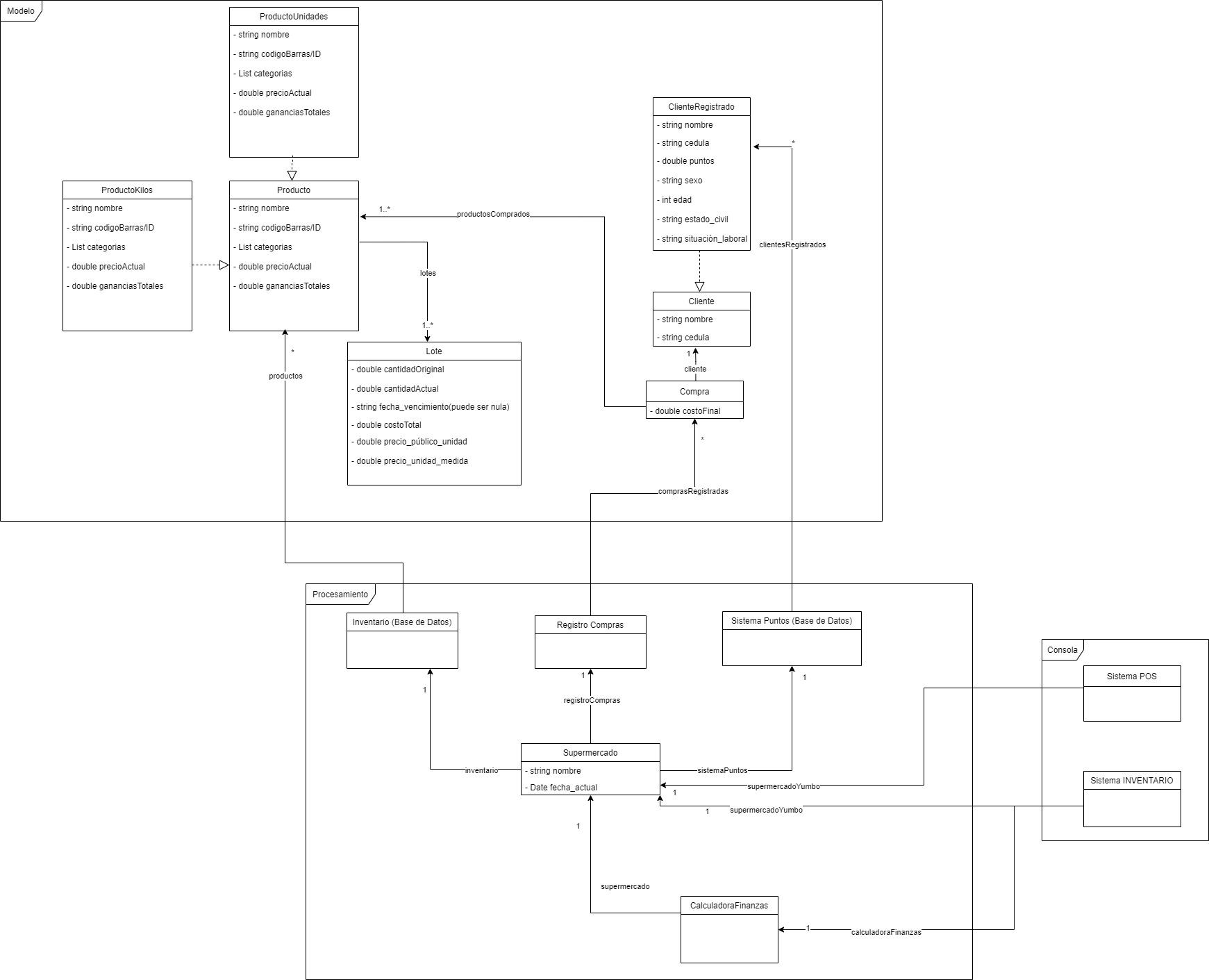
**Modelo:**

* Lote
* Producto
* Cliente
* Compra

1. Un diagrama de clases de diseño que incluya todas las clases, incluyendo sus las relaciones, atributos y métodos.

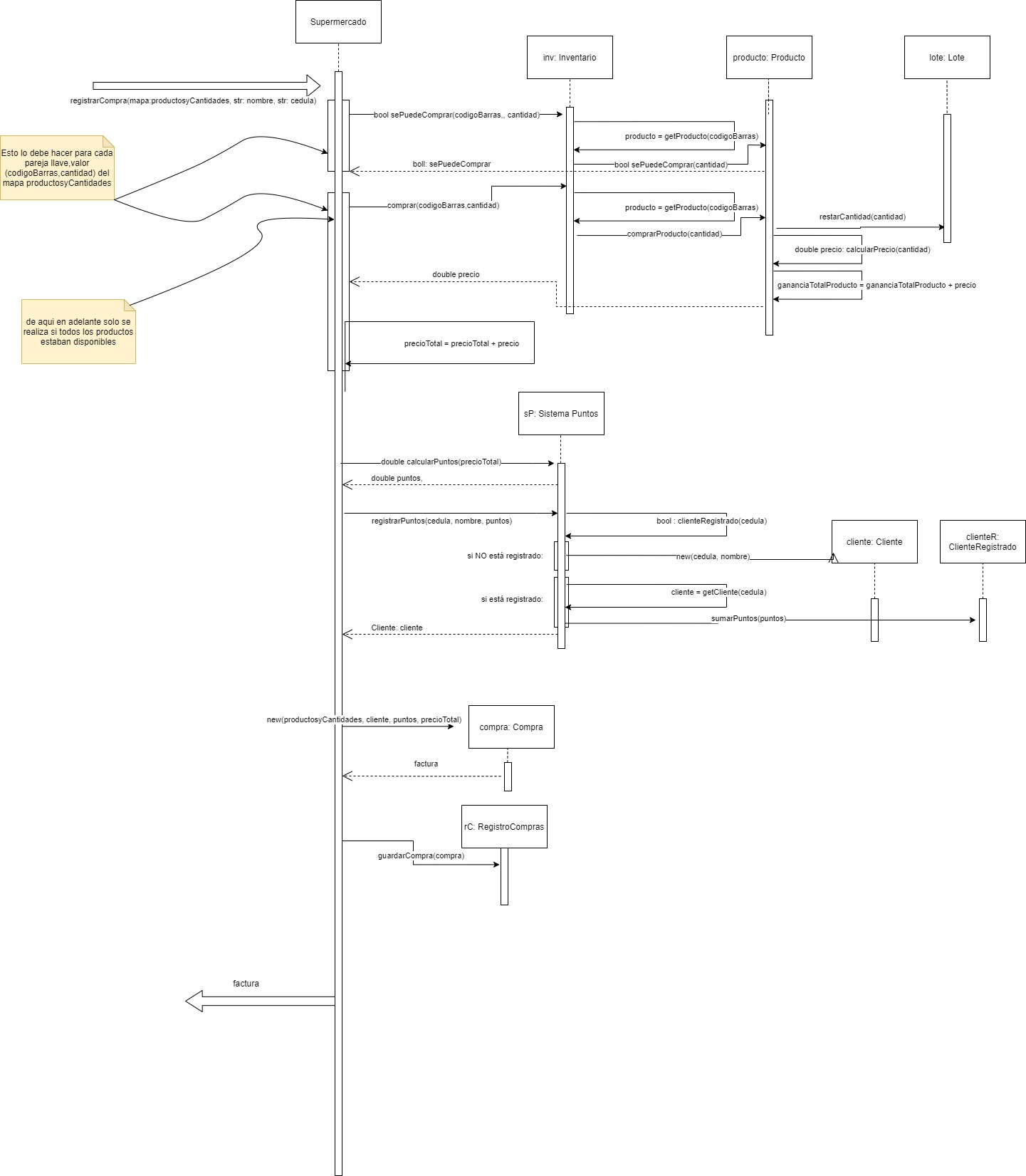


1. Un diagrama de clases de alto nivel, que incluya todas las clases y sus relaciones, pero no todos los métodos ni atributos. Este diagrama facilitará entender las relaciones entre clases.

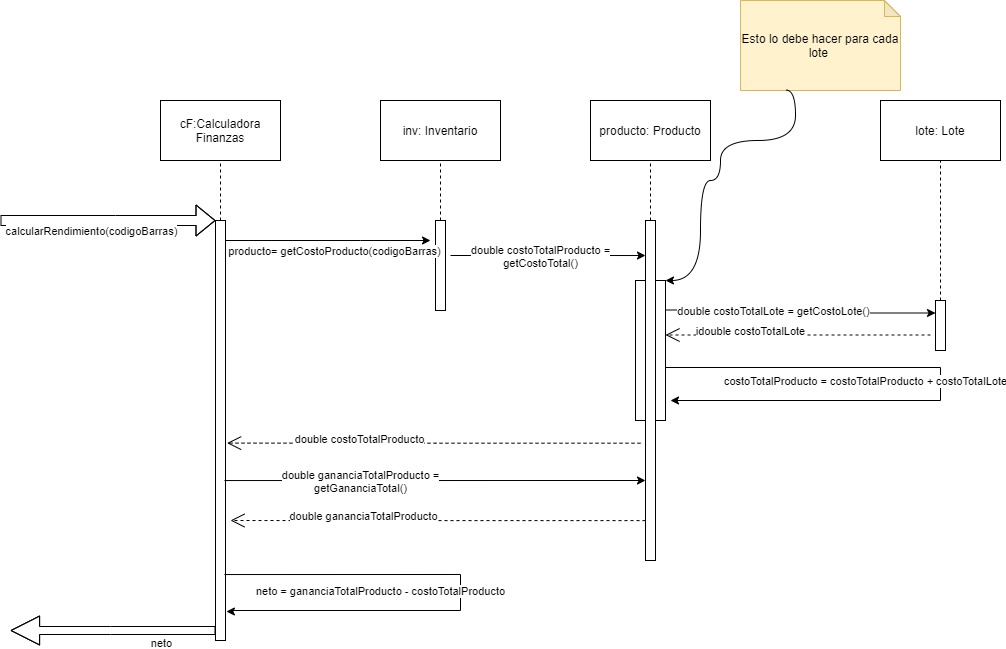


1. Diagramas de secuencia para las funcionalidades que usted considere críticas:

* Registrar compra.



* Calcular desempeño financiero de producto.



(Los diagramas también se encuentran en la carpeta del proyecto)

**Lista de responsabilidades:**

Sistema puntos:

* Registrar a un cliente en el sistema de puntos
* Conocer los clientes registrados en el sistema de puntos
* Calcular puntos generados por compra (parte de realizar compra)

Registro Compras:

* Conocer las compras registradas

Compra:

* Registrar la compra con el cliente así este inscrito en el sistema de puntos o no (parte de realizar compra)
* Generar o mostrar un recibo del resumen de la compra (parte de realizar compra)

Inventario:

* Agregar productos (incluyendo agregar categorías de productos), cada producto viene en lotes con cierta cantidad de instancias del mismo, una fecha de vencimiento, un precio de pago al proveedor, un precio de venta al público (podría incluir fecha de adquisición, pero no se incluirá porque no parece ser pertinente). Al agregarse un lote, se cambia el precio de venta al público de ese producto. Esta entrada de lotes de producto no se hace lote por lote por consola, sino que se debe leer un archivo que puede tener varios lotes de distintos productos. (Todo el archivo podría ser una sola fecha de adquisición de los lotes que contiene)
* Consultar la información del inventario disponible de cada producto, es decir conocer la fecha de vencimiento de cada lote y ver cuántos productos quedan de cada lote.
* Eliminar los lotes vencidos de un producto
* Tener en cuenta que no se permite comprar productos que no están en inventario (los vencidos si se pueden comprar si ningún encargado de inventario los ha eliminado del inventario) (parte de realizar compra)

Lote:

* Actualizar inventario restando los productos comprados (parte de realizar compra)

Calculadora Finanzas:

* Consultar el desempeño financiero de cada producto (historial de ganancias y pérdidas)

Producto:

* Conocer (y actualizar cuando llegan nuevos) sus lotes incluyendo sus costos
* Conocer (y actualizar cuando se realiza compra) sus ganancias totales
* Tener en cuenta que no se permite comprar productos agotados (que la cantidad de la compra es mayor a la disponible)

Cliente:

* Saber sus datos y su cantidad de puntos (y actualizarlos con cada compra)

Supermercado:

* Tener acceso a: inventario, registro de compras y sistema de puntos.
* Registrar compras en el registro de compras

Realizar compra será algo dividido entre distintos elementos por lo que necesitará de colaboraciones:

* Realizar compra:
  + Registrar la compra con el cliente así este inscrito en el sistema de puntos o no (RC-> compra)
  + Calcular puntos generados por compra (SP)
  + Generar o mostrar un recibo del resumen de la compra (RC-> compra)
  + Actualizar inventario restando los productos comprados (Inventario-> producto-> lote)
  + Tener en cuenta que no se permite comprar productos que no están en inventario (los vencidos si se pueden comprar si ningún encargado de inventario los ha eliminado del inventario) (Inventario-> producto)

**Argumentaciones de Decisiones de Diseño:**

Probablemente la decisión más básica del diseño, que podría pasar desapercibida por ser tan intuitiva, es separar los datos en las tres clases de tipo bases de datos sobre clientes(puntos), inventario y compras dentro del mismo supermercado. Esto se hace para lograr encapsulamiento de las bases de datos debido a que tenemos dos partidas de inicio (dos consolas por dos usuarios distintos: POS, INVENTARIO), así dependiendo de lo que se quisiera hacer para cualquiera de los dos sistemas se deben cambiar solamente sus bases de datos correspondientes (*Inventario* se usa en el sistema INVENTARIO, *Sistema Puntos* en sistema POS y *Registro Compras* en ambos).

Dentro del módulo *Modelo* hemos agrupado las clases que buscan modelar elementos del mismo problema del mundo a través de la abstracción. Estos serán los objetos puntuales que estarán guardados en las bases de datos agrupando la información deseada. Los métodos que consultan o cambian información sobre las bases de datos en *Supermercado* en *procesamiento*, como por ejemplo *registrarCompra* en *Supermercado* o *calcularRendimiento* en *CalculadoraFinanzas*,finalmente parte de lo que hacen es consultar o cambiar información en las clases de *modelo* que representan a elementos puntuales del mundo real. Cuando se hace esto, llamamos a métodos de las clases de *modelo* dentro del método que se corre en las clases de *procesamiento*.

Una decisión muy importante que hemos tomado con respecto al diseño de la clase *Lote* es que esta no tendrá como atributo a un conjunto de productos que resultarían todos con las mismas propiedades, sino que tendrá un solo producto que representara toda la información que sería constante entre todos los productos del lote y otro atributo que será la cantidad que tiene de estos productos. ¿Esto porque se puede hacer? Aunque sea un cambio en el diseño de la clase *Lote*, es consecuencia de la forma en la que se ve la clase *Producto*: en este diseño una instancia de la clase *Producto* es un producto de la vida real representado por un código de barras. Por ejemplo: una pasta de dientes de marca X sería una instancia de la clase *Producto*; la pasta de dientes de marca X en la góndola Z y la pasta de dientes de marca X en las manos de un empleado del supermercado NO serían dos instancias de la clase *Producto* (se podrían abstraer y entonces corresponderían a la misma instancia). Esto implica como muestra el ejemplo que no podemos diferenciar entre dos “pastas de dientes de la marca X” distintas en nuestro programa, lo cual por ahora no es necesario y estamos apostando a que tampoco lo será en las próximas actualizaciones (proyecto 2 y 3).

Otra decisión de diseño que se hizo fue que toda compra tiene un cliente. Un cliente solamente debe consistir en una cédula y un nombre junto unos puntos, pero en caso de que esté registrado entonces debe tener mucha más información que es la que el supermercado exige para la inscripción en el sistema de puntos. Se pudo haber hecho que la clase *Cliente* fuera en realidad la clase *ClienteRegistrado* y las compras no tuvieran un cliente sino solo una cedula y un nombre de cliente y los puntos de la compra, pero decidimos hacerlo así para agrupar esta información sabiendo que corresponde a la información del cliente de la compra y la posibilidad de herencia facilita tener la versión específica de los clientes registrados. Además, tenemos el método sumar puntos que siempre se ejecutara en el registro de una compra, y al tener dos clases distintas para el cliente registrado y el no registrado, el método en la clase del no registrado no hará nada por dentro por ahora, y el método de la clase del registrado le sumara a su registro de puntos.